613622

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

Al Globus, James Crawford, Jason Lohn and Robert Morris

Abstract

We describe work in progress concerning multi-instrument, multi-satellite scheduling. Most, although not all, Earth observing instruments currently in orbit are unique. In the relatively near future, however, we expect to see fleets of Earth observing spacecraft, many carrying nearly identical instruments. This presents a substantially new scheduling challenge. Inspired by successful commercial applications of evolutionary algorithms in scheduling domains, this paper presents work in progress regarding the use of evolutionary algorithms to solve a set of Earth observing related model problems. Both the model problems and the software are described. Since the larger problems will require substantial computation and evolutionary algorithms are embarrassingly parallel, we discuss our parallelization techniques using dedicated and cycle-scavenged workstations.

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. Image collection for these satellites is planned and scheduled by a variety of software systems (Muraoka et al. 1998, Potter and Gasch 1998, Sherwood et al. 1998, and others). Science activities on different satellites or even different instruments on the same satellite are typically scheduled independently of one another, requiring the manual coordination of observations by communicating teams of mission planners. As the number of satellites and the number of observation requests grow large, manual coordination will no longer be possible. A more effective way to manage observation scheduling is by allowing customers of the data to request data products from a central authority instead of an individual satellite or mission. Customer preferences will constrain which satellite or satellites will be used to collect the data. Automated techniques can schedule the necessary resources. This should enable more efficient management of a fleet of satellites. 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).

Scheduling EOS is complicated by a number of important constraints. Potin 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

Potin also notes that "ASAR offers, by exploiting the combinations of polarizations and incidence angles, 37 different and mutually exclusive high rate operating modes" (Potin 1998) and Yamaguchi et al. note that "ASTER could collect approximately 1.7 million scenes of full-mode data. *In practice, there will be factors that will decrease this amount, such as scheduling inefficiencies*" (italics added) (Yamaguchi et al. 1998), suggesting that even scheduling a single instrument can be challenging. ASAR is an Advanced Synthetic Aperture Radar featuring enhanced capability in terms of coverage, range of incidence angles, polarisation, and modes of operation. ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) is one of several imaging instruments on Terra, launched in 1999. 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 requires a representative set of problems and software to solve them. We are developing a set of such problems that can be generated by AGI's Satellite Tool Kit (http://www.stk.com/) combined with small amounts of custom software. We are also developing an evolutionary algorithm and custom constraint system to solve EOS scheduling problems.

The next section describes the model problems. This is followed by a description of the evolutionary software and constraint system under development. Finally, since complex problems sometimes require substantial computation, the final section describes our approach to parallelization of the computation on a large number of dedicated and cycle-scavenged CPUs.

Model Problems

Since our project is designed to consider the scheduling of a parameterizable generic system, not any particular spacecraft, sensor, or satellite constellation, it is important to develop a set of model problems that exhibit the important aspects of EOS scheduling now and in the future. In all cases we attempt to base our model sensors and satellites on hardware currently in orbit, although the association is quite loose and the parameters are meant to be representative, not accurate. We have identified and begun to scope six problems:

1. A single satellite with a slewable instrument. This problem exercises slew scheduling on an instrument modeled on ASTER slewing (Muraoka et al. 1998, Yamaguchi et al. 1998) and the Landsat ETM instrument (Potter and Gasch 1998) for other characteristics. We are particularly interested in minimizing slew time, since the ASTER instrument has a limited lifetime slew budget. Minimizing slew while maximizing the number of images taken leads to a multi-objective optimization problem. The Landsat program has orbited a series of Earth imaging satellites, including the first one, and the ETM (Enhanced Thematic Mapper) is the main instrument on the more recent satellites.

2. A single agile satellite with one instrument. This is the same as problem one except that we assume the whole spacecraft is slewed, rather than the instrument relative to the spacecraft. This allows more complex pointing behavior (three axis instead of one). Lamaitre et al. compared constraint satisfaction and local search on a variant of this problem and found that their local search algorithm out-performed constraint satisfaction (Lamaitre et al. 2000).

3. A single satellite with multiple instruments (one slewable). This exercises multiple instruments sharing satellite resources such as power and SSR (Solid State Recorder – memory used to save images until they can be sent to the ground). Lemaitre et al. found that global optimization could out-perform static quotas for the French SPOT satellite shared between two users (Lemaitre et al. 1998). Problem three addresses similar issues in a more complex environment loosely based on the Aqua satellite (http://aqua.nasa.gov).

4. A large constellation of single- and multiple-instrument satellites communicating directly with the ground. This seeks to mimic hypothetical future sensor webs, large constellations of Earth imaging satellites competing for ground station time. The same sensor is replicated on multiple satellites to reduce the time-between-images on the same request and increase the total number of images that can be taken. To model stereo pair problems, three pairs of satellites with shared sensors will orbit one minute apart and a fraction of their requests will be shared, stereo pair requests. One of these pairs will include a multiple-instrument satellite.

5. A large constellation of single-instrument slewable satellites communicating with an in-orbit communications system based on high-data-rate lasers. This problem assumes a robust inter-satellite communication system and a network of communication satellites to reduce ground station contention and limit on-board memory requirements.

6. The same as problem five, but with a much larger number of satellites, multiple instruments, and requirements to image the same target, at the same time, from multiple angles and with different instruments. This problem presumes much cheaper and more reliable launch.

In all of the problems, we represent a request for an image as a point where the point is assumed to be in the center of the area to be imaged. Imaging time is proportional to the length of the imaging area along the satellite ground track and depends on the problem. In some problems the imaging time varies. In all cases, the number of requests is chosen to be large enough so that all instruments should be over-subscribed. Requests will be randomly generated and assigned a random priority. Note that some missions aim for world-wide repeated coverage over time and others are demand driven, which may require somewhat different request sets. Since some sensors are sensitive to clouds, and clouds are not randomly distributed, cloud cover probabilities should be calculated from historical data.

Table 1 summarizes the problems, Table 2 summarizes the satellites, and Table 3 summarizes the instruments. Note that we have not yet found all the data necessary for these models.

AIRS, AMSU, HSB, MODIS and AMSR are all instruments on NASA's Aqua satellite, launched 4 May

ID	tests	schedule time	ground stations	satellites
1	slewing	1 week	1	A
2	agile	1 week	1	B
3	multiple instruments	2 days	2	C
4	sensor web	2 days	6	10D + 2C
5	sensor web	2 days	N/A	10B
6	sensor web	1 day	N/A	50B + 50B'

Table 1: Model problem summary.

ID	modeled after	instrument(s)	SSR (bits)	Power(kw)
A	Aster/Landsat	1	375G	1.55
B	Ikonos	3	12G	
C	Aqua	1,2,3,4,5	136G	4.6
D	sensor web	any one of 1-5	350G	1.55

Table 2: Satellite Summary. All satellites are in sun-synchronous orbits. B' is the same as B but images different spectral bands where some imaging requires both types of sensors simultaneously.

ID	1	2	3	4	5
loosely modeled after	ETM-ASTER	AIRS-AMSU-HSB	IKONOS	MODIS	AMSR
expected images per day	250	100	70	150	200
requests per day	300	150	100	250	350
time for request (sec)	24	10-30	90		
data rate (bits/sec)	150M	1.5M		7M	88K
swath (km)	185	1650	13	2330	1445
cross track FOV (degrees)	7.47	49			
FOV for point requests (+/- degrees)	25				
cross track slew limits (+/- degrees)	24	N/A	none	N/A	N/A
slew rate (degree/sec)	1	N/A	6 pitch, 3 roll	N/A	N/A
lighting	day	any	day	any	day
clouds ok	no	yes	no	no	no
warm-up time (sec)	72				
power (W)		220		147	350

Table 3: Instrument characteristics. The AMSR instrument has five incompatible modes with one minute switching time. The requests are randomly assigned between modes.

2002. The AIRS, AMSU, and HSB all have similar operational characteristics and are designed to work together, measuring different aspects of the same area. The IKONOS instrument is modeled after a high-resolution, agile commercial EOS satellite.

Evolutionary Algorithms and EOS Scheduling

There are a number of evolutionary algorithms in the literature. We are using a genetic algorithm (GA) to address EOS scheduling. 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 or a Gantt chart; each schedule is called an individual
2. Generate a population of individuals with random characteristics
3. Calculate the fitness of each individual
4. Repeat
   (a) Randomly select parents with a bias towards better fitness
   (b) Produce children from the parents with either:
      i. crossover that combines parts of two parents into a child
      ii. mutation that modifies a single parent
      iii. or some combination of the two
   (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

Evolutionary algorithms, particularly the genetic algorithm, have been used to schedule a wide variety of tasks. For example, Syswerda and Palmucci scheduled the U.S. Navy's System Integration Test Station laboratory for F-14 jet fighters using a GA with a permutation of tasks representation and a fast greedy scheduler to place tasks, one at a time, in the schedule (Syswerda and Palmucci 1991). Wolfe and Sorensen compared three scheduling algorithms, including GA, for EOS scheduling problems and found that GA produced the best schedules, albeit with a substantial CPU time penalty (Wolfe and Sorensen 2000). Philip Husbands provides a good, if somewhat dated, survey of GA for scheduling problems (Husbands 1994).

Evolution is guided by a fitness function. The fitness function must provide a fitness for any possible individual, no matter how bad, and distinguish between any two individuals, no matter how close they are. For EOS scheduling, the fitness function is multi-objective. These objectives include:

1. Maximize the number, quality and importance of the images taken (takeImages). For scientific applications the importance can be measured by priority. For commercial applications the importance can be measured by dollar value.
2. For images that require certain weather conditions, e.g., minimal clouds, maximize image taking redundancy.
3. Minimize total slewing (slew motors wear out).

To investigate GA applied to EOS scheduling, we are developing software to 1) compare a permutation representation to a Gantt chart representation and 2) compare squeaky-wheel versus blind transmission operators. Squeaky-wheel and blind operators are described below. We also intend to compare GA scheduling with an HBSS (Heuristic Biased Stochastic Sampling) EOS scheduler under development at NASA Ames (Frank et al. 2002) and are investigating a comparison with the ASPEN scheduler from JPL (Sherwood et al. 1998).

One of the key issues for any evolutionary algorithm is problem representation. We are currently investigating two representations for the scheduling problem: permutation and Gantt chart.

In the permutation representation, each individual is a permutation of the requested takeImages. A greedy scheduler attempts to schedule the requested takeImages in the order indicated by the permutation. The first greedy scheduler was a minor modification of the HBSS algorithm using the Europa constraint system (Frank and Jonsson 2002) described in (Frank et al. 2002), but this software currently has performance problems when scheduling thousands of takeImages. When these problems are resolved we will use it for HBSS/GA comparisons. In addition, we are developing a custom greedy scheduler as an extension to the JavaGenes software (Globus et al. 2000). This scheduler currently implements a permutation representation, earliest-first scheduling heuristics, and sensor availability and slewing constraints. This is a work-in-progress paper, and there has not been enough time to solve any of the model problems with this software. Permutation is a well-studied GA representation for scheduling (e.g., Whitley et al. 1989, Syswerda and Palmucci 1991, Montana 2001, and others) and there are many transmission operators in the literature. We are currently using Syswerda and Palmucci's order-based mutation and position-based crossover.

The Gantt chart representation is an extension of the permutation representation where the scheduled location of tasks in the parents is used in crossover and mutation. Specifically, rather than use heuristics to schedule each task, the parental placement is attempted first. In the crossover case, the parent to use may be chosen at random.

In order to enforce the constraints, fast constraint evaluation is necessary. Assuming digitized time, and that each takeImage should only be taken once (i.e., for sensors insensitive to clouds), all opportunities for

one takeImage form a mutually exclusive set (only one need be taken). Furthermore, for each time step the possible takeImages from a single sensor form another mutually exclusive set. All impossible slews between takeImages can be computed and turned into binary mutual exclusions (mutexes). The mutually exclusive sets may be implemented as sets where only one value is allowed, or by a set of mutexes. Note that this approach to EOS scheduling does not automatically schedule a takeImage when there is only one opportunity as this may prevent an earlier-in-the-permutation takeImage placement due to constraint violation.

If time is continuous and the satellite is agile (slews cross- and along-track), the number of takeImage opportunities grows very large and precomputing the sensor availability and slewing constraints becomes impractical. One approach to handling the constraints in this case is to represent each resource as a timeline. Each timeline then takes on appropriate values (e.g., in use for a sensor, slew motor setting, power, SSR memory available, etc.) at different times. Since values must be frequently inserted into the timeline, a doubly linked list is an appropriate data structure. However, finding a particular time in a long inked list is very slow. This can be solved by an array where each element points to the linked list node at the time associated with that array element, and the array elements are associated with fixed time intervals. Thus, to find the node at time 10,483 one simply calculates the appropriate array index, a constant-time operation, rather than traverse the linked list, an $O(n)$ operation where n is the number of nodes. As long as the interval represented by each array element is not significantly longer than the time represented by a typical linked list node, this double data structure should be fast for insertion and locating a particular time. This is the constraint data structure currently being developed.

In most evolutionary algorithm representations, it is difficult or impossible to determine which part of the representation is responsible for improvement or degradation to the fitness. However, for both EOS scheduling representations this is not the case. For the permutation representation, those takeImages that are not scheduled are a drag on the fitness. In the Gantt chart representation different time periods can exhibit high or low fitness and time can be divided into intra-dependent time periods — e.g., the times between data dumps to the ground. Thus, rather than use traditional blind GA transmission operators — which do not evaluate different parts of the representation — it is possible to use squeaky wheel (Joslin and Clements 1999) transmission operators where the operator knows which part of the representation should be modified. For example, in the permutation case an unscheduled takeImage could be mutated forward in the permutation (Joslin and Clements 1999).

Parallelization

The vast majority of CPU time is expected to be spent checking constraints, where the processing of each individual is independent of others in the population. Thus, the most compute-intensive portions of the GA can run in parallel. Furthermore, to evaluate any stochastic technique one must have multiple runs and make statistical comparisons. Also, all GA runs have parameters, e.g., population size, mix of crossover vs. mutation, etc. It is rarely obvious, a priori, what the parameters should be. A tedious hand search through this parameter space can be avoided by running many GA jobs with randomized parameters. If the machine time is low cost, e.g., if the cycles are scavenged from otherwise idle work stations, GA parameter randomization can be very effective (Globus, Menon, and Srivastava 2002). We have software systems in place that parallelize at both levels across runs and across individuals within runs.

To parallelize across runs, we simply use scripts to start many jobs on many machines and collect the results on disk. Across individual parallelization is via a master/slave architecture. The master is implemented as a set of PHP programs running inside a web server. The PHP programs use a mySQL database to maintain the population. The slaves can run on any machine with access to the web server. Slaves pick up individuals from the master via http requests. Slaves send results to the master for storage in the mySQL database.

In the future, we plan to combine these approaches by using the master to mediate immigration between populations. For additional details, see (Globus 2001).

Slaves are run on a 72 CPU Beowulf cluster and/or any available workstation. We plan to run a cycle-scavenger on the 350+ workstations in our division to run jobs nights, weekends and other times the workstations are idle. The Condor system (Litzkow, et al. 1988) has been effective in this role in our previous genetic algorithm work (Globus et al. 2000).

Summary

Earth imaging satellite constellation scheduling is a complex task with many variables and interacting constraints. We are defining a set of representative model problems intended to exercise scheduling software in the relevant dimensions. We hypothesize that evolutionary programming can solve the EOS scheduling problem effectively and have begun the development of software to test this hypothesis on the model problems. This software is also being designed to compare permutation vs. Gantt chart representations and squeaky-wheel vs. blind transmission operators.

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 Pro-

gram. 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 Jennifer Dungan, Jeremy Frank, and Bonnie Klein for reviewing this paper and to Jennifer Dungan, Jeremy Frank and David Smith for many helpful discussions.

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, A.; Menon, M.; Srivastava, D. 2002. JavaGenes: Evolving Molecular Force Field Parameters with Genetic Algorithm, to appear in Computer Modeling in Engineering and Science.

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.

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.