# Scheduling Earth Observing Fleets Using Evolutionary Algorithms: Problem Description and Approach

Al Globus, James Crawford, Jason Lohn and Anna Pryor

March 6, 2003

## 1 Abstract

We hypothesize that evolutionary algorithms can effectively schedule coordinated fleets of Earth observing satellites. The constraints are complex and the bottlenecks are not well understood, a condition where evolutionary algorithms are often effective. This is, in part, because evolutionary algorithms require only that one can represent solutions, modify solutions, and evaluate solution fitness.

To test the hypothesis we have developed a representative set of problems, optimization software (in Java) to solve them, and run experiments. This paper presents initial results including a comparison of separate vs. integrated scheduling of a two satellite constellation and a comparison of several evolutionary and other optimization techniques; namely genetic algorithms, simulated annealing, squeaky wheel optimization, and stochastic hill climbing.

## 2 Introduction

A growing fleet of NASA, commercial, and foreign Earth observing satellites (EOS) uses a variety of sensing technologies for scientific, mapping, defense and commercial activities. As the number of satellites (now around 60) increases, the system as a whole will begin to approximate a sensor web. Image collection for these satellites is planned and scheduled by a variety of techniques (Muraoka et al. 1998, Potter and Gasch 1998, Sherwood et al. 1998, and others), but nearly always as separate satellites or even instruments, not as an integrated sensor web. Since activities on different satellites or even different instruments on the same satellite are typically scheduled independently of one another, the manual coordination of observations by communicating teams of mission planners is required. As sensor webs with large numbers of satellites and observation requests develop, manual coordination will no longer be possible. Schedulers that treat the entire web as a collection of resources will become necessary. There has been some work toward automatic scheduling of satellite fleets, e.g., Rao, et al. reported scheduling ground station use, but not imaging activity, for a fleet of seven Indian Earth imaging satellites (Rao et al. 1998), and we present our initial results here.

Scheduling EOS is complicated by a number of important constraints. Potin (Potin 1998) lists some of these constraints as:

1. Power and thermal availability.
2. Limited imaging segments per orbit.
3. Time required to take each image.
4. Limited on-board data storage.
5. Transition time between look angles (slewing).
6. Revisit limitations.
7. Cloud cover.
8. Stereo pair acquisition.
9. Ground station availability, especially playback opportunities.
10. Coordination of multiple satellites.

For further detail on the EOS scheduling problem see Sherwood et al. 1998 and Frank et al. 2002.

We hypothesize that evolutionary algorithms can effectively schedule Earth imaging satellites, both single satellites and cooperating fleets. The constraints on such fleets are complex and the bottlenecks are not always well understood, a condition where evolutionary algorithms are often more effective than traditional techniques. Traditional techniques often require a detailed understanding of the bottlenecks, whereas evolutionary programming requires only that one can represent solutions, modify solutions, and evaluate solution fitness, not actually understand how to reason about the problem or which direction to modify solutions (no gradient information is required, although it can be used).

To test the hypothesis we have developed a (hopefully) representative set of problems and software to compare solutions generated by various evolutionary and other optimization techniques. We also present data comparing scheduling a two satellite constellation as a (small) sensor web vs. as separate systems.

The next section describes the model problems. This is followed by a description of the optimization technique comparison software, the results, and future plans. Further details on the model problems and software, including parallelization techniques, may be found in (Globus et al. 2002).

## 3 Model Problems

Since our project is designed to consider the scheduling of a parameterizable generic system, not any particular spacecraft, sensor, or sensor web, it is important to develop a set of model problems that exhibit the important aspects of EOS scheduling now and in the future. We have attempted to base our model sensors and satellites on hardware currently in orbit. We have identified and begun to scope seven problems:

1. A single satellite with a single cross-track slewable instrument.
2. A two satellite constellation with identical satellites from problem one.
3. A single agile satellite with one instrument. Here the whole spacecraft is slewed, rather than the instrument relative to the spacecraft. This allows slewing both cross-track and along-track.
4. A single satellite with multiple instruments (one of which is slewable).
5. A sensor web of single- and multiple-instrument satellites communicating directly with the ground.
6. A sensor web of single-instrument agile satellites communicating with an in-orbit communications system based on high-data-rate lasers.
7. A sensor web with a very large number of satellites including satellites with multiple instruments. This problem presumes much cheaper and more reliable launch.

Problems 1 and 2 have been implemented. The Results section compares a number of search techniques against problem 2 with the following characteristics

1. one week of satellite operations.
2. two satellites in sun synchronous orbit one minute apart.
3. each satellite had the same single instrument.
4. instruments could slew up to 48 degrees cross-track in either direction at a rate of 50 seconds/degree.
5. 4200 imaging targets (takeImages) were randomly distributed around the globe; 123 of these never came into view of either satellite.
6. each takeImage requires 24 seconds of viewing.
7. each takeImage had a priority between 1 and 5 (higher priority is more important).
8. each run (a combination of a particular search technique and transmission operators) was repeated 94 times (94 jobs).
9. 100,000 schedules were produced and scored for fitness by each job.

## 4 EOS Scheduling by Evolutionary Algorithms and Other Optimization Techniques

There are a number of optimization (evolutionary and otherwise) algorithms in the literature. We compare a genetic algorithm (GA), simulated annealing (SA), and stochastic hill climbing (HC) to address EOS scheduling. In addition, we compare random and squeaky wheel (SW) transmission operators. Note that although these techniques can be thought of as evolutionary algorithms, not all practitioners would classify them this way.

GAs seek to mimic natural evolution's ability to produce highly functional objects. Natural evolution produces organisms, whereas GAs can produce schedules, programs, molecular designs, and many other structures. Our GA employs the following algorithm:

1. Represent each schedule with a permutation (ordering) of the imaging tasks (explained below)
2. Generate a population of random permutations
3. Calculate the fitness of each permutation
4. Repeat
   (a) Randomly select parent permutations with a bias towards better fitness
   (b) Produce child permutations from the parents with a transmission operator:
      i. crossover that combines parts of two parents into a child, or
      ii. mutation that modifies a single parent
   (c) Calculate the fitness of the child
   (d) Randomly replace individuals of less fitness in the population with the children
5. Until satisfied according to some minimal convergence criteria

In this paper we compare three search algorithms:

1. GA (the genetic algorithm described above)
2. HC (stochastic hill climbing) can be thought of as a genetic algorithm with a population of one where only fitter children are allowed to replace the parent. Two cases are investigated: five restarts per run and no restarts.
3. SA (simulated annealing) is similar to HC except less fit children can replaces the parent with probability $p = e^{\frac{-\triangle F}{T}}$ where $\triangle F$ how much less fit the child is, and the temperature $T$ starts at 100 and is multiplied by 0.92 every 1000 children (100,000 children are generated per run).

Evolution is guided by a fitness function. The fitness function must provide a fitness for any possible permutation, no matter how bad the schedule is, and distinguish between any two permutations, no matter how close they are. Our fitness function is multi-objective. The objectives include:

1. Maximize the sum of the priority of the images taken (takeImages).
2. Minimize total time spent slewing (slew motors wear out).
3. Minimize the sum of the slew angles for the images taken (small slews improve image resolution).

These are manipulated so that lower values are better fitness and then combined into a weighted sum.

We represent schedules as an permutation, or ordering, of the image requests (takeImages). A simple greedy scheduler assigns times and resources to the requested takeImages in the order indicated by the permutation. This scheduler currently assigns takeImages using earliest-first scheduling heuristics; maintaining consistency with sensor availability, onboard memory (SSR) and slewing constraints. If a takeImage cannot be scheduled without violating constraints, it is left unscheduled and the scheduler attempts to schedule the next takeImage in the permutation. Simple earliest-first scheduling had some problems, and we discovered that the algorithm works better if 'earliest-first' starts with a particular imaging window (period where the satellite is within sight of a target; most takeImages have several windows in our week-long problem) rather than epoch (time = 0); wrapping around if the takeImage cannot be scheduled before the end of time. The takeImages's scheduled imaging window is remembered and used by children when they generate schedules. This approach may work better than true earliest-first because of the additional scheduling flexibility provided by the window information.

Constraints are enforced by representing each resource as a timeline. Each timeline then takes on appropriate values (i.e., in use for a sensor, slew motor setting, SSR memory available) at different times. A takeImage is inserted at the first time examined and available in all the required resource timelines.

Evolution requires mutation and crossover operators to create children from parents. This paper compares four mutation operators and one crossover operator. The mutation operators are:

1. Random swap. Two permutation locations are chosen at random and the takeImages swapped. Swaps are executed $n$ times per mutation. A single random swap is called order-based mutation (Syswerda and Palmucci 1991).

2. Squeaky swap. This is the same as random swap except that the takeImages to swap are chosen more carefully. Specifically, a tournament of size $n$ selects both takeImages. One takeImage that 'should' be moved forward in the permutation is chosen. The winning takeImage is (in this order):

   (a) unscheduled rather than scheduled

   (b) higher priority

   (c) later in the permutation

   The other takeImage is chosen assuming it should be moved back in the permutation. This tournament winner has the opposite characteristics. Although the takeImages to swap are chosen because one 'should' move forward in the permutation and the other 'should' move back, this is not enforced. Experiment determined that the desired direction of the swap did not actually occur nearly as much as expected, sometimes less than half the time!

3. Placed squeaky swap. Here the direction is enforced. A separate tournament is conducted for each takeImage. The takeImage to move forward is forced to be in the last half of the permutation. The takeImage to move back is then forced to be at least half way towards the front.

4. Cut and rearrange. The permutation is cut into $n$ pieces and these are put back together in a random order. This is similar to the cut-set based operators used in the traveling salesman problem community.

The crossover operator is only used in the genetic algorithm. The operator is Syswerda and Palmucci's position-based crossover (Syswerda and Palmucci 1991). N permutation positions are chosen. These are copied from the father to the child. The remaining task numbers fill in the other positions in the order they appear in the mother.

In many runs, several different transmission operators were used. In these cases, each child was produced by a randomly chosen transmission operator.

## 5 Results

A comparison of search techniques and transmission operators (various forms of mutation and crossover) can be found in table 5. The techniques at the top of the table produce the best schedules, the techniques at the bottom the worst. A few observations:

1. Simulated annealing is clearly the best search technique.

2. Random swap mutations beat the smarter 'squeaky' mutation where the tasks to swap are chosen more carefully (a counter intuitive result).

3. Multiple swaps are better than a single swap, probably because some moves are impossible with a single swap.

4. Ordering techniques by priority or takeImage rather than fitness doesn't make any difference for the best techniques, and much of the difference that does occur is not statistically significant.

These observations should not, however, be considered definitive. First of all, this is a single problem and results may vary when the full range of the model problems are addressed. Second, the squeaky algorithms can stand

| search algorithm | transmission operators | fitness | priority | takeImage |
|---|---|---|---|---|
| SA | 1-9 swap | 2171 | 1873 | 1199 |
| SA | 1 swap | 2354 | 2077 | 1295 |
| HC 5 restarts | 1-9 swaps | 2539 | 2287 | 1415 |
| HC 5 restarts | 1 swap | 2564 | 2313 | 1429 |
| HC 0 restarts | 1 swap | 2575 | 2327 | 1436 |
| SA | 1 squeaky swap | 2772 | 2527 | 1615 |
| SA | 1 placed squeaky swap | 2814 | 2559 | 1579 |
| HC | 1 squeaky swap | 2868 | 2625 | 1623 |
| GA population = 100 | crossover and 1 swap | 3007 | 2759 | 1558 |
| GA population = 100 | 1-5 cut and rearranges | 3008 | 2754 | 1526 |
| SA | 1-5 cut and rearranges | 3012 | 2737 | 1439 |

Table 1: Comparison of search techniques. Search-technique/transmission operator pairs ordered by mean fitness. Techniques are ordered by fitness (low values are better schedules for all measures). Priority is the sum of the priority of all unscheduled tasks. TakeImage is the number of unscheduled takeImages. All data are the mean of 94 searches. Values are rounded of to the next lowest whole number. All differences are statistically significant (as measured by Student's T-Test) except for fitness: HC with 0 and 5 restarts with 1 swap, and the worst three; priority: only the worst three; and several of the takeImage comparisons.

improvement and may someday outperform the random operators. Nonetheless, if these results stand up, there are some important implications.

1. Simulated annealing requires less memory than the genetic algorithm and does not require crossover operators or a population, making it better performing, more efficient, and easier to implement.
2. Random swaps out perform the 'smarter' squeaky swaps, making random swaps better performing, faster, and easier to implement.
3. One should allow multiple random swaps, in spite of the minor increase in code complexity.

Figure 1 shows the evolutionary history of the best individuals for the best schedules evolved by simulated annealing (SA) , hill climbing (HC), and the genetic algorithm (GA) using the one random swap mutation operator. Notice that although simulated annealing wins in the end, it trails GA until about generation 50 and trails HC until about generation 70. SA seems to be doing a better job of finding and then exploiting a deep minimum. Notice also that all three techniques are still improving the schedule at the end of the run, suggesting that additional evolution (more than 100,000 children) would be rewarded with better schedules.

One unexpected property of the schedules generated was the slewing. Specifically, in order to minimize total slewing time the schedules tended to place takeImages such that the instrument is slewed to extremes (see figure 2). This could perhaps be reduced if the fitness function gave more weight to minimizing the sum of the slews or if the instruments slewed faster (which would be more realistic).

A second experiment compared GA with one swap operator on two problems: in the first, each satellite was randomly assigned half of 4,200 the takeImages; in the second, either satellite was allowed to execute any takeImage. As might be expected, the case where any satellite could take any image produced superior schedules. Specifically, the shared case was able to take about 15% more images. The scheduler computed total slew time for both runs, but there was no statistically significant difference even though the shared case executed more takeImages.

## 6 Future Work

Future work will be focused on expanding table 1 to include more problems and techniques. Specifically, we intend to add:

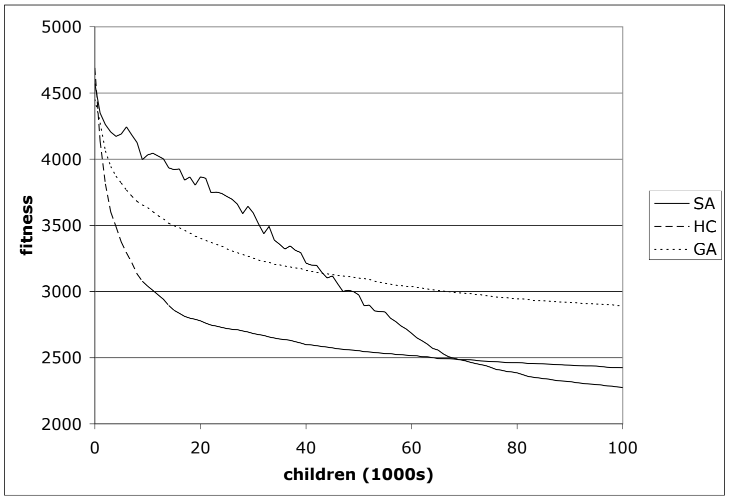

Figure 1: A comparison of the evolutionary history of simulated annealing, hill climbing, and the genetic algorithm. Lower fitness values indicate better schedules.

| scheduling | mean unscheduled takeImages | standard deviation |
|---|---|---|
| shared | 2027 | 80 |
| separate | 2312 | 28 |

Table 2: Comparison of unscheduled takeImage for GA with single swap mutation. In the shared case, either satellite could execute any of 4,200 takeImages. In the separate case, each satellite was randomly assigned 2,100 takeImages. The shared case is about 15% better.

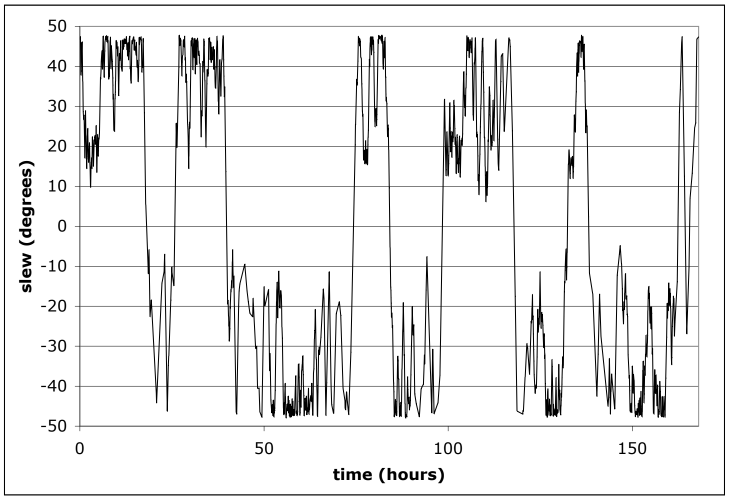

Figure 2: The slew history for one satellite in the best schedule generated. The horizontal axis is time; a total of one week. The vertical axis is the amount of cross-track slew necessary to execute the scheduled takeImages for this satellite. Note the preference for extreme slews. The extreme slews apparently minimize the total slewing time sufficiently to overcome the fitness pressure towards small slews.

1. additional model problems.

2. a duty cycle constraint. This constraint requires that an instrument is not used for more than $u$ seconds in any $t$ second time period.

3. improved squeaky operators; in particular, shifting a high priority, unscheduled takeImage forward, rather than swapping with a scheduled, low priority takeImage.

4. a random swap operator where the number of swaps is a probabilistic function of the number of children that have been produced. As evolution proceeds, the number of swaps is reduced. This encourages large steps in the beginning of evolution and smaller refinement steps near the end.

5. transmission operator evolution; where transmission operators that have done well early in evolution are more likely to be used.

6. additional forms of local search.

7. a multi-objective co-evolution genetic algorithm JASON REFERENCE HERE.

8. HBSS (Heuristic Biased Stochastic Search) with contention based heuristics similar to Frank et al. 2002.

## 7 Summary

Earth imaging satellite constellation scheduling is a complex task with many variables and interacting constraints. We hypothesize that evolutionary programming can solve the EOS scheduling problem effectively and have begun to test various evolutionary search techniques and transmission operators. To date, simulated annealing combined with multiple-iteration random swap operators are the best. We have also shown that scheduling a small fleet as a combined resource outperforms separate scheduling for each satellite by about 15%.

## 8 Acknowledgements

This work was funded by NASA's Computing, Information, & Communications Technology Program, Advanced Information Systems Technology Program (contract AIST-0042), and by the Intelligent Systems Program. Thanks to Alex Herz, Orbit Logic, Inc. and Jerald Arp, PhD., Manager of Technical Support, Space Imaging LLC for information regarding current EOS systems. Thanks also to ??? for reviewing this paper and to Jennifer Dungan, Jeremy Frank, Robert Morris and David Smith for many helpful discussions. Finally, thanks to the developers of the excellent Colt open source libraries for high performance scientific and technical computing in Java (http://hoschek.home.cern.ch/hoschek/colt).

## 9 References

Frank, J.; Jonsson, A.; Morris, R.; and Smith, D. 2002. Planning and Scheduling for Fleets of Earth Observing Satellites. In proceedings of the 6th International Symposium on Artificial Intelligence, Robotics, Automation and Space 2002 Montreal, June 18-22.

Frank, J.; and Jonsson, A. 2002. Constraint-based Attribute and Interval Planning, to appear in the Journal of Constraints, Special Issue on Constraints and Planning.

Globus, A.; Langhirt E.; Livny M.; Ramamurthy R.; Solomon M.; Traugott S. 2000. JavaGenes and Condor: Cycle-Scavenging Genetic Algorithms, Java Grande 2000, sponsored by ACM SIGPLAN, San Francisco, California, 3-4 June 2000.

Globus, A. 2001. Towards 100,000 CPU Cycle-Scavenging by Genetic Algorithms, NAS technical report NAS-0-011, October 2001.

[Globus et al. 2001] Al Globus, James Crawford, Jason Lohn, and Robert Morris, "Scheduling Earth Observing Fleets Using Evolutionary Algorithms: Problem Description and Approach," *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, NASA, Houston Texas, 27-29 October, 2002.

[Globus et al. 2002] Al Globus, Madhu Menon, Deepak Srivastava "JavaGenes: Evolving Molecular Force Field Parameters with Genetic Algorithm," *Computer Modeling* *in Engineering and Science*, volume 3, number 5, pages 557-576,

Husbands P. 1994. Genetic Algorithms for Scheduling, AISR Quarterly, number 89.

Joslin, D.E.; Clements D.P. 1999 .Squeaky Wheel Optimization, Journal of Artificial Intelligence Research, Volume 10, pages 353-373.

Lamaitre, M.; Verfaillie, G.; Bataille, N. 1998. Sharing the Use of a Satellite: an Overview of Methods, SpaceOps 1998, 1-5 June, Tokyo, Japan.

Lamaitre, M.; Verfaillie, G.; Frank, J.; Lachiver, J.; Bataille, N. 2000. How to Manage the New Generation of Agile Earth Observation Satellites, SpaceOps 2000, sponsored by the Centre National dEstudes Spatialles (CNES) of France.

Litzkow, M.; Livny, M.; Mutka, M.W. 1988. Condor - a Hunter of Idle Workstations, Proceedings of the 8th International Conference of Distributed Computing Systems, pages 104-111.

J.D. Lohn, G.L. Haith, S.P. Colombano, D. Stassinopoulos, ”A Comparison of Dyna mic Fitness Schedules for Evolutionary Design of Amplifiers,” Proc. of the Firs t NASA/DoD Workshop on Evolvable Hardware, Pasadena, CA, IEEE Computer Society P ress, 1999, pp. 87-92.

Montana, D.J. 2001. A Reconfigurable Optimizing Scheduler, Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, California, 7-11 July 2001, pages 1159-1166.

Muraoka, H.; Cohen, R.H.; Ohno T.; Doi, N. 1998. Aster Observation Scheduling Algorithm, SpaceOps 1998, 1-5 June, Tokyo, Japan.

Rao, J.D.; Soma, P.; Padmashree, G.S. 1998. Multi-Satellite Scheduling System for LEO Satellite Operations, SpaceOps 1998, 1-5 June, Tokyo, Japan.

Potin, P. 1998. End-To-End Planning Approach for Earth Observation Mission Exploitation, SpaceOps 1998, 1-5 June, Tokyo, Japan.

Potter W.; Gasch, J. 1998. A Photo Album of Earth: Scheduling Landsat 7 Mission Daily Activities, SpaceOps 1998, 1-5 June, Tokyo, Japan.

Sherwood, R.; Govindjee, A.; Yan, D.; Rabideau, G.; Chien, S.; Fukunaga, A. 1998. Using ASPEN to Automate EO-1 Activity Planning, Proceedings of the 1998 IEEE Aerospace Conference, Aspen, CO, March.

Syswerda G.; Palmucci, J. 1991. The Application of Genetic Algorithms to Resource Scheduling, Proceedings of the Fourth International Conference on Genetic Algorithms, University of California, San Diego, 13-16 July 1991, Richard K. Belew and Lashon B. Booker, editors, pages 502-508.

Whitley, D.; Starkweather, T.; Fuquay, D. 1990. Scheduling Problems and Traveling Salesmen: the Genetic Edge Recombination Operator, Proceedings of the Third International Conference on Genetic Algorithms, pages 133-140.

Wolfe, W.J.; Sorensen, S.E. 2000. Three Scheduling Algorithms Applied to the Earth Observing Systems Domain, Management Science, volume 46, number 1, January 2000, pages 148-168.

Yamaguchi, Y.; Kawakami, T.; Kahle, A.B.; Pniel, M.; Tsu, H. 1998. Aster Mission Planning and Operations Concept, SpaceOps 1998, 1-5 June, Tokyo, Japan.