ASYNCHRONOUS GLOBAL OPTIMIZATION TECHNIQUES FOR MEDIUM AND LARGE INVERSION PROBLEMS*

V. Pereyra, M. Koshy, Weidlinger Associates, Los Altos, California, and
J. Meza, Sandia National Laboratory, Livermore, California

SUMMARY

We discuss global optimization procedures adequate for seismic inversion problems. We explain how to save function evaluations (which may involve large scale ray tracing or other expensive operations) by creating a data base of information on what parts of parameter space have already been inspected. It is also shown how a correct parallel implementation using PVM speeds up the process almost linearly with respect to the number of processors, provided that the function evaluations are expensive enough to offset the communication overhead.

INTRODUCTION

Geophysical seismic inversion attempts to find the parameters in a model of the earth that best fit a set of observable seismic data. The purpose of these models is to provide images of the earth interior for applications in oil and mineral prospecting, reservoir delineation and management of enhanced recovery processes, among others.

The seismic inversion problem has several characteristics which makes it an interesting, but difficult, problem to solve. As noted by Tarantola [15], inversion problems, by their nature, often overdetermine some model parameters while leaving others underdetermined. This, along with the noisy and band-limited nature of seismic data (*c.f.* [14]), yields problems which may have many possible solutions. That is, there may be many points in parameter space which satisfy the necessary conditions to be *local* solutions to a seismic inversion problem (or that look as such to a numerical optimization algorithm), and have similar misfit values. We call such points, "numerical stationary points".

The challenge we face is that of providing tools that will facilitate a more comprehensive exploration of parameter space in order to identify the best possible solution of geophysical interest that a data set can support. Current single applications of local optimization techniques have shown to be inadequate for this purpose, not only in real inversion problems arising from geophysical models of actual earth regions, but even in relatively simple synthetic ones. These techniques tend to require a fairly precise a priori knowledge of the target model (i.e., good initial guess) in order to converge at all to an useful solution.

This has hampered the routine use of inversion techniques in the industry. It is therefore important to improve the robustness, efficiency and trustworthiness of these techniques if we expect them to be useful in a production environment. In this paper we consider techniques of global optimization coupled with distributed computing as a first step to attack medium to large scale seismic inversion problems. In subsequent work we will use these techniques, together with decomposition or blocking techniques [10], to solve nonlinear travel time inversion problems in three dimensions based on our forward and inverse modeling approach [9, 11, 6].

GLOBAL OPTIMIZATION OVERVIEW

The global optimization problem can be stated as

$$\min_{x \in D \subset \Re^n} f(x) \tag{1}$$

where x represents a set of n unknown parameters, D is defined by a set of bounds for each element of x, and $f : \Re^n \to \Re$ is a twice continuously differentiable nonlinear function of the parameters. We assume that the global minimizer is in the interior of D and that there may be many points in D which are local

*This research was partially supported by the National Science Foundation under SBIR Grant III-9300992

minimizers of f. Frequently, when problem (1) is encountered in the literature, a local minimizer is deemed satisfactory. In this research, we are interested in finding either the global minimizer, or a local solution that we feel is sufficiently close to the global minimizer, or the best local minimizer available after a given amount of computational time or number of function evaluations.

Many methods have been proposed for the global optimization problem including the tunneling method [7], [8], differential equation techniques [1], [2], interval methods [12], and stochastic methods [5], [3], and [13]. These techniques are computationally intensive, often requiring thousands of function evaluations even on small dimensional test problems. The following characteristics are required for any method to be viable on the seismic inversion problem:

- the number of function evaluations must be kept to a minimum,
- as many of the function evaluations as possible should be performed in parallel.

Algorithms designed with these two properties as motivation will allow us to attempt to solve the seismic inversion problem for a "global" solution as defined above.

Most global optimization algorithms consist of two phases: a *minimization* phase (usually local), and an *escape from a local minimizer* phase. The minimization phase often employs a state–of–the–art local optimization technique, but may be as simple as discarding the elements of a sample which correspond to the highest function values. The escape from minimizer phase can be as simple as a restart from a random point in the domain or as complex as that for the tunneling algorithm.

This two-phase framework is natural: the minimization phase *efficiently* locates a point or region in the feasible domain, and the escape phase tries to improve upon the minimization results by initiating a search for another local minimizer, preferably with a lower function value. The two-phases should be executed sequentially, but multiple copies can be executed in parallel and it may be possible to execute the internal steps of each phase in parallel.

The stochastic methods of [5], [3], and [13] form the basis for the techniques that we have explored. We have considered two algorithms in this research. The first is an algorithm that is very similar to those proposed and tested by [3], and [13]. The second extends the first by adding a feature, referred to as a data base of reversed trust region constraints, which is intended to reduce the total number of function evaluations.

REVERSED TRUST REGION CONSTRAINTS

Most current approaches for finding a local solution to optimization and/or nonlinear systems of equations problems are based on the following general ideas for determining a step from the current iterate, x_c:

line search: choose a direction and then calculate a step length in that direction which satisfies the conditions for ultimate convergence.

trust region: bound the step length, then find a direction and length which satisfies the conditions for ultimate convergence.

Thus, at each iteration of a trust region algorithm there is a subproblem of the form

$$\begin{array}{ll} \min\limits_{x \in \Re^n} & q(x) \\ \text{subject to} & ||x - x_c|| \leq \Delta \end{array}$$

where q is a quadratic model of f based on information at x_c, and $\Delta > 0$ is the trust region radius, whithin which it is estimated that the quadratic model approximates well the actual nonlinear function being minimized.

Assuming that a trust region algorithm generates a sequence of iterates that converges to a local minimizer, then executing this algorithm will yield a set of n-dimensional spheres within which we can obtain an approximate minimizer for a quadratic model based on information at the center. Since the center of the $i + 1^{st}$ sphere is within Δ_i of the i^{th} sphere's center, the spheres are connected and form a wormlike trajectory from the initial guess to a local minimizer. Also, the spheres corresponding to the last several steps of a converging algorithm will overlap significantly since, during convergence, the steps become small and the trust region radius is not updated.

Observe that this behavior will also occur in false convergence situations, in which the algorithm slows down to a crawl although it might not be close to a mathematical stationary point. Thus, we are including "numerical or algorithmic stationary points" in this discussion.

Figure 1 displays the first four trust regions for a function having the displayed contours in a 2-dimensional region. Here, thicker lines represent higher function values and the trust regions are displayed with shaded circles. If the trust region information generated during the execution of a trust region algorithm is recorded, then this information can be used to avoid function evaluations for any additional points that fall in these trust regions in subsequent phases of the global optimization algorithm. The assumption here being that an iteration that starts in the "worm" associated with a local minimum will converge to the same local minimum (or numerical stationary point). This process will allow more effective use of function evaluations during a global optimization algorithm.

The idea of creating a database of constraints based on trust region information to reduce the number of function evaluations can be employed in both the escape and minimization phases of any global optimization method. In the escape phase of a sampling restart technique, a step is added which guaranties that sample points do not fall in any previously computed trust region. That is, sample points must satisfy the condition:

- if $x \in S$, then for each x_c, $\Delta_c > 0$ in the trust region constraint database we have $||x - x_c|| > \Delta_c$.

This is the trust region constraint described earlier with the inequality reversed, and thus we call it a *reversed trust region constraint.*

For the minimization phase, if x_k^p is the accepted iterate for the k^{th} step of the p^{th} minimization process, then we will continue the p^{th} process only if

$$||x_k^p - x_c^q|| > \Delta_q + \Delta_p$$

where $q \neq p$ is any reversed trust region constraint from any process other than the p^{th} process. Reversed trust region constraints from process p are not included to avoid incorrect preemptive termination of the p^{th} minimization process. The trust region diameter Δ_p associated with the current iterate x^p is added, since what we are trying to prevent is the overlap of trust worms.

In a parallel asynchronous implementation, all the concurrent local optimizations will contribute to the common database of reversed trust region constraints. These processes may also interrogate the database in order to avoid repeated calculations and therefore they will be effectively cooperating asynchronously. When a process is truncated because one of its iterate's trust region impinges on an existing trust region, its trust region "worm" will be added to the other process' worm, forming a dendritic structure as shown in Figure 2. The object of this is, of course, to try to cover as much volume of the parameter space as possible with a minimum of function evaluations.

Figure 2 displays the level sets for the two-dimensional function

$$f(x) = \frac{cos(y_1)}{y_1} \times \frac{cos(y_2)}{y_2} + 0.1 * (2.0 - x_1)^2 + 0.01 * (2.0 - x_2)^2$$

where, $y_1 = \pi(x_1 + 0.001)$ and $y_2 = \pi(x_2 + x_1)$, for $x \in \Re^2$ in the domain $x_1 \in [0.7,\ 3.5]$, $x_2 \in [0.0,\ 4.5]$. Higher function values are represented with thicker lines. Also displayed are the trust regions from the SUMSL package [4] for several initial points. The trust regions resulting from SUMSL executed from a particular initial point are represented with identically shaded circles and the trajectories associated with the SUMSL iterates are displayed as vectors from one iterate to the next. Each minimization process was halted if it generated an iterate whose trust region impinged on the trust region worm from any other process. The dendritic structure of the reversed trust region constraints discussed above is readily apparent.

The primary motivation for the various steps of the algorithms is to reduce the total number of function evaluations, to perform as many of the function evaluations as possible in parallel, and to retain the convergence analysis. Our baseline global optimization algorithm (i.e., not employing the database of constraints) is described below. Each iteration has a sampling phase and a minimization phase. The sampling phase of iteration $i + 1$ can be executed concurrently with the minimization phase of iteration i. That is, there is no synchronization necessary at the end of each iteration.

Algorithm 1:

Until *convergence* do:

1. Generate a sample in the domain: $S = \{x_i \mid 1 \leq i \leq \eta\}$
2. Define a cutoff distance: $\delta > 0$
3. Prune S such that if $x_i,\ x_j \in S$, and $i \neq j$, then $||x_i - x_j|| > \delta$.
4. Evaluate $f(x)$ for all $x \in S$.
5. Prune S such that if $x \in S$, then $f(x) < f_c$, where $f_c = f_{min} + \max\{\rho, |f_{min}|\}$ with f_{min} the minimum of all computed function values, and $\rho > 0$.

6. Use each $x \in S$ as a starting point for a (local) minimization algorithm.

In our implementation of this algorithm, we have generated the sample randomly. We are also considering a systematic sampling of the domain space and the theoretical consequences of such an approach.

There are some differences between algorithm 1 and algorithm 2.1 of [13]. In step 3 of algorithm 1 we reduce the sample by minimizing the number of elements in it, whereas, in algorithm 2.1 of [13], the function values are calculated for each x and the point with the minimum value is retained in the sample. Also, Smith et. al. subdivide the domain to concentrate their efforts on productive regions and use an oversampling technique to handle boundary conditions.

The second algorithm is derived from the first with the addition of a database of reversed trust region constraints. This requires that we modify steps 2, 3, and 6 when such a database of constraints is added to the algorithm. In step 6, the minimization method is modified to halt if x_+ violates any reversed trust region constraint. Here, x_+ is an *accepted* iterate in the algorithm. Note that the trust region becomes inoperative in the convergent stages of trust region minimization algorithms, i.e., steps are generated inside the trust region and the trust region radius is not updated. Consequently, only reversed trust region constraints not associated with the current minimization process are considered. This allows the trust region algorithm to converge to a solution.

The cutoff distance in step 2 can be defined based on the δ's used in the database, e.g., we can use an average of the constraint radii. Step 3 is modified to also eliminate sample points which violate any database constraint. These changes result in the following algorithm:

Algorithm 2:

Until *convergence* do:

1. Generate a sample in the domain: $S = \{x_i \mid 1 \leq i \leq \eta\}$
2. Define a cutoff distance: $\delta > 0$
3. Prune S such that
   - $x \in S \quad \Rightarrow x$ does not violate any database constraint, and
   - if $x_i,\ x_j \in S$, and $i \neq j$, then $||x_i - x_j|| > \delta$.
4. Evaluate $f(x)$ for all $x \in S$.
5. Prune S such that if $x \in S$, then $f(x) < f_c$, where $f_c = f_{min} + \max\{\rho, |f_{min}|\}$ with f_{min} the minimum of all computed function values, and $\rho > 0$.
6. Use each $x \in S$ as a starting point for a (local) minimization algorithm modified to halt if a database constraint is violated.
7. At each successful iterate of local minimization process p, verify if the iterate violates any reverse trust region constraint corresponding to a different process; if so, interrupt p:

$$if \quad ||x_k^p - x_c^q|| \leq \delta_q + \delta_p \quad stop$$

   If not, then record $x_k^p,\ and\ \delta_p$ in the data base and continue.

Comparing algorithms 1 and 2 will yield a measure of how effective the reversed trust region constraint database is on a set of test problems. That is, we will have information on the number of function evaluations which are avoided, the size of the database, and the access time for the database.

The details of the implementations of each of these algorithms are discussed below together with the parallelization techniques employed.

IMPLEMENTATION DETAILS

We have implemented algorithms 1 and 2 described above in order to perform feasibility and performance studies. We have defined a data base system containing iterates and trust region balls, that can be used to interrupt any searches or optimizations that lead to a previously explored region of parameter space. We have produced a fairly successful algorithm for problems with a moderate number of isolated minima (local or global).

The data base has been implemented with a distributed computing model in mind, where several processors are operating in parallel, contributing information to the data base and accessing asynchronously the information provided by other processes. In this way we have designed a cooperative approach to the global optimization problem based on a network computing paradigm.

In this implementation we have used PVM (Parallel Virtual Machine), a language for distributed computing which has all the functionality necessary to experiment with the parallel approach in a network of workstations, without compromising portability to other systems.

We have created an extensible library of global optimization test problems. Currently, the library contains 18 test problems: the 16 problems described in [7], the problem defined in [13], and the problem displayed in Figure 2 of this paper and described in the associated text. Explicit function and derivatives are available for each test problem.

Testing the code on this set of challenging problems from the global optimization literature, we have also seen that as the number of minima (local or global) increases, the effectiveness of the technique diminishes, especially if these minima are evenly distibuted in parameter space.

When solving seismic inversion problems by conventional local optimization techniques we frequently encounter full affine subspaces of "numerical stationary points" resulting from ill-conditioning. By identifying and separating low dimensional subsets of parameters, from the best determined to the less determined ones, we expect to be in a situation where few such numerical stationary points appear and the global optimization approach becomes feasible.

This implementation treats the local optimization procedure as a nearly black box. For this purpose we obtained the SUMSL code (Secant Unconstrained Minimization SoLver) due to David Gay from *netlib*. We then modified SUMSL by adding a call to subroutine `GOCHCK` after a successful iterate is found in the minimization process, and continuing or halting SUMSL as appropriate. `GOCHCK` is designed to be called from the local optimization procedure to check if a point violates any reversed trust region constraint and/or to add a database constraint for this point with an associated trust region radius. That is, the appropriate database routines for a specified operation are invoked from `GOCHCK`.

Although SUMSL is a general unconstrained minimization code and we are interested in seismic inversion by nonlinear least squares, the techniques that we use in the latter problems are similar enough, so that a reversed trust region database library can be applied in the same way to trust region methods for nonlinear least squares problems. In particular, the routines we have developed for seismic inversion will need to be modified only as much as we have modified the SUMSL code.

This development includes driver routines for algorithm 1 and 2. The driver for algorithm 2 uses PVM. A sequential version of algorithm 2 can be executed by employing a single CPU. The PVM framework is still required in a sequential execution to provide access to the database handling process.

We have parallelized only steps 4 and 6 of algorithm 2 since these are the function evaluation and function minimization steps. It is possible to parallelize each of the other steps but we have not done so in this phase of our work since our emphasis has been on reducing and parallelizing function evaluations. In fact, it would be possible to run iterations of algorithm 2 in parallel (more than 1 iteration executing concurrently) but, again, we have not pursued this yet.

Random sampling is used to generate the samples for each algorithm. There may be problems involved in using random sampling for generating points in an n-dimensional space so that they provide an "uniform" coverage and it is our intention to use a hybrid systematic and random sampling technique in subsequent work. Eventually, the data base itself can be distributed, if enough processors are available and the size of the problem warrants it.

NUMERICAL RESULTS AND COMPARISONS

Our first implementation stored the reverse trust region constraints in a central location and each local optimization process interrogated the data base to check if a current iterate violated an existing constraint. As many messages as constraints currently in the data base were issued. If a constraint was violated then the corresponding local optimization process was terminated prematurely, thus saving function evaluations.

We were pleased to verify that the method produced the desired results: for most problems we got a decrease in the number of function evaluations without loosing any minimae. The time savings, however, were not impressive, since the function evaluation costs for these test problems were small. Also, when we tried a distributed version with several processors, we found that the speed-up was not very significant. We suspected that the implementation of the static data base, with the number of short messages that it implied, was conspiring to ruin our parallel performance, i.e., communication time was dominant.

In the current implementation we decided to create the data base as a live process, handling all the operations, including updates and checks for constraint violations. We call this an *active data base server.*

This strategy reduced dramatically the number of messages between the optimization procedures and the data base, as compared with our earlier implementation, which we will refer to as a *passive data base server.*

In a first test we failed to see the expected improvement in time, even in cases where the number of function evaluations were reduced substantially. We associated this failure with the fact that for this test problem the function evaluation cost was very low, contrary to what we can expect in realistic inversion problems, and therefore, the reduction in the number of function evaluations could not offset the overhead cost of using the data base. Thus we introduced an artificial number of floating point operations in the function evaluations, to make the problem more similar to a real seismic inversion problem.

We give in Table 1 the results of applying the active data base algorithm to problem 5 of [8] with padding in the function evaluations. We use a single optimization processor and include as reference the times for algorithm 1, the baseline global optimization algorithm without data base constraint checking. We see that in fact the new algorithm reduces substantially the number of function evaluations and the computing time.

We also obtained a considerable speed up when using several processors, as shown for Problem 5 in Table 2. The three processors we used have a combined computing power equivalent to eight SUN IPC's and the speed up was a factor of eight for the no data base algorithm, and 6.9 for the active data base algorithm. A similar behavior was observed for the other test problems.

In conclusion, we can say that an active data base approach is the correct way to implement this type of procedure, and that it definitely will save function evaluations and computer time. When applied to realistic inversion problems, with their expensive function evaluations, we expect to see good efficiency as we use multiple processors.

Algorithm	Function evaluation	Time (sec)	Processor
No data base	1166	440.78	SUN IPC
Active data base	431	188.12	SUN IPC

Table 1. Results for problem 5 with function padding, one processor.

Algorithm	Function evaluation	Time (sec)	Processors
No data base	1166	55.03	SUN 5, SUN 10, IPC
Active data base	401	27.26	SUN 5, SUN 10, IPC

Table 2. Results for problem 5 with function padding, three processors.

References

[1] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. "Global optimization and stochastic differential equations." *J. Optim. Theo. Appl.*, 47:1-16, 1986.

[2] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. "A global optimization algorithm using stochastic differential equations." *ACM Transactions on Mathematical Software,* 14:345-365, 1988.

[3] R. H. Byrd, C. L. Dert, A. H. G. Rinnooy Kan, and R. B. Schnabel. "Concurrent stochastic methods for global optimization". Technical Report CU-CS-338-86, Department of Computer Science - University of Colorado, Boulder, June 1986. To appear in *Mathematical Programming.*

[4] D. Gay. "SUMSL: General unconstrained minimization code." *nanet,* 1982.

[5] A.H.G. Rinnooy Kan and G.T. Timmer. "A stochastic approach to global optimization". In *Numerical Optimization 1984* (P.T. Boggs, R.H. Byrd, and R.B. Schnabel editors), pp. 245-262. SIAM, Philadelphia, 1984.

[6] M. Koshy, V. Pereyra, and J.C. Meza. "Distributed computing applications in forward and inverse geophysical modeling". Society of Exploration Geophysicists 61st Annual Meeting, Houston, TX. Expanded Abstracts, 349-352, 1991.

[7] A.V. Levy and S. Gomez. "The tunneling method applied to global optimization". In *Numerical Optimization 1984* (P.T. Boggs, R.H. Byrd, and R.B. Schnabel, editors), pp. 213-244. SIAM, Philadelphia, 1984.

[8] A.V. Levy and A. Montalvo. "The tunneling algorithm for the global minimization of functions". *SIAM J. Sci. Stat. Comput.*, 6:15-29, 1985.

[9] V. Pereyra. "Two-point ray tracing in general 3D media". *Geophysical Prospecting*, 40:267–287, 1992.

[10] V. Pereyra. "Parallel block inversion of geophysical data". In *Mathematical Methods in Geophysical Imaging.* SPIE, 2033:192-198, 1993.

[11] V. Pereyra and S. J. Wright. "Three–dimensional inversion of travel–time data for structurally complex geology". In J. Bee Bednar, L.R. Lines, R.H. Stolt, and A.B. Weiglein, editors, *Geophysical Inversion*, pp 137–157. SIAM, 1992.

[12] H. Ratschek, and J. Rokne. *New Computer Methods for Global Optimization.* Halsted Press, Chichester, 1988.

[13] Sharon L. Smith, Elizabeth Eskow, and Robert B. Schnabel. "Large adaptive, asynchronous global optimization algorithms for sequential and parallel computation". In Thomas F. Coleman and Yuying Li, editors, *Large-Scale Numerical Optimization*, pages 207–227. SIAM, Philadelphia, 1990.

[14] R. H. Stolt. "Seismic inversion revisited". In J. Bee Bednar, L. R. Lines, R. H. Stolt, and A. B. Weglein, editors, *Geophysical Inversion*, pages 3–19. SIAM, Amsterdam, 1992.

[15] Albert Tarantola. *Inverse Problem Theory.* Elsevier Science Publishers, New York, 1987.

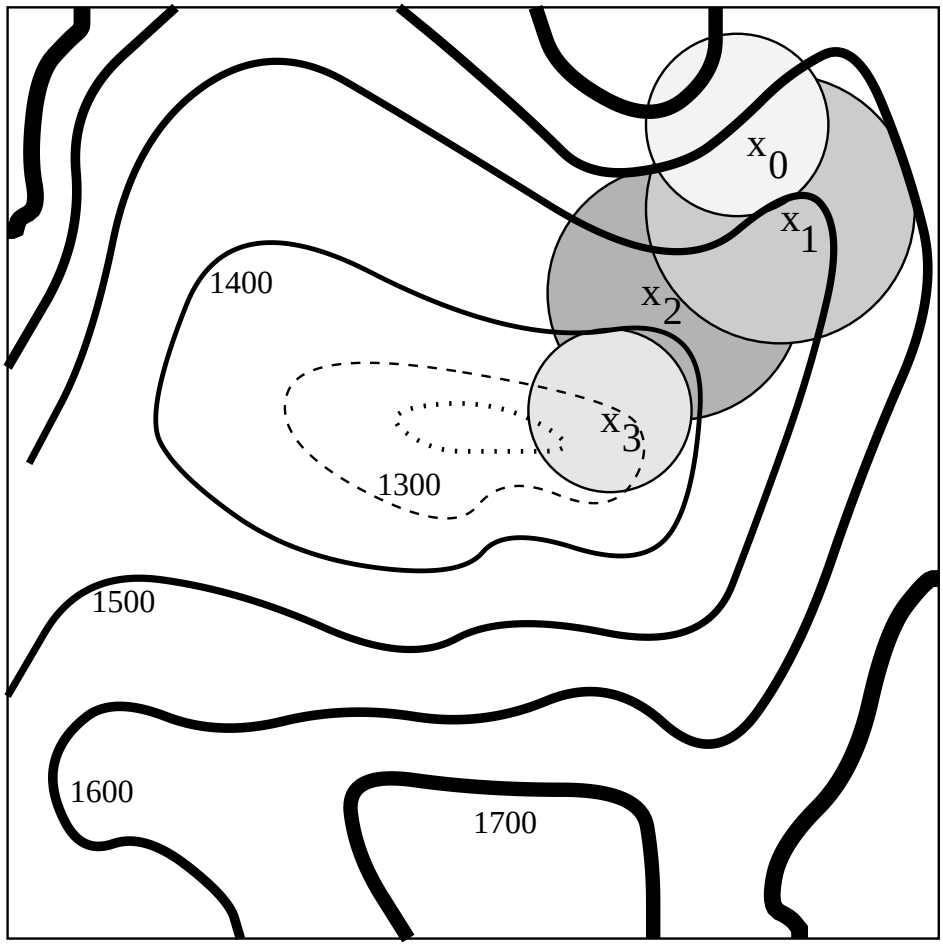

Figure 1: Function contours and 4 trust regions.

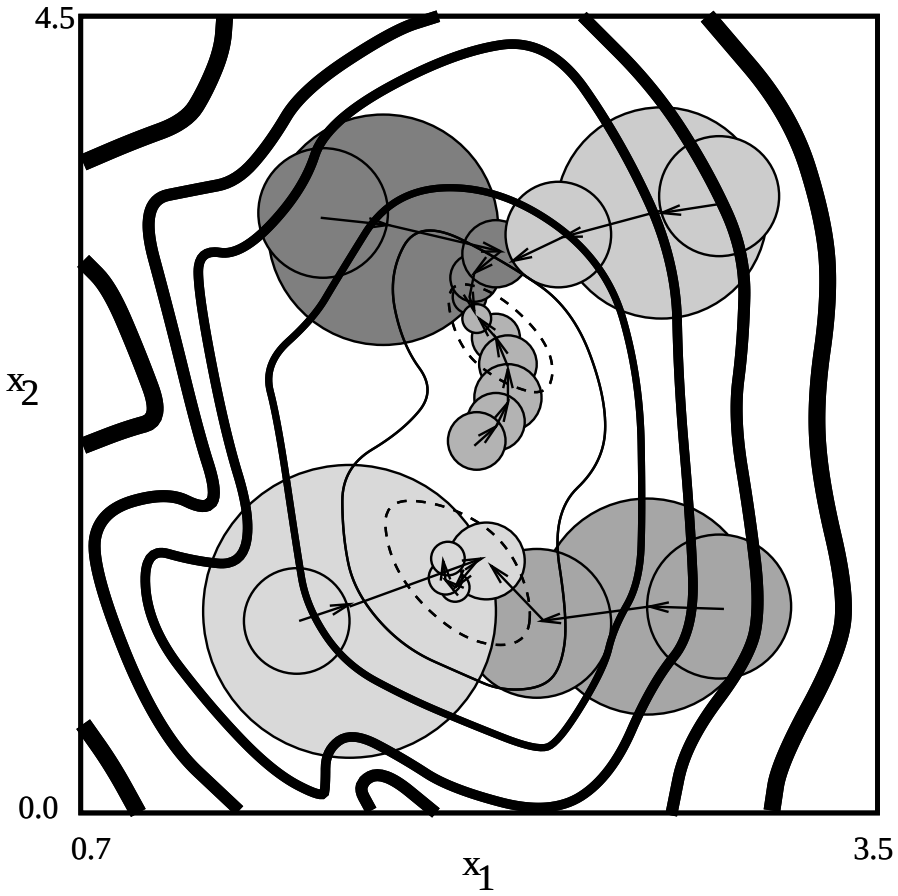

Figure 2: Function contours and trust region dendrites.