# Selective Search for Global Optimization of Zero or Small Residual Least-Squares Problems: A Numerical Study

*L. Velázquez, G. Phillips Jr., R. Tapia, and Y. Zhang*

**CRPC-TR99794**
**September 1999**

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Submitted November 1999

# Selective Search for Global Optimization of Zero or Small Residual Least-Squares Problems: A Numerical Study

L. Velázquez* G. Phillips Jr.† R. Tapia‡ Y. Zhang§

September, 1999

**Abstract**

In this paper, we consider approximating global minima of zero or small residual, nonlinear least-squares problems. We propose a selective search approach based on the concept of selective minimization recently introduced in Zhang *et al* [14]. To test the viability of the proposed approach, we construct a simple implementation using a Levenberg-Marquardt type method combined with a multi-start scheme, and compare it with several existing global optimization techniques. Numerical experiments were performed on zero residual nonlinear least-squares problems chosen from structural biology applications and from the literature. On the problems of significant sizes, the performance of the new approach compared favorably with other tested methods, indicating that the new approach is promising for the intended class of problems.

**Keywords:** Global minimization, zero or small residual least-squares problems, selective minimization, Levenberg-Marquardt method, multi-start.

## 1 Introduction

This paper concerns searching for global minima of least-squares problems whose function values at the globally optimal solutions are zero or very small. This class of problems arises in some data-fitting applications in engineering and science.

Recently, Zhang, Tapia and Velázquez [14] introduced a concept called selective minimization and showed, among other things, that under certain conditions minimization methods for nonlinear least-squares problems, such as the Gauss-Newton and the Levenberg-Marquardt methods, can possess a desirable property called selective minimization. Roughly

*Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005. (Email: leti@caam.rice.edu) This author was supported in part by NSF RTG Grant BIR-94-13229 (the W. M. Keck Center on Computational Biology) and Sloan Foundation Grant 94-12-12.

†Department of Biochemistry and Cell Biology, Rice University, Houston, Texas 77005. This author was supported in part by Welch Foundation Grant C-1142 and NSF RTG Grant BIR-94-13229 (the W.M. Keck Center for Computational Biology).

‡Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005. This author was supported in part by DE-FG03-93ER25178 and DOE/LANL Contract 03891-99-23.

§Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005. This author was supported in part by DOE Grant DE-FG03-97ER25331, DOE/LANL Contract 03891-99-23 and NSF Grant DMS-9973339.

speaking, for zero or very small residual least-squares problems, iterates generated by these methods are locally attracted to minima with sufficiently low function values, but often repelled from local minima with high function values. It is suggested in [14] that the selective minimization property could be a useful tool for constructing or improving global minimization techniques.

This paper is an attempt to test the viability of the selective minimization approach in a global minimization setting. We construct a simple technique based on a combination of a Levenberg-Marquardt type method and a multi-start strategy. Numerical experiments are performed to compare the behavior of the new technique with that of several existing global optimization techniques on test problems from the literature and, in particular, on two model problems from structural biology — the distance geometry problem and the phase-retrieval problem.

In most least-squares data-fitting applications, optimal residuals are generally nonzero and not necessarily small. For those least-squares problems, the approach presented in this paper is not directly applicable. The current work is only a starting point and further research is certainly needed to extend the applicability of the approach.

## 2 Nonlinear Least-Squares Problems

We are interested in finding global minima of a multi-variable function that may possess many local minima. Let us start with the following unconstrained minimization problem:

$$\min f(x), \tag{1}$$

where $x \in \Re^n$ and $f(x) : \Re^n \to \Re$ is at least twice-continuously differentiable. The global optimization problem is to find a point $x^*$ such that

$$f(x^*) \leq f(x) \text{ for all } x \in \Re^n.$$

Such a point $x^*$ is called a global minimizer of the function $f$.

In general, global optimization problems are intractable in terms of computational complexity. However, efficient solutions to specific problems of interesting sizes are still possible through the development of problem-specific algorithms that can effectively exploit the problem structures.

In this paper, we focus on problems whose objective function $f(x)$ in (1) can be formulated as a sum-of-squares of errors or residuals, i.e.,

$$f(x) = \frac{1}{2}\sum_{i=1}^{m} r_i^2(x) = \frac{1}{2}R(x)^T R(x), \tag{2}$$

where for some $m > n$

$$R(x) = [r_1(x), r_2(x), \ldots, r_m(x)]^T \in \Re^m.$$

Let $x^*$ be a global minimizer of a least-squares optimization problem. A problem for which $R(x^*) = 0$, and hence $f(x^*) = 0$, is called a zero-residual problem. A problem is called

a small-residual problem if these quantities are small. (Obviously, smallness is a problem-dependent concept.)

The gradient and Hessian of $f(x)$, as defined in (2), are

$$\nabla f(x) = \sum_{i=1}^{m} r_i(x)\nabla r_i(x) = J(x)^T R(x),$$

and

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x),$$

where $J(x) \in \Re^{m\times n}$ is the Jacobian

$$[J(x)]_{i,j} = \frac{\partial r_i(x)}{\partial x_j},\ i = 1,\ldots,m,\ j = 1,\ldots,n.$$

In least-squares applications, one seeks to find a set of variables that minimizes the error between experimental data and values calculated from a model $\mathcal{M}(x,t)$, where $\mathcal{M}(x,t)$ models a biological or physical system with system variables $t$ and model parameters $x$. The objective is to choose $x$ so that

$$\mathcal{M}(x,t_i) \approx y_i, \quad i = 1,2,\ldots,m.$$

The residuals are defined by

$$r_i(x) = \mathcal{M}(x,t_i) - y_i,\ i = 1,2,\ldots,m, \tag{3}$$

that measure the discrepancy between predicted and observed values.

As a first step, in this paper we consider mostly the ideal cases where the residuals are zero or very small. There do exist applications from which zero-residual problems arise naturally, but they are not common. Future research is needed to extend our approach in this paper to problems with substantial residuals.

## 2.1 Weighted Least-Squares Problems

We now consider a zero-residual problems for which there exists a point $x^* \in \Re^n$ such that all $r_i(x^*) = 0$ in (3), i.e.,

$$\mathcal{M}_i(x^*) = y_i\ \ i = 1,2\ldots,m.$$

By a weighted least-squares problem we mean

$$\min f_w(x) = \frac{1}{2}\sum_{i=1}^{m} w_i^2(\mathcal{M}_i(x) - y_i)^2, \tag{4}$$

where $w_i \neq 0, i = 1,2,\ldots,m$. Obviously, the original formulation of the least-squares problem corresponds to the weights $w_i \equiv 1, i = 1,2,\ldots,m$. The stationary points of $f_w(x)$ satisfy the equation

$$\nabla f_w(x) \equiv J(x)^T W^2(\mathcal{M}(x) - y) = 0,$$

where $W = \mathrm{Diag}(w) \in \Re^{m\times m}$.

Some observations on zero-residual, weighted least-squares problems are listed below.

(i) In general, the set of stationary points of $f_w(x)$, $\{x \in \Re^n : \nabla f_w(x) = 0\}$, changes with $w$; however for a zero-residual problem the set of global minimizers of the weighted least-squares problem is invariant with respect to nonzero weights.

(ii) A proper choice of the weights for a zero-residual least-squares problem can make the global minimization problem easier to solve by local minimization techniques.

Although weighting does not affect global minimizers, it can change the number of stationary points, in particular the number of local minimizers. To see this, let us construct a trivial example of size $m = 2$ and $n = 1$ as follows,

$$\mathcal{M}(x) = \begin{pmatrix} \cos(x) \\ \sin(8x) \end{pmatrix}, \quad \text{and} \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Obviously, the unique global minimizer is $x^* = 0$ for $x \in [-\pi/2, \pi/2]$.

In Figure 1, we plot the function $f_w(x)$ for $w = (1,\ a)^T$ in the interval $[-\pi/2, \pi/2]$ for $a = 1$, 0.5 and 0.005, respectively. As we can see from Figure 1, the number of local minima

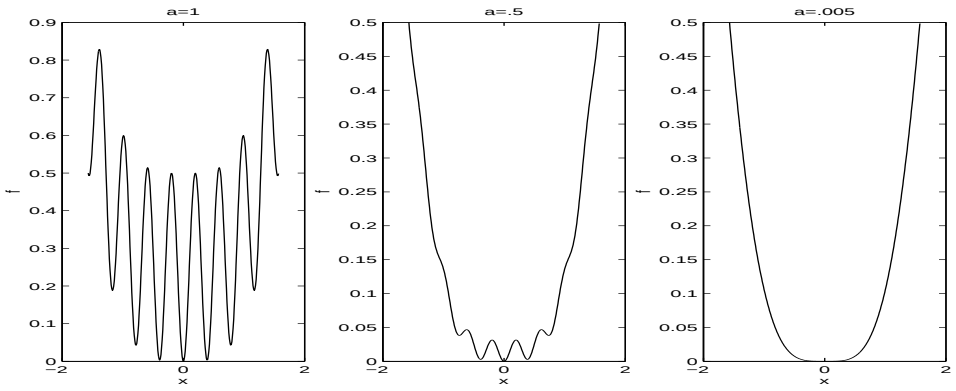


Figure 1: Functions corresponding to different weights.

is reduced as less weight is given to the high frequency component, $\mathcal{M}_2(x)$, while the global minimum remains unaltered at $x = 0$.

In general, properly chosen weights for a zero-residual least-squares problem can reduce the number of local minima, thus making the problem more tractable. Of course, the choice of weights depends greatly on the specifics of a particular problem. For the phase-retrieval problem in X-ray Crystallography, which will be introduced later, the basic guideline is simply to put more weight on low frequency components and less on high frequency ones as in the above example.

## 3 Multi-Start, Selective Search Strategy

We introduce a new global optimization approach based on an old strategy: the multi-start strategy. The classical multi-start strategy is a simple, easy to implement global optimization strategy which combines random sampling with a local minimization algorithm. Each randomly sampled point is a starting point from which one seeks a local minimizer via a

local minimization algorithm. At the end, the local minimizer with lowest function value is taken as an approximation to the global minimizer. Schematically, the classical multi-start paradigm is

$$\text{Classical Multi-Start} = \text{Random Sample} + \text{Local Search}.$$

In contrast, the proposed new multi-start paradigm is

$$\text{New Multi-Start} = \text{Random Sample} + \text{Selective Search}.$$

The key difference in the above two paradigms is that instead of searching for a local minimizer starting from a random point, a selective minimization algorithm is employed to try to find a better minimizer than the closest local minimizer. This selective minimization algorithm has the property that it is attracted to minimizers with sufficiently low function values, including the global minimizers of course, but in general not to minimizers with high function values. Such selective minimization algorithms do exist in some cases (see [14] for relevant theoretical results). In particular, they exist for least-squares problems whose global optimal value is zero or very small; i.e., zero or small residual least-squares problems. The extra information about the residual size plays a crucial role in the selective minimization process.

We now describe the main components of the proposed multi-start, selective search strategy.

### 3.1 Random Sampling

Over the years, considerable research has been done in developing effective random sampling techniques for multi-start methods (see, for example Becker and Lago [2], Locatelli [11], and Törn [13]). Since the random-sample phase is not the focus of the present work, we will use the simplest possible sampling method in our implementation – the uniform sampling, while fully aware of the availability of more sophisticated methods.

### 3.2 Selective Search

The main ingredient of the selective strategy implemented in this paper is the use of a Levenberg-Marquardt type without enforcing descent.

The Levenberg-Marquardt method is a popular technique for finding local minima of least-squares problems (see, for example Björck [3], and Dennis and Schnabel [5]). Given a current iterate $x_c$, a usual implementation of Levenberg-Marquardt method calculates the next iterate $x_+$ as

$$x_+ = x_c - \alpha \left( J(x_c)^T J(x_c) + \mu_c I \right)^{-1} J(x_c)^T R(x_c)$$

for some $\mu_c \geq 0$ and some step length $\alpha \in (0, 1]$. If $\mu_c \equiv 0$, the method becomes the well-known Gauss-Newton method. The reason we choose to use the Levenberg-Marquardt instead of the Gauss-Newton method is that the former is numerically more stable than the latter.

In classical multi-start techniques, a variable steplength approach is implemented in order to guarantee that at each iteration the function value is decreased sufficiently. The

Levenberg-Marquardt method always generates descent directions. A steplength control mechanism such as a line search or a trust region technique is a necessary component to enhance convergence to a local minimizer (see, for example, Dennis and Schnabel [5]).

On the contrary, in the proposed new strategy, a unit steplength is always used regardless whether the function value is decreased or not. This feature is the key in the proposed multi-start strategy that prevents the algorithm from getting trapped easily at local minima, while still letting the algorithm converge to global minima or local minima of very small function values. As a result, the algorithm will more likely locate a global minimum instead of a local minimum of large function values. In our computational experience, we have indeed observed that iterates were often repelled from undesirable local minima which classical multi-start would force iterates to converge to, but still attracted to global minima or those local minima with close to zero function values. This phenomenon is called selective minimization and studied in Zhang, Tapia and Velázquez [14].

### 3.3 The Algorithm MS3

We formally state an algorithmic framework based on the new Multi-Start, Selective Search (MSSS) strategy denoted by Algorithm MS3. This algorithm is designed to take advantage of the special structure of least-squares and the fact that the problem is a zero-residual problem. The new multi-start algorithm is as follows:

**Algorithm MS3**:
Set upper bounds for the number of random initial points $l_{max}$
and iterations $k_{max}$ allowed.
Choose a tolerance $\epsilon > 0$, a parameter $\tau > 0$, and set $l = 0$.
**Until** ( $\|R(x_k)\|^2 < \epsilon$ or $l > l_{max}$ ) **do**
    (1) Select a random starting point $x_0$ and set $k = 0$.
    (2) Increment $l = l + 1$.
    **Until** ( $\|J(x_k)^T R(x_k)\| \leq \epsilon$ or $k > k_{max}$ ) **do**
        (a) Set $\mu_k = \tau \|R(x_k)\|$.
        (b) Compute $s_k = -(J(x_k)^T J(x_k) + \mu_k I)^{-1} J(x_k)^T R(x_k)$.
        (c) Set $x_{k+1} = x_k + s_k$ and increment $k$.
    **End**
**End**

In our implementation, the following parameter values were chosen: $k_{max} = 200$, $l_{max} = 10$, $\epsilon = 10^{-9}$ and $\tau = 0.1$.

We stress that Algorithm MS3 was not meant to be a sophisticated, full-featured global optimization algorithm, but a test bed for the viability of our selective search approach. Hence, we have purposely kept Algorithm MS3 in an extremely simple form.

## 4 Test Problems

In this section we describe three sets of test problems on which our numerical experimentation was conducted. The first set of test problems is from the literature described in

Floudas and Pardalos [6], and Levy and Gómez [9], the second set consists of distance geometry problems generated from the model described in Moré and Wu [12], and finally the third set of problems are created based on a phase-retrieval model problem. For the last two test sets, artificial data was used instead of real-world experimental data in order to create zero-residual problems. It should be emphasized that we are not solving real-world application problems, but rather experimenting with idealized models where the "observed" data were generated from known solutions.

### 4.1 Test Set I: Test Problems from the Literature

A set of ten nonlinear least-squares problems was selected from the literature where the number of variables ranges from 2 to 100.

1. Test Problem No. 1 (Floudas and Pardalos [6]). The function to be minimized is

$$f(x) = \sum_{i=1}^{11} \left( \frac{\delta_i - x_1 b_i^2 + b_i x_2}{b_i^2 + b_i x_3 + x_4} \right)^2$$

subject to

$$0 \geq x_i \geq 0.42$$

where
$\delta = [0.195, 0.194, 0.173, 0.16, 0.844, 0.627, 0.456, 0.342, 0.323, 0.235, 0.246]$,
$b = [4, 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}]$, $x^* = (0.1928, 0.1908, 0.1231, 0.1357)$,
and $f(x^*) = 3.07486e - 4$.

2. Test Problems No. 2-4 (Levy and Gómez [9]). The general function is

$$f(x) = \frac{\pi}{n} \left( 10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} [(y_i - 1)^2 (1 + 10 \sin^2(\pi y_{i+1}))] + (y_n - 1)^2 \right),$$

$$y_i = 1 + 0.25(x_i - 1), \ -10 \geq x_i \geq 10, \ i = 1, 2, \ldots, n$$

where $n$ denotes the dimensionality of the problem. This function has many local minima but only one of them is also a global minimum,

$$x_i^* = 1, \ f(x^*) = 0, \ i = 1, 2, \ldots, n.$$

The test problems 2-4 are created by setting the parameter $n$ to 2, 3, and 4 respectively.

3. Test Problems No. 5-7 (Levy and Gómez [9]). The function to be minimized is

$$f(x) = \frac{\pi}{n} \left( 10 \sin^2(\pi x_1) + \sum_{i=1}^{n-1} [(x_i - 1)^2 (1 + 10 \sin^2(\pi x_{i+1}))] + (x_n - 1)^2 \right),$$

$$-10 \geq x_i \geq 10, \ i = 1, 2, \ldots, n$$

where $n$ denotes the dimensionality of the problem. This function has many local minima but only the following one of them is a global minimum,

$$x_i^* = 1, \ f(x^*) = 0, \ i = 1, 2, \ldots, n.$$

The test problems 5-7 are created by setting the parameter $n$ to 5, 8 and 10 respectively.

4. Test Problems No. 8-10 (Levy and Montalvo [10]).
   The general function used for test problems 5-7 is minimized. Test cases are selected by choosing n= 30, 50 and 100.

### 4.2 Test Set II: Distance Geometry Model Problems

Distance geometry problems arise in the field of computational biology. They can be posed as an unconstrained minimization problem, i.e. given a molecule with $m$ atoms, and distances $\delta_{i,j}$ between atoms $i$ and $j$ for some subset $S$ of the atoms pairs, find positions $x_1, \ldots, x_m \in \Re^3$ that solve the nonlinear least squares problem

$$\min \ f(x_1, x_2, \ldots, x_m) = \sum_{i,j \in S} (\|x_i - x_j\|^2 - \delta_{i,j})^2. \tag{5}$$

The test problems are generated based on one of the model problems described in Moré and Wu [12]. This model simulates a molecule with $m = s^3$ atoms located in the three-dimensional integer lattice

$$(x, y, z) : 0 \leq x < s, 0 \leq y < s, 0 \leq z < s$$

for some integer $s > 1$, where $x, y, z$ are integers.

The following strategy describes how the subset $S$ has been selected: specify an ordering for the set of atoms representing a given "molecule" by letting atom $i$ be the atoms at position $(x, y, z)$, where

$$i = 1 + x + sy + s^2 z,$$

and then define $S$ in terms of an integer $r$ by

$$S = \{(i, j) : |i - j| \geq r\}.$$

This model has distance data for both near and relatively far away atoms for $r = s^2$, and it contains about $rm$ pairs of distances.

### 4.3 Test Set III: Phase-Retrieval Model Problems

Phase-retrieval (more precisely, phase-retrieval refinement) problems are formulated as global minimization problems where the objective functions are expressed in a least-squares format. A simplified model is used that do not take into account all the biological details behind this problem. Even further, we may start with a randomly generated known "solution" and evaluate the model on the solution to generate the "observed" data for the least-square problem. This way, an exact fit always exists which is the known solution.

The problem formulation is as follows: Let $\hat{h}_i = (h_i, k_i, l_i)$, $r_j = (x_j, y_j, z_j)$, $r = (r_1, r_2, \ldots, r_n)^T$ where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Given experimental intensities $I_{\hat{h}_i} = |F^o_{\hat{h}_i}|^2$, the model for the phase-retrieval problem can be written as a least-squares problem:

$$\min \sum_{i=1}^{m} \left( |F_{\hat{h}_i}(r)|^2 - |F^o_{\hat{h}_i}|^2 \right)^2 \tag{6}$$

where

$$|F_{\hat{h}_i}(r)|^2 = \left(\sum_{j=1}^{n} \cos(2\pi\ \hat{h}_i^T r_j)\right)^2 + \left(\sum_{j=1}^{n} \sin\,(2\pi\ \hat{h}_i^T r_j)\right)^2. \tag{7}$$

In our simplified setting, the objective is to locate the atomic positions $r$ as the global solution of the above unconstrained minimization problem.

The function to be minimized is highly nonlinear. As a result, there exist many local minimizers which makes it difficult to locate the desired global minimizer. In our numerical experimentation, we generate five instances of phase-retrieval model problems with the number of atoms $n = 4$, 5, 10, 21 and 39, respectively. The corresponding number of variables is $3(n-1)$ since one atom is fixed. A set of integer lattice vector, $\{\hat{h}_i \in \Re^3,\ i = 1, \ldots, m\}$, is generated where each component of $\hat{h}_i$ varies between 0 and $\pm 18$. The "observed" data set $\{|F^o_{\hat{h}_i}|^2\}$ is generated by evaluating $|F_{\hat{h}_i}(r)|^2$ at known positions $r^*$. In this way, we create a zero residual least-squares problem which attains its global optimum at $r^*$.

For instances of the model problems with 4, 5 and 10 atoms, the known solutions $r^*$ are generated randomly. The two instances with 21 atoms and 39 atoms are created by using actual atomic positions that resemble a part of the known structures of aspirin and caffeine proteins, respectively.

## 4.4 Weighted Phase-Retrieval Problem

A weighting scheme for the phase-retrieval problem formulation (6) is derived in order to reduce the number of local minima. Let $y_i = |F^o_{\hat{h}_i}|^2$ and $\mathcal{M}_i(r) = |F_{\hat{h}_i}(r)|^2$.

Applying the binomial expansion to (7), and invoking the identities $\sin^2 a + \cos^2 a = 1$, and $\cos(a-b) = \cos a \cos b + \sin a \sin b$, we have

$$\mathcal{M}_i(r) = 2\sum_{j=1}^{n}\sum_{k=j+1}^{n} \cos\left(2\pi \hat{h}_i^T (r_j - r_k)\right) + n. \tag{8}$$

Let us denote the first term by

$$p_i(r) = 2\sum_{j=1}^{n}\sum_{k=j+1}^{n} \cos\left(2\pi \hat{h}_i^T (r_j - r_k)\right). \tag{9}$$

Then the residual functions have the form

$$\mathcal{M}_i(r) - y_i = p_i(r) + (n - y_i). \tag{10}$$

Now we try to smooth out the function $p_i(r)$ by taking its average over some tube.

**Proposition 4.1** *Let $p_i(r)$ be defined as in (9). The average function of $p_i(r)$ over a cube centered at $r$ with length $2t$ is given by*

$$\bar{p}_i(r) = w_i p_i(r),$$

*where*

$$w_i \equiv w(t\hat{h}_i) = \left(\prod_{j=1}^{3} \frac{\sin(2\pi \hat{h}_i(j)t)}{2\pi \hat{h}_i(j)t}\right)^2, \tag{11}$$

*and $\hat{h}_i(j)$ represents the $j$-th component of $\hat{h}_i$ for $j = 1, 2$ or 3.*

**Proof:** Recall that the average of a function $f(r) : \Re^n \to \Re$ on a hypercube centered at $r$ with length $2t$ is determined as follows:

$$f_{\text{ave}}(r) = \frac{1}{(2t)^n} \int_{-t}^{t} \cdots \int_{-t}^{t} \int_{-t}^{t} f(r+s) ds_1 ds_2 ds_3 \cdots ds_n \tag{12}$$

where $(2t)^n$ is the volume of the hypercube. For each term in the sum of (9) that involves six variables in $r_j$ and $r_k$, the average is the following six-time integral

$$\int_{-t}^{t}\int_{-t}^{t}\int_{-t}^{t}\left(\int_{-t}^{t}\int_{-t}^{t}\int_{-t}^{t} 2\cos(2\pi \hat{h}_i^T((r_j+s_j)-(r_k+s_k)) ds_j^1 ds_j^2 ds_j^3\right) ds_k^1 ds_k^2 ds_k^3$$

divided by $(2t)^6$, where $s_\ell = (s_\ell^1,\ s_\ell^2,\ s_\ell^3)^T$ for $\ell = j$ and $k$. Using the formula

$$\int_{-t}^{t} \cos(\alpha + \beta\tau) d\tau = \frac{2\ \sin(\beta t)}{\beta} \cos\alpha,$$

one can verify that the average function of $p_i(r)$ over a hypercube is nothing but

$$\bar{p}_i(r) = w_i p_i(r),$$

where

$$w_i \ \equiv\ w(t\hat{h}_i) = \frac{1}{(2t)^6}\left[\left(\frac{2\sin(2\pi\hat{h}_i(1)t)}{2\pi\hat{h}_i(1)}\right)\left(\frac{2\sin(2\pi\hat{h}_i(2)t)}{2\pi\hat{h}_i(2)}\right)\left(\frac{2\sin(2\pi\hat{h}_i(3)t)}{2\pi\hat{h}_i(3)t}\right)\right]^2,$$

$$= \left(\prod_{j=1}^{3} \frac{\sin(2\pi\hat{h}_i(j)t)}{2\pi\hat{h}_i(j)t}\right)^2.$$

We called this factor $w_i$ the weighting factor. If the constant term $n - y_i$ in (10) is also equally weighted, then the weighted residual terms

$$w_i(\mathcal{M}_i(x) - y_i) = \bar{p}_i(r) + w_i(n - y_i)$$

are smoother than the unweighted ones $\mathcal{M}_i(r) - y_i$. Consequently, with an appropriately chosen value $t$, the weighted sum of squares $f_w(r)$ is smoother for $w_i = w(t\hat{h}_i)$ than for $w_i = 1$.

Observe that for any fixed $t$, $w(t\hat{h}_i) \to 1$ if $\|\hat{h}_i\|_\infty \to 0$, and $|w(t\hat{h}_i)|$ is smaller for larger $\|\hat{h}_i\|_\infty$. Therefore, this choice of weights does have the desirable property that it weights the low frequency terms more heavily than the high frequency ones. The level of bias in this weighting scheme can be controlled by the parameter $t$. In our numerical experiments, we have set $t = 0.8$.

# 5 Numerical Experimentation

In this section, we provide information on the numerical experimentation conducted with five global optimization techniques on the test problems described in the last section. The main objective of the experiments is to see how Algorithm MS3 performs in comparison with four other existing global optimization techniques: a multi-start damped Gauss-Newton method, a simulated annealing technique, and the classical and exponential tunneling methods.

### 5.1 Five Global Optimization Techniques

We present the five global optimization techniques and some implementation details below.

1. MGN: Classical Multi-Start Damped Gauss-Newton.
   This algorithm, written in Matlab, is a specific implementation of the classical multi-start approach. The structure of this algorithm is closely related to the new multi-start Algorithm MS3. The main differences are that damped steps are taken using a backtracking strategy that enforces descent on the objective function, and that the Gauss-Newton method is selected as the local minimization method. The same stopping criteria, sampling technique, and parameters $l_{max} = 10$, $k_{max} = 200$, and $\epsilon = 10^{-9}$ as in Algorithm MS3 are implemented.

2. SA: Simulated Annealing.
   Simulated annealing is a popular global optimization technique in the computational biology community and other application fields. Simulated annealing is a probabilistic approach introduced by Kirkpatrick, Gelatt and Vecchi [8]. It works by emulating the physical process that occurs when a solid is heated with high temperature until all particles randomly arrange in a liquid phase, and then is slowly cooled down by decreasing the temperature so that when its structure is frozen at low temperature, a minimum energy crystalline configuration is obtained. This approach achieves solutions of low function values by employing a random search which not only accepts changes that decrease the objective function, but also some changes that increase it.

   The algorithm starts by choosing a high temperature $T_0$ and an initial configuration $x_0$ with an energy (function) value $E_0$. It randomly searches a neighborhood of $x^0$ until it accepts a new configuration $x^1$ with energy value $E_1$. Then the temperature is decreased by a given scheme and the process is repeated until a prescribed low temperature is reached. A new configuration $x^1$ is always accepted if $\Delta E = E_1 - E_0 < 0$, i.e. $x^1$ has a lower energy value; otherwise it is accepted with a probability $p = \exp(-\Delta E/T_0)$ (or with some other probability distributions). The latter feature allows the algorithm to move away from local minima.

   The software used in our experiments is called *Simann*. It is a Fortran program described in Goffe, Ferrier and Rogers [7]. In the numerical experimentation, the algorithm terminates if the maximum number of 20000 iterations is exceeded or a given stopping criteria is met. The internal parameters are chosen as suggested in the code.

3-4. $\mathrm{T_C}$: Classical Tunneling and $\mathrm{T_E}$: Exponential Tunneling.
   The tunneling method described in Levy and Gómez [9] and Levy and Montalvo [10] is an iterative procedure for solving global unconstrained minimization problems. This method consists of two phases. The first phase involves finding a local minimizer via a local minimization technique, and the second phase, tunneling, finds a new point with a function value no greater than the previous minimum found.

   The algorithm is described as follows: in the minimization phase, given a current point $x_c$, use any unconstrained minimization method until a local minimizer $x^*$ of

$f(x)$ is found. Then initiate the tunneling phase, by finding a new $x_c$ such that

$$f(x_c) \leq f(x^*),\ x_c \neq x^*.$$

In order to move away from $x^*$, one minimizes the classical tunneling function,

$$T_c(x, x^*, \eta^*) = \frac{f(x) - f(x^*)}{[(x - x^*)^T(x - x^*)]^{\eta^*}},$$

or the exponential tunneling function,

$$T_e(x, x^*, \eta^*) = [f(x) - f(x^*)]\ \exp\left(\frac{\eta^*}{[(x - x^*)^T(x - x^*)]^{\eta^*}}\right)$$

both of which create a pole at $x^*$ with a pole strength $\eta^*$. To calculate $\eta^*$, small increments, e.g $\Delta\eta = 1.0$, are added to $\eta_0 = 0$. This pole is needed to move iterates away from the current local minimizer.

Now, we set $x_0 = x^* + \epsilon$ where $\epsilon$ is a small random perturbation. As long as $T_{(c,e)}(x_k, x^*, \eta^*) > 0$ or a maximum number of iterations is not reached, the algorithm generates a sequence

$$x_{k+1} = x_k + \alpha s_k$$

by using the descent direction for $T_{(c,e)}(x, x^*, \eta^*)$ at $x_k$ defined by

$$s_k = -T_{(c,e)}(x_k, x^*, \eta^*)\frac{\nabla T_{(c,e)}(x_k, x^*, \eta^*)}{||\nabla T_{(c,e)}(x_k, x^*, \eta^*)||^2}.$$

The steplength $\alpha \in (0, 1]$ is determined from a bisection technique.

The exponential tunneling function is preferred over the classical tunneling function because it can create a pole at a local minimizer independent of the precision of the local minimum found (see Barron and Gómez [1]).

The experimentation was done using the Tunneling Fortran code, and the internal parameters of this algorithm are set as suggested by Castellanos and Gómez [4].

5. MS3: Multi-Start, Selective Search.
   The Algorithm MS3 was written in Matlab, and the code follows the pseudo code described in Section 3.3.

The computation was performed on a SUN SparcStation 4 computer running Solaris 2.6 with 64 Megabytes of memory. The results for each global optimization technique were recorded in terms of the CPU time in seconds for it to solve each test problem from different initial points. The algorithms were run with the same starting points for each test problem, and the starting points were either chosen randomly or specified in the original references.

For each problem in Set I, the five algorithms were run four times with the same set of initial points. The starting points for test problem 1 were created randomly, and for test problems 2-10 were the same points listed in Levy and Gómez [9], and Levy and Montalvo [10]). In the case of Set II, only three runs for each of problem were performed because of the computational effort required. For Set III, four different runs for each of the five model problems were performed with the five different techniques.

For all the problems, we stop the algorithms when one of the following criteria is met:

i) The tolerance $f(x^*) \leq 1e-9$ is met, indicating convergence to a global minimizer $x^*$.

ii) The run time of the algorithms exceeds a CPU time of 500 or 1000 seconds for Set I and Set II, respectively. For Set III, a total CPU time of 2000 seconds is allowed.

We set the limits on the maximum CPU time in order to have a timely completion of our numerical experimentation. However, the simulated annealing code may still stop itself even when neither of the above two criteria is satisfied.

## 5.2 Numerical Results

Table 1 summarizes the numerical results obtained on the test problems in Set I. In this table, the first column gives the problem number, the second the number of variables, and the third the number of local minima. The next five columns specify the total CPU time in seconds required by the five codes, MGN, SA, $T_c$, $T_e$ and MS3, respectively, to reach the global minimum starting from four different initial points. The subscript "$*$" denotes that the simulated annealing code stopped itself at a point $x$ with $f(x) \approx 1.0^{-4}$. The notation "$> 500$" indicates that a code did not converge to a global minimum within the limit of 500 CPU-seconds from each of the four initial points[1].

As can be seen from Table 1, for lower dimensional problems, i.e., problems 1-7, most of the codes were able to find the global minima, except for the code MGN which had difficulty solving even small-sized problems within the time limit of 500 CPU seconds.

For higher dimensional problems, i.e., problems 8-10, Algorithms $T_c$ and $T_e$ did not solve the problems within the CPU time limit, while the SA code stopped at some suboptimal solutions. Algorithm MS3 is the only one that solved all the problems to the prescribed accuracy within the time limit.

Table 2 shows the numerical results obtained with three codes, SA, $T_e$ and MS3, on Test Set II — distance geometry problems. The other two techniques were not competitive on these problems, hence excluded. The first column indicates the number of variables of the test problem. These three test problems are generated by setting $s = 3, 4, 5$, respectively. Three runs were performed for each test problem and are denoted by R1, R2, R3. The code terminates if it exceeds the run time limit of 1000 seconds for each run. If the code converges to the global minimum, its run time in CPU second is reported. The subscript "$*$" denotes that the simulated annealing code stopped itself at some suboptimal points. The notation "$> 1K$" indicates that a code did not converge to a global minimum within the limit of 1000 CPU-seconds.

As can be seen from Table 2, on the first problem all three codes were able to successfully terminate before the time limit, though the simulated annealing code did not quite find a global minimum. For this problem, the tunneling approach was much faster than the other two. As the dimensionality of the problem increases, the tunneling and simulated annealing codes did not find a global minimum within the time limit. On the other hand, in most cases the MS3 code was able to reach a global minimum.

Problems in Test Set III turn out to be more difficult. The small phase-retrieval model problems of 4, 5 and 10 atoms are already a greater challenge than one would expect.

[1] In these experiments, it never occurred that a code stopped at the 500-second limit from one of the four starting point, but converged to a global minimum from another one.

Table 1: CPU time for Test Set I

| Prob. No. | No. of Variables | No. of local Minima | MGN time | SA time | $T_c$ time | $T_e$ time | MS3 time |
|---|---|---|---|---|---|---|---|
| 1 | 4 | | $> 500$ | 1.98 | 5.22 | 6.10 | 6.67 |
| 2 | 2 | 25 | $> 500$ | 1.74 | 2.48 | 7.89 | 5.45 |
| 3 | 3 | 125 | 2.05 | 3.28 | 8.90 | 7.89 | 6.61 |
| 4 | 4 | 625 | 2.15 | 3.97 | 10.81 | 8.49 | 7.44 |
| 5 | 5 | $10^5$ | 55.3 | 4.86 | 9.52 | 4.76 | 14.39 |
| 6 | 8 | $10^8$ | $> 500$ | 9.38 | 24.35 | 27.99 | 23.81 |
| 7 | 10 | $10^{10}$ | $> 500$ | 13.82 | 34.64 | 40.53 | 19.83 |
| 8 | 30 | $10^{30}$ | $> 500$ | $144.42_*$ | $> 500$ | $> 500$ | 29.57 |
| 9 | 50 | $10^{50}$ | $> 500$ | $422.85_*$ | $> 500$ | $> 500$ | 46.97 |
| 10 | 100 | $10^{100}$ | $> 500$ | $427.218_*$ | $> 500$ | $> 500$ | 175.65 |

Table 1 illustrates the total CPU time in seconds that each code took to converge to a global minima from four initial points. The subscript "$*$" indicates that the simulated annealing code stopped itself at a point $x$ with $f(x) \approx 1.0^{-4}$, and the notation "$>500$" indicates that a code did not find a global minimum within the limit of 500 CPU seconds.

Table 2: CPU time for Test Set II

| No. of. Variables | SA | | | $T_e$ | | | MS3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| 81 | $147*$ | $153_*$ | $159_*$ | 2.53 | 3.43 | 3.22 | 59 | 64 | 61 |
| 192 | $> 1K$ | $> 1K$ | $342_*$ | $> 1K$ | $> 1K$ | $> 1K$ | 366 | 400 | 516 |
| 375 | $> 1K$ | $> 1K$ | $> 1K$ | $> 1K$ | $> 1K$ | $> 1K$ | 810 | $> 1K$ | 845 |

Table 2 reports the computing time in CPU second per run that each code took to terminate, successfully or otherwise, for three separate runs denoted by R1, R2 and R3. The subscript "$*$" indicates that the simulated annealing code stopped itself at a local minimum, and the notation "$>1K$" states that a code did not find a global minimum within the limit of 1000 CPU seconds.

Algorithm MS3 was the only code that was able to solve the problems within the time limit of 2000 CPU-seconds from a set of random starting points. The model problem of 21 atoms, shown in Figure 2, was also solved by MS3 after several attempts. However, even the MS3 code was not successful on the larger model problem of 39 atoms.

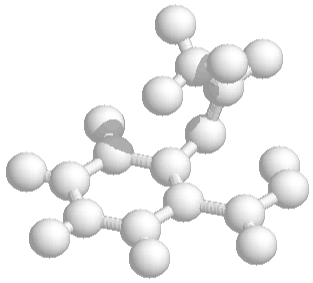

Figure 2: A Structure of 21 Atoms

## 6 Concluding Remarks

Simulated annealing and tunneling methods are general-purpose global optimization techniques which do not take into account the information on the size of residuals available for our test problems. On the other hand, the selective search technique, Algorithm MS3, is designed specifically for the class of zero or small residual least-squares problems. We do not make general claims on the relative efficiency of the tested methods based on the limited scope of our tests. However, our numerical experiments do indicate that the selective search strategy is a viable and promising approach to solving the zero or very small residuals least-squares problems. Further research in this direction is needed to develop more robust algorithms and to extend the applicability of this approach.

## Acknowledgment

We sincerely thank Susana Gómez for generously providing us with their Tunneling Fortran program.

## References

[1] C. Barron and S. Gómez. The exponential tunneling method. Technical report, IIMAS-UNAM, 1991.

[2] R.W. Becker and G.V. Lago. A global optimization algorithm. *In Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pages 3–12, 1970.

[3] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[4] L. Castellanos and S. Gómez. Tunnel: A fortran subroutine for global minimization with bounds on the variables using the tunneling method. Technical Report (In preparation), IIMAS-UNAM, 1998.

[5] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics (SIAM): Classics in Applied Mathematics, Philadelphia, PA, 1996.

[6] C.A. Floudas and P.M. Pardalos. *Recent Advances in Global Optimization*. Princeton Press, Inc., Englenwood Cliffs, New Jersey, 1992.

[7] B. Goffe, Ferrier, and Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 30(1-2):65-100, Jan/Feb 1994.

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

[9] A.V. Levy and S. Gómez. *The Tunneling Method Applied to Global Optimization*, pages 213-244. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1985.

[10] A.V. Levy and A. Montalvo. The tunneling algorithm for the global minimization of functions. *SIAM J. Sci. Stat. Comput.*, 6(1):15-29, 1985.

[11] M. Locatelli. Relaxing the assumptions of the multilevel single linkage algorithm. *J. of Global Opt.*, 13:25-42, 1998.

[12] J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM J. on Optimization*, 7(3):814-836, 1997.

[13] A.A. Törn. Global optimization as a combination of global and local search. *In Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economic Models*, pages 191-206, 1972.

[14] Y. Zhang, R.A. Tapia, and L. Velázquez. On convergence of minimization methods: Attraction, repulsion and selection. Technical Report TR99-12, Rice University, Department of Computational and Applied Mathematics, MS-134, Houston, TX 77005, 1999. Submitted to *SIAM Journal on Optimization*.