Journal of Global Optimization, , 1–12 (2005)

Nonconvex Piecewise-Quadratic Underestimation for Global Minimization

O. L. MANGASARIAN olvi@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706
Department of Mathematics
University of California at San Diego
La Jolla, CA 92093

J. B. ROSEN jbrosen@cs.ucsd.edu
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093

M. E. THOMPSON thompson@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Received April 2005

Editor: Panos Pardalos

Abstract. Motivated by the fact that important real-life problems, such as the protein docking problem, can be accurately modeled by minimizing a nonconvex piecewise-quadratic function, a nonconvex underestimator is constructed as the minimum of a finite number of strictly convex quadratic functions. The nonconvex underestimator is generated by minimizing a linear function on a reverse convex region and utilizes sample points from a given complex function to be minimized. The global solution of the piecewise-quadratic underestimator is known exactly and gives an approximation to the global minimum of the original function. Successive shrinking of the initial search region to which this procedure is applied leads to fairly accurate estimates, within 0.0060%, of the global minima of synthetic nonconvex functions for which the global minima are known. Furthermore, this process can approximate a nonconvex protein docking function global minimum within four-figure relative accuracy in six refinement steps. This is less than half the number of refinement steps required by previous models such as the convex kernel underestimator [4] and produces higher accuracy here.

1. Introduction

Our principal concern in this work is the global unconstrained minimization of a nonconvex function with multiple local minima. Such NP-hard problems occur in real-life problems such as protein docking [5]. Although there is no guaranteed finite method that can solve this problem in reasonable time, interesting effective approaches for attacking this problem have been recently proposed that obtain a global solution for a number of synthetic problems with multiple minima. In [2, 6, 9] the nonconvex function is successively bounded below by a strictly convex quadratic function whose minimum gives improved estimates of the global minimum of the nonconvex function. In [3] the nonconvex function is underestimated by a piecewise linear function and in [4] by a convex kernel function [11, 1, 10] and a global solution of the underestimator is found. All these methods, though effective in their own way, do not take advantage of the important fact that the original nonconvex function is very closely modeled by a nonconvex function which itself is the minimum of a finite number of convex functions. The global minimum of this latter nonconvex function is easily computed as the least of the minima of the convex functions constituting it. It is precisely this approach that we shall utilize in this paper which will lead to an approximate global solution in a finite number of steps.

We outline the contents of the paper now. In Section 2 we formulate our problem as that of obtaining a close underestimator that is the minimum of a finite number of strictly convex piecewise-quadratic functions. In Section 3 we state our algorithm which consists of minimizing a linear function on a polyhedral reverse convex region and establish its termination in a finite number of steps at a stationary point. In Section 4 we give results for numerical testing of our algorithm on various sized nonconvex synthetic test problems including a model for a protein docking problem. In all instances tested, the global minimum value was attained within 0.0060%. Section 5 concludes the paper.

A word about our notation and background material follows. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. Superscripts will typically denote instances of matrices, vectors or scalars such as H^i, c^i or α^i. A column vector of ones of arbitrary dimension will be denoted by e, while the identity matrix of arbitrary dimension will be denoted by I. For a concave function $f : R^n \rightarrow R$ the supergradient $\partial f(x)$ of f at x is a vector in R^n satisfying

$$f(y) - f(x) \leq \partial f(x)(y - x) \tag{1}$$

for any $y \in R^n$. The set $D(f(x))$ of supergradients of f at the point x is nonemepty, convex, compact and reduces to the ordinary gradient $\nabla f(x)$, when f is differentiable at x [7, 8]. The notation := will denote a definition of a term on the left of the symbol and =: will denote a definition of a term on the right of the symbol, while $\arg\min_S c's$ will denote the set of minimizers of $c's$ in the set S.

2. Piecewise-Quadratic Underestimation

The problem we are interested in is to find the global minimum of a function $f : R^n \longrightarrow R$, given m function evaluations of $f(x)$, that is:

$$y^j = f(x^j),\ j = 1, \ldots, m. \tag{2}$$

In [9] a strictly convex quadratic underestimate:

$$q(\alpha, c, H, x) = \alpha + c'x + \frac{1}{2}x'Hx,\ H \text{ symmetric positive definite,} \tag{3}$$

is first obtained by solving the mathematical program:

$$\begin{array}{lrcl} \min\limits_{\alpha,c,H} & \sum\limits_{j=1}^{m}(y^j - q(\alpha,c,H;x^j)) & & \\ \text{s.t.} & q(\alpha,c,H;x^j) & \leq & y^j,\ j=1,\dots,m, \\ & & H & \text{symmetric positive definite,} \end{array} \tag{4}$$

where $\alpha \in R$, $c \in R^n$ and $H \in R^{n\times n}$, and then minimizing $q(\alpha,c,H;x)$ over $x \in R^n$.

Our proposed approach is to replace the strictly convex quadratic, possibly inaccurate, underestimator (3) by the much more accurate nonconvex function which is the minimum of ℓ strictly convex quadratic functions:

$$q(p;x) := \min_{1\leq i\leq \ell} \alpha^i + {c^i}'x + \frac{1}{2}x'H^ix,\ H^i \text{ symmetric positive definite,} \tag{5}$$

where, for simplicity, p represents the variables (α^i, c^i, H^i) as follows:

$$p^i := \begin{bmatrix} \alpha^i \\ c^i \\ H^i \end{bmatrix}, i=1,\dots,\ell,\ \ p := \begin{bmatrix} p^1 \\ \vdots \\ p^\ell \end{bmatrix}. \tag{6}$$

Our approximation for an underestimator of $f(x)$ is obtained by solving the following maximization problem.

$$\begin{array}{ll} \max\limits_{p} & \sum\limits_{j=1}^{m}(\min\limits_{1\leq i\leq \ell} \alpha^i + {c^i}'x^j + \frac{1}{2}{x^j}'H^ix^j) \\ \text{s.t. } y_j & \geq \min\limits_{1\leq i\leq \ell}(\alpha^i + {c^i}'x^j + \frac{1}{2}{x^j}'H^ix^j),\ j=1,\dots,m, \end{array} \tag{7}$$

which can be rewritten in the following equivalent form:

$$\begin{array}{rl} \max\limits_{(p,\gamma)} & \sum\limits_{j=1}^{m}\gamma_j \\ \text{s.t.}\ \min\limits_{1\leq i\leq \ell} & (\alpha^i + {c^i}'x^j + \frac{1}{2}{x^j}'H^ix^j) - y_j \leq 0,\ j=1,\dots,m, \\ \gamma_j & -\alpha^i - {c^i}'x^j - \frac{1}{2}{x^j}'H^ix^j \leq 0,\ i=1,\dots,\ell,\ j=1,\dots,m, \\ \gamma_j & -y_j \leq 0,\ j=1,\dots,m. \end{array} \tag{8}$$

We note that in this reformulation of (7), the last constraint of the reformulation (8) is completely redundant. However, it is added in order to generate a convex polyhedral set containing our nonconvex feasible region when the first set of constraints of (8) are dropped. We also note note that the nonconvexity of the feasible region is caused by the "reverse convex" first set of constraints of (8), each component of which generates a feasible region whose complement is convex, as depicted in Figure 1 which utilizes the following notation. By defining:

$$s := \begin{bmatrix} p \\ \gamma \end{bmatrix} = \begin{bmatrix} p \\ \gamma_1 \\ \vdots \\ \gamma_m \end{bmatrix}, \tag{9}$$

our optimization problem (8) can be written as follows.

$$\begin{array}{ll} \max\limits_{s\in S} & d's \\ \text{s.t.} & h(s) \le 0 \end{array} \tag{10}$$

where $d := \left[\begin{smallmatrix} 0 \\ e \end{smallmatrix}\right]$, $h(s)$ and S, a convex polyhedral set, are defined as follows.

$$h_j(s) := \min_{1\le i\le \ell} (\alpha^i + c^{i\prime} x^j + \frac{1}{2} x^{j\prime} H^i x^j) - y_j,\ j = 1, \dots, m,$$

$$S := \left\{ s \;\middle|\; \begin{bmatrix} \gamma_j & -\alpha^i - c^{i\prime}x^j - \frac{1}{2}x^{j\prime}H^i x^j \\ \gamma_j - y_j & \end{bmatrix} \le 0,\ i = 1, \dots, \ell,\ j = 1, \dots, m \right\}. \tag{11}$$

We note that the nonconvexity in the minimization problem (10) is caused by the reverse convex constraint $h(s) \le 0$ while the set S is a convex polyhedral set. See Figure 1. We turn now into our algorithmic part of the paper.

3. The Successive Linearization Algorithm

The basic idea of the algorithm is to first find a solution $\hat{s}$ of the linear program $\max\limits_{s\in S} d's$ and then to solve a succession of linear programs whose feasible regions are generated by adding to the polyhedral constraint $s \in S$ appropriate supporting planes to the complement of the reverse convex set $\{s \mid h_j(s) \le 0\}$ for some j. The concavity of $h_j(s)$ ensures that the resulting polyhedral set is contained in the original feasible region of (10):

$$T := \{s \mid h(s) \le 0, s \in S\}. \tag{12}$$

Before stating our algorithm it is convenient to define the active linear parts $h_{ji}(s)$ of the constraint $h_j(s) \le 0$ for $j = 1, \dots, m$ at the point s as follows.

$$\begin{aligned} h_j(s) &:= \min_{1\le i\le \ell} (\alpha^i + c^{i\prime} x^j + \frac{1}{2} x^{j\prime} H^i x^j) - y_j, \\ &= (\alpha^i + c^{i\prime} x^j + \tfrac{1}{2} x^{j\prime} H^i x^j) - y_j,\quad i \in I(j) \subset \{1, \dots, \ell\} \\ &=: h_{ji}(s),\quad i \in I(j) \subset \{1, \dots, \ell\}. \end{aligned} \tag{13}$$

We now state our algorithm.

Algorithm 1 Feasible Successive Linearization Algorithm (FSLA)
Let $\hat{s}$ be a solution of the linear program $\max\limits_{s\in S} d's$ and let $s^0 \in T$. Having s^k determine $s^{k+1} \in T$ as follows.

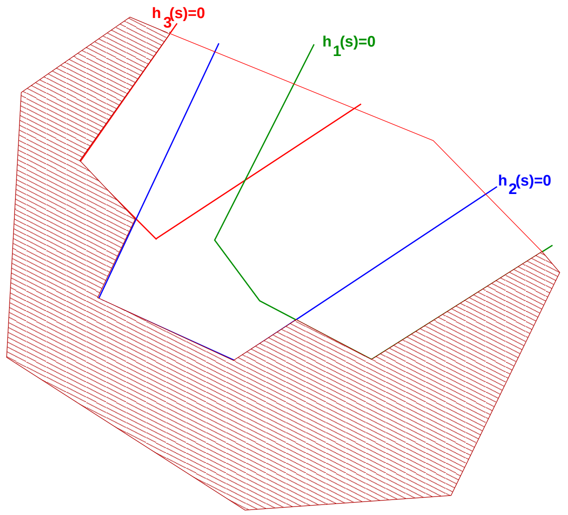


Figure 1. **The crosshatched reverse convex feasible region T (12) of the optimization problem (10) consisting of the intersection of the polyhedral set S with the reverse convex sets generated by $h(s) \leq 0$.**

(i) Let $\hat{s}^k = s^k + \lambda^k(\hat{s} - s^k)$ where:

$$\lambda^k := \min_{1\leq j\leq m}(arg \min_{0\leq\lambda\leq 1}\{\lambda \mid h_j(s^k + \lambda(\hat{s} - s^k)) = 0\}), \tag{14}$$

and let the active set of constraints at $\hat{s}^k$ be defined as:

$$J(\hat{s}^k) := arg \min_{1\leq j\leq m}(arg \min_{0\leq\lambda\leq 1}\{\lambda \mid h_j(s^k + \lambda(\hat{s} - s^k)) = 0\}). \tag{15}$$

Note: $\hat{s}^k \in T$.

(ii)

$$s^{k+1} \in arg \min_{s\in S}\left\{ d's \;\middle|\; \begin{array}{rcl} h_j(\hat{s}^k) + \partial h_j(\hat{s}^k)(s - \hat{s}^k) & \leq & 0,\ \ j \in J(\hat{s}^k) \\ \sum_{i\in I(j)} h_{ji}(s) & \leq & 0,\ \ j \notin J(\hat{s}^k) \end{array} \right\} \tag{16}$$

Note: $s^{k+1} \in T$ because of the concavity of $h(s)$ and the fact that the second set of constraints of (16) are supergradients of the concave constraints $h_j(s) \leq 0$, $j \notin J(\hat{s}^k)$. Note also that $h_j(\hat{s}^k) = 0$ for $j \in J(\hat{s}^k)$ in (16).

(iii) Stop if $s^{k+1} = \hat{s}^k$. Else $k + 1 \to k$ and go to (i).
Note: $d's^k \leq d'\hat{s}^k \leq d's^{k+1} \leq d'\hat{s}^{k+1}$.
Hence, the nondecreasing bounded above sequences $\{d's^k\}$ and $\{d'\hat{s}^k\}$ converge provided the feasible region T is bounded.

We state now our finite termination result for our FSLA Algorithm at a KKT point of our original problem (10).

Proposition 2 Finite Termination of FSLA Algorithm *Under the nondegeneracy assumption that each $\hat{s}^k$ lies on a nonrepeating unique intersection of the boundary planes of the feasible region T (12), the sequence $\{\hat{s}^k\}$ terminates at a KKT point of the original problem (10).*

Proof At $\hat{s}^k$ the solution s^{k+1} satisfies the following KKT conditions with multipliers u and v, where we have assumed that the polyhedral region S is defined by:

$$S := \{s \mid As \le b\}, \tag{17}$$

for some A and b:

$$\begin{array}{rcl} d + \displaystyle\sum_{j\in J(\hat{s}^k)} v_j \partial h_j(\hat{s}^k) & + & \displaystyle\sum_{j\notin J(\hat{s}^k), i\in I(j)} v_j \nabla h_{ji}(s^{k+1}) + A'u = 0 \\ 0 \le v_j & \cdot & \partial h_j(\hat{s}^k)(s^{k+1} - \hat{s}^k) \le 0,\ j \in J(\hat{s}^k), \\ 0 \le v_j & \cdot & \displaystyle\sum_{i\in I(j)} h_{ji}(s^{k+1}) \le 0,\ j \notin J(\hat{s}^k), \\ 0 \le u & \cdot & (As^{k+1} - b) \le 0. \end{array} \tag{18}$$

The dot denotes a scalar product in the last condition above. Since there is a finite number of unique intersections of boundary planes, the sequence $\{\hat{s}^k\}$ must terminate. But, the only way for $\{\hat{s}^k\}$ to terminate is that $\hat{s}^k = s^{k+1}$. However in that case the KKT conditions (18) degenerate to the KKT conditions of the original problem (10). $\square$

Figure 2 depicts the various iterates of the algorithm and its termination in this very simple case at a global solution.

We note that the nondegeneracy assumption required for the proof of Proposition 2 is rather strong, but was satisfied most of the time in our numerical tests. In the few occasions when it was violated, our algorithm did terminate at a local minimum.

We turn now to our numerical results.

4. Numerical Testing

In our numerical implementation, we enforced the requirement of (5) that each H^i be symmetric positive definite by requiring that each H^i be strictly diagonally dominant as follows. We let L be a lower triangular matrix, set $H^i = L + L'$ and imposed the constraint:

$$L_{ii} \ge 0.1 + \sum_{j<i} |L_{ij}| + \sum_{j>i} |L_{ji}|. \tag{19}$$

In our testing, we found that the success of Algorithm 1 in producing an accurate approximation of the original function was significantly dependent on the initial

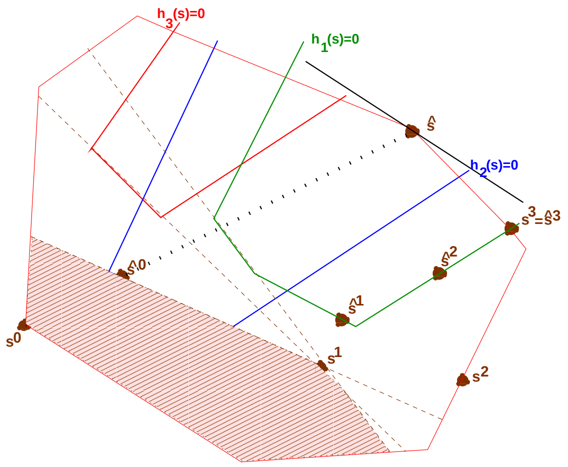


Figure 2. **Iterates of of Algorithm 1 for a simple example. The crosshatched area constitutes the feasible region of problem (16) for the first iteration (ii), that is $k = 0$.**

feasible approximation, s^0. Hence, we divided the given data points (2) into ℓ randomly chosen, possibly overlapping, subsets each consisting of $2+n+n^2$ points. We then fit a quadratic to each subset using Algorithm 1 with $\ell = 1$ with a trivial initial approximation. Each quadratic formed a piece of our initial approximation for the original problem, and was shifted down if need be in order to maintain feasibility. That is, the piecewise-quadratic function consisting of the minimum of the ℓ quadratic functions lay on or below the given m function values y^j, $j = 1,\ldots,m$. Table 1 shows a sequence of objective values for a given s^k starting with s^0 generated in the above manner.

Table 1. **Results from using Algorithm 1 on a synthetic test problem from (20) in R with $k_1 = 1000$, $k_2 = 0.1$ using $m = 7$ initial points and $\ell = 6$ pieces. The actual solution is a minimum value of -1000 at $x = 0$. Note that solving $\max_{s\in S} d's$ gives an upper bound of $d'\hat{s} = -3443.25$**

Iteration k	0	1	2	3	4
$d's^k$	-10354.02	-3444.66	-3444.28	-3443.28	-3443.28
$d'\hat{s}^k$	-3601.05	-3444.66	-3444.28	-3443.28	-3443.28
$\min_x q(p^k;x)$	-1986.406	-999.999	-999.963	-1000.000	-1000.00
$\arg\min_x q(p^k;x)$	0.036	-0.007	0.003	0.001	0.001

To produce our numerical results, we used a refinement process similar to the one used in [3, 4] to reduce the size of the search region while searching for an approximate minima. In it, we approximate the function based on m points uniformly distributed across the search space. We then re-center our search space around the approximated minimum and reduce the size of the search space in each dimension by a specified refinement rate. For example, a refinement rate of 0.25 produces a search space with edges that are 25% of the length of those of the search space of the previous iteration.

Using this process, we tested Algorithm 1 on six nonconvex functions on R^n with $n = 1, \ldots, 6$ defined as follows.

$$y(x) = \frac{1}{2}\|x\|^2 - k_1 cos(k_2 e'x), \; k_1, k_2 \in R \tag{20}$$

Note that this function attains a minimum value of $-k_1$ at $x = 0$. Figure 3 shows an example of this function in R^2 and an approximation of it using the minimum of $\ell = 9$ strictly convex quadratic functions. Figure 4 shows some sample iterates of the refinement process of Algorithm 1 on the function (20) on R.

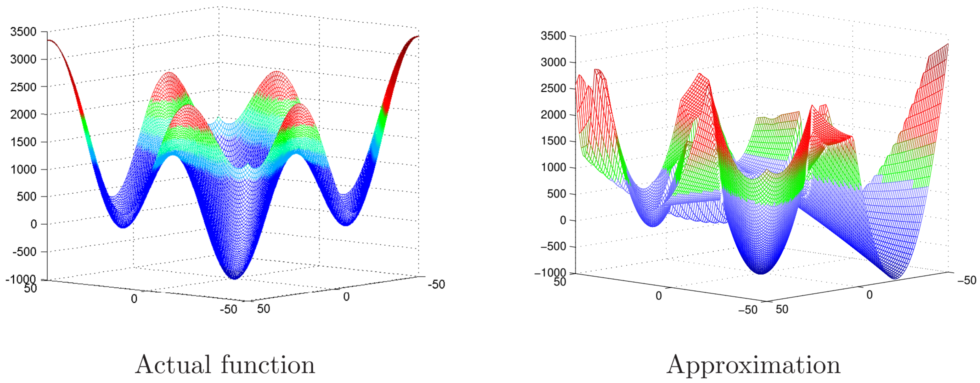


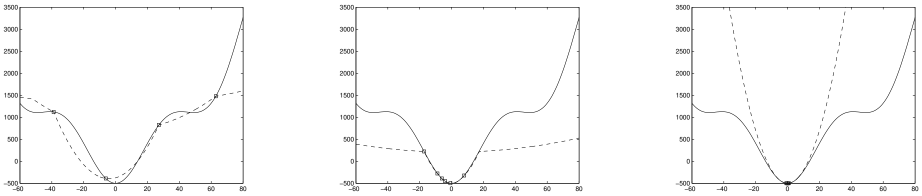
Figure 3. **The function (20) with $k_1 = 1000$ and $k_2 = 0.1$ on R^2 and its approximation using the minimum of $\ell = 9$ strictly convex quadratic functions.**

We also tested the algorithm using six nonconvex piecewise-quadratic functions on R^n with $n = 1, \ldots, 6$ defined as follows.

$$y(x) = \min_{j \in \{1,\ldots,r\}} g_j(x), \tag{21}$$

where $g_j(x)$, $j = 1, \ldots, r$ are arbitrary strictly convex quadratic functions, such as:

$$g_j(x) = \beta^j + {z^j}'x + \frac{1}{2}x'(0.5I + {M^j}'M^j)x, \; j = 1, \ldots, r, \tag{22}$$



Iteration 1 Iteration 2 Iteration 4

Figure 4. **Selected iterations from the refinement process of Algorithm 1 using the minimum of the minima of three strictly convex quadratic functions, that is $\ell = 3$, to find the global minimum of the one-dimensional function defined by (20) with constants $k_1 = 500$ and $k_2 = 0.1$. Our dashed-line approximation underestimates the given solid-line function (20) values at m given points at each iteration.**

Here, $\beta^j \in R$, $z^j \in R^n$ and $M^j \in R^{n\times n}$ are randomly chosen. In our testing, we used $r = 5$. An exact global minimum solution of (21) can be computed as described in [3, Section 4, Proposition 1].

We also tested our algorithm on a synthetic protein docking (SPD) problem generated from real docking data [5, 6]. For the SPD problem, we used the model (21) with $r = 5$ and $(0.5I + M^{j'}M^j)$ replaced by a random diagonal matrix $D^j \in R^6$ with element values between 0.6 and 140. We then computed β^j and z^j such that each $g_j(x)$ is a strictly convex quadratic function with a pre-determined minimum solution corresponding to actual local minima of the docking problem energy function [5]. That is, given local minima at x^j with minimum value v^j, set $z^j = -D^j x^j$ and $\beta^j = v^j - \frac{1}{2}z^{j'}x^j$. Results of these tests are presented in Table 2 for the function of (20) and in Table 3 for the piecewise-quadratic function of (21) including the SPD problem.

It should be noted that we do not use any information about the functions (20) and (21) except their values at m points, selected to be underestimated by the piecewise-quadratic underestimating functions (11). This corresponds to the situation in real docking problems. Typically, in computer models of the docking energy surface (for which we want to locate the global minimum) it is only possible to compute the value $f(x^j)$ (2) of the energy surface at specified $x^j,\ j = 1,\ldots,m$, where x^j represents a possible conformation of a protein-ligand docked pair [5]. Each such value generally requires a significant amount of computation.

We make the following observations regarding our numerical results:

(i) The percent error in the minimum value for all examples including the real-world simulation of the protein docking SPD problem was no more than 0.0060%.

(ii) The percent 1-norm error in the solution vector for all the synthetic problems of Table 3 including the SPD problem was no more than 0.0790%.

(iii) The times for the 6-dimensional problems were no more than two and a half hours. Since this is the largest dimensional fixed docking problem in previous

Table 2. **Results from Algorithm 1 on six synthetic test problems in R^n from (20) with $k_1 = 500, k_2 = 0.1$. The true minimum value is -500, attained at 0. %Error is rounded to 4 decimal places.**

n	1	2	3	4	5	6
m	4	16	64	81	243	729
Refinement rate	0.25	0.25	0.25	0.25	0.25	0.25
Iterate tolerance	0.001	0.01	0.01	0.1	0.1	0.1
ℓ	9	9	5	4	3	3
Computed min	-500.00	-500.00	-500.00	-500.02	-500.02	-500.03
%Error in min	0.0000%	0.0000%	0.0000%	-0.0036%	-0.0046%	-0.0059%
Error in soln (1-norm)	0.0001	0.0000	0.0062	0.0582	0.0442	0.0791
No. refinements	3	5	9	14	11	13
Time (s)	3.65	8.24	38.37	57.99	143.51	4842.12
Time per refinement	1.22	1.65	4.26	4.14	13.05	372.47

Table 3. **Results from Algorithm 1 on six piecewise-quadratic synthetic test problems and the synthetic protein docking (SPD) problem. %Error is rounded to 4 decimal places.**

	Synthetic Problems in R^n						SPD in R^6
n	1	2	3	4	5	6	6
m	4	9	27	81	243	729	729
Refinement rate	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Iterate tolerance	0.1	0.1	0.1	0.1	0.01	0.1	0.1
ℓ	6	3	4	4	3	3	3
True min	-384.6	-12451.8	-20148.6	-23199.1	-6019.2	-7252.0	-42.7
Computed min	-384.6	-12451.8	-20148.6	-23199.1	-6019.3	-7252.5	-42.7
Error in min	0.0	0.0	0.0	0.0	-0.1	-0.4	0.0
%Error in min	0.0000%	0.0000%	0.0000%	0.0000%	-0.0010%	-0.0060%	0.0000%
Error in soln (1-norm)	0.0000	0.0000	0.0000	0.0000	0.0103	0.0281	0.0000
%Error in soln	0.0000%	0.0000%	0.0000%	0.0000%	0.0276%	0.0790%	0.0000%
No. refinements	5	11	7	6	16	13	6
Time (s)	3.56	7.61	10.89	40.90	316.85	8786.54	3005.71
Time per refinement	0.71	0.69	1.56	6.82	19.80	675.89	500.95

work [2, 5, 6] that needs to be handled, our method can reasonably accommodate these problems. In order to handle larger dimensional problems in R^n, one could use m randomly generated points of the order n^3, for example.

(iv) The time, error rates, and number of refinements on the synthetic problems of Table 3 were better than those of [3] for problems in R^2 and greater, including the SPD problem.

(v) The error rates and number of refinements on the problems of Table 3 were better than those of the linear-quadratic kernel of [4]. Also, the number of refinements were fewer than those of the Gaussian kernel of [4], with comparable error rates. The improvement in the number of refinements reflect the fact that Algorithm 1 produces a closer approximation of the original function than the the method of [4]. This is important when actual data is difficult to obtain, and we would like to limit the total number of sample points used. For instance, the SPD problem required 11664 points using the algorithm from [4] with a Gaussian kernel, but only 4374 using Algorithm 1 of this paper. A comparison of the error rates and number of refinements for these different methods can be found in Table 4.

Table 4. **Comparison of Piecewise-Quadratic, Convex Kernel [4], and Piecewise-Linear Underestimation [3] on the SPD Problem in R^6. All algorithms used 729 sample points per refinement step.**

Underestimator	%Error in Soln (1-Norm)	%Error in Min	No. Refinements
Piecewise-Quadratic	0.0000%	0.0000%	6
Gaussian Kernel	0.0085%	0.0000%	16
Quadratic Kernel	0.7767%	0.9290%	12
Piecewise-Linear	0.083%	0.014%	26

Similar to previous work [3, 4], our computations were performed on machines utilizing an 800 Mhz Pentium III processor and 256MB of memory running Tao Linux 1.0, with MATLAB 7.0 installed.

5. Conclusion

We have proposed a method for finding an accurate estimate of the global minimum of a nonconvex function by underestimating the function by a nonconvex piecewise-quadratic function and then finding the easily computable global minimum of the underestimator. The method gives accurate estimates of the global minima for nonconvex functions with multiple minima including a class of synthetic nonconvex functions that closely model protein docking problems. An interesting problem for future consideration is that of approximating nonconvex protein docking energy functions by the minimum of a finite number of convex kernel functions instead of the single convex kernel function used in [4]. This might provide a highly accurate underestimator with an easily computable global minimum.

Acknowledgments

The research described in this Data Mining Institute Report 05-01, March 2005, was supported by National Science Foundation Grants CCR-0138308, ITR-0082146, 051312, by the Microsoft Corporation and by ExxonMobil.

References

1. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, MA, 2000.
2. K. A. Dill, A. T. Phillips, and J. B. Rosen. CGU: An algorithm for molecular structure prediction. In L. T. Biegler et al, editor, *IMA Volumes in Mathematics and its Applications: Large Scale Optimization with Applications III: Molecular Structure and Optimization*, pages 1–22, 1997.
3. O. L. Mangasarian, J. B. Rosen, and M. E. Thompson. Global minimization via piecewise-linear underestimation. Technical Report 03-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, June 2003. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/03-03.ps. *Journal of Global Optimization*, to appear.
4. O. L. Mangasarian, J. B. Rosen, and M. E. Thompson. Convex kernel underestimation of functions with multiple minima. Technical Report 04-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, May 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/04-02.ps. *Computational Optimization and Applications*, submitted.
5. J. C. Mitchell, A. T. Phillips, J. B. Rosen, and L. F. Ten Eyck. Coupled optimization in protein docking. In *Optimization in Computational Chemistry and Molecular Biology*, pages 191–207, Dordrecht, Netherlands, 2000. Kluwer Academic Publishers.
6. A. T. Phillips, J. B. Rosen, and K. A. Dill. Convex global underestimation for molecular structure prediction. In P. M. Pardalos et al, editor, *From Local to Global Optimization*, pages 1–18, Dordrecht, Netherlands, 2001. Kluwer Academic Publishers.
7. B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.
8. R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.
9. J. B. Rosen and R. F. Marcia. Convex quadratic approximation. *Computational Optimization and Applications*, 28:173–184, 2004.
10. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
11. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.