Towards a Self-Configuring Optimization System for Spacecraft Design

Alex S. Fukunaga Andre Stechert Steve Chien

Jet Propulsion Laboratory, MS 525-3660
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
{alex.fukunaga,andre.stechert,steve.chien}@jpl.nasa.gov

Abstract

Spacecraft design optimization is a difficult problem, due to the complexity of optimization cost surfaces and the human expertise in optimization that is necessary in order to achieve good results. In this paper, we propose the use of a set of generic, metaheuristic optimization algorithms (e.g., genetic algorithms, simulated annealing), which is configured for a particular optimization problem by an adaptive problem solver based on artificial intelligence and machine learning techniques. We describe work in progress on OASIS, a self-configuring optimization system based on these principles.

1 Introduction

Many engineering design optimization problems are instances of constrained optimization problems. Given a set of decision variables X and a set of constraints C on X, the constrained optimization is the problem of assigning values to X to minimize or maximize an objective function $F(X)$, subject to the constraints C.

Although constrained optimization is a mature field that has been studied extensively by researcher, there are a number of open, fundamental problems in the practical application of optimization techniques. In particular, the problem of selecting and configuring an optimization algorithm for an arbitrary problem is a significant obstacle to the application of state of the art algorithms to real world problems.

We are currently developing the *Optimization Assistant (OASIS)* system, an automated, self-configuring optimization tool for spacecraft design these two issues. The goal of OASIS is to facilitate rapid "what-if" analysis of spacecraft design by developing a widely applicable, spacecraft design optimization system that maximizes the automation of the optimization process and minimizes the user effort required to configure the system for a particular optimization problem instance. In the rest of this paper, we describe initial work on OASIS.

2 Resource-Bounded Black Box Optimization

The problem of global optimization on difficult, arbitrary cost surfaces is still poorly understood. The optimization of smooth, convex cost functions is well understood, and efficient algorithms for optimization on these surfaces have been developed. However, these traditional approaches often perform poorly on cost surfaces with many local optima, since they tend to get stuck on local optima. Unfortunately, many real-world optimization problems have such a "rugged" cost surface and are thus difficult problems for traditional approaches to optimization.

In addition, many real-world optimization problems are *black-box optimization problems*, in which the structure of the cost function is opaque. That is, it is not possible to directly analyze the cost surface by analytic means in order to guide an optimization algorithm. For example, $F(X)$ can be computed by a complex simulation about which the optimization algorithm has no information (e.g., to evaluate a candidate spacecraft design, we could simulate its operations using legacy FORTRAN code about which very little is known to the optimizer except for its I/O specifications). Black-box optimization problems are therefore challenging because currently known algorithms for black-box optimization are essentially "blind" search algorithms-instead of being guided by direct analysis of the cost surface, they must sample the cost surface in order to indirectly obtain useful information about the cost surface.

Recently, there has been much research activity in so-called *metaheuristic* algorithms such as simulated annealing [5], tabu search [2, 3] and genetic algorithms [4] for global optimization. These are loosely defined, "general-purpose" heuristics for optimization that proceed by iteratively sampling a cost surface, and they implement various mechanisms for escaping local optima. Although these algorithms have been shown to be successful on numerous applications with difficult cost surfaces, the behavior of these algorithms is still poorly un-

derstood. Successful application of these metaheuristics[1] to a particular problem requires:

- Selection of the most appropriate metaheuristic for the problem, and
- Intelligent configuration of the metaheuristic by selecting appropriate values for various control parameters (e.g., temperature cooling schedule for simulated annealing).

Currently, successful applications of metaheuristics are often the result of an iterative cycle in which a researcher or practitioner selects and adjusts a number of different metaheuristic/control parameter combinations on a problem, observes the results, and repeats this process until satisfactory results are obtained. This process of selecting and configuring a metaheuristic to obtain good results on a given problem is usually time-consuming, and requires a significant amount of optimization expertise (which is often very costly to obtain). As a result, in many cases, the cost of successfully applying metaheuristic techniques on black-box problems can be prohibitively expensive.

One might wonder whether there is some super-metaheuristic which can be configured so that it outperforms all others algorithm configurations for all problems of interest, or whether it is at least possible to characterize the performance of metaheuristic configurations in general. The current conventional wisdom in the optimization research community is that this possibility is extremely unlikely (although it not likely that this can ever be formally proved, due to the empirical nature of the question). This is supported by related recent theoretical work such as [7], which shows that over all possible cost surfaces, the expected performances of all optimization algorithms are exactly equal. Although it is possible that "all problems of interest" (in our context, all nontrivial spacecraft design optimization problems) reflect a particular subset of all possible cost surfaces for which some metaheuristic configuration's performance dominates that of all others, we strongly believe that this is not the case. Thus, to obtain the best quality solution, it is necessary to select and configure a metaheuristic so that it matches the structure of the problem instance cost surface.

The *resource-bounded black-box optimization problem* can be stated as follows: Let OS be a configurable optimization system, with several control points, $CP_1 \ldots CP_n$ (where each control point CP_i corresponds to a particular control decision), and a set of values for each control point, $\{M_{i,1} \ldots M_{i,k}\}$,[2] a *configuration* is an assignment of values to control points that defines the overall behavior of the problem solver. Given a time bound T, and d, a problem instance. Let $U(OS, d, T)$, be a real valued utility function that is a measure of the goodness of the behavior of the optimization system on d after executing for time T. The task of resource-bounded black-box optimization is to maximize U (i.e., obtain the best possible solution quality within a given time). In principle, this can be accomplished by choosing the optimal configuration of OS for the particular problem instance. In practice, determining the optimal configuration is difficult, so the system needs to maximize its performance by a heuristically guided meta-level search through the space of possible configurations of OS. That is, the system reconfigures itself dynamically in order to maximize performance.

[1] In the rest of the paper, we use the terms *metaheuristic* and *metaheuristic algorithm* interchangeably.

[2] Note that a method may consist of smaller elements so that a method may be a set of control rules or a combination of metaheuristics.

3 OASIS Architecture

OASIS (Optimization Assistant) is an integrated software architecture for spacecraft design optimization currently being developed at JPL that supports self-configuring black-box optimization.

The three major components of OASIS are:

- A spacecraft design model,
- A suite of configurable metaheuristics, and
- A self-configuring optimizer.

We describe each of these in the following discussion.

3.1 Spacecraft Design Model

The spacecraft design model is a software simulation of a spacecraft design. The design model takes as input decision variables to be optimized, and outputs an objective function value, which is assigned as the result of an arbitrarily complex computation (i.e., the simulator is a *black-box simulation*).

Thus, the design model is the component of OASIS that is the most domain-specific, and is provided by the end users, i.e., spacecraft designers. In order for an optimization system such as OASIS to be useful in practice, it must support a wide range of design models, which may consist of models implemented using various languages on different platforms.

The Multidisciplinary Integrated Design Assistant for Spacecraft (MIDAS) [1] is a graphical design environment that allows a user to integrate a system of possibly distributed design model components together using a graphical diagram representing the data flow of the system. Each node in the diagram corresponds to a design model component, which may be one of 1) a model in a commercial design tool such as IDEAS, NASTRAN, or SPICE, 2) a program written in C, C++, or FORTRAN, or 3) an embedded methogram (i.e., this allows methograms to have a hierarchical structure). Inputs

to nodes in the methogram correspond to input parameters for the component represented by the node, and outputs from a methogram node correspond to output values computed by the component. MIDAS is implemented as a CORBA object, and supports a wide variety of methods that can be used by external client systems (e.g., a GUI) to manipulate the methograms. This essentially provides an optimization system with a uniform interface for any design model encapsulated in MIDAS. Therefore, our solution to the problem of supporting a wide range of design models is to support an interface to MIDAS. That is, OASIS is designed to be an optimization system that can be used to optimize any MIDAS model.

Thus, the design model, which constitutes the user input to the OASIS system, is composed of the following:

- A MIDAS diagram that encapsulates the design model,
- A list of decision variables, as well as ranges of their possible values (may be continuous or discrete), and
- An output from a methogram node that corresponds the user's objective function value.[3]

Figure 1 shows part of a MIDAS methogram for the Neptune Orbiter model (see Section 4).

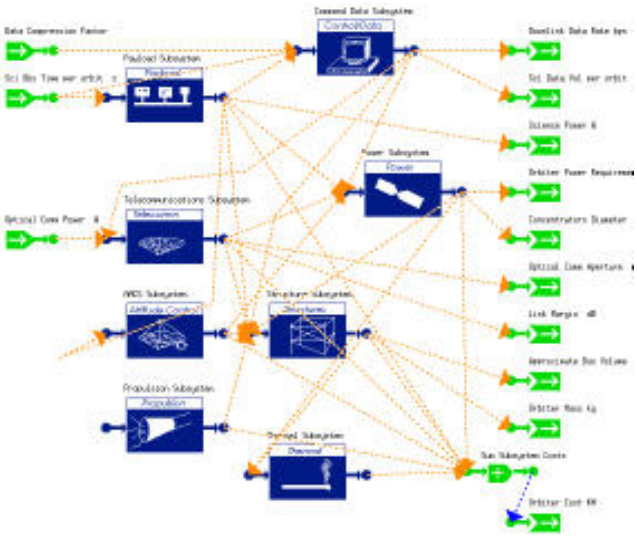

Figure 1: Screen shot of a MIDAS methogram (part of a Neptune Orbiter model).

3.2 Black-Box Optimization Algorithm Suite

OASIS includes a set of *configurable black-box optimization algorithms*, which are generic implementations of optimization algorithms that provide an interface for dynamic reconfiguration of their control points at runtime. Currently, this consists of a reconfigurable genetic algorithm [4] a reconfigurable simulated annealing algorithm [5], and some variants of traditional local optimization algorithms (Powell's Method and conjugate gradient algorithm) [6] with random restarts.

3.3 Self-Configuring Optimization System

Given a spacecraft design optimization problem instance in the form of a design model, the self-configuring optimization component of OASIS selects and configures a metaheuristics from its suite in order to maximize some utility measure (usually, this is the quality of the design found by the system).

Our approach to optimizer configuration is to view it as a meta-level heuristic search through the space of possible problem solver configurations, where candidate configurations are evaluated with respect to a utility measure, and the goal is find a configuration that maximizes this utility measure. In principle, it is possible to do a brute-force search through the space of possible problem solver configurations. This method is clearly intractable in general, since the number of configurations is exponential in the number of control points. Consider a problem solver with c control points, each with v values; there are c^v problem solver configurations to be considered.

Given the enormous computational expense of searching through the space of problem solver configurations, one might wonder whether the search should/could be avoided altogether. To avoid search completely, there are two alternatives. The first is to find a metaheuristic that outperforms all others for all problem instances (and thereby avoiding the problem of optimizer configuration altogether). As discussed in Section 2, we reject this solution as infeasible. The second alternative is a syntactic, "lookup-table" approach: classify the problem instance as a member of some class of problems, then apply the metaheuristic configuration that is known to work well for this class of problems. This method can work very well if we happen to have studied the class of problems to which the particular instance belongs, and we have available a good technique for classifying the instance as a member of the class. This approach, however, is of limited utility if we encounter an instance of a class that we know nothing about, or if we cannot correctly classify the problem as one that belongs to a class for which we have a good metaheuristic configuration.[4] Thus, a purely syntactic approach does not suffice. A self-configuring optimization system needs to search the space of possible metaheuristic configurations-the challenge is to discover and apply enough heuristic knowledge to the task to make it more tractable.

[3]The objective function could either be obtained directly from one of the existing outputs in the methogram, or it could be computed by adding a new node that computes, e.g., a weighted linear combination of some set of output nodes.

[4]Indeed, the problems of defining useful notions of classes of problem instances, and classifying a problem instance as belonging to some particular class is a challenging pattern recognition problem in itself.

What types of heuristic knowledge are available to be either acquired from a human, or through knowledge discovery/machine learning techniques? We identify three general classes of knowledge which are applicable in this context:

- domain-dependent knowledge about the behavior of metaheuristics in a given domain,
- domain-independent, meta-knowledge about optimization, and
- domain-independent, but system-dependent structural knowledge.

Each of these types of knowledge are discussed below:

3.3.1 Domain-dependent knowledge

This includes knowledge about the structure of particular classes of problems, and the behavior of metaheuristics on particular problem instances or classes of instances. Examples of heuristics that can be derive from this type of knowledge include:

- If a problem instance is in the problem instance class I, then configuration C is promising;
- If a problem instance i is in the problem instance class I, then it is likely that its cost surface is of type S;
- If the behavior of a configuration C is poor, then the instance is likely to be in instance class I;
- If the cost surface of the instance belongs to class S, then the instance is likely to be in instance class I.

In addition, this class includes any knowledge that can be used to classify a problem instance as belonging to a particular class of problems, including pattern classification heuristics that can be used by a problem instance analysis module.

3.3.2 Domain-independent knowledge

This is a formalization of the knowledge that human optimization experts possess about optimization in general. It includes knowledge about the behavior of metaheuristics on particular classes of cost surfaces,[5] and knowledge about the behavior of metaheuristics in general. Classes of cost surfaces can be defined by attributes such as the number of local minima, distance relationships between local minima, etc. Examples of heuristics that can be derived from this type of knowledge include:

- If a surface of class S, configuration C is promising;
- If the behavior of a configuration C is poor, then, configuration C' is promising;
- Given some features of the observed behavior of configuration C, it is likely that the cost surface is in class S.

This class of knowledge may not be as powerful as domain-dependent knowledge by itself, since it is more abstract in nature and more difficult to apply. For example, analysis of the cost surface is more computationally expensive than syntactic analysis of the instance, since it requires expensive runs of the black-box simulation to evaluate the cost surface. However, human optimization experts who may not be domain experts for a particular problem that they are given must often rely on their body of domain-independent knowledge, in combination to what domain-dependent knowledge they can obtain. The apparent success of these optimization experts indicates that this type of knowledge can be a very useful means of controlling search.

3.3.3 Domain-independent, system-dependent structural knowledge

It may be possible to exploit the syntactic structure of the problem representation in a particular adaptive problem solver. For instance, in OASIS, for a particular problem instance, it may be possible to analyze the structure of its dataflow graph in the MIDAS methogram to identify, e.g., decision variables that affect a relatively large number of other nodes in the graph. Hence, it may be more important to focus on nodes of high degree than a node than a node of low degree. At this time, it seems that useful knowledge of this type may be extremely difficult to acquire, although this is an area of research that seems particularly interesting from the design/human-computer interface perspectives, since it entails the understanding of how engineers structure simulations from a graph-theoretic point of view.

The various types of knowledge described above can be applied either directly or indirectly (through chains of inference) as variable/value ordering heuristics[6] in a search algorithm that searches the space of metaheuristic configurations.

Thus, the central research problem that is currently being addressed in the OASIS project is the representation and learning of the classes of knowledge described above. We are currently exploring the application of neural network and Bayesian learning algorithms in order to learn domain-dependent knowledge from large databases of performance data.[7]

[5] Note the difference between *classes of problem instances* and *classes of cost surfaces.* Cost surfaces are a more abstract, and classes of cost surfaces can encompass many classes of problems. For instance, the class of cost surfaces that are "bumpy or rugged" includes cost surfaces from a very large number of classes of problem instances.

[6] Variable ordering heuristics guide the choice of control points to change; value ordering heuristics guide the choice of control point values to try.

[7] Run-time performance data (such as solution quality as a function of time) is stored for every optimization algorithm run.

4 Example Spacecraft Design Optimization Problems

In this section, we describe two specific spacecraft design optimization problems to which we are currently applying the OASIS system. The first is a low-level optimization of the physical dimensions of a soil penetrator microprobe. The second is a system-level optimization of the configuration of the communication system of an orbiter spacecraft. These examples are illustrative of the range of different optimization problems that arise in spacecraft design.

4.1 The Mars Soil Penetrator Microprobe

As part of the NASA New Millennium program, two microprobes, each consisting of a very low-mass aeroshell and penetrator system, are planned to launch in January, 1999 (attached to the Mars Surveyor lander), to arrive at Mars in December, 1999. The probes will ballistically enter the Martian atmosphere and passively orient themselves to meet peak heating and impact requirements. Upon impacting the Martian surface, the probes will punch through the entry aeroshell and separate into a fore- and aftbody system. The forebody will reach a depth of 0.5 to 2 meters, while the aftbody will remain on the surface for communications.

Each penetrator system includes a suite of highly miniaturized components needed for future micropenetrator networks: ultra low temperature batteries, power microelectronics, and advanced micro-controller, a microtelecommunications system and a science payload package (a microlaser system for detecting subsurface water).

The optimization of physical design parameters for a soil penetrator based on these Mars microprobe is the first testbed for the OASIS system. The microprobe optimization domain in its entirety is very complex, involving a three-stage simulation:

- Separation analysis (i.e., separation from the Mars Surveyor),
- Aerodynamical simulation,
- Soil impact and penetration.

To illustrate the utility of adaptive problem solving, we now briefly describe current work on a simplified version of the soil penetration stage.

Given a number of parameters describing the initial conditions including the angle of attack of the penetrator, the impact velocity, and the hardness of the target surface, the optimization problem is to select the total length and outer diameter of the penetrator, where the objective is to maximize the ratio of the depth of penetration to the length of the penetrator. We maximize this ratio, rather than simply maximizing the depth of penetration, since for the Mars microprobe science mission, the depth of penetration should ideally penetrate at least the length of the entire penetrator).

One of the initial condition parameters that has a significant impact on the structure of the cost surface for this optimization problem is the soil number, which indicates the hardness of the target surface. Intuitively, one would expect this to be an important parameter, since, for example, it is clearly more difficult to penetrate harder targets (the penetrator could bounce off the target, for example).

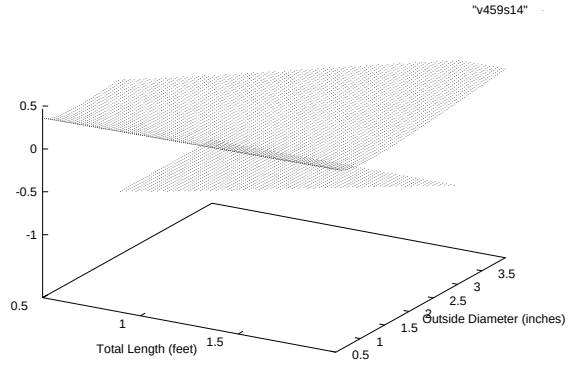

Figure 2: Sample points from cost surface for soil penetrator microprobe model. Plot of ratio of depth of penetration to length of penetrator. Soil number = 13 (soft soil). The z-axis represents the fitness value.

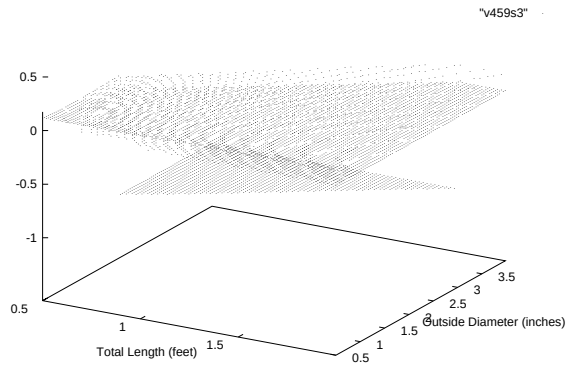

Figure 3: Sample points from cost surface for soil penetrator microprobe model. Plot of ratio of depth of penetration to length of penetrator. Soil number = 7 (hard soil).The z-axis represents the fitness value.

Figures 2 and 3 show plots of sample points from the cost

surface of this simplified penetration problem for two different soil numbers, *soillNum=7* (hard), and *soilNum=13* (soft). The cost surface for *soilNum=13* is a relatively smooth surface, while the cost surface for *soilNum=7* is a much more rugged surface Because of the larger number of discontinuities in the cost surface for *soilNum=7*, optimization algorithms are more likely to get stuck in local maxima in this cost surface. We would expect that a greedy metaheuristic would be very successful for the soft surface, while a successful metaheuristic for the hard surface would require some mechanism to escape local minima. Therefore, to obtain the best performance on a similar problem instance (given a different soil number, for example), one should choose and configure a metaheuristic to exploit this knowledge appropriately; this process would benefit from the application of our adaptive problem solving approach. The soil number is therefore a problem parameter that the OASIS system can use as a feature with which to classify problem instances (i.e., into problems with soft and hard soil numbers).

4.2 The Neptune Orbiter

Neptune Orbiter is a mission concept currently being studied under the Outer Planet Orbital Express program at the Jet Propulsion Laboratory. The goals of the mission are to put a spacecraft in orbit around Neptune using state-of-the-art technologies in the areas of telecommunications, propulsion, orbit insertion, and autonomous operations. The spacecraft is expected to arrive at Neptune five years after launch in 2005 using a Delta launch vehicle. The subsystem requirements include 100 kbps data rate, solar electric propulsion, solar concentrator power source and a cost of less than $400M[8].

For the initial phase of the optimization demonstration, the focus is on the orbital operations of Neptune Orbiter. The launch and cruise phases of the mission will be included in the optimization once the orbiter problem is well understood. The driving constraints of the orbiter problem are the optical communication aperture, transmit power, and spacecraft mass. The transmit power is a direct input into the integrated spacecraft design model. The other inputs include the science observation time per orbit and the data compression factor. The output of the model that is being maximized is the science data volume per orbit. For designs in which the spacecraft mass is greater than 260kg, the data volume output is zero. A spacecraft with a dry mass of greater than 260kg is too heavy to lift on the target launch vehicle. Thus the mass limit bounds the optimization problem. Currently, we are using cost models in conjunction with the simulation of the orbiter as described above to obtain our cost function-a quantitative estimate of the science return (measured in, e.g., volume of science data obtained per dollar cost of the spacecraft).

[8] In Fiscal Year 1994 dollars.

5 Summary and Conclusion

Designing a widely applicable tool for spacecraft design optimization is a significant technical challenge. In this paper, we have proposed the use of metaheuristic optimization algorithms, which are customized for particular problem instances by a process of adaptive problem solving, and we have described OASIS, our current implementation of a system based on these principles, and discussed many of the technical issues that have arisen in its design. By this, we hope to provide a design optimization tool that can provide spacecraft designers with the ability to perform successful design optimization with minimal human effort.

Acknowledgments

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Bog Glaser and Celeste Satter provided the spacecraft models for the Mars Microprobe and Neptune Orbiter, respectively. Thanks to Julia Dunphy and Jose Salcedo for assistance with MIDAS.

References

[1] J. George, J. Peterson, and S. Southard. Multidisciplinary integrated design assistant for spacecraft (MIDAS). In *Proceedings of American Institute of Aeronautics and Astronautics (AIAA)*, 1995.

[2] F. Glover. Tabu search – part i. *Operations Research Society of America (ORSA) J. Computing*, 1:190–206, 1989.

[3] F. Glover. Tabu search – part ii. *Operations Research Society of America (ORSA) J. Computing*, 2:4–32, 1990.

[4] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[5] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition.* Cambridge University Press, 1992.

[7] D.H. Wolpert and W.G. Macready. The mathematics of search. SFI-TR-95-02-010, 1994.