PRODUCING LARGE OBSERVATION CAMPAIGNS USING COMPRESSED PROBLEM REPRESENTATIONS

Russell Knight and Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
{*firstname.lastname*}@jpl.nasa.gov

Abstract: *We present a technique for solving observation campaign problems with over one million observations. Our approach takes advantage of the cyclic nature of such problems, and compresses the problem into a smaller representation. We show that this can be interleaved with a non-compressed representation while still providing better performance than using a completely non-compressed representation alone.*

1 Introduction

Consider the problem of scheduling the orientation and on/off times of a spacecraft or collection of spacecraft such that we adequately cover as many target points as possible. But, we must not oversubscribe memory or energy.

Many aspects of the problem need to be considered.

- Cost in terms of battery and energy when the instrument is on or off.
- Time to transition from one orientation to the next.
- Rate, time, and duration of downlinks and sun exposures.

For small versions of this problem, we can employ standard scheduling techniques. But as the problem size grows, standard propagation techniques are too costly, and even estimating the final quality of a partial schedule becomes practically intractable.

We note that many of these problems are cyclic in nature. That is, for spacecraft, we often have "repeat passes" where we overfly the same points at some predetermined interval. If we can take advantage of this cyclic nature in our representations, then we can compress the problem into a single-cycle problem. Unfortunately, memory and energy profiles are not guaranteed to match any cyclic representation, and either need to be represented explicitly or compressed as well.

2 Problem Description

The Cyclic Large Observation Campaign problem is the problem of selecting from a candidate set of observations those observations that cover as many target points as possible without oversubscribing energy and memory budgets. Observation opportunities repeat, therefore we are given a cycle bound in which to work.

More formally, given:

- a set of points P,
- a set of observations opportunities O each $o \in O$ consists of a start ($o.start$), and a duration ($o.duration$),
- a function $obs(p \in P)$ that returns the subset of O whose elements positively affect coverage of p,
- a function $coverage(p, sol \subseteq obs(p))$ that determines the level of coverage of p given sol,
- a bound on coverage *minQuality* that must be met for a point to be considered covered,
- a bound on memory *memory*,
- a bound on energy *energy*,
- a rate at which memory is used while the instrument is on *memRate*,
- a rate at which energy is used while the instrument is on *useWatts*,
- a rate at which memory is recovered during a downlink *downRate*,
- a rate at which energy is collected while in sunlight *sunWatts*,
- a function that returns the minimum duration to transition from one observation orientation to another $trans(o_1 \in O, o_2 \in O)$,
- a set of sun exposures S where each $s \in S$ consists of a start ($s.start$) and a duration ($s.duration$),
- a set of downlinks D where each $d \in D$ consists of a start ($d.start$) and a duration ($d.duration$),

select $p.sol$ subset of $obs(p)$ for each $p \in P$ such that the number of $p \in P$ that $coverage(p, p.sol) \geq minQuality$ is maximized, yet at no time do we oversubscribe memory or energy.

Since *coverage* is a black-box function, optimizing this in general is problematic. But if we make the assumption that coverage is monotone (adding another o

to *p.sol* never makes it worse), then we can perform local optimization based on approximating that adding the *o* that gives us the best improvement now might give us good improvement overall. If *coverage* is linear, all the better.

3 Solution Techniques

Our approach is to apply squeaky wheel optimization to the scheduling problem. Squeaky wheel optimization is an iterative greedy algorithm.

1. All of the points are sorted according to priority.
2. Each point is scheduled, greedily, in order of priority.
3. Priorities are increased for those points that failed to schedule.
4. Iterate by repeating from step 1.

The result of this is that points that are not scheduled have a better chance of being scheduled as we iterate because each time the point is not scheduled, its priority increases, until it is the first point scheduled.

At each iteration, we go through each point p (in priority order) and add observations o to $p.pos$ from $obs(p)$ (in descending order of $coverage(p, p.pos \cup \{o\})$ until $coverage(p, p.pos) \geq minQuality$. We only consider observations that cannot oversubscribe memory, cannot oversubscribe energy, and respect transition duration constraints.

The asymptotic time complexity of this algorithm is $O(n \log n)$ per iteration, where n is the number of points. This is basically the time required to sort the points according to priority.

To be able to reason about multiple cycles, we use a technique of using a single cycle for observation representations, but multiple cycles for energy and memory representations. This saves us the space required to represent all possible observations over all cycles, and instead we represent a single cycle of observations with tags that represent which cycle the observation belongs to. In this way, we maintain linear memory for observation encoding in the number of total observations. Note that this abstracts away the possibility of multiple observations at the same point in the cycle. The assumption here is that an observation from the same angle will not give us considerable return. An extension to our scheme allows for counting repeat passes on cycles. We lose the linear relationship of the encoding, but we can a faithful representation for multiple observations and we remain linear in the number of actual observations taken.

The reason we cannot do the same encoding trick with resources (energy and memory) is that the effects of these resources is not locally bounded. Since usage of each persists over time, we need to propagate these values over time, and across downlink opportunities and sunlight, so that the actual profiles can be computed. Encoding these using the same compression technique would mean that we can charge the battery in the future for use now, even though the battery is currently empty. Obviously, this encoding is too relaxed to produce good schedules.

4 Results

First, we report the performance of the squeaky wheel optimizer for a global coverage problem where points are selected from points on land on Earth. We narrow our purview for one day's worth of observations. We see that both solution time and setup time increase with quality. This makes sense in that the higher the quality, the more information that needs to be propagated for resources. Figure 1 shows the solution and setup times on the vertical axis, with the iteration along the horizontal axis. Figure 2 shows the solution quality along the vertical axis, with the iteration along the horizontal axis.

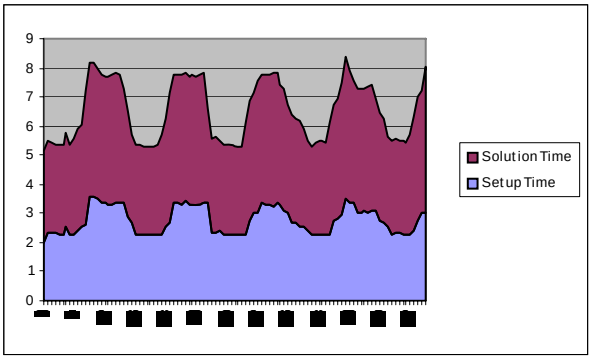

Figure 1 Solution and Setup times

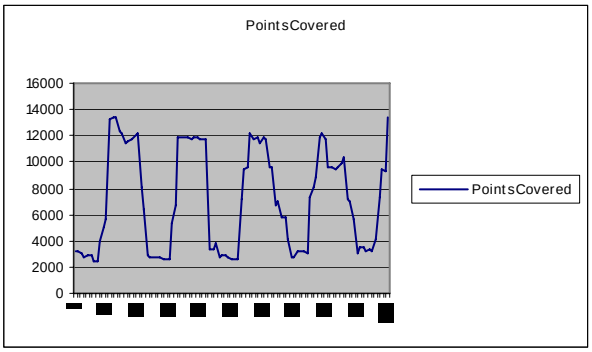

Figure 2 Solution Quality

We see that solution quality seems to be cyclic. This makes sense given the nature of the squeaky wheel algorithm makes it somewhat cyclic. But we also see improvement over time.

When solving the larger problem of the entire schedule over 330 days (30 cycles), we see that all possible points that can be covered are, and this solution only requires 15 minutes to setup and solve. This is a greater than 20 fold increase in speed over using non-compressed representations, and an even greater speed-

up over using other systems, which require about a week to produce a single schedule.

5 Related Work

[Knight 2005] solves smaller problems of this sort optimally using a combination of branch and bound techniques combined with flow network approximations.

For a good example of a polyhedral solution to a combinatorial optimization problem having to do with satellite scheduling (formulated as a pick-up and delivery problem), see [Ruland 1986].

[Oddi 2003] solves a constrained-memory domain with fewer types of constraints called the Mars Express Memory Dumping Problem. The system uses a portfolio approach to solving the problem as formulated in a constraint-based framework. The portfolio consists of a tabu search strategy, a random sampling strategy, and a greedy strategy.

More general constraint-based frameworks for scheduling that have been applied to spacecraft operations include that of [Dungan 2002], [Ghallab 1994], and [Chien 2000]. In each of these, the problem is expressed as a set of constraints to be satisfied. In the case of [Dungan 2002], and [Ghallab 1994], the systems search the feasible space of domains in the constraint space. In the case of [Chien 2000] the system searches both the infeasible and feasible space of value assignments, using randomized local search.

6 Acknowledgements

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[Chien 2000] S. Chien, G. Rabideu, R. Knight, R. Sherwood, B Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. “Automating space mission operations using automated planning and scheduling.” In *Proc. SpaceOps*, 2000.

[Corman *et al*] Thomas H. Corman, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[Dungan 2002] J. Dungan, J. Frank, A. Jonsson, R. Morris, and D. Smith. “Advances in Planning and Scheduling of Remote Sensing Instruments for Fleets of Earth Orbiting Satellites.” *Earth Science Technology Conference*, 2002. Pasadena, California.

[Frank 2000] J. Frank. "SOFIA's Choice: Automating the Scheduling of Airborne Observations" Proceedings of the 2d NASA Workshop on Planning and Scheduling for Space, March 2000.

[Garey and Johnson 1979] M.R. Garey and D.S. Johnson. Computers and Intractability. W.H. Freeman and Company, San Francisco, 1979.

[Ghallab 1994] M. Ghallab and H. Laruelle, “Representation and control in IxTeT, a temporal planner”, in *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems* (AIPS-94), pp. 61-67, Chicago, IL, (1994). AAAI Press, Menlo Park.

[Knight 2005] R. Knight, “Solving the Constrained Coverage Problem using Flow Networks as Linear Program Approximations,” Ph.D. Dissertation, University of California, Los Angeles, 2005.

[Karp 1972] R. M. Karp “Reducibility among combinatorial problems.” In R. E. Miller and J. W. Thatcher (eds.) *Complexity of Computer Computations*, Plenum Press, New York, 85-103.

[Muraoka 1998] H. Muraoka, R. H. Cohen, T. Ohno, and N.Doi. “ASTER Observation Scheduling Algorithm.” SpaceOps 98. 1998, 1-5 June, Tokyo, Japan.

[Oddi 2003] A. Oddi, N. Policella, A. Cesta and G. Cortellessa. “Generating High Quality Schedules for Spacecraft Memory Downlink Problems.” *Ninth International Conference on Principles and Practice of Constraint Programming*, 29 September - 3 October, 2003, Kinsale, County Cork, Ireland.

[Papadimitriou and Steiglitz 1982] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Ruland 1986] K. Ruland. Polyhedral solution to the pickup and delivery problem. Washington University, Sever Institute of Systems Science and Mathematics. http://rodin.wustl.edu/~kevin/dissert/dissert.html (Dissertation). St. Louis Missouri, 1995.

[Schrijver 1986] A. Schrijver. Theory of Linear and Integer Programming, Wiley, 1986.

[Zhang 2000] W. Zhang. “Depth-First Branch-and-Bound versus Local Search: A Case Study.” In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000), pages. 930-935, Austin, Texas, 2000.