Parallel Continuous Optimization*

J. E. Dennis Jr.† and Zhijun Wu‡

Abstract. Parallel continuous optimization methods are motivated here by applications in science and engineering. The key issues are addressed at different computational levels including local and global optimization as well as strategies for large, sparse versus small but expensive problems. Topics covered include global optimization, direct search with and without surrogates, optimization of linked subsystems, and variable and constraint distribution. Finally, there is a discussion of future research directions.

Key Words. Parallel optimization, local and global optimization, large-scale optimization, direct search methods, surrogate optimization, optimization of linked subsystems, design optimization, cluster simulation, macromolecular modeling

1 Introduction

Optimization has broad applications in engineering, science, and management. Many of these applications either have large numbers of variables or require expensive function evaluations. In some cases, there may be many local minimizers, and the user naturally wants to know how solutions found by the algorithm compare to other local solutions. These factors contribute

†Department of Computational and Applied Mathematics, and Center for Research on Parallel Computation, Rice University, Houston, Texas, 77005. Work supported by DOE DEFG03-95ER25257, AFOSR F49620-98-1-0267, The Boeing Company, Sandia LG-4253, Mobil and CRPC CCR-9120008.

‡Department of Computational and Applied Mathematics, Center for Research on Parallel Computation, and Keck Center for Computational Biology, Rice University, Houston, Texas, 77005. Work supported in part by DOE/LANL Contract 03891-99-23, the Keck Center for Computational Biology, the Robert A. Welch Foundation and NSF training grant 9413229.

to the need for more intensive computation than traditional architectures can support. High-performance computing provides powerful tools for solving these problems with a degree of practicality that would otherwise be impossible.

Example applications where parallel optimization can play an important role include aircraft shape design (Cramer, Dennis, et al [29]) and macromolecular modeling (Moré and Wu [64]).

In aircraft shape design, one attempts to match an ideal pressure distribution by manipulating the shape variables. The number of shape variables is in the order of hundreds at most, but they are constrained by at least two systems of PDEs. This is typical of many important applied optimization problems, there may not be so many decision variables for the optimizer, but there may be many ancillary variables that must be determined to compute the objective function and constraints. In order to obtain a feasible solution, the systems must match the input of each with the output of the others, in addition to satisfying side constraints such as range. The systems require expensive PDE solves for millions of grid points, different grids for different PDEs, and at least, there is one PDE to be solved for the air flow and one to be solved for the structural deflection.

This problem is computationally intensive because there is a great deal of linear and nonlinear algebra going on at each function and constraint evaluation. We will describe some domain decomposition type methods for this problem. As in that case, the *sequential* efficiency of the parallel optimization procedure can be better than more traditional methods.

In macromolecular modeling, one attempts to determine molecular structure by minimizing a given potential energy function. One of the most important applications is the determination of protein structures in structural molecular biology. The challenge in solving this problem is that the potential energy function has many local minimizers, while the structure to be determined is believed to correspond to a global or nearly global optimal solution to the minimization problem. Global optimization algorithms have been developed to solve the problem. Not surprisingly, they rely heavily on using computing power that only parallel high-performance architectures can provide.

Substantial research efforts on parallel optimization have been made for twenty years, and in the past ten years or so, some have born fruit by focusing on special applications and others by exploring more general parallel schemes.

Optimization has close relationships with numerical linear algebra and partial differential equations. For example, a typical optimization procedure requires solving a linear system in every iteration to predict a step to a better approximate solution; function or constraint evaluation often requires solving a partial differential equation. Thus, parallel optimization algorithms and software development certainly benefits from advances in parallel numerical linear algebra and partial differential equations.

General algorithms have also been developed such as parallel direct search methods by Dennis and Torczon [90, 34] and Torczon [90], parallel methods for optimization of linked subsystems by Dennis and Lewis [32] and Dennis , Li, and Williamson [33], and variable and constraint distribution schemes by Ferris and Mangasarian [37, 38].

Parallel global optimization has been one of the most active areas in parallel continuous optimization. Work in this area is motivated by important applications in chemical and biological disciplines such as cluster simulation and protein modeling. Algorithms and software developed in recent years include parallel stochastic global optimization algorithms for molecular conformation and protein folding by Byrd and Schnabel [21, 19], parallel global continuation software DGSOL for protein structure determination with NMR distance data by Moré and Wu [65, 64, 68, 66], and parallel effective energy simulated annealing for protein potential energy minimization by Coleman, Shalloway, and Wu [25, 26].

Optimization problems take different forms arising from the motivating applications. They can be linear or nonlinear, constrained or unconstrained, and local or global. They can be either large, sparse or small but very expensive to evaluate. This means that quite different parallel algorithms may be required and quite different architectures may be appropriate. For example, if the problem is large but sparse, a shared-memory system may be a good choice, for otherwise the distribution of a sparse, irregular structure over multiprocessors may cause load imbalance and severe communication overhead. On the other hand, most global optimization algorithms are coarsely parallel. They can be implemented on distributed-memory architectures, or even loosely connected networks of workstations, and still maintain scalability.

In the following sections, we discuss various parallel optimization methods in greater detail. We describe optimization problems and algorithms and their associated parallelism at different computational levels: function evaluation, algebraic calculation, and optimization. In particular, we review

parallel methods for local and global optimization, and compare strategies for large, sparse versus small but expensive problems. Parallel techniques including parallel direct search, optimization of linked subsystems, and variable and constraint distribution are introduced. Future research directions are discussed in the end.

2 Local Optimization

Let us consider the problem of minimizing a nonlinear function, $f(x)$, where f is continuous and differentiable for all $x \in R^n$. Generally, we would be given some incumbent approximate minimizer x^0. The most popular methods for this problem construct a quadratic model for the objective function (and a linear model of the constraints if they are present). This model problem is intended to represent the problem of interest in some neighborhood of x^0. Generally this is true because the model is built by using at least the $1st$ order Taylor series term. Often finite difference approximations to the derivatives are used, and this is an obvious opportunity for parallelism. In fact, the kind of parallelism used is one of the most useful for optimization, it is that values of the true function are obtained in parallel (see discussions in Byrd, Schnabel, and Shultz [23]).

Since the model is thought to represent the problem locally, one hopes that by finding a really good minimizer for the model, one will obtain a point that improves the real objective function. Thus, Newton or quasi-Newton algorithms choose a putative next iterate by solving the model problem. The difficulty with this procedure is that the solution to the model problem may be outside the region about x^0 where the model represents the problem well.

If the iterate found in this way is a better solution, then one moves to it and iterates the procedure. It will not surprise the reader that this procedure is likely to find the bottom of the same function valley one starts in, i.e., a nearby local minimizer x^*, assuming there is one, in the sense that for any x in a neighborhood of x^*, $f(x)$ is greater than or equal to $f(x^*)$.

If the pure iteration does not succeed in finding a better point, then it resorts to a globalization strategy. In this sense, globalization means convergence to some solution from any point, not solution to the global minimizer.

The two main classes of globalization algorithms for this problem are line searches and trust regions. Trust regions adaptively estimate a region

in which the local model can be "trusted" to adequately represent the true function. The next iterate is chosen by approximately minimizing the model over the trust region. Line search algorithms backtrack (usually) from the solution to the model problem along the direction from the incumbent. Each approach has its place in the optimization toolbox, and each has its own opportunities for parallelism.

Trust region algorithms can use parallelism in the linear algebra needed to solve the trust region subproblem - minimize the model in the region where it is trusted to represent the function (see Santos and Sorensen [78] and Rendl and Wolkowicz [75]). Line search algorithms can use parallel linear algebra to compute the solution to the model problem, and they can also use parallel function evaluations to find the best step along the direction they compute. Parallel multiple line searches (Nash and Sofer [70]), and parallel inexact Newton step computation (Nash and Sofer [69]) can be applied here.

For large-scale optimization, it is often useful to take advantage of the property of partial separability. That is, the objective function can be written in the following form,

$$f(x) = \sum_{i=1}^{m} f_i(x), \tag{1}$$

where f_i is called the element function of f and depend only on a small subset of the variables. This class of functions can be computed in parallel by distributing the element functions to the processors. Each processor will then be responsible for computing only the contributions of the element functions to the whole function, gradient, and Hessian.

Let processor i compute element functions $f_{i_1}, \ldots, f_{i_{max}}$. Then the function, gradient, and Hessian can be computed in the following procedure,

initialize $f,\ \nabla f,\ \nabla^2 f$
on processor i:
    do for $j \in \{i_1, \ldots, i_{max}\}$
        $f = f + f_j$
        $\nabla f = \nabla f + \nabla f_j$
        $\nabla^2 f = \nabla^2 f + \nabla^2 f_j$
    end do
end

where updates to f, ∇f, and $\nabla^2 f$ require global reduction on distributed-memory machines or access to shared-variables on shared-memory machines. However, the updates for the gradient and the Hessian can be done efficiently by updating only the elements for which the corresponding elements of ∇f_j and $\nabla^2 f_j$ are non-zeros (Averick and Moré [7] and Moré, Walenz, and Wu [63]).

The computation of the step, or its direction, with methods using the Hessian or Hessian approximations can be parallelized in several ways. In general, this is a place where a "plug-and-play" approach can be used by calling existing parallel linear algebra software such as LAPACK or SCALAPACK [3, 10]. For example, a parallel direct solver with Cholesky factorization can be used to compute the search direction $[B(x^i)]^{-1}\nabla f(x^i)$ at any iterate x^i if the Hessian or its approximation $B(x^i)$ is symmetric positive definite; a parallel matrix-vector multiplication routine can be used for computing all matrix-vector products in the truncated Newton or trust region subproblem solves. Byrd, Schnabel, and Shultz [23] showed that the quasi-Newton step can be obtained by using the inverse BFGS updates which then require only matrix-vector multiplications and can be parallelized straightforwardly with a parallel matrix-vector multiplication routine.

If one wishes to exploit sparsity, the above parallelization becomes more complicated. Several issues arise: First, an iterative solver can always be used for either a line search or trust region algorithm in the truncated Newton's method. This requires a pre-conditioner, which not only depends on the problem but also is more difficult to parallelize. Work on this issue can be found in Gropp, Smith, and Curfman [8] and Jones and Plassman [50, 51, 53, 52], who developed a parallel incomplete Cholesky factorization algorithm that seems efficient in practice.

Second, parallel direct sparse solves are difficult on distributed-memory machines, because data and computation are tricky to distribute to balance the load among processors. A symbolic factorization phase is a potential serial bottle neck in addition to the sparse triangular system solves. Coleman and Sun [27] developed a group of parallel direct sparse solvers for optimization using a multi-frontal approach.

Bokhari and Mavriplis [12], Feo, Kahan, and Wu [36], and Zaslavsky, Kahan, et al [96] demonstrated that the Tera multi-threaded architecture is particularly good for parallel sparse and irregular calculations. However, there is no general sparse matrix software available yet on this architec-

ture. Finally, sparsity patterns often change from application to application. Classes of optimization problems having the same sparsity patterns, like some large linear programming problems, need to be identified, and special parallel sparse solvers targeted to these classes of problems can then be developed. Work in this direction includes Bixby and Martin [9], Schneider and Wise [82], and Coleman and Wright [24].

3 Global Optimization

Research on global optimization has increased dramatically in recent years. An important reason is that the increasing power of parallel high-performance architectures makes it possible to attack many large, difficult global optimization problems of practical interest. Ten years ago, work in this area was still limited to toy problems of about 10 variables, but now, with the help of parallel computing, advanced algorithms have been developed and applied to problems with hundreds or even thousands of variables in such applications as cluster simulation (Byrd and Schnabel [21, 18, 19, 86, 20], Rosen and Xue [95, 94], Coleman, Shalloway, and Wu [25, 26]), protein folding (Byrd and Schnabel [22, 31], Scheraga [62, 74, 73, 54, 55], Shalloway [85, 84, 72], Coleman and Wu [28], and molecular docking (Meza, Plantenga, and Judson [48], Dill, Phillips, and Rosen [35]).

A global optimization problem requires a local minimizer with the lowest function value among all local optimizers. Certain classes of problems, like convex programming problems, have only one local minimum, but most function arising in applications are non-convex and they may have many local minima, constrained or unconstrained.

It is quite easy to see that general smooth continuous global optimization problems are intractable. Furthermore, even if one had the global minimizer in hand, it is an intractable problem to verify that it is anything more than a local minimizer.

Nevertheless, just because a problem is impossible to solve in general does not precludes useful research in the area. Often practical problems are posed as global optimization problems because that is the nearest model problem in the optimization toolkit to what the user really wants, and modern global optimization methods can find valuable solutions that satisfy the user.

This point is far less subtle than it may seem at first. To see this, consider

a hypothetical problem in engineering design. The designer wishes to find the best design for a widget in terms of a single design variable which is constrained to lie in a bounded interval. Suppose that there are three local minimizers as in the figure below. The left hand local minimizer and the wide shallow middle minimizer will be found by a good global optimization algorithm. On the other hand, the wide shallow middle minimizer is likely to be the only one found by a local algorithm. However the narrow right hand local minimizer, which is also the global minimizer, is unlikely to be found without an impractical amount of effort by any algorithm.

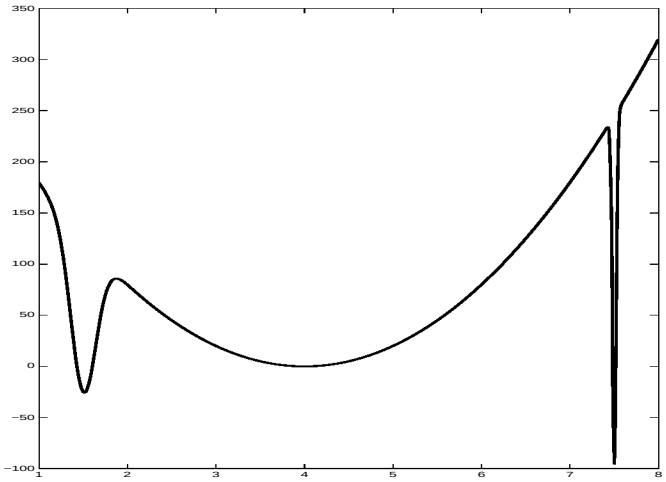

Figure 1: A function with three local minimizers

In practice, this may not be important at all, such a narrow minimum for the function is likely to have little practical value because if the process for manufacturing the widget leads to any variability in the decision variable, then the actual design criterion for the finished good will end up high up on the narrow valley at a much worse value than that in the more stable left hand valley. Of course, any decision maker would want to make that decision for themselves in possession of the location of the true global optimizer, but our point is that the more difficult a given global optimizer is to find, the less important it is likely to be to find it.

We describe some applications areas and related parallel global optimization work below.

Protein Folding

Protein folding is a fundamental unsolved problem in structural molecular biology. The problem is to determine how the protein amino acids fold to a unique three-dimensional structure. There are no direct physical means to detect this. X-ray crystallography and NMR spectroscopy have been used to derive approximate structures, but this requires months or even years of laboratory efforts for each protein.

The goal is to determine the structure with only the knowledge of the amino acid sequence of the protein by finding a structure corresponding to the global potential energy minimum. While this is possible in theory, it is computationally intense since it requires solving a global optimization problem with many thousands of degrees of freedom.

The potential energy function usually is given in an empirical form. It includes energy terms for such atomic interactions in proteins as electrostatic, van der Waal's, bonded, torsional, etc. Typically, the total energy E has the following form.

$$E = E_{elec} + E_{vand} + E_{bond} + E_{angl} + E_{tors}, \tag{2}$$

where

$$E_{elec} = \sum_{ij/electro} \frac{q_i q_j}{\epsilon r_{ij}}, \tag{3}$$

$$E_{vand} = \sum_{ij/vander} \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - 2\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}, \tag{4}$$

$$E_{bond} = \sum_{ij/bonded} k_{ij}(r_{ij} - r_{ij}^0)^2, \tag{5}$$

$$E_{angl} = \sum_{\theta/bonded} k_\theta(\theta - \theta^0)^2, \tag{6}$$

$$E_{tors} = \sum_{\phi/torsional} k_\phi[1 + \cos(n\phi - \phi^0)], \tag{7}$$

where θ, ϕ, and r_{ij} are bond angle, torsional angle, and pairwise distance and depend on the atomic positions, and everything else are given parameters.

Note that the potential energy function is defined in terms of the atomic positions x_i, $i = 1, \ldots, n$, where n is the number of the atoms in the protein and usually is in the range of 1,000 to 100,000. Recent work to develop

special methods for this problem includes Scheraga et al [73, 54, 55, 81], Straub et al [89], Coleman, Shalloway, and Wu [25, 26], Byrd and Schnabel [21, 22, 86, 31].

Cluster Simulation

Another class of global optimization problems comes from the emerging field of cluster science (Reynolds [76] and Haberland [40]). Clusters important for material design include argon clusters (Hoare and Pal [45, 44, 43]), various metal clusters (Jellinek [39, 49]), and clusters of carbon molecules such as the famous carbon 60, the Buckyball (Smalley [56]). A key research problem is to find the most stable configuration for any given cluster. The clusters may not exist in nature or be hard to observe. However, given a potential energy function, its global minimizer corresponds to the most stable configuration for that model of potential energy. As an example, the potential energy function for simulating argon clusters is:

$$E_{argon} = \sum_{ij} \frac{1}{\|x_i - x_j\|^{12}} - \frac{2}{\|x_i - x_j\|^6}, \tag{8}$$

where x_i and x_j are positions of the atoms.

Note that this function is very similar to the van der Waal's term in the protein potential energy function. As a matter of fact, they are indeed models of the same type of potentials due to the so-called van der Waal's weak forces between pairs of atoms. The potential energy function for the argon cluster is simpler than for proteins, but it is by no means easy to minimize. Hoare and Pal [45] estimated that this function has exponentially many local minimizers which grow as a function e^{n^2} of the number of atoms n in the cluster. Recent work on this problem includes Northby [71], Xue [94], Byrd and Schnabel [19], Coleman, Shalloway, and Wu [25, 26], and many others.

Distance Geometry

Strong motivation for solving distance geometry problems is their application in NMR macromolecular modeling, where a protein structure can be determined by solving a distance geometry problem using the NMR distance data.

They can be formulated as global nonlinear least-squares optimization problems (Crippen and Havel [30]), or, from a graph-theoretic point of view,

they are a class of NP-complete graph-embedding problems (Hendrickson [41], Saxe [79, 80], Moré and Wu [64]). Recent attempts to solve these problems on parallel high-performance architectures are by (Hendrickson [42], Moré and Wu [66], Byrd, Schnabel et al [97], etc).

A simple version of the distance geometry problem is to find a set of points to realize a given set of distances between some of the points. A more general version is to satisfy a given set of bounds on the distances. Mathematically, the problem is to find a set of points $x_i \in R^3,\ i = 1, \ldots, n$ such that the distance $\|x_i - x_j\|$ between points x_i and x_j is equal to a given distance d_{ij} or in between a given pair of bounds l_{ij} and u_{ij} of the distance. It can be formulated as a global optimization problem as follows. If d_{ij} are given,

$$\textbf{min} \sum_{(i,j)\in S} (\|x_i - x_j\|^2 - d_{ij}^2)^2, \tag{9}$$

where S is a given set of (i, j) pairs. If l_{ij} and u_{ij} are given,

$$\textbf{min} \sum_{(i,j)\in S} \min^2 \left\{ \frac{\|x_i - x_j\|^2 - l_{ij}^2}{l_{ij}^2}, 0 \right\} + \max^2 \left\{ \frac{\|x_i - x_j\|^2 - u_{ij}^2}{u_{ij}^2}, 0 \right\}. \tag{10}$$

Note that S may have (i, j) ranging from only a few to all possible pairs. For less than n pairs, the problem can be trivial. For all possible pairs, the problem still can be solved in polynomial time (Blumenthal [11], Crippen and Havel [30]). However, in practice, S is sparse, and the problem is hard to solve.

Stochastic Global Optimization

A stochastic global optimization method was proposed by Kan and Timmer [77]. Byrd and Schnabel [21] developed a parallel version. The method has these basic steps. A set of points is chosen in the problem domain, and the objective function is evaluated at the points. A subset of the points with low function values are selected as starting points for local minimization, which then is performed.

If one of the local minimizers is accepted as the global minimizer, the algorithm stops. Otherwise the process is repeated. Each time the starting points are selected from all previous, as well as current, sampled points. Therefore, as the algorithm proceeds, more and more points are sampled, and there are increasing chances to find the global minimizer.

Kan and Timmer [77] showed that with probability one the algorithm converges to a global minimizer in a finite number of iterations. Byrd and Schnabel [21] developed a parallel version of the algorithm by sampling starting points and performing local minimizations all in parallel. The problem domain is divided into smaller regions, each of which is assigned to a processor. Some regions are further refined if lower function values or local minima are found, and the subregions are reassigned to other processors when necessary to achieve load balance.

Although it requires dynamic load balancing, the stochastic global optimization algorithm is easy to parallelize and performs well on both shared- and distributed-memory architectures. Byrd and Schnabel [21] reported the development of the algorithm on KSR-1 and IBM SP2 and the performance results on protein and related molecular conformation problems. The algorithm has also been used by research groups in other institutions, including some at CRPC.

Effective Energy Simulated Annealing

The effective energy simulated annealing algorithm was developed by Coleman, Shalloway, and Wu [25]. The algorithm was parallelized and implemented on Intel iPSC/860 and IBM SP2. We describe the parallel implementation of this algorithm to show a general parallelization strategy for all simulated annealing type algorithms.

A physical annealing process starts from a high temperature, and then cools down by stages gradually to the zero temperature where the system reaches the ground state. The process has to proceed slowly so that at each cooling stage the system has enough time to reach equilibrium, for otherwise it will be trapped in a local state.

A simulated annealing algorithm tries to mimic this process by considering the objective function of the global minimization problem as the energy function of a simulated system. A parameter corresponds to the temperature and is decreased by stages. At each stage, function values are randomly sampled. Each time a point of lower potential energy is found, it is accepted as the current point. Otherwise a point is accepted or rejected randomly using the Metropolis criterion, which depends on the temperature: If the temperature is higher, the probability of accepting the point is also higher. This property allows the algorithm to sample and accept more points at high

temperature, while gradually settling down at lower temperatures to smaller regions where the lowest point of potential energy may be located. It has been proved that the sequence of the points sampled by the simulated annealing algorithm form a Boltzmann distribution and converges to a global minimizer with probability one as the temperature goes to zero (Aarts and Korst [1]).

The effective energy simulated annealing algorithm is similar to the simulated annealing algorithm except that a class of objective functions, called effective energy functions, are used, one at each temperature. The higher the temperature, the smoother the corresponding objective function. Coleman, Shalloway, and Wu [25] demonstrated experimentally that this algorithm converges faster with fewer function evaluations than the standard simulated annealing algorithm.

As in all simulated annealing type algorithms, the effective energy simulated annealing algorithm can be parallelized by sampling and evaluating all the points in parallel at every cooling stage. A general strategy is that at each cooling stage, each processor generates its own sequence of points, or in other words, random walks, compares the results with other processors, and chooses the lowest point among all processors as the starting point in the next stage. Coleman, Shalloway, and Wu [25] demonstrated scalable performance of the algorithm using this strategy on the Intel iPSC/860 with application to molecular cluster conformation problems.

Global Continuation

Global continuation algorithms, as named in Moré and Wu [65], are a class of homotopy-type algorithms applied to global optimization problems. A special integral transform is used to generate the homotopy. A set of curves tracing the solutions to the homotopy at each parameter value is then traced to locate a global solution at the end. The special transform makes the function smoother with fewer local minimizers. Also, the local minimizers are concentrated in regions with low function values, where a global minimizer is likely to be located. Therefore, by tracing the local minimizers on the smoothed functions back to the original function, there is a good chance that at least one curve will lead to a global minimizer of the original function (Wu [93], Moré and Wu [65]).

The global continuation algorithms have been studied by several research

groups including Scheraga et al [54], Shalloway [85, 84], Coleman, et al [25], Straub [88, 89], Moré and Wu [65, 64, 68, 66]), and Byrd and Schnabel [86], each having slightly different transforms. In particular, Moré and Wu [65, 67] developed a class of parallel global continuation algorithms for solving distance geometry problems with application to NMR macromolecular modeling.

The algorithms are embarrassingly parallel: multiple solution curves are traced in parallel. The best solution found by the processors is selected at the end. The computation on each processor is intensive since it involves a sequence of local minimizations. However, the load on all processors is almost the same, and little communication is required except for at the beginning and end of the computation. The algorithms have been implemented on several parallel architectures as a parallel software package called DGSOL available through the NEOS, the Network Enabled Optimization System, at Argonne National Lab. This parallel implementation has been used to solve large distance geometry problems of practical interest.

4 Direct Search Methods

Direct search (pattern search) methods are longtime favorites of users, but they have only recently become interesting to optimization researchers. One of the reasons is that direct search methods are more interesting to try to parallelize than Newton methods. The work in a Newton's method usually is dominated by the cost of the function evaluations and the linear algebra required to solve the underlying local model problem. Direct search methods require essentially no linear algebra, but they are profligate users of function evaluations. This is because there is no underlying local model to suggest where a better next iterate is likely to be found as in a Newton method. Instead, direct search methods sample the function to find the next iterate. Of course, Newton methods use function evaluations to confirm or reject a suggested next iterate, but direct search methods use function values to explore for a next iterate. This exploration phase makes it possible to invent new intrinsically parallel direct search methods rather than simply to parallelize an existing method as in Newton's method.

This approach led to the parallel direct search method PDS [34, 90], and to the novel convergence theory for more general pattern search methods

[91, 92, 57, 58, 61, 60]. Audet [4] showed by several examples that Lewis and Torczon's convergence analysis in [57] is sharp. Lewis and Torczon [59] and Audet and Dennis [5, 6] extended these methods and the convergence analysis to nonlinear problems with discrete as well as continuous variables and to those with general constraints. Hough and Meza [47] proposed using the PDS method for the trust-region subproblem and developed a parallel trust-region algorithm for nonlinear optimization. Hough, Kolda, and Torczon [46] also developed an asynchronous version of PDS, which demonstrated better parallel performance than the standard PDS in real applications.

Indeed, the sequential version of PDS called MDS or multidirectional direct search was a "sequentialization" of PDS rather than the usual way around. In order to discuss parallel aspects of the algorithms, we will give the algorithm for continuous variables and no constraints. Thus, we consider

$$\min_{x \in R^n} f(x)$$

The following formulation of Generalized Pattern Search (GPS) is from [14]. It differs from Torczon's original formulation in [92], but it is equivalent. For simplicity, we say that if there are constraints, and if x either is infeasible or if $f(x)$ cannot be evaluated successfully, then we set $f(x) = \infty$. Note that both steps have ample opportunities for parallel evaluations of the objective function and constraints. Indeed, one would certainly tailor the search step to the number of function evaluations it would be convenient to compute in parallel. There is also a nice place here for hierarchical parallelism if the evaluation of a single $f(x)$ is already a parallel program.

GPS: Let M_0 denote a mesh on R^n and suppose that $x_0 \in M_0$ has been given. (Typically, $x_0 \approx x^*$, where x^* is a preliminary baseline solution, but any choice of $x_0 \in M_0$ is allowed.) Let $X_0 \subset M_0$ contain x_0 and any $2n$ points adjacent to x_0 for which the differences between those points and x_0 form a maximal positive basis (composed of multiples of the coordinate vectors) for R^n. As the algorithm generates $x_k \in M_k$, let $X_k \subset M_k$ be defined in the same way. For $k = 0, 1, \ldots$, do:

1. **Search**: Employ some finite strategy to try to choose $x_{k+1} \in M_k$ such that $f(x_{k+1}) < f(x_k)$. If such an x_{k+1} is found, declare the **Search** successful, set $M_{k+1} = M_k$, and increment k;

2. else **Poll**:
   if x_k minimizes $f(x)$ for $x \in X_k$, then declare the **Poll** unsuccessful, set $x_{k+1} = x_k$, and refine M_k to obtain M_{k+1} by halving the mesh size (write this as $M_{k+1} = M_k/2$);
   else declare the **Poll** successful, set x_{k+1} to a point in X_k at which $f(x_{k+1}) < f(x_k)$, and set $M_{k+1} = M_k$.
   Increment k.

Step 2 provides the safeguard that guarantees convergence, as in the following result from [5] which extends [92]. The extension of GPS to GMIPS (x has some discrete components) differs from GPS in the definition of the poll set X_k, and the convergence result is a bit different, though equally satisfying:

If f is continuously differentiable, then there are infinitely many unsuccessful iterates produced by any GPS method, and some limit point of the unsuccessful iterates is a stationary point for problem (4).

This result says that one needs only monitor the unsuccessful iterates of GPS to find a stationary point, and this is without regard to how naive the search strategy is in Step 1. In practice, of course, the search strategy matters a lot to the number of function values required to find a good optimizer. We now turn to using global model functions as surrogates for $f(x)$ to try to **Search** with greater parsimony and thereby reduce the total number of objective function evaluations, or parallel objective function evaluations. Intuitively, surrogate methods use global models to predict where to find a successful next iterate in just the way that Newton methods use local models. Of course, the local models must be first order accurate for Newton methods to work - but then they work very well indeed. It is unrealistic to expect much accuracy of a global model, and that is one reason why we avoid calling them approximations. The poll step has the same opportunities for parallelism as before, but parallel function evaluations of the inexpensive surrogate can allow a sort of rough global surrogate optimization in the search strategy. One can return a number of candidates for x_{k+1}, evaluate them in parallel, and accept the best.

The Surrogate Management Framework

The description of SMF that we present here is a set of strategies for using approximations in both the **Search** and **Poll** steps of a GPS algorithm. For greater clarity, we have also identified a separate **Evaluate/Calibrate** step. In what follows, we assume that a family of approximating functions has been specified, that an initial approximation has been constructed, and that an algorithm to re-calibrate the approximation is available. (See [14], [16], [15], [83] for more details.)

SMF: Given s_0, an initial approximation of f, and $x_0 \in M_0$, let $X_0 \subset M_0$ contain x_0 and any $2n$ points adjacent to x_0 for which the differences between those points and x_0 form a maximal positive basis (composed of multiples of the coordinate vectors) for R^n. As the algorithm generates $x_k \in M_k$, let $X_k \subset M_k$ be defined in the same way. For $k = 0, 1, \ldots$, do:

1. **Search**: Use any method to choose a trial set $T_k \subset M_k$. If $T_k \neq \emptyset$ is chosen, then it is required to contain at least one point at which $f(x)$ is not known. If $T_k = \emptyset$, then go to **Poll**.

2. **Evaluate/Calibrate**: Evaluate f on elements in T_k until either it is found that x_k minimizes f on T_k or until $x_{k+1} \in T_k$ is identified for which $f(x_{k+1}) < f(x_k)$. If such an x_{k+1} is found, then declare the **Search** successful. Re-calibrate s_k with the new values of f computed at points in T_k.

3. If **Search** was successful, then set $s_{k+1} = s_k$, $M_{k+1} = M_k$, and increment k;
   else return to **Search** with the re-calibrated s_k, but without incrementing k.

4. **Poll**:
   If x_k minimizes $f(x)$ for $x \in X_k$, then declare the **Poll** unsuccessful, set $x_{k+1} = x_k$, and set $M_{k+1} = M_k/2$;
   else declare the **Poll** successful, set x_{k+1} to a point in X_k at which $f(x_{k+1}) < f(x_k)$, and set $M_{k+1} = M_k$.
   Re-calibrate s_k with the new values of f computed at points in X_k. Set $s_{k+1} = s_k$.
   Increment k.

Asynchronous Parallel Search

In general, the PDS algorithm assumes a homogeneous and tightly-coupled parallel system and it synchronizes in every iteration to compare the function values among all processors. The problem is that in practice, the machines available may be loosely-coupled and heterogenneous. Synchronization may force many of the processors idle while others are busy. The problem can be more serious when the cost for function evaluation also varies with varying evaluation points and the load will not be balanced among processors.

To address this problem, Hough, Kolda, and Torczon [46] developed an asynchronous version of PDS and obtained better performance than the PDS algorithm on standard test problems and some problems arising in practical applications.

The idea of the asynchronous algorithm is that in the PDS algorithm, each processor takes an independent direction to search for a decreasing point; it broadcasts the point when it finds one, or returns to next iteration when informed that a decreasing point has been found elsewhere. The communication among the processors are managed by some daemon processes and the cost is justified by much better balanced computation across all processors.

The following is an outline of the algorithm. For more details, readers are referred to [46].

APDS: On each processor, define x_+, x_{best}, x_{trial} to be current, best, and trial iterates, respectively, and let f_+, f_{best}, f_{trial} and $\Delta_+, \Delta_{best}, \Delta_{trial}$ be corresponding function values and step sizes. Let *tol* be a small tolerance for the step size.

1. Consider each incoming triplet $\{x_+, f_+, \Delta_+\}$ received from another processor. If $f_+ < f_{best}$, then $\{x_{best}, f_{best}, \Delta_+\} \leftarrow \{x_+, f_+, \Delta_+\},\ \Delta_{trial} \leftarrow \Delta_{best}$.

2. Compute $x_{trial} \leftarrow x_{best} + \Delta_{trial} d$ and evaluate $f_{trial} = f(x_{trial})$, where d is the local direction.

3. Set $\{x_+, f_+, \Delta_+\} \leftarrow \{x_{trial}, f_{trial}, \Delta_{trial}\}$.

4. If $f_+ < f_{best}$, then $\{x_{best}, f_{best}, \Delta_{best}\} \leftarrow \{x_+, f_+, \Delta_+\},\ \Delta_{trial} \leftarrow \Delta_{best}$, and broadcast $\{x_{best}, f_{best}, \Delta_{best}\}$. Else $\Delta_{trial} \leftarrow \frac{1}{2}\Delta_{trial}$.

5. If $\Delta_{trial} > tol$, goto Step 1. Else broadcast a local covergence message.

6. Wait until either (a) enough of processes have converged for this point or (b) a better point is received. In case (a), exit. In case (b), goto Step 1.

5 Optimization of Linked Subsystems

The formulation we will discuss in this section applies to a class of optimization problems arising in multidisciplinary design optimization (MDO). In MDO, the design variables and the system variables are correlated through coupled nonlinear subsystems each of which may involve expensive calculations such as PDE solves. The idea here is that parallelism can be exploited at the coupling level where the subsystems can be solved independently, if an appropriate MDO formulation is employed. The technique we employ is related closely to domain decomposition for PDE or multiple shooting for ODE.

A general MDO problem can be formulated as the following nonlinear optimization problem.

$$\begin{array}{ll} \textbf{min} & f(x, u(x)) \\ \textbf{st.} & g(x, u(x)) \geq 0, \end{array} \tag{11}$$

where x is a set of design variables and $u(x)$, the vector of system or ancillary variables, is defined implicitly by the blocked system of equations:

$$\begin{array}{rcl} A_1(x, u_1(x), \ldots, u_N(x)) & = & 0 \\ & \vdots & \\ A_N(x, u_1(x), \ldots, u_N(x)) & = & 0 \ . \end{array} \tag{12}$$

This system represents the linking of all the subsystems, and the act of solving it numerically for $u(x)$, given x, is known as multidisciplinary analysis (MDA). This terminology is in line with standard engineering terminology that a disciplinary *analysis* is a single disciplinary simulation run.

The most conventional approach to (11) is sometimes called the *control theory* or *closed equations* or *black box* approach which formulates the problem as:

$$\begin{array}{ll} \textbf{min} & \hat{f}(x) \\ \textbf{st.} & \hat{g}(x) \geq 0, \end{array} \tag{13}$$

where $\hat{f}(x) \equiv f(x, u(x))$ and $\hat{g}(x) \equiv g(x, u(x))$.

At each iteration of an optimization procedure applied to (13), any call to the function routines causes the design variable x to be passed to the MDA solver and the linked system (12) is solved for $u(x)$. This reduces the optimization problem to its essential decision variables x, which can be large when x is a distributed parameter, but it is often an order of magnitude lower dimensional than the dimension of u.

To get more of an idea of the expense of solving (12), think of MDA as solving a perhaps huge nonlinear system of equations, which will have to be done iteratively, whose residuals at a given iterate $u(x)_k$ can only be evaluated in blocks, and where to evaluate the ith block may require doing a single discipline analysis for the discipline represented by A_i perhaps several times in every iteration. Even more scary is to think about the problem of computing derivative approximations in order to use a Newton-type method for MDA. Even getting derivatives to use in an optimization algorithm applied to (13) is expensive. For example, if one is to use finite differences, then $\nabla \hat{f}(x)$ for any x will cost $\dim(x)$ MDA solves. One can try to find and use adjoint formulations, but that is generally not practical if the dimension of the range of g is at all large.

The MDA system (12) is naturally decomposed into disciplinary equations, which can be distributed to multiprocessors and solved in parallel. For example, A_1 is solved for u_1 on processor 1, A_2 for u_2 on processor 2, etc. However, given x, u_1 depends on x as well as other u_i, and so on for $u_2, \ldots, u_N$. Thus, this procedure is just a block nonlinear Jacobi iteration, which is problematic at best, though it easily allows parallel single disciplinary analyses. Of course, in many cases, load balancing is a problem because of the different cost of executing different single disciplinary analyses.

Probably the most often used procedure is the almost equally problematic, and less parallel, Gauss-Seidel or successive replacements procedure; that is, first assume some values for x and u_i, solve A_1 for u_1; then with the new value for u_1 along with given values for x and other u_i, solve A_2 for u_2, and so on until a new set of values for all u_i are found. The procedure then repeats until the whole system reaches equilibrium, or in other words, converges to u that satisfies all the equations. This method can only be executed sequentially, and as with the Jacobi procedure, there is no reason to believe it will converge for a given problem. Still, the method does not require a system Jacobian for (12), and sometimes intuition helps to order the single discipline solves

to obtain convergence.

At the other end of the spectrum of formulations is the *simultaneous analysis and design* or *nonlinear programming* or *open equations* formulations. This formulation can best be seen by rewriting (11) as:

$$\begin{array}{ll} \textbf{min} & f(x,u) \\ \textbf{st.} & g(x,u) \geq 0, \\ \textbf{and} & A(x,u) = 0 \end{array} \tag{14}$$

where x, u are both treated as independent variables, and the inclusion of the MDA equations as a constraint ensures that $u = u(x)$ at all feasible points. There are many reasons why this is the ideal formulation for most problems, but, it is likely to be extremely large and to need special linear algebra to handle the linear algebra needed for a sequential quadratic programming implementation. But, a major difficulty is that one must be able to open up the single discipline analysis codes and extract the residual computations for the equations solved by that code.

Recent work on MDO, for example, Cramer et al [29], Dennis and Lewis [32], and Alexandrov and Lewis [2], demonstrated that the MDA equations as well as the MDO problem can be solved in parallel if appropriate formulations and algorithms are used. We describe some of the ideas in the following.

For simplicity, consider a two-discipline MDO problem,

$$\begin{array}{ll} \textbf{min} & f(x_0; R_1(u_1); R_2(u_2)) \\ \textbf{st.} & g_0(x_0; S_1(u_1); S_2(u_2)) \geq 0 \\ & g_1(x_0; x_1; u_1) \geq 0 \\ & g_2(x_0; x_2; u_2) \geq 0, \end{array} \tag{15}$$

where, u_1 and u_2 depend on x_0, x_1, and x_2 through the MDA system,

$$A_1(x_0; x_1; u_1; T_1(u_2)) = 0 \tag{16}$$

$$A_2(x_0; x_2; u_2; T_2(u_1)) = 0. \tag{17}$$

Here the design variable x is partitioned into $x = (x_0; x_1; x_2)$ with x_1 and x_2 specific to discipline 1 and 2, respectively, and x_0 shared by both. The function g_0 is called the design constraint, g_1 and g_2 are the disciplinary design constraints, and A_1 and A_2 are the disciplinary analysis constraints.

As we have discussed before, a major difficulty is that the disciplinary analysis constraints are coupled through variables u_1 and u_2. A very important property that generally holds is that the coupling through T_1, T_2 may involve small subvectors of u_1 and u_2. This is analogous to the domain decomposition approach to PDE solutions where at most a band around the boundary values at the subdomain interfaces are exchanged between subdomain solves. Thus, without making the problem too much larger than (13), we can introduce new variables u_{12} and u_{21} to replace $T_1(u_2)$ and $T_2(u_1)$ and add new constraints to make u_{12} equal to $T_1(u_2)$ and u_{21} to $T_2(u_1)$. Again, a feasible point will satisfy (12). This gives one of the [29] IDF formulations for the MDO problem:

$$\begin{aligned}
\textbf{min} \quad & f(x_0; R_1(u_1); R_2(u_2)) && (18)\\
\textbf{st.} \quad & g_0(x_0; S_1(u_1); S_2(u_2)) \geq 0 && (19)\\
& g_1(x_0; x_1; u_1) \geq 0 && (20)\\
& g_2(x_0; x_2; u_2) \geq 0 && (21)\\
& u_{12} - T_1(u_2) = 0 && (22)\\
& u_{21} - T_2(u_1) = 0, && (23)
\end{aligned}$$

where, u_1 and u_2 depend on x_0, x_1, and x_2 through the MDA system,

$$\begin{aligned}
A_1(x_0; x_1; u_1; u_{12}) &= 0 && (24)\\
A_2(x_0; x_2; u_2; u_{21}) &= 0. && (25)
\end{aligned}$$

Note that in this formulation, u_{12} and u_{21} are considered as independent variables, therefore, given x_0, x_1, x_2, u_{12} and u_{21}, A_1 and A_2 can be solved in parallel for u_1 and u_2. The coupling between the two equations is handled by the consistency constraints (22) and (23) at the MDO level. Thus, a complete MDA is not required at each iteration of a standard SQP optimizer. A major point is that the individual discipline solver codes are used as they are.

The model we show above is for a two-discipline MDO problem, but the technique for decoupling or decomposing the MDO/MDA system into parallel, independent subsystems can be extended to problems with more than two disciplines especially when they are loosely-coupled, i.e., each equation/constraint is connected with only a few other equations/constraints, and thus only a small number of auxilliary variables will be required.

MDO or linked subsystems problems are one of the "grand challenges" of scientific computation. There is little hope of solving realistic problems without significant advances in automatic differentiation.

Finally, we remark that Braun [17] and Sobieski and Kroo [87] suggested a way of posing linked subsystem problems called *collaborative optimization*. Only in special circumstances is this problem equivalent to (11), but there are other ways to pose optimization with linked subsystems than the straightforward (11), and "CO" has the comforting feature of mimicing the way "parallel" teams of disciplinary specialists would attach such problems.

6 Variable and Constraint Distribution

Ferris and Mangasarian [37, 38] developed two classes of parallel algorithms for constrained optimization problems. Algorithms of the first class distribute the variables on multiple processors. Each processor updates its own variables in parallel while allowing the other variables to change in a restricted fashion. Once a new step is obtained, all processors communicate and combine the steps to obtain the new iterate in the whole space. The second class of algorithms distribute the constraints over the processors instead. Each processor then solves a subproblem with a subset of constraints and a modified objective function. The processors then exchange Lagrange multipliers and repeat.

Ferris and Mangasarian [37, 38] presented algorithms of these classes designed for various types of optimization problems, gave a convergence theory, and provided preliminary performance results. We refer the reader there for more details.

Variable Distribution

Consider the problem

$$\mathbf{min}_{x \in \mathcal{X}} \; f(x) \tag{26}$$

where $\mathcal{X}$ is a nonempty closed convex set in $\mathcal{R}^n$ and f a continuous and differentiable function. The variable distribution algorithm first distributes p blocks $x_1, \ldots, x_p$ of variable x, where $x_l \in \mathcal{R}^{n_l}$, $\sum_{l=1}^{p} n_l = n$, over p processors. At iteration i with an iterate $x^i \in \mathcal{R}^n$, processor l updates block x_l^i by

solving a subproblem,

$$\mathbf{min}_{x_l,\lambda_l} \quad f(x_l, x_{\bar{l}}^i + D_{\bar{l}}^i \lambda_{\bar{l}}) \tag{27}$$

$$\mathbf{s.t.} \quad (x_l, x_{\bar{l}}^i + D_{\bar{l}}^i \lambda_{\bar{l}}) \in \mathcal{X} \tag{28}$$

where $\bar{l}$ denotes the complement of l in $1, \ldots, p$, $\lambda_{\bar{l}} \in \mathcal{R}^{p-1}$. The matrix $D_{\bar{l}}^i$ is an $n_{\bar{l}}$-by-$(p-1)$ matrix. It is formed by taking arbitrary direction $d^i \in \mathcal{R}^n$, breaking it into blocks of $d_l^i \in \mathcal{R}^{n_l}$, $l = 1, \ldots, p$, consistent with the distribution of the variables, and placing these vectors along the block diagonal of $D_{\bar{l}}^i$,

$$D_{\bar{l}}^i = \mathbf{diag}(d_1^i, \ldots, d_{l-1}^i, d_{l+1}^i, \ldots, d_p^i). \tag{29}$$

Let $(y_l^i, \lambda_{\bar{l}}^i)$ be the optimal solution of problem (27), and $x^{i_l} = (y_l^i, \lambda_{\bar{l}}^i)$. Then after all processors obtain their x^{i_l}, $l = 1, \ldots, p$, the next iterate x^{i+1} for the original problem (26) can be obtained by solving the subproblem,

$$\mathbf{min}_{\mu_0,\ldots,\mu_p} \quad f(\mu_0 x^i + \sum_{k=1}^{p} \mu_k x^{i_k}) \tag{30}$$

$$\mathbf{s.t.} \quad \mu_0 x^i + \sum_{k=1}^{p} \mu_k x^{i_k} \in \mathcal{X}, \quad \sum_{k=0}^{p} \mu_k = 1, \tag{31}$$

with x^{i+1} set to $\mu_0 x^i + \sum_{k=1}^{p} \mu_k x^{i_k}$.

Note that the subproblem (27) is to solve the problem in the subspace spanned by its allocated variables. Since each involves only its own local variables, all can be solved in parallel. The subproblem (30), is again to solve the problem on a subspace, but now it is the subspace spanned by the steps from the current iterate to each of the subspace optima found at the previous level. Since these steps were computed on different processors, a synchronization step among processors is required for solving (30). Clearly, this process can be applied to generate multiple levels until a good fit is found for the given problem on the given machine.

The variable distribution method can be used for unconstrained optimization problems and problems with block separable constraints. Ferris and Mangasarian [38] showed that the algorithms for these problems converge with certain optimality conditions. They also tested the algorithms with a subset of optimization problems in **CUTE** [13] and obtained reasonable speedups on CM-5 with up to 32 processors.

Constraint Distribution

The constraint distribution method applies to quadratic programs with strictly convex objective functions. It can also be extended to general convex programs, but with relatively weaker convergence results.

In general, consider the following convex program,

$$\mathbf{min} \quad f(x) \tag{32}$$

$$\mathbf{s.t.} \quad g_1(x) \leq 0, \ldots, g_p(x) \leq 0, \tag{33}$$

where f is a strictly convex function, and g_l are convex functions from $\mathcal{R}^n$ to $\mathcal{R}^{m_l}$, $l = 1, \ldots, p$. The method distributes the block constraints to p processors. On processor l, a subproblem with only constraint block $g_l(x) \leq 0$ and a modified objective function is solved. Then the solutions and the Lagrange multipliers are shared among processors, and the whole process is repeated. Note that the modified objective function on one processor is composed of the original function plus some augmented Lagrangian terms formed by the constraints assigned to other processors.

For illustrative purposes, consider a quadratic program with 3 blocks of inequality constraints,

$$\mathbf{min} \quad c^T x + \frac{1}{2} x^T Q x \tag{34}$$

$$\mathbf{s.t.} \quad A_l x \leq a_l, l = 1, 2, 3, \tag{35}$$

where $c \in \mathcal{R}^n$, $Q \in \mathcal{R}^{m_l \times n}$, $a_l \in \mathcal{R}^{m_l}$, and Q is symmetric and positive definite. A constraint distribution algorithm for this problem would first distribute the constraints to 3 processors, with constraint l to processor l. Then at iteration i, a subproblem can be solved on each processor in parallel, that is, on processor l:

$$\mathbf{min}_{x_l} \quad c^T x_l + \frac{1}{2} x_l^T Q x_l + \frac{1}{2\gamma} \left[\sum_{j=1, j \neq l}^{3} \| (\gamma(A_j x_l - a_j) + p_{jl}^i \|^2 \right] + x_l^T r_l^i \tag{36}$$

$$\mathbf{s.t.} \quad A_l x_l \leq a_l, \tag{37}$$

where γ is a positive number and p_{jl}^i and r_l^i, $j, l = 1, 2, 3$ are parameters to be determined. The p_{jl}^i play the roles of the multipliers and converge to the optimal multipliers eventually, while r_l^i replaces estimates of the multipliers

by their most recent values obtained from each of the other subproblems. Note that the objective functions for the subproblems in (36) are quadratic augmented Lagrangian functions perturbed by the linear terms $x_l^T r_l^i$. Thus in each subproblem, some constraints are treated explicitly as constraints while the remaining ones are terms in the augmented Lagrangian objective function.

Given the values for all the parameters, each subproblem in (36) can be solved in parallel. However, the parameters are updated using their most recent values from other processors. Therefore, communication is required at certain points. Ferris and Mangasarian [37] showed the convergence results for the constraint distribution algorithm for strictly convex quadratic programs and extended them to general convex programs. Five small quadratic programming problems were tested with the algorithm on the Sequent Symmetry S-81. Encouraging results were obtained.

7 Future Research and Development

Despite much effort and some solid developments, the use of parallelism in general optimization has not been as fruitful as its use in other areas of numerical computation, like numerical linear algebra. There are special successful applications, and some software packages available, but not much performance analysis or benchmarking work. One of possible reasons is that practical optimization problems often have many ancillary variables, but only a few decision variables. The great opportunities for finding parallelism might then lie in parallelizing the computation of the ancillary variables by using domain decomposition to solve a PDE for example. If tools for hierarchical parallelism become more generally available, this situation may change.

Another reason for our slow progress may be the tradition favoring inherently sequential Newton-like methods where one carefully builds local models and extracts all the information one can before evaluating a trial step. After all, in locally modeling methods, there is generally a clearly preferred trial step, and if that is not successful, then the fall back strategies use information gotten from the failure. Methods such as parallel line searches or sector searches have not been great successes, probably because they kluge one paradigm onto another rather than find a single consistent algorithmic paradigm.

Global optimization methods have attracted more and more attention as usable parallel and high performance computing resources became available. Indeed, there are many cases where scientific problems of an interesting size have been solved by these methods. Still, the general global optimization problem is intractable, even for infinitely smooth functions. Empirically, the computing time needed to get a reasonable solution using a general global optimization algorithm seems to grow exponentially with the problem size while the speed-up can at best be counted upon to be linear with the number of processors. Thus, the future of global optimization is in the development of efficient and reliable algorithms for specific classes of problems. Without such algorithms, problem sizes will remain limited despite gains from parallel computation.

Parallel direct search methods are another successful development in the quest for parallel optimization algorithms. The theory is developing rapidly, and they are easy to use either as sequential or parallel algorithms. There are many successful applications, but the methods are slower than Newton methods, and as with all derivative-free methods, it is difficult to know when to terminate. Thus the algorithm is more suitable for small problems with uncertain accuracy in the function. Constraints are problematic for these algorithms as well. If one has no derivatives, then Lagrange multipliers, a mainstay of constrained optimization are not available. However, algorithms for constraints and large-scale applications are interesting research directions.

We call the problem class MDO in Section 5, but in fact, it is much more general than design. As simulation is used to aid decision makers in more and more areas, such as crisis management, instances of these problems will arise. Picture a library of standard simulation codes, like fluid flow, thermal conductivity, structures, etc. One might want to make decisions concerning systems governed by coupling various choices from among these systems. It may not be practical to have in the library special simulations for all these combinations. Here, we provide a completely equivalent formulation for the original problem for which this would not be necessary because the separate "closed" subsystems could be linked numerically for each required x without recoding to obtain a solver that works for any x. However, since the method is relatively new, and the computational demands for MDO are so high, computational experiments are limited. Indeed, this field is in its infancy.

The variable and constraint distribution algorithms are interesting. Different from many other algorithms, which are obtained by parallelizing their

serial counterparts, these algorithms are developed with parallel computation in mind. Therefore, standard optimization components, like computation of a search direction, are designed as parallel procedures. Convergence results have also been established for the algorithms. They have not been extensively tested or applied in practice. Further research on these algorithms and their applications can be promising and fruitful. For example, parallel variable distribution and parallel direct searches seem an interesting pairing for extending the latter to larger problems. Partial separability seems also to be clearly related to parallel variable distribution.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, 1989.

[2] N. M. Alexandrov and R. M. Lewis. Comparative properties of collaborative optimization to mdo. Technical Report 99–14, ICASE, NASA Langley Research Center, Hampton, VA, July 1999.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM Publications, 1995.

[4] C. Audet. Convergence results for pattern search algorithms are tight. Technical Report CRPC-TR98779, CRPC, Rice University, 1998.

[5] C. Audet and J. E. Dennis Jr. On the convergence of mixed integer pattern search algorithms. Technical Report CRPC-TR98785, CRPC, Rice University, 1999.

[6] C. Audet and J. E. Dennis Jr. A pattern search filter method for nonlinear programming without derivatives. Technical Report TR00-09, Department of Computational and Applied Mathematics, Rice University, 2000.

[7] B. M. Averick and J. J. Moré. Evaluation of large-scale optimization problems on vector and parallel architectures. *SIAM J. Optimization*, 4:708–721, 1994.

[8] S. Balay, W. Gropp, L. McInnes, and B. Smith. Efficient management of parallelism in object-oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*. Birkhauser Press, 1997.

[9] R. E. Bixby and A. Martin. Parallelizing the dual simplex method. Technical Report CRPC-TR95706, CRPC, Rice University, 1997.

[10] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM Publishers, 1997.

[11] L. M. Blumenthal. *Theory and Applications of Distance Geometry*. Oxford University Press, 1953.

[12] S. H. Bokhari and D. J. Mavriplis. The Tera multithreaded architecture and unstructured meshes. Technical Report ICASE Interim Report No. 33, ICASE, NASA Langeley Research Center, Hampton, VA, 1998.

[13] I. Bongartz, A. R. Conn, N. Gould, and P. Toint. Constrained and unconstraned optimization testing environment. Technical Report 93/10, Départment de Mathématique, Facultés Universitaires De Namur, 1993.

[14] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B.Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 1999.

[15] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. W. Moore, and D. B. Serafini. Managing surrogate objectives to optimize a helicopter rotor design - further experiments. In *Proceedings of the Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.

[16] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, and V. Torczon. Optimization using surrogate objectives on a helicopter test example. In *Optimal Design*, 1998.

[17] R. D. Braun. *An architecture for large-scale distributed design.* PhD thesis, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 1996.

[18] R. H. Byrd, T. Derby, E. Eskow, K. P. B. Oldenkamp, and R. B. Schnabel. A new stochastic/perturbation method for large-scale gobal optimization and its application to water cluster problems. In W. Hager, D. Hearn, and P. Pardalos, editors, *Large-Scale Optimization: State of the Art.* Kluwer Academic Publishers, 1994.

[19] R. H. Byrd, E. Eskow, and R. B. Schnabel. A new large-scale global optimization method and its application to Lennard-Jones problems. Technical report, Department of Computer Science, University of Colorado, Boulder, CO, 1995.

[20] R. H. Byrd, E. Eskow, and R. B. Schnabel. A large-scale stochastic-perturbation global optimization method molecular cluster problems. Technical report, Department of Computer Science, University of Colorado, Boulder, CO, 1999.

[21] R. H. Byrd, E. Eskow, R. B. Schnabel, and S. L. Smith. Parallel global optimization: Numerical methods, dynamic scheduling methods, and application to molecular configuration. In B. Ford and A. Fincham, editors, *Parallel Computation*, pages 187–207. Oxford University Press, 1993.

[22] R. H. Byrd, E. Eskow, A. van der Hoek, R. B. Schnabel, C. Shao, and Z. Zou. Global optimization methods for protein folding problems. In P. M. Pardalos, D. Shalloway, and G. Xue, editors, *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding.* American Mathematical Society, 1996.

[23] R. H. Byrd, R. B. Schnabel, and M. H. Shultz. Parallel quasi-newton methods for unconstrained optimization. Technical report, Department of Computer Science, University of Colorado, Boulder, CO, 1990.

[24] T. F. Coleman, J. Czyzyk, C. Sun, M. Wagner, and S. J. Wright. ppcx: Parallel software for linear programming. In *Proceedings of the Eighth*

SIAM Conference on Parallel Processing in Scientific Computing. SIAM Publications, 1997.

[25] T. F. Coleman, D. Shalloway, and Z. Wu. Isotropic effective energy simulated annealing searches for low energy molecular cluster states. *Comp. Optim. Applications*, 2:145–170, 1993.

[26] T. F. Coleman, D. Shalloway, and Z. Wu. A parallel build-up algorithm for global energy minimizations of molecular clusters using effective energy simulated annealing. *J. Global Optim.*, 4:171–185, 1994.

[27] T. F. Coleman and C. Sun. Parallel orthogonal factorizations of large sparse matrices on distributed-memory multiprocessors. In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM Publications, 1993.

[28] T. F. Coleman and Z. Wu. Parallel continuation-based global optimization for molecular conformation and protein folding. *J. Global Opt*, 8:49–65, 1996.

[29] E. J. Cramer, J. E. Dennis, Jr, P. D. Frank, R. M. Lewis, and G. R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal of Optimization*, 4(4):754–776, 1994.

[30] G. M. Crippen and T. F. Havel. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, 1988.

[31] S. Crivelli, R. H. Byrd, E. Eskow, R. B. Schnabel, R. Yu, T. Phillips, and T. Head-Gordon. A global optimization strategy for predicting protein tertiary structure: α-helical proteins. Technical report, Department of Computer Science, University of Colorado, Boulder, CO, 1998.

[32] J. E. Dennis and M. Lewis. Problem formulation and other optimization issues in multidisciplinary optimization. Technical Report CRPC-TR92277, CRPC, Rice University, 1992.

[33] J. E. Dennis, Jr. and R. M. Lewis. Problem formulation and other optimization issues in multidisciplinary optimization. Technical Report CRPC-TR94469, CRPC, Rice University, Houston, TX, 1994.

[34] J. E. Dennis, Jr. and V. Torczon. Direct search methods on parallel machines. *SIAM J. Optimization*, 1(4):448–474, November 1991.

[35] K. A. Dill, A. T. Phillips, and J. B. Rosen. CGU: An algorthm for molecular structure prediction. In L. T. Biegler, T. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large-Scale Optimization and Applications Part III: Molecular Structure and Optimization*. Springer, 1997.

[36] J. Feo, S. Kahan, and Z. Wu. Crash analysis on the Tera MTA. *IEEE Computational Science and Engineering*, 5:53–59, 1998.

[37] M. C. Ferris and O. L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, 1:487–500, 1991.

[38] M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.

[39] I. L. Garzón and J. Jellinek. Melting of gold microclusters. *Z. Phys. D - Atoms, Molecules and Clusters*, 20:235–238, 1991.

[40] H. Haberland, editor. *Clusters of Atoms and Molecules, Springer Series in Chemical Physics*, volume 52. Springer-Verlag, 1994.

[41] B. A. Hendrickson. *The molecule problem: Determining conformation from pairwise distances.* PhD thesis, Cornell University, Ithaca, New York, 1991.

[42] Bruce A. Hendrickson. The molecule problem: Exploiting structure in global optimization. *SIAM J. Optimization*, 5:835–857, 1995.

[43] M. R. Hoare. Structure and dynamics of simple microclusters. *Advances in Chemical Physics*, 40:49–135, 1979.

[44] M. R. Hoare and J. McInnes. Statistical mechanics and morphology of very small atomic clusters. *Faraday Discuss. Chem. Soc.*, 61:12–24, 1976.

[45] M. R. Hoare and P. Pal. Statistics and stability of small assemblies of atoms. *J. Cryst. Growth*, 17:77–96, 1972.

[46] P. D. Hough, T. G. Kolda, and V. J. Torczon. Asynchronous parallel pattern search for nonlinear optimization. Technical Report Sand2000-8213, Sandia National Laboratories, 2000.

[47] P. D. Hough and J. C. Meza. A class of trust-region methods for parallel optimization. Technical Report Sand98-8245, Sandia National Laboratories, 1998.

[48] T. D. Plantenga J. C. Meza and R. S. Judson. Novel applications of optimization to molecular design. In L. T. Biegler, T. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large-Scale Optimization and Applications Part III: Molecular Structure and Optimization*. Springer, 1997.

[49] J. Jellinek. Theoretical dynamical studies of metal clusters and cluster-ligand systems. In N. Russo, editor, *Metal-Ligand Interactions: Structure and Reactivity*. Kluwer Academic Publishers, 1995 (in press).

[50] M. T. Jones and P. E. Plassmann. An efficient parallel iterative solver for large sparse linear systems. In A. George, J. Gilbert, and J. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 229–245. Springer-Verlag, 1993.

[51] M. T. Jones and P. E. Plassmann. Scalable iterative solution of sparse linear systems. *Parallel Computing*, pages 753–773, 1994.

[52] M. T. Jones and P. E. Plassmann. Algorithm 740: Fortran subroutines to compute improved incomplete cholesky factorizations. *ACM Trans. on Mathematical Software*, pages 18–19, 1995.

[53] M. T. Jones and P. E. Plassmann. An improved incomplete cholesky factorization. *ACM Trans. on Mathematical Software*, pages 5–17, 1995.

[54] J. Kostrowicki, L. Piela, B. J. Cherayil, and H. A. Scheraga. Performance of the diffusion equation method in searches for optimum structures of clusters of Lennard-Jones atoms. *J. Phys. Chem.*, 95:4113–4119, 1991.

[55] J. Kostrowicki and H. A. Scheraga. Application of the diffusion equation method for global optimization to oligopeptides. *J. Phys. Chem.*, 96:7442–7449, 1992.

[56] H. W. Kroto, J. R. Heath, S. C. O'Brien, F. Curl, and R. E. Smalley. C60: Buckminsterfullerene. *Nature*, 318, 1985.

[57] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. Technical Report 96–20, ICASE, NASA Langley Research Center, Hampton, VA, 1996.

[58] R. M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, ICASE, NASA Langley Research Center, Hampton, VA, 1996.

[59] R. M. Lewis and V. Torczon. A global convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. Technical Report TR 98-31, ICASE NASA Langley Research Center, 1998.

[60] R. M. Lewis and V. J. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. Technical Report 98–31, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1998.

[61] R. M. Lewis and V. J. Torczon. Pattern search methods for linearly constrained minimization. Technical Report 98–3, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1998.

[62] Z. Li and H. A. Scheraga. Monte Carlo approach to the multiple-minima problem in protein folding. *Proc. Natl. Acad. Sci.*, 84:15–29, 1987.

[63] J. J. Moré, B. Walenz, and Z. Wu. Configuration of large, confined ionic systems by potential energy minimization. Working paper, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1996.

[64] J. J. Moré and Z. Wu. ε-optimal solutions to distance geometry problems via global continuation. In P. M. Pardalos, D. Shalloway, and G. Xue, editors, *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding*, pages 151–168. American Mathemtical Society, 1995.

[65] J. J. Moré and Z. Wu. Global continuation for distance geometry problems. Preprint MCS-P505-0395, Argonne National Laboratory, Argonne, IL, 1995.

[66] J. J. Moré and Z. Wu. Issues in large-scale global molecular optimization. Preprint MCS-P539-1095, Argonne National Laboratory, Argonne, IL, 1995.

[67] J. J. Moré and Z. Wu. Distance geometry optimization for protein structures. Preprint MCS-P628-1296, Argonne National Laboratory, Argonne, IL, 1996.

[68] J. J. Moré and Z. Wu. Smoothing techniques for macromolecular global optimization. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*, pages 297–312. Plenum Press, 1996.

[69] S. G. Nash and A. Sofer. Block truncated-newton methods for parallel optimization. *Mathematical Programming*, 45:529–546, 1989.

[70] S. G. Nash and A. Sofer. A general purpose parallel algorithm for unconstrained optimization. *SIAM J. Optimization*, 1:530–547, 1991.

[71] J. A. Northby. Structure and binding of Lennard-Jones clusters: $13 \leq n \leq 147$. *Journal of Chemical Physics*, 87:6166–6177, 1987.

[72] Matei Orešič and D. Shalloway. Hierarchical characterization of energy landscapes using Gaussian packet states. *J. Chem. Phys.*, 101:9844–9857, 1994.

[73] L. Piela, J. Kostrowicki, and H. A. Scheraga. The multiple-minima problem in the conformational analysis of molecules: Deformation of the protein energy hypersurface by the diffusion equation method. *J. Phys. Chem.*, 93:3339–3346, 1989.

[74] E. O. Purisima and H. A. Scheraga. An approach to the multiple-minima problem in protein folding by relaxing dimensionality. *J. Mol. Biol.*, 196:697–709, 1987.

[75] F. Rendl and H. Wolkowicz. A semidefinite framework to trust-region subproblem with applications to large-scale minimization. Technical

Report CORR 94-32, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada, 1994.

[76] P. J. Reynolds, editor. *On Clusters and Clustering*. North-Holland, 1993.

[77] A. H. G. Rinnooy Kan and G. T. Timmer. Global optimization. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, pages 631–662. North-Holland, 1989.

[78] S. A Santos and D. C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. Technical Report TR95-20, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1995.

[79] J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. Technical report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1979.

[80] J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proc. 17th Allerton Conference in Communications, Controli and Computing*, pages 480–489, 1979.

[81] H. A. Scheraga. Predicting three-dimensional structures of oligopeptides. In Kenny B. Lipkowitz and Donald B. Boyd, editors, *Reviews in Computational Chemistry*, volume 3, pages 73–142. VCH Publishers, 1992.

[82] J. Schneider and T. H. Wise. Airline crew scheduling: supercomputers and algorithms. In Greg Astfalk, editor, *Applications on Advanced Architecture Computers*. SIAM Publications, 1996.

[83] D. B. Serafini. A framework for managing models in optimization for computationally expensive functions. Ph.D. Thesis 90–7, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005–1892, 1997.

[84] D. Shalloway. Application of the renormalization group to deterministic global minimization of molecular conformation energy functions. *J. Global Optim.*, 2:281–311, 1992.

[85] D. Shalloway. Packet annealing: A deterministic method for global minimization, application to molecular conformation. In C. Floudas and P. Pardalos, editors, *Recent Advances in Global Optimization*, pages 433–477. Princeton University Press, 1992.

[86] C. Shao, R. H. Byrd, E. Eskow, and R. B. Schnabel. Global optimization for molecular clusters using a new smoothing approach. In L. T. Biegler, T. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large-Scale Optimization and Applications Part III: Molecular Structure and Optimization*. Springer, 1997.

[87] I. Sobieski and I. Kroo. Aircraft design using collaborative optimization. AIAA Paper 96-0715, The 34th AIAA Aerospace Sciences Meeting, Reno, NV, 1996.

[88] J. E. Straub. Optimization techniques with applications to proteins. Preprint, Department of Chemistry, Boston University, Boston, MA, 1994.

[89] J. E. Straub, J. Ma, and P. Amara. Simulated anealing using coarse-grained clasical dynamics: Fokker-Planck and Smoluchowski dynamics in the Gaussian density approximation. *J. Chem. Phys.*, 103:1574–1581, 1995.

[90] V. Torczon. Multi-directional search: a direct search algorithm for parallel machines. Ph.D. Thesis 90–7, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1990.

[91] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM J. Optimization*, 1(1):123–145, February 1991.

[92] V. Torczon. On the convergence of pattern search methods. *SIAM J. Optimization*, 7(1):1–25, February 1997.

[93] Z. Wu. The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM J. Opt*, 6:748–768, 1996.

[94] G. L. Xue. Improvement on the Northby algorithm for molecular conformation: Better solutions. *J. Global. Optim.*, 4:425–440, 1994.

[95] G. L. Xue, R. S. Maier, and J. B. Rosen. Minimizing the Lennard-Jones potential function on a massively parallel computer. Preprint 91-115, AHPCRC, University of Minnesota, Minneapolis, MN, 1991.

[96] L. Zaslavsky, S. Kahan, B. Elton, K. Macshoff, and L. Stern. A scalable approach for solving irregular sparse linear systems on the Tera MTA multithreaded parallel shared-memory computer. In *The Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM Publications, 1999.

[97] Z. Zou, R. H. Byrd, and R. B. Schnabel. A stochastic/perturbation global optimization algorithm for distance geometry problems. Technical report, Department of Computer Science, University of Colorado, Boulder, CO, 1996.