SHERPA – An Efficient and Robust Optimization/Search Algorithm

Introduction

Numerical design optimization is now an industry-accepted practice for more quickly identifying designs that meet increasingly stringent performance specifications and cost targets. Rather than manually iterate on design parameters in the hope of finding a design that meets the required specifications, automated numerical optimization algorithms can yield much better designs in much less time. These algorithms work with existing analysis tools, which predict how well a design performs. So the final result of an optimization run is an analyzed model of the best design and its predicted response characteristics.

One of the keys to a successful optimization study is the effectiveness of the search algorithm used. This paper provides brief answers to the following questions about optimization algorithms:

- What does it mean for an algorithm to be efficient and robust, and why is it important?
- How do various algorithms compare on these important characteristics?
- What makes the algorithm SHERPA so effective?

1. Efficiency and Robustness of Optimization Algorithms

Optimization algorithms use the results from numerical analyses and simulations, herein called “evaluations,” to guide the search for an optimal design. For example, a finite element analysis of a particular design candidate would be called an evaluation. An algorithm’s efficiency is measured in terms of the total number of evaluations required to find the optimal design or a design of a specified performance level. Using fewer evaluations is important because often each evaluation can require a significant amount of CPU time. For example, many nonlinear finite element simulations require from several hours to several days of CPU time. So reducing the total number of evaluations needed has a large impact on the time required to find an optimized design.

The search path taken by an optimization algorithm will generally be different in each run. This means that the number of evaluations required to achieve a given level of design performance can be quite different from run to run. More importantly, the final results of several runs using the same algorithm may not be the same – that is, each run may fall short in some way from finding an optimal solution.

These differences depend upon the starting conditions of the search, such as the initial or baseline design in a gradient-based algorithm or the initial population of designs in an evolutionary algorithm. Ideally, the performance of an optimization algorithm should be similar under all sorts of different starting conditions. Such an algorithm is said to be robust. This property is important for instilling confidence in the results of an algorithm and for approximating the number of evaluations needed to identify a good design.

2. A Simple Benchmark Problem

The true test of any optimization algorithm is its overall performance on numerous problems of varying type and complexity. But all algorithms considered for general use should at least be effective at solving rather simple problems such as the one considered here.

We consider a cantilevered I-beam subjected to a tip load, as shown in Figure 1. The goal is to design the cross-sectional shape of the I-beam such that a minimum mass solution is found that also satisfies constraints on the stress and deflection. Mathematically, this is expressed as:

Minimize: $m\ (H, h_1, b_1, b_2)$

such that: $\sigma_{\max} \le \sigma_{all}$

$u_{\max} \le u_{all}$

where (H, h_1, b_1, b_2) are the design variables, m is the mass of the beam, $\sigma_{\max}$ is the maximum bending stress, $\sigma_{all} = 5000$ psi is the allowable bending stress, $u_{\max}$ is the maximum deflection, and $u_{all} = 0.1$ inches is the allowable deflection.

The variables are allowed to vary as follows:

$$3.0 \le H \le 7.0$$
$$0.1 \le h_1 \le 1.0$$
$$2.0 \le b_1 \le 12.0$$
$$0.1 \le b_2 \le 2.0$$

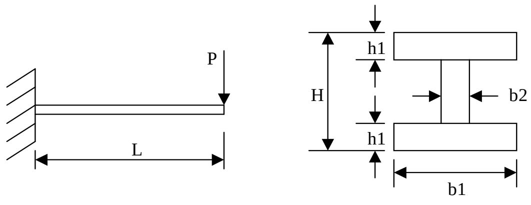

Figure 1. *(a) Cantilever beam with a tip load. (b) Cross sectional shape variables in the I-beam.*

Five different algorithms were investigated. Four of the algorithms were selected because they are the most widely used methods today, and are available in most commercial optimization packages. The fifth algorithm is SHERPA, which is a proprietary method available in the software package HEEDS. The algorithms considered are:

SHERPA – simultaneous hybrid exploration that is robust, progressive and adaptive. This is a multi-point, hybrid adaptive algorithm described below in Section 3.

GA – genetic algorithm. This is a multi-point, evolutionary search method that performs global exploration of the design space while searching for an optimal solution. It does not require the calculation of solution gradients.

SA – simulated annealing. This is a single-point algorithm that is capable of finding global optima but often finds a local optimal solution. It is not dependent on solution gradients.

NLSQP – non-linear sequential quadratic programming. This is a single-point, gradient- based approach that typically exhibits good convergence toward the nearest local optimal solution.

RSM – response surface method. This method searches a surrogate approximation model that is generated by fitting a chosen function to a set of evaluation data points. Here, a quadratic least squares surface was used.

This study investigated the ability of each algorithm to find the known optimal (lowest mass) solution using a specified number of evaluations. For a given number of allowable evaluations, each algorithm was run fifty (50) times from different (random) starting conditions. The best solution found in each run was recorded, and the average of these best solutions was calculated and then normalized by the known optimal solution. These results are shown in Figure 2. The standard deviation of these solutions was also calculated, as shown in Figure 3.

Prior to executing the final runs for this study, an effort was made to “tune” the parameters in each algorithm in order to increase the effectiveness of the search for this particular problem. The exception was SHERPA, which has no tunable parameters. All of its parameters are automatically adjusted in an adaptive manner, as discussed in Section 3.

In Figure 2, it can be seen that SHERPA required many fewer evaluations than the other algorithms did to find nearly optimal solutions. Alternatively, it may be said that for a given number of allowable evaluations, SHERPA found much better designs than the other algorithms did. On this particular problem, SHERPA is evidently much more efficient than the other algorithms considered, even after tuning their parameters to improve performance. Similar conclusions have been reached for many other problems.

The results in Figure 3 indicate that SHERPA is also very robust for this problem, having the least amount of variation in the final solutions found for all levels of evaluations performed. This increased robustness is due in part to the hybrid nature of SHERPA, as described below.

3. An Overview of SHERPA

HEEDS contains a unique search strategy called SHERPA, which stands for Simultaneous Hybrid Exploration that is Robust, Progressive and Adaptive.

During a single search, SHERPA uses multiple search methods simultaneously (not sequentially). This approach takes advantage of the best attributes of each method, and reduces a method’s participation in the search if/when it is determined to be ineffective.

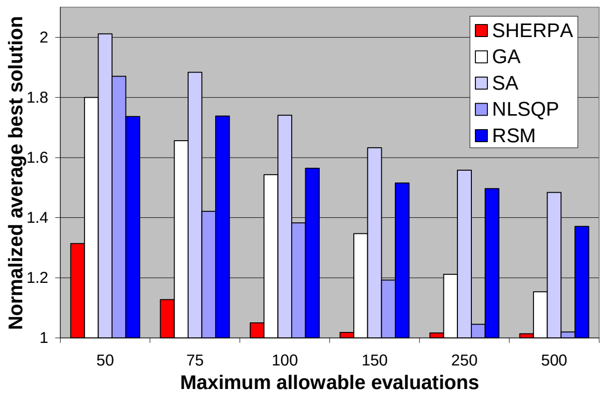

Figure 2. Average best solution found over 50 runs versus the number of allowable evaluations. All solutions are normalized by the known optimal solution.

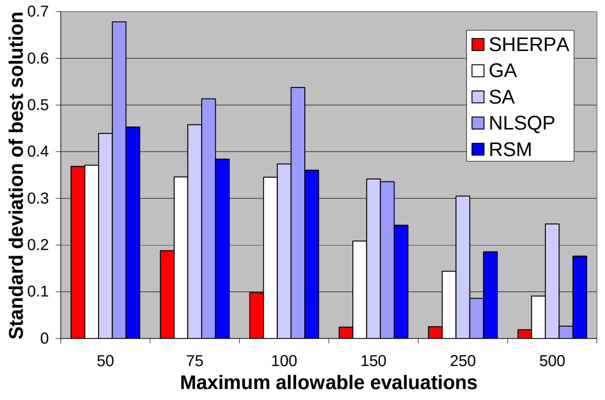

Figure 3. Standard deviation of the best solutions found over 50 runs versus the number of allowable evaluations.

A combination of global and local search methods is used, with the number of different methods used at any time ranging between two and ten.

Each method contains tuning parameters that are modified automatically during the search according to knowledge gained about the nature of the design space. This evolving knowledge about the design space also determines when and to what extent each method is used. In other words, SHERPA efficiently learns about the design space and adapts itself so as to effectively search all sorts of design spaces, even very complicated ones. Naturally, there is no claim that this approach is better for all problems than some other approach might be, or that it will always find a global optimal solution, but it has been shown to work very effectively and efficiently for many practical engineering design problems.

Aside from being very efficient, robust and effective, the main advantages of this approach include:

- Users need not spend time and effort trying to understand their design space before choosing a suitable algorithm for an optimization run. SHERPA will learn about the design space and employ the appropriate algorithms as it proceeds toward finding an optimized solution.
- Users do not need much, if any, expertise in optimization algorithms and applications, because SHERPA makes all of the decisions about which methods to use and how to tune them.
- Users can define a problem realistically, based on actual engineering or business costs and benefits, without feeling constrained by the capabilities of a particular search method. Problem definitions can be much broader and include a larger number of variables.

4. Summary

The importance of efficiency and robustness in optimization algorithms was discussed, and the results of a simple benchmark example were described to highlight the performance of some commonly used algorithms. For this problem, the new hybrid adaptive algorithm SHERPA was shown to be far superior to the other algorithms in terms of both efficiency and robustness.