Experience with Approximations in the Trust-Region Parallel Direct Search Algorithm[*]

S.M. Shontz[1,**], V.E. Howle[2], and P.D. Hough[3]

[1] Department of Computer Science and Engineering
The Pennsylvania State University,
University Park, PA 16802
shontz@cse.psu.edu

[2] Department of Mathematics and Statistics
Texas Tech University Lubbock, TX 79409
victoria.howle@ttu.edu

[3] Advanced Software Research and Development Department
Sandia National Laboratories
Livermore, CA 94551
pdhough@sandia.gov

Abstract. Recent years have seen growth in the number of algorithms designed to solve challenging simulation-based nonlinear optimization problems. One such algorithm is the Trust-Region Parallel Direct Search (TRPDS) method developed by Hough and Meza. In this paper, we take advantage of the theoretical properties of TRPDS to make use of approximation models in order to reduce the computational cost of simulation-based optimization. We describe the extension, which we call mTRPDS, and present the results of a case study for two earth penetrator design problems. In the case study, we conduct computational experiments with an array of approximations within the mTRPDS algorithm and compare the numerical results to the original TRPDS algorithm and a trust-region method implemented using the speculative gradient approach described by Byrd, Schnabel, and Shultz. The results suggest new ways to improve the algorithm.

Keywords: approximation models, parallel optimization, nonlinear programming.

1 Introduction

Coupling optimization software with simulations has become a common way to address optimal design and analysis questions. For example, one may want to

[*] Most of this work was performed while all of the authors were members of the Computational Sciences and Mathematics Research group at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

[**] Part of this work was performed while the first author was a member of the Center for Applied Mathematics at Cornell University, supported by the National Physical Science Consortium.

identify parameters for a new model such that simulation results most closely match experimental results or to determine an optimal device design. It is well-known that simulation-based optimization problems pose many challenges. We consider gradient-based methods in which finite-difference approximations to the gradient are used because analytic gradients are not available. We consider problems with a small number of variables, so the dominant optimization cost is the function evaluation cost. Our goal is to reduce the number of function evaluations performed by leveraging parallelism and approximation models that incur less computational expense.

There are many ways of parallelizing optimization algorithms such as [1]-[5]. We focus on two variants of trust-region optimization methods. The first is a speculative gradient technique introduced in [6]. The key assumption of this method is that in a classical Newton method (with either line search or trust region), the initial trial point at each iteration is usually accepted. Since small clusters of processors are commonly available, the additional processors can be used to begin computing finite-difference gradient components at the trial point while the function is being evaluated. If the trial point is rejected, nothing is lost. However, since it is usually accepted, this approach results in substantial computational savings. The other approach we employ is the Trust-Region Parallel Direct Search (TRPDS) algorithm developed in [7]. This method combines the trust-region version of a Newton method with a parallel direct search (PDS) method in a way that retains the best properties of both. In particular, PDS is used to augment the set of search directions to offset inaccuracies in the numerical gradient approximations. Since PDS is inherently parallel, this can be done without any additional cost when multiple processors are available.

Approximation models are another approach to reducing the overall cost of the optimization. There are many ways in which an approximation to a high-accuracy model can be constructed, such as response surface and spacing mapping techniques for constructing surrogates in [9]. Examples of optimization strategies that make use of approximations are described in [10]-[13]. In this work, we consider the use of approximation models within the TRPDS algorithm. Unlike [10] and [13], this approach incorporates the use of numerical gradients. In addition, we do not assume that trial iterates satisfy decrease conditions by construction as is true in classical trust-region methods and in [12]. Our work is related to [11] in that both fall into a general class of trust-region algorithms described in [8], and both leverage the flexibility of this class of algorithms by using approximation models to reduce computational cost. The primary distinction is that our approach seeks to find decrease quickly at each iteration using the approximation rather than optimizing the approximation at each iteration as in [11]. We will present a case study using TRPDS in conjunction with approximations obtained by reducing accuracy of the physics model and by constructing quadratic representations. Results suggest ways of improving the algorithm.

2 Speculative Gradients

Recall that a trust-region method is an optimization method in which each iteration entails constructing a quadratic Taylor series expansion of the function and minimizing that expansion over a region in which it is expected to be a good approximation to the function. We refer the reader to standard references like [19] for details, but we present a summary of the speculative gradient version of the algorithm here.

We first note that a function gradient is required to construct each Taylor series expansion and is computed using finite-difference calculations since analytic gradient information is not available. Furthermore, let us assume that we have p processors at our disposal. For simplicity, the algorithm is described assuming that a function evaluation uses one processor and forward/backward finite-differences are being performed. Generalizing to multiprocessor function evaluations and central differences is straightforward. To maximize the use of available processors, one processor is used to evaluate the trial point, and the remaining $p-1$ processors are used to calculate up to $p-1$ components of the finite-difference gradient. If the trial point is accepted, we have $p-1$ components of the gradient available and only need to calculate the remaining $n-(p-1)$ components, where n is the problem dimension. If the trial point is not accepted, no time is lost because the function evaluation is required regardless. The speculative-gradient trust-region algorithm is shown below in Algorithm 1. Here $\mathbf{x}_k$ is the current iterate, $\mathbf{s}$ is the trial step, $g(\mathbf{x}_k)$ is the gradient of f at the current point, $H_k \approx \nabla^2 f(\mathbf{x}_k)$ is the Hessian approximation at the current point, δ_k is the size of the trust region, and

$$\psi(\mathbf{s}) = g(\mathbf{x}_k)^T\mathbf{s} + \frac{1}{2}\mathbf{s}^T H_k \mathbf{s}. \tag{1}$$

Algorithm 1. Trust-Region Method with Speculative Gradients

```
Given $p$ processors, $\mathbf{x}_0$, $\mathbf{g}_0$, $H_0$, $\delta_0$, and $\eta \in (0,1)$
for $k = 0, 1, \ldots$ until convergence do
    for $i = 0, 1, \ldots$ until step accepted do
        1. Find $\mathbf{s}_i$ that approximately solves the quadratic subproblem
        2. Processor 0: evaluate $f(\mathbf{x}_k + \mathbf{s}_i)$
           Processor $j$: evaluate $g_{j-1}(\mathbf{x}_k + \mathbf{s}_i)$ for $j = 1, \ldots, p-1$
        3. Compute $\rho = (f(\mathbf{x}_k + \mathbf{s}_i) - f(\mathbf{x}_k))/\psi(\mathbf{s}_i)$
        if $\rho > \eta$ then
            4. Accept step, set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_i$
            5. Processor $j$: for $j = 0, \ldots, p-1$:
            for $l = j+1, \ldots, n-(p-1)$, step $p$, do
                6. Evaluate $g_{l+(p-1)}(\mathbf{x}_k + \mathbf{s}_i)$
            7. Update $H_k$
        else
            8. Reject step
        9. Update $\delta_k$
```

3 TRPDS with Generalized Approximation Models

In 2002, Hough and Meza developed the Trust Region-Parallel Direct Search (TRPDS) algorithm [7]. TRPDS employs the trust-region framework but uses the PDS algorithm of Dennis and Torczon [1] to solve a non-standard subproblem, the PDS subproblem, at each iteration. Solving this subproblem entails using PDS to minimize the function itself subject to the trust-region constraint and a fraction of Cauchy decrease constraint. See [7] for more details.

The original motivation for TRPDS was to combine the desirable convergence properties of trust-region methods with the robustness of PDS for low-accuracy functions. However, this algorithmic combination has a great deal of additional flexibility, which we leverage by extending the subproblem solution to a two-phase approach that incorporates the use of an approximation model. The first phase consists of using PDS to find the j best solutions to the following problem:

$$\begin{aligned} \min_{\mathbf{s}\in\mathbb{R}^n} \quad & m(\mathbf{x}_k + \mathbf{s}) \\ \text{s. t.} \quad & \|\mathbf{s}\|_2 \le 2\delta_k, \\ & \psi(\mathbf{s}) \le \beta \|g(\mathbf{x}_k)\| \min\left(\delta_k, \frac{\|g(\mathbf{x}_k)\|}{C}\right), \end{aligned} \tag{2}$$

where j is an integer, m is a computationally inexpensive approximation to the objective function, $\mathbf{x}_k$ is the current iterate, $\mathbf{s}$ is the trial step, δ_k is the size of the trust region, $\beta > 0$, $C > 0$, and $\psi(\mathbf{s})$ is defined in (1). This resembles the PDS subproblem except the objective function has been replaced by an approximation model. Also, the constraint on $\psi(\mathbf{s})$ enforces the fraction of Cauchy decrease condition. We are using p processors and are taking $j = p$ for simplicity of description. In the second phase, each processor evaluates the objective function at one of these j trial points. The point that yields the lowest function value is returned to and processed by the trust-region framework. This variation of TRPDS, referred to as mTRPDS, is given in Algorithm 2 below.

Algorithm 2. mTRPDS

```
Given $p$ processors, $\mathbf{x}_0$, $\mathbf{g}_0$, $H_0$, $\delta_0$, and $\eta \in (0,1)$
for $k = 0, 1, \ldots$ until convergence do
    1. Solve $H_k \mathbf{s}_N = -\mathbf{g}_k$
    for $i = 0, 1, \ldots$ until step accepted do
        2. Form an initial simplex using $\mathbf{s}_N$
        3. Compute the $p$ best approximate solutions $\mathbf{s}_1,\ldots,\mathbf{s}_p$ to (2) using PDS
        4. Determine $\mathbf{s} \in \{\mathbf{s}_1, \ldots, \mathbf{s}_p\}$ that minimizes $f(\mathbf{x}_k + \mathbf{s})$
        5. Compute $\rho = (f(\mathbf{x}_k + \mathbf{s}_i) - f(\mathbf{x}_k))/\psi(\mathbf{s}_i)$
        if $\rho > \eta$ then
            6. Accept step and set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_i$, evaluate $\mathbf{g}_{k+1}$, $\mathbf{H}_{k+1}$
        else
            7. Reject step
        8. Update $\delta$
```

In [7], it was observed that the TRPDS class of algorithms fits into the generalized trust-region framework of Alexandrov et al. [8], which provides much flexibility in the choice of trust-region model and in the step computation. In particular, let a be an approximation to the objective function, f. If $a(\mathbf{x}_k) = f(\mathbf{x}_k)$, and $\nabla a(\mathbf{x}_k) = \nabla f(\mathbf{x}_k)$, and the sequence of iterates generated during the optimization satisfies a fraction of Cauchy decrease condition according to a, then the standard trust-region convergence theory implies this class of methods will converge to a local minimizer of f [8]. Note that a is different from the approximation, m, described in mTRPDS. The former could be any model that satisfies the conditions above; however in mTRPDS, as in TRPDS, we fix $a = \psi$, where ψ is defined in (1). Furthermore, the fraction of Cauchy decrease condition is enforced by the second constraint in (2), so the iterates generated by mTRPDS also satisfy the above assumptions. Thus, as with TRPDS, mTRPDS is guaranteed to converge according to the theory in [8].

Now that we have established the convergence properties of our TRPDS modification, we present a case study for an earth penetrator design problem.

4 Case Study for Earth Penetrator Design

To evaluate the mTRPDS algorithm, we consider two problems in earth penetrator design, a problem of long-standing interest to the Departments of Energy and Defense. We consider the scenario in which there is a pre-existing hole in the target, as depicted in Fig. 1.

The penetrator model is fairly simple in that we consider it to be a solid "egg" made of one material. The penetrator shaft is divided into three sections, and their lengths are varied independently. The radius is held constant. In the first problem, we wish to find lengths that minimize the maximum acceleration

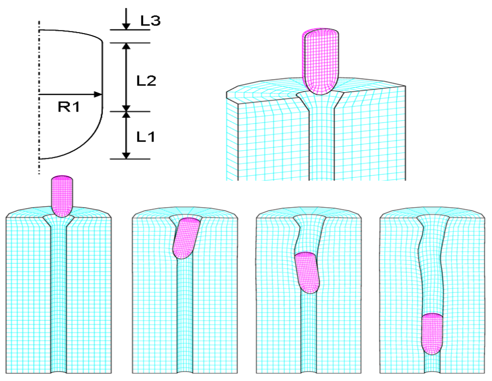

Fig. 1. The earth penetrator radius is held fixed, while the lengths are varied independently. The goal is to find section lengths that will optimize mission performance.

subject to bounds on the length parameters. Our optimization problem is the following:

$$\begin{aligned} \min_{\mathbf{L} \in \mathbb{R}^3} \quad & F(\mathbf{L}) = \max(\text{acceleration}) \\ \text{s.t.} \quad & l_i \leq L_i \leq u_i, i = 1 \ldots 3, \end{aligned} \tag{3}$$

where $\mathbf{L}$ is the vector containing the three unknown length parameters, L_i, and l_i and u_i are the lower and upper bounds, respectively. In the second problem, we wish to find lengths resulting in maximal penetration depth subject to bounds on the length parameters. Our optimization problem is the following:

$$\begin{aligned} \min_{\mathbf{L} \in \mathbb{R}^3} \quad & F(\mathbf{L}) = -(\text{depth of penetration}) \\ \text{s.t.} \quad & l_i \leq L_i \leq u_i, i = 1 \ldots 3. \end{aligned} \tag{4}$$

Presto, a Sandia-developed three-dimensional explicit transient dynamics code that is implemented using Lagrangian finite elements [14], is used to model the mechanical deformation of the penetrator upon impact. The ACME library [15] is used for the contact algorithms. We use a finite element model that represents a solid, homogeneous body. The penetrator is modeled as an elastic material, and a Mohr-Coulomb soil constitutive model is used to represent the target. The penetrator and target models are axisymmetric. CUBIT [16] was used to develop a parametric mesh model that was used to generate the finite element mesh for each set of length parameters. Meshes consist of eight-node hex elements, and the time step is chosen to satisfy the Courant stability condition.

To compare the approaches, we ran a set of computational experiments on a Linux cluster with dual 3.6 GHz Intel EM64T processors with 6 GB RAM. Each node runs Red Hat Enterprise Linux WS 4 and MPICH over an Infiniband network. For mTRPDS, we chose a range of approximations constructed in the following ways: 1) altering the mesh discretization, 2) altering the amount of event time simulated, and 3) using a Taylor series to construct a quadratic model of the function. The results were compared to those obtained using TRPDS and the trust-region method with speculative gradients. In all cases, the gradient was approximated by central differences, and a BFGS approximation to the Hessian was employed.

We chose the number of processors to be that which is ideal for speculative gradient computation. The penetrator design problem has three variables, so seven function evaluations need to be done simultaneously to compute the objective and central difference gradient at the trial iterate. We used 16 processors for each simulation and one for the optimization process, totaling 113 processors. Thus, we also used 113 processors for the mTRPDS computational experiments. The ideal settings for mTRPDS on 113 processors are a search pattern size (sps) of 7 and $j = 7$. The optimization algorithms are implemented in OPT++, and additional algorithmic parameters were set to their default values shown in [17].

Table 1 shows the set of algorithms and models used in the experiments and the wall clock time for a single execution of each model. The approximation models were named according to the following convention: MeshXk refers to the

Table 1. This table shows the algorithm-model combinations used in experiments and the time required for a single execution of each model. SpecGrad and TRPDS use the truth model, where mTRPDS uses the truth model along with the specified approximation models.

Key	**Algorithm**	**Model**	**Time for Single Model Execution**
1	SpecGrad	Truth (Mesh640k, Time25ms)	2 – 3 hours
2	TRPDS	Truth (Mesh640k, Time25ms)	2 – 3 hours
3	mTRPDS	QuadraticModel	negligible
4	mTRPDS	Mesh10k	0.8 – 1.3 hours
5	mTRPDS	Mesh80k	1.3 – 1.8 hours
6	mTRPDS	Time6.25ms	1.1 – 1.6 hours
7	mTRPDS	Time12.5ms	1.7 – 2.3 hours

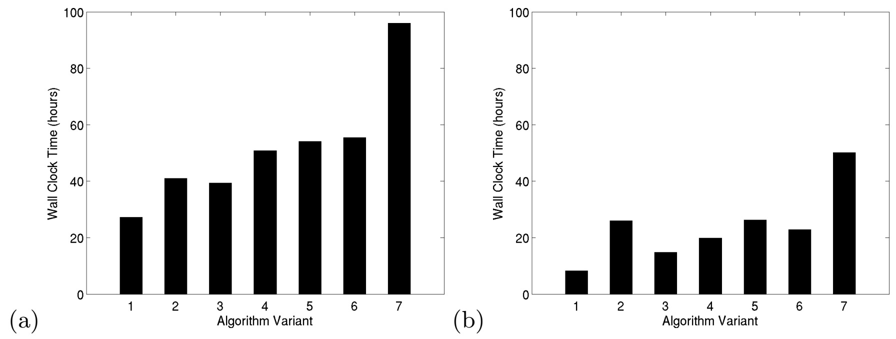

Fig. 2. Key is given in Table 1. (a) This figure shows the wall clock time required to achieve a 0.1% change in the function value for (3). Variations in results are due primarily to different numbers of iterations. (b) This figure shows the wall clock time required to achieve an 0.1% change in the function value for (4). Variations in results are due primarily to different costs per iteration.

model using a mesh with X thousand elements; TimeYms refers to the model in which the event time simulated is Y milliseconds, and QuadraticModel is the quadratic Taylor series expansion for f. The results of the experiments appear in Figs. 2 through 4. Figure 2 shows the wall clock times to reach a 0.1% change in the function value for the algorithms and approximations tested. We use this criteria because investigation into behavior of the algorithms after this point uncovered the need for further research into useful numerical stopping criteria.

For problem (4), shown in Fig. 2b, all experiments took approximately the same number of iterations. The variations in wall clock times, therefore, are due primarily to differences in the average wall clock time per iteration. We see from Fig. 3b that there can be a notable difference in time per iteration. Further investigation into these differences revealed opportunities to improve the computational efficiency of the mTRPDS algorithm. For the depth of penetration, the

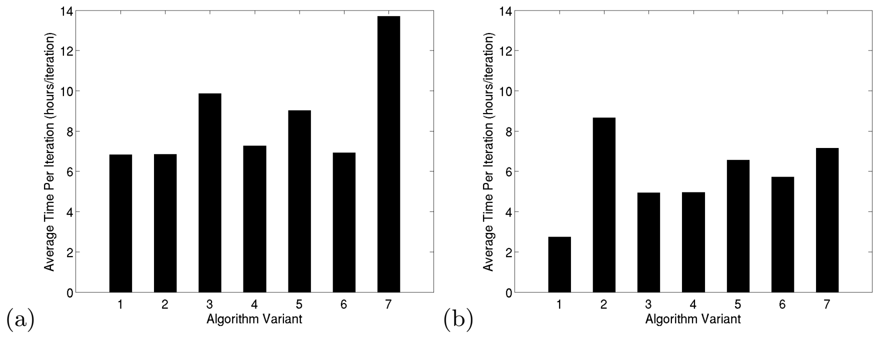

Fig. 3. Key is given in Table 1. (a) This figure shows the average time per iteration for (3). Results suggested a need for better characterization of algorithm and approximation performance. (b) This figure shows the average time per iteration for (4). Results suggested specific improvements in computational efficiency for mTRPDS.

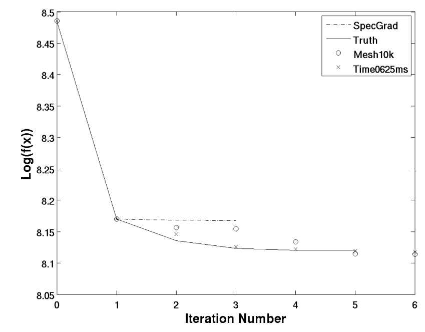

Fig. 4. Log of function value vs. iteration for the four algorithms with comparable average time per iteration for problem (3). TRPDS-based algorithms move to a solution with a lower function value, thereby taking more iterations.

approximation models track the truth fairly well (see [18]). This indicates that we should reduce the value of j and incorporate the computation of speculative finite-difference gradients for those j points. It is possible that the best choice is $j = 0$. This would reduce the number of truth evaluations needed at each iteration, thereby reducing the total time. We also found that PDS was doing approximation evaluations that make little or no apparent contribution to the progress of the optimization algorithm. Even though PDS is operating on the approximation models, this can add up to notable time. We would like to eliminate that by developing a dynamic scheme for managing the amount of work done by PDS based on the quality of the approximation evaluations it does.

Further investigation into the results for problem (3) show that the primary difference in wall clock times (shown in Fig. 2a) is due to the algorithms taking different numbers of iterations. We see in Fig. 3a, however, that several algorithm-model combinations have comparable average times per iteration. To get further insight, we plot the function value versus iteration number for those variants in Fig. 4. We see that the TRPDS-based algorithms start out the same as the speculative gradient algorithm but then move toward solutions with lower function values, thereby taking longer. To better understand the reasons for this behavior, further characterization of the effects of problem features on algorithm performance is needed. Such characterization requires a set of computationally expensive, physics-based test problems. Test problems with these characteristics are hard to come by, as standard test sets do not meet these criteria.

5 Conclusions and Future Work

We have extended the TRPDS algorithm of Hough and Meza to include the use of an approximation model in solving the PDS subproblem. This approach, which we call mTRPDS, uses the approximation to identify several candidates for the trial iterate and takes advantage of parallel processing to evaluate them. The algorithm was studied, together with TRPDS and a speculative gradient implementation of the classical trust-region method, in the context of two earth penetrator optimal design problems but is generally applicable to any simulation-based unconstrained or bound-constrained nonlinear optimization problem.

The empirical results we collected from the numerical tests suggested both short-term, concrete areas for improving the computational efficiency of mTRPDS and longer-term research areas. To improve computational efficiency, we will further examine the choice of j to reduce the number of truth evaluations needed. We will also develop a means of dynamically managing the amount of work PDS performs using the approximation model. Longer-term research includes developing meaningful numerical stopping criteria for optimization algorithms and characterizing the effects of problem characteristics on algorithm performance.

Acknowledgments: The authors wish to thank John Dennis, Juan Meza, and Virginia Torczon for many helpful discussions and suggestions on the TRPDS-based algorithms. They would also like to thank Monica Martinez-Canales, Genetha Gray, Laura Swiler, Mike Chiesa, and Randy Settgast for assistance in implementing and understanding the earth penetrator application.

References

1. Dennis Jr., J.E., Torczon, V.: Direct Search Methods on Parallel Machines. SIAM J. Optimiz. 1(4), 448–474 (1991)
2. Ingber, L.: Simulated Annealing: Practice Versus Theory. Math. Comput. Model. 18(11), 29–57 (1993)
3. Laarhoven, P.J.M.: Parallel Variable Metric Algorithms for Unconstrained Optimization. Math. Program. 33, 68–81 (1985)
4. Phua, P.K.-H., Zeng, Y.: Parallel Quasi-Newton Algorithms for Large-Scale Optimization. Tech. Report TRB2/95, National Univ. of Singapore, Singapore (1995)

5. Straeter, T.A.: A Parallel Variable Metric Optimization Algorithm. Tech. Report NASA TN D-7329, NASA, Langley Research Center, Hampton, VA (1973)
6. Byrd, R.H., Schnabel, R.B., Shultz, G.A.: Parallel Quasi-Newton Methods For Unconstrained Optimization. Math. Program. 42, 273–306 (1988)
7. Hough, P.D., Meza, J.C.: A Class of Trust-Region Methods for Parallel Optimization. SIAM J. Optimiz. 13(1), 264–282 (2002)
8. Alexandrov, N., Dennis Jr., J.E., Lewis, R.M., Torczon, V.: A Trust Region Framework for Managing the Use of Approximation Models in Optimization. J. Struct. Optimiz. 15(1), 16–23 (1998)
9. Dennis Jr., J.E.: Surrogate Modelling and Space Mapping for Engineering Optimization: A Summary of the Danish Technical University November 2000 Workshop. Tech. Report CAAM-TR00-35, Rice University, Houston, TX (2000)
10. Booker, A.J., Dennis Jr., J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A Rigorous Framework for Optimization of Expensive Functions by Surrogates. J. Struct. Optimiz. 17(1), 1–13 (1999)
11. Giunta, A.A., Eldred, M.S.: Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit. In: Proceedings of 8th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA (2000)
12. Nash, S.G.: A Multigrid Approach to Discretized Optimization Problems. Optim. Method. Softw. 14, 99–116 (2000)
13. Siefert, C.M.: Model-Assisted Pattern Search. Honors Thesis, Department of Computer Science, College of William & Mary, Williamsburg, VA (2000), `http://www.cs.wm.edu/~va/CS495/siefert.ps.gz`
14. Koteras, J.R., Guillerud, A.S., Crane, N.K., Hales, J.D., Reinert, R.K.: Presto Users Guide 2.7. Tech. Report SAND2007-3749, Sandia National Laboratories, Albuquerque, NM (2007)
15. Brown, K.H., Summers, R.M., Glass, M.W., Guillerud, A.S., Heinstein, M.W., Jones, R.E.: ACME: Algorithms for Contact in a Multiphysics Environment, API Version 1.0. Tech. Report SAND2001-3318, Sandia National Laboratories, Albuquerque, NM (2001)
16. Owen, S.J.: CUBIT Geometry and Mesh Generation Toolkit (2006), `http://www.cubit.sandia.gov`
17. Meza, J.C., Hough, P.D., Williams, P.J., Oliva, R.A.: OPT++ 2.4 Documentation (2007), `http://csmr.ca.sandia.gov/opt++/opt++2.4_doc/html/index.html`
18. Martinez-Canales, M.L., Swiler, L.P., Hough, P.D., Gray, G.A., Chiesa, M.L., Heaphy, R., Thomas, S.W., Trucano, T.G., Lee, H.K.H., Taddy, M., Gramacy, R.B.: Penetrator Reliability Investigation and Design Exploration. Tech. Report SAND2006-7669, Sandia National Laboratories, Livermore, CA (2006)
19. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, London (1981)