Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Task Scheduling on Spacecraft by Hybrid Genetic Algorithms

Il-Jun Jeong[1], George Papavassilopoulos[1], and David S. Bayard[2]

[1] Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089

[2] Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

Abstract

A genetic algorithm is developed for scheduling tasks on a spacecraft having constrained resources. Due to the complex nature of the constraints, it is necessary to take a hybrid approach where the genetic algorithm is combined with a decoder routine to generate feasible solutions. The performance of the hybrid algorithm is demonstrated by example. Results are very encouraging, demonstrating an improvement relative to earlier results, and providing a flexible capability to modify existing sequences in near-real-time to comply with time-varying goals and constraints.

1 Introduction

This paper addresses the problems of scheduling tasks on a spacecraft having constrained resources. The main emphasis is on developing fast and effective on-board algorithms that allow the spacecraft to schedule its own activities, and operate autonomously in a time-varying and/or uncertain environment.

Typically, spacecraft resources of interest include power, propellant, computer memory (RAM, magnetic tape, solid-state mass data storage), computer throughput (CPU time), accumulated wheel momentum, science instrument utilization, cryogen utilization, communication (telemetry bandwidth, transmission time), or any other resource of finite quantity that impacts in-flight operations.

Many times in the course of a spacecraft mission, an unexpected event, anomaly, or unanticipated science opportunity arises. The standard response is to have spacecraft activities rescheduled on the ground (Earth), suffering delays from both ground latencies (associated with setting up and optimizing the modified sequences), as well as a two-way communication delay in receiving and sending relevant information through space. This approach is costly and inefficient, and must be supported by a fully staffed ground infrastructure. Furthermore, in certain cases of time-critical events, an immediate response is required and such delays cannot be tolerated. An alternative approach is to develop the capability for on-board scheduling of activities.

The capability for on-board sequencing is a long term goal of the present research effort, and is in keeping with recent trends toward developing smarter and more effective "autonomous" spacecraft [3, 11]. In this report, we select genetic algorithms [7] to solve these problems. The reasons that we select genetic algorithms are their robustness and good performance. They are also quite widely applicable and can be implemented as parallel algorithms [7].

Genetic algorithms are population based optimization algorithms. Each individual of the population has a fitness value, which indicates the quality of the individual for a given function. From the population, the next generation is produced by three operators: selection, crossover, and mutation operator. Generally, the population size is taken to be the same from generation to generation.

The common procedures of genetic algorithms are as follows. First, calculate the fitness values of randomly generated individuals. Then, select superior individuals based on their fitness values. Modify the selected individuals a little by crossover and mutation. Then, recalculate new fitness values for the modified individuals. The above procedures are repeated until certain stopping criteria are met. Detailed explanations can be found in [7, 10].

The idea of the crossover resembles the DNA's crossover in nature. Keep in mind that the crossover of the genetic algorithms does not automatically generate better individuals than the previous ones. It only produces different individuals. The mutation operator alters each individual randomly, and the number of the mutation operations is governed by a mutation rate. It is shown that if the mutation rate is not zero and the best individual is preserved, genetic algorithms asymptotically converge to a global optimum [12].

0-7803-5180-0-5/99 $10.00 © 1999 IEEE

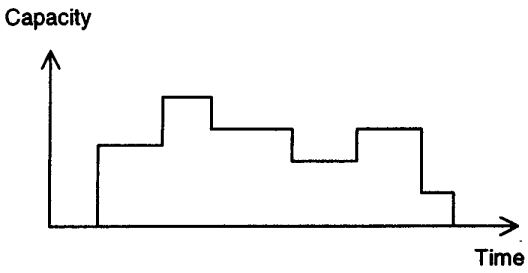

Figure 1 Resource capacity is represented by a stairstep function in this paper. The resource demand of a task also can be described by the stairstep function.

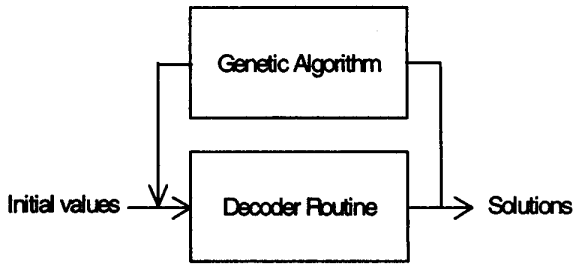

Figure 2 Decoder routine generates feasible solutions for genetic algorithms. Genetic algorithms produce the next input for the decoder routine.

For constrained optimization problems, the efficiency of genetic algorithms can be greatly improved by adding a decoder routine [10]. We suggest one decoder routine generating feasible solutions. The routine serves as a function calculating fitness values of individuals. The results of our methods are quite promising.

The structure of this paper is as follows. In Section2, we define scheduling problems of a spacecraft. In Section 3, we present a decoder routine and explain an order-based genetic algorithm. Two cases of problems are solved by our hybrid genetic algorithm in Section 4. Finally, brief concluding remarks are given in Section 5.

2 Problem Statement

Let us assume that there are N tasks and M resources. Each task T_i requires execution time d_i and resource demand $r_{i,j}$ on resource R_j. Now, a task T_i can be specified by a function $f_{i,j}(t - t_i)$:

$$f_{i,j}(t-t_i)=\begin{cases} r_{i,j} & t_i \le t \le t_i + d_i \\ 0 & otherwise. \end{cases} \quad (1)$$

where, $i=1,2,..,N$ and $j=1,2,...,M$. The t_i is the start time of the task T_i. By arranging the start time t_i, we should place all N tasks within a given interval D and maximum resource capacity $C_j(t)$ of resource R_j:

$$\sum_{i=1}^{N} f_{i,j}(t-t_i) \le C_j(t) \quad (2)$$

where $t_i \in [0,D]$, $t \in [0,D]$, and $j = 1, 2, ..., M$. The maximum resource capacity $C_j(t)$ is generally continuous function and is converted to a stairstep function in this paper (figure 1).

Sometimes, not all tasks can be scheduled while satisfying all constraints. For such case, some less important tasks should be removed from scheduling. To distinguish the importance of each task, a utility value $w_i \ge 0$ is used. By convention, the higher the w_i, the more important the task. Now, the scheduling problem becomes:

Problem Find $e_i \in \{0,1\}$ and $t_i \in [0,D]$, so that

$$\max_{t_i, e_i} \sum_{i=1}^{N} w_i e_i \quad (3)$$

subject to

$$\sum_{i=1}^{N} e_i f_{i,j}(t-t_i) \le C_j(t) \quad (4)$$

where $w_i \ge 0$, $t \in [0,D]$, $i = 1,2,...,N$, and $j=1,2,..,M$. The e_i indicates whether the task T_i is selected or not: $e_i = 1$ means the task T_i is selected, and $e_i = 0$ means the task T_i is not selected.

Three types of ordering constraints are considered. A *sequencing constraint* makes sure that some tasks are performed before others. If a task T_i should be completed before a task T_i^*, then following constraint is necessary:

$$t_i + d_i + \tau_i \le t_i^* \quad (5)$$

where a parameter $\tau_i \ge 0$ is used for an additional interval between two tasks. A *non-overlapping constraint* does not permit overlapping of two tasks. It can be expressed by:

$$t_j + a_j + \tau_i \le t_i^* \text{ or } t_i^* + a_i^* + \tau_i^* \le t_i \quad (6)$$

```
input = order of tasks
output = start times of the tasks

for time = first to last
  for task = first to last
    if task satisfies ordering constraints
      if task satisfies resource constraints
        start time of the task = time;
      end
    end
  end
end
```

Figure 3 Algorithm of a decoder routine generating feasible solutions for genetic algorithms.

With this constraint, task T_i and task T_i^* cannot be executed at the same time. Finally, some tasks require an *absolute start time constraint*:

$$t_i = t \tag{7}$$

For example, many observations need to be performed at certain times because of the spacecraft's location.

3 Genetic Algorithms for Constraints

For constrained optimization problems, the efficiency of genetic algorithms can be greatly improved by adding a decoder routine. The decoder routine generates feasible solutions, which satisfy all constraints. In Figure 2, one method of a hybrid genetic algorithm, which takes advantage of the decoder routine, is suggested. The decoder routine generates feasible solutions, and the genetic algorithm produces the next input for the decoder routine.

3.1 Decoder Routine

One decoder routine is suggested and its algorithm is shown in Figure 3. The routine takes an order of tasks as input and generates start times of the tasks as output. From the earliest scheduling time and from the first task of the given order, the routine checks whether the task satisfies both ordering constraints and resource constraints. If the task satisfies both constraints, then the routine places the task at the time. The placed task affects the resource constraints and possibly ordering constraints for other tasks. The above procedures continue until no more tasks are available (i.e. all tasks are selected) or the time reaches the end of the intended period.

The order of tasks determines the quality of output. Thus, we use a genetic algorithm to find the best order. Generally, tasks appearing earlier in the ordering have a better chance of being selected.

Other methods can be used to define a decoder routine. For example, we can randomly generate a schedule, then remove tasks violating constraints. It is also possible that after removing the tasks, we can relocate unselected tasks to any remaining resource.

3.2 Order-Based Genetic Algorithms

Because, our decoder routine requires the order of tasks as input, an order-based genetic algorithm [5] is used. If the number of tasks is N, then an individual will be a permutation of $\{1, 2, ..., N\}$. For example:

$$X = [2\ 3\ 4\ 7\ 1\ 8\ 5\ 6\ 9]$$

is a typical example of the individual for nine tasks.

Based on the rank of fitness values [1], the individuals of the next population are selected by the roulette wheel selection method [7]. If N_p is the number of individuals in the population and R is the rank ($R = 1$ means the worst individual, and $R = N_p$ means the best individual) of an individual, then the probability of selecting the individual will be:

$$Prob(R) = \frac{2R}{N_p(N_p+1)}.$$

We select the order based crossover operator [13]. The order based crossover operator chooses some elements of one parent, and reorders them according to the order of the other parent. For example, suppose that two parents are:

$$X_1 = [9\ 8\ 2\ 1\ 3\ 4\ 5\ 6\ 7]$$
$$X_2 = [4\ 6\ 5\ 2\ 8\ 7\ 9\ 1\ 3].$$

Assume that the 3rd, 5th, and 6th elements are selected for the crossover operation. They are underlined below:

$$X_1 = [9\ 8\ \underline{2}\ 1\ \underline{3}\ \underline{4}\ 5\ 6\ 7]$$
$$X_2 = [4\ 6\ \underline{5}\ 2\ \underline{8}\ \underline{7}\ 9\ 1\ 3].$$

The elements 8, 5, and 7 of X_1 are replaced by the underlined elements 5, 8, and 7 of X_2 respectively. Similarly, the elements 4, 2, and 3 of X_2 are replaced by the underlined elements 2, 3, and 4 of X_1. Thus, new offsprings are:

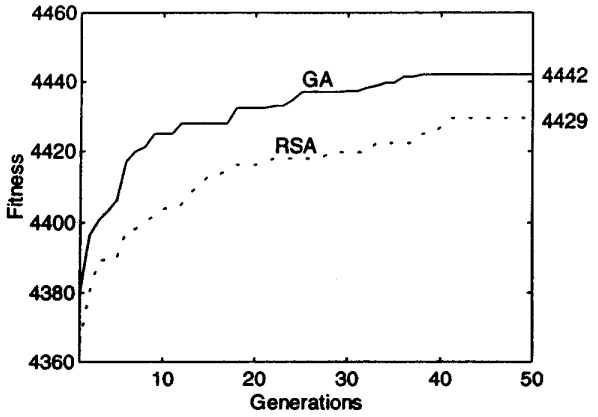

Figure 4 The results of genetic algorithms (GA) and random search algorithms (RSA) of case I. The average values of 10 runs are used to generate graphs.

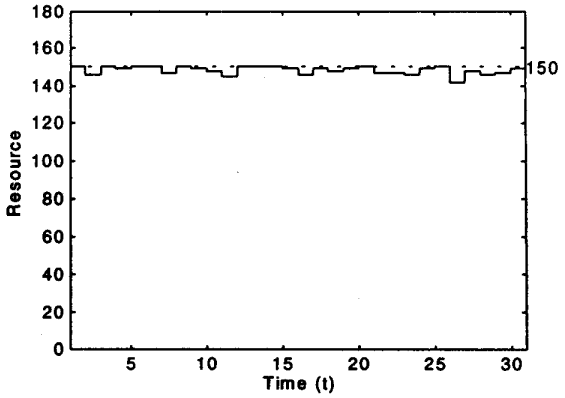

Figure 5 One scheduling result by our hybrid genetic algorithm. The dotted line shows maximum resource capacity (resource constraint) and the solid lines shows the total resource demands of tasks at each time unit.

$$Y_1 = [9\ 5\ 2\ 1\ 3\ 4\ 8\ 6\ 7]$$
$$Y_2 = [2\ 6\ 5\ 3\ 8\ 7\ 9\ 1\ 4].$$

We select the exchange mutation operator [2], which simply exchanges the position of two randomly selected elements. For example, if 5 and 3 are selected for mutation, then:

$$X = [9\ \underline{5}\ 4\ 7\ 1\ 8\ 2\ 6\ \underline{3}] \quad \text{(before mutation)}$$
$$Y = [9\ \underline{3}\ 4\ 7\ 1\ 8\ 2\ 6\ \underline{5}]. \quad \text{(after mutation)}$$

4 Experimental Studies

We test our method with two cases. The first case involves scheduling 100 tasks under one resource constraint and 14 ordering constraints. The second case involves scheduling tasks under a dynamic environment: tasks are canceled and added, resource capacities vary, and ordering constraints appear during scheduling.

The results of the genetic algorithm are compared with the results of a random search algorithm, which randomly generates solutions and keeps the best for each generation. Note that random search algorithm also uses the decoder algorithm. Thus, the comparison does not reflect the effectiveness of the decoder algorithm.

4.1 Case I

The first case involves placing 100 tasks within 30 horizontal locations (time) and 150 vertical locations (resource). A task T_i requires one resource demand r_i, which is a value between 1 and 10, and time duration d_i, which is a number between 1 to 50.

The utility value w_i is calculated by multiplying r_i and d_i. The fitness value of each individual is evaluated by

$$\sum_{i=1}^{N} w_i e_i \ . \qquad (8)$$

The maximum fitness value cannot exceed 30 x 150 = 4500, because the fitness value is an area occupied by selected tasks. The actual optimum fitness value is indeed 4500, when all 100 tasks are selected.

In addition to the resource constraint, each individual should satisfy the following ordering constraints:

$$t_{44} + d_{44} \le t_{43} \qquad t_{57} + d_{57} \le t_{51}$$
$$t_{20} + d_{20} \le t_2 \qquad t_6 + d_6 \le t_{97}$$
$$t_4 + d_4 \le t_{49} \qquad t_{44} + d_{44} \le t_{71}$$
$$t_{36} + d_{36} \le t_{85} \qquad t_{76} + d_{76} \le t_{65}$$
$$t_9 + d_9 \le t_{37} \qquad t_{93} + d_{93} \le t_{16}$$
$$t_{10} = 20 \qquad t_{50} = 15$$
$$t_1 + d_1 \le t_2 \text{ or } t_2 + d_2 \le t_1$$
$$t_{28} + d_{28} \le t_{77} \text{ or } t_{77} + d_{77} \le t_{28}.$$

The results of the genetic algorithm (solid line) and the random search algorithm (dotted line) are shown in Figure 4. The results are based on the average values of the 10 runs for each algorithm. In Figure 5, one scheduling result by our method (solid line) is shown. Only two tasks are not selected for this result. The fitness value 4446 of this result is 98.8% of the optimal value 4500. Although the result of the random search algorithm is worse than the result of the genetic algorithm, they both provide good solutions due to the use of the decoder algorithm.

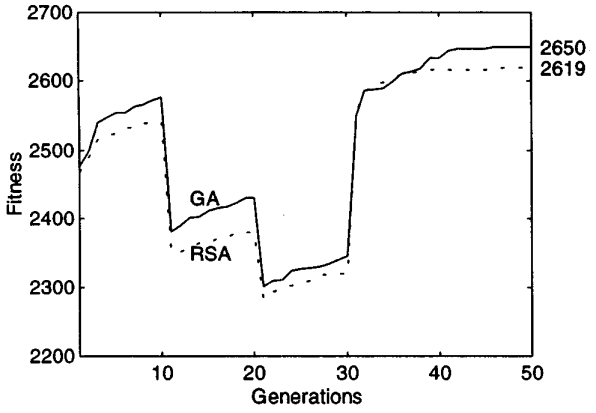

Figure 6 The results of genetic algorithms (GA) and random search algorithms (RSA) of case I. The average values of 10 runs are used.

4.2 Case II

During a space mission, sudden changes can occur due to unexpected circumstances and/or a changing space environment. For example, tasks may have to be rescheduled in light of failed observations (i.e., lack of target acquisition), or conversely, the desire to take additional science observations of unexpected target opportunities. In addition, resource constraints may change unexpectedly due to component failures. This simulation covers these cases.

For this problem, a task T_i requires four resource demands $r_{i,1}$, $r_{i,2}$, $r_{i,3}$, $r_{i,4}$, which are randomly generated between 0 and 5, and time duration d_i, which is randomly generated between 1 to 5. The w_i is calculated by

$$w_i = (r_{i,1} + r_{i,2} + r_{i,3} + r_{i,4}) \times d_i \qquad (9)$$

Like the previous problem, the fitness value of each individual is given by (8). The number of tasks is 100 and the number of horizon locations is 30.

We assume that changes occur during scheduling operation: more specifically at the 11th, 21st, and 31st generations.

1. At the beginning:

$C_1(t) = 25 \quad 1 \le t \le 30$

$C_2(t) = 20 \quad 1 \le t \le 5, 11 \le t \le 15, 21 \le t \le 25$

$\quad\quad 30 \quad 6 \le t \le 10, 16 \le t \le 20, 26 \le t \le 30$

$C_3(t) = 35 \quad 1 \le t \le 10, 21 \le t \le 30$

$\quad\quad 15 \quad 11 \le t \le 20$

$C_4(t) = 40 - t \quad 1 \le t \le 30$

No ordering constraint

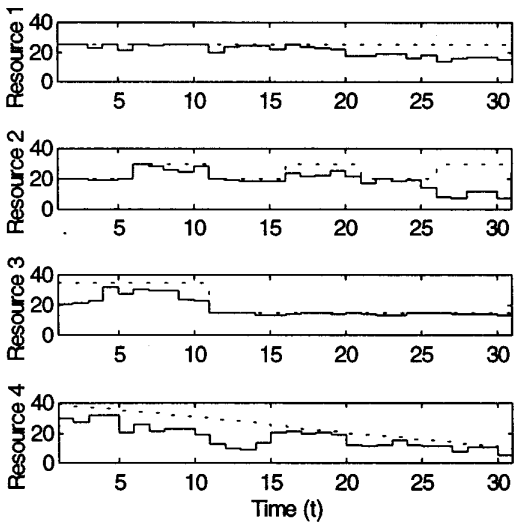

Figure 7 One example of scheduling by hybrid genetic algorithm. The dotted line shows maximum resource capacity and the solid lines shows the summation of resource demands of selected tasks.

2. At the 11th generation: Resource 3 changes

$C_3(t) = 35 \quad 1 \le t \le 10$

$\quad\quad 15 \quad 11 \le t \le 30$

3. At the 21st generation: New ordering constraints appear

$$t_{20} + d_{20} + \tau_{20} \le t_{19} + d_{19} + \tau_{19} \le \cdots \le t_2 + d_2 + \tau_2 \le t_1$$

where, $\tau_{20} = \tau_{19} = \ldots = \tau_2 = 1$

4. At the 31st generation:

Task 51 to 60 are cancelled

New Tasks 101 to 105 are assigned

From Figure 6, we can notice severe transitions at the 11th, 21st, and 31st generation. They are caused by the above changes. These results also show that the genetic algorithm performs better than the random search algorithm.

The four constraints are plotted in Figure 7 as dotted lines along with one scheduling result of our hybrid genetic algorithm (solid line). For multiple resource constraint problems, some resources can be more dominant during a certain period. For example, for $11 \le t \le 30$, placing tasks is mostly restricted by the availability of resource 3.

It is hard to measure the quality of the scheduling result, because the optimal fitness value is not known in

this case. However, because only small space is left especially under the dominant resources, we may assume that the solution is more than acceptable.

5 Conclusion

A method for scheduling tasks has been developed relevant to spacecraft applications. Such problems inherently involve both resource constraints and task ordering constraints. For simple genetic algorithms, those constraints result in many infeasible individuals, which decrease the efficiency of the algorithms. Thus, to recover the efficiency, a decoder routine that generates feasible individuals is suggested. Because, our decoder routine takes an ordering of tasks as input, an order-based genetic algorithm is selected and combined with the decoder routine.

This hybrid genetic algorithm was applied to two experimental problems. Results indicate that the algorithm finds good solutions quickly, on problems of reasonable size. However, actual spacecraft scheduling problems can be of much larger dimension, and this remains as an area for further investigation. Comparing to earlier results on similar size problems [3], the use of the decoder algorithm in combination with the GA runs significantly faster than the GA used alone, and with improved performance. Overall, the results of the study are very encouraging and take us one step closer towards providing the capability for a spacecraft to schedule its own activities, and to modify existing sequences in near-real-time to comply with time-varying goals and constraints. Practical applications that could benefit from this adaptive resequencing capability are numerous and include autonomous rover exploration, planetary flybys, maintaining autonomous outposts in space, and serendipitous science imaging.

Acknowledgments

We are grateful to David H. Collins and William "Curt" Eggemeyer of the Jet Propulsion Laboratory for helpful discussions, and to Hamid Kohen of JPL for additional software support on this project. The research described in this document was carried out for the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] J. E. Baker, "Adaptive selection methods for genetic algorithms," *Proceedings of an International Conference on Genetic Algorithms and Their Applications,* pp 101-111, 1985.

[2] W. Banzhaf, "The 'molecular' traveling salesman," *Bio. Cybern.,* vol. 64, pp. 7-14, 1990.

[3] D. S. Bayard, G. Papavassilopoulos, and H. Kohen, *Genetic algorithms for space autonomy: An optimization approach to on-board sequencing,* JPL Internal Document JPL D-15347, 1997.

[4] S. Chien, D. DeCoste, R. Doyle, and P. Stolorz, "Making an Impact: Artificial Intelligence at the Jet Propulsion Laboratory," American Association for Artificial Intelligence, *AI Magazine,* pp. 103-122, Spring 1997.

[5] L. Davis, Ed., *Handbook of Genetic Algorithms,* New York: Van Nostrand Reinhold, 1991.

[6] K. A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems,* Doctoral dissertation, University of Michigan, 1975.

[7] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning,* Addison Wesley, 1989.

[8] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," *Proceedings of an International Conference on Genetic Algorithms and Their Applications,* pp 154-159, 1985.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems,* The University of Michigan Press, 1975.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* Springer-Verlag, N.Y., 1994.

[11] N. Muscettola, C. Fry, K. Rajan, B. Smith, S. Chien, G. Rabideau, and D. Yan, "On-Board Planning for New Millennium Deep Space One Autonomy," *Proceedings of IEEE Aerospace Conference,* Snowmass, CO., 1997.

[12] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Transactions on Neural Networks,* vol. 5, no. 1, pp. 96-101, 1994.

[13] G. Syswerda, "Scheduling optimization using genetic algorithms," in [5], pp. 332-349.