SPT: An improved tunneling method for large global optimization problems.*

David B. Reister
Center for Engineering Science Advanced Research (CESAR)
Computer Science and Mathematics Division
Oak Ridge National Laboratory

Outline

1. Introduction
2. The SPT Algorithm
3. Example Problem
4. Results

*This work is sponsored by the Office of Basic Energy Sciences, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

Introduction - Problem Class

Large problems with hundreds or thousands of parameters.

The function is evaluated by performing a multistep calculation using a computer code.

Calculation of the function is expensive and calculation of the gradient of the function is very expensive.

We will use several of the problem classification methods discussed in Torn and Zilinkas (1989) to introduce our method.

Introduction - Deterministic

Grid search with M evaluations in each dimension.
Function evaluations for N parameters is M^N.
If M = 1000 and N = 10, evaluations are 10^{30}.
If each function evaluation requires 1000 flops,
gigaflop computer: $3\ 10^{13}$ evaluations per year
and grid search would require $3\ 10^{16}$ years.
IBM to develop a petaflop computer: $3\ 10^{10}$ years.

Clearly, grid searches are not practical for problems with hundreds or thousands of parameters.

A trajectory search evaluates the function at points along a trajectory that goes through all local minima.

Deterministic methods are impractical for problems with many parameters.

Introduction - Probabilistic

Line Searches

Random line searches, but not lines in random directions.
We do line searches for a random choice of parameter.
Number of directions increases exponentially with N,
Number of parameters increases linearly.

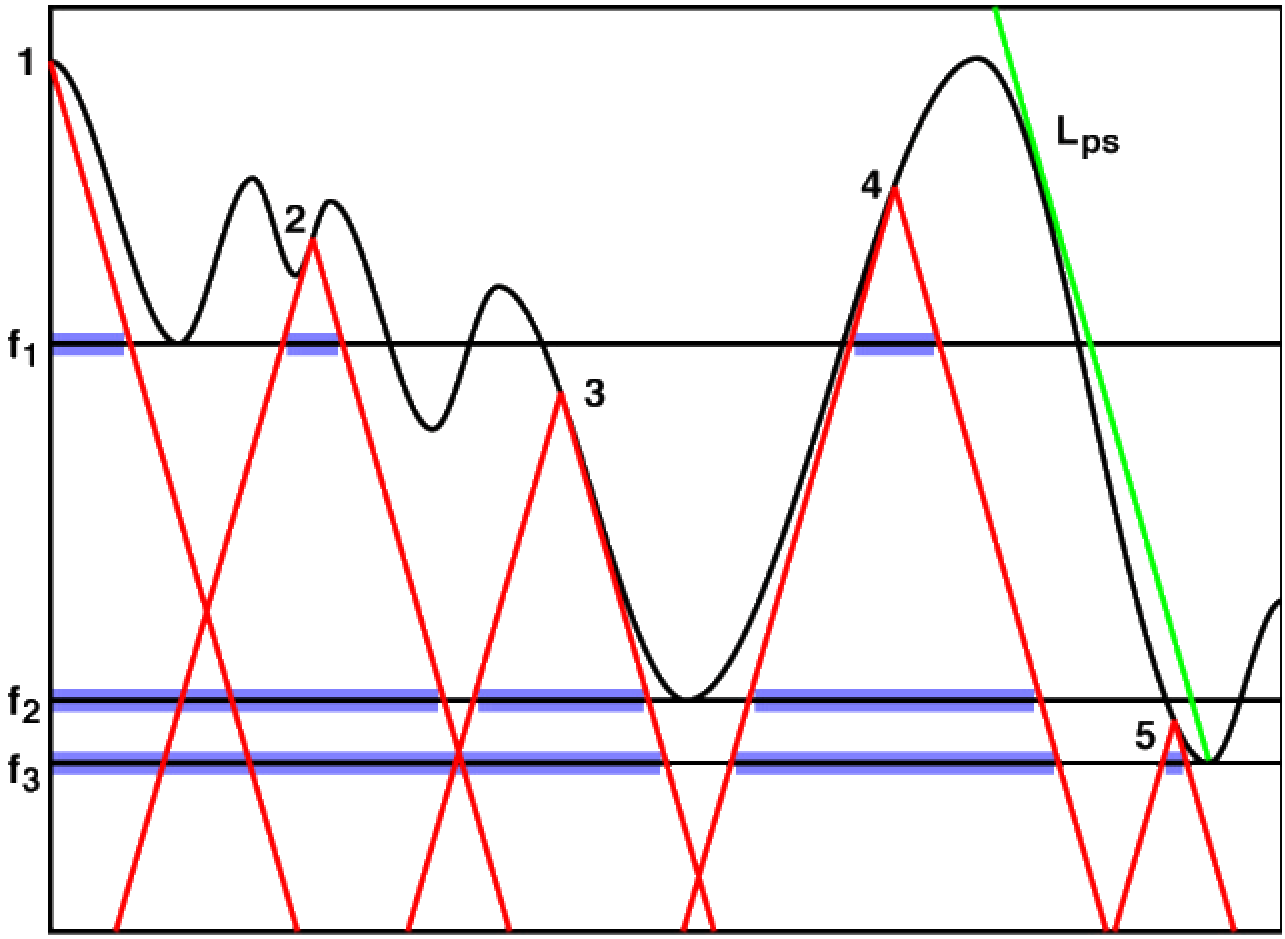
During line search for a chosen parameter,
use a covering method to detect subregions that cannot contain a local minimum and excludes them from further exploration.

The goal of clustering methods is to prevent the rediscovery of local minima.
We descend to a local minimum and do not descend again until we find a point that is lower that the best current minimum (in a new basin of attraction).
Our method is a generalized descent method.

Optimization problem - Sample Problem

Optimization problem: given an objective function [f(x) where x is an N dimensional vector that is constrained to lie in a domain (P)], find the global minimum (x_g).
Find a point x_g in P such that $f(x_g)$ £ f(x) for all x in P.

Sample problem: 1D function displayed with five minima.



Optimization problem - TRUST

TRUST [Barhen et al. (1997)] finds the global minimum by executing a series of cycles of descent and tunneling.

Begin at point x_1, descend to the first local minimum f_1.
Tunnel (move in the positive x direction) until find a value smaller than f_1 and then we would descend to the next local minimum.
Continue to tunnel and descend until reach the upper bound on x.
The lowest value would be the global minimum.

A rigorous proof [Cetin et al. (1993)] has shown that TRUST will find the global minimum for the 1D case.

Cannot use the 1D trajectory method for large problems because the number of function evaluations increases exponentially with the number of dimensions.

TRUST - Line Search

TRUST had good success using many 1D paths to search for the global minimum of a large problem.
From a local minimum, we explore each of the N dimensions of the problem one at a time.
When we find a point where with a lower value than the current local minimum, we descend to the next local minimum.
If we have explored all of the N directions and have not found a lower value, we stop the algorithm.

The choice of the dimension to search next can be numerical order or random.

We have worked on another global optimization problem (protein folding) that cannot be solved by line searches.
It requires an exhaustive search of an M^N grid.

The SPT Algorithm - Introduction

SPT (Stochastic Pijavskij Tunneling) is a new algorithm that greatly speeds up the 1D tunneling phase.
The algorithm is stochastic because the function is evaluated at random points rather than at every point on a grid.
The algorithm was developed by Oblow (1999) and uses Pijavskij cones to eliminate regions of the 1D space

[for a discussion of Pijavskij's algorithm see: Pijavskij (1967), Shubert (1972), and Torn (1989)].

Before we discuss SPT, we will discuss two features of the algorithm: descent to a local minimum and the Lipschitz constant.

The SPT Algorithm - Descent

Calculation of the function is expensive and calculation of the gradient of the function is very expensive.
Gradient required to descend to a local minimum.
Cost high but benefit is high.
When find a lower value than all previous, are in a new basin of attraction.
Continually descend toward the global minimum.

An algorithm that descends from many random starting points can repeatedly descend into a previously identified local minimum.

Two benefits of descent:
No gradients while searching and use gradients only when descending to a new and lower local minimum.

The SPT Algorithm - Lipschitz constant

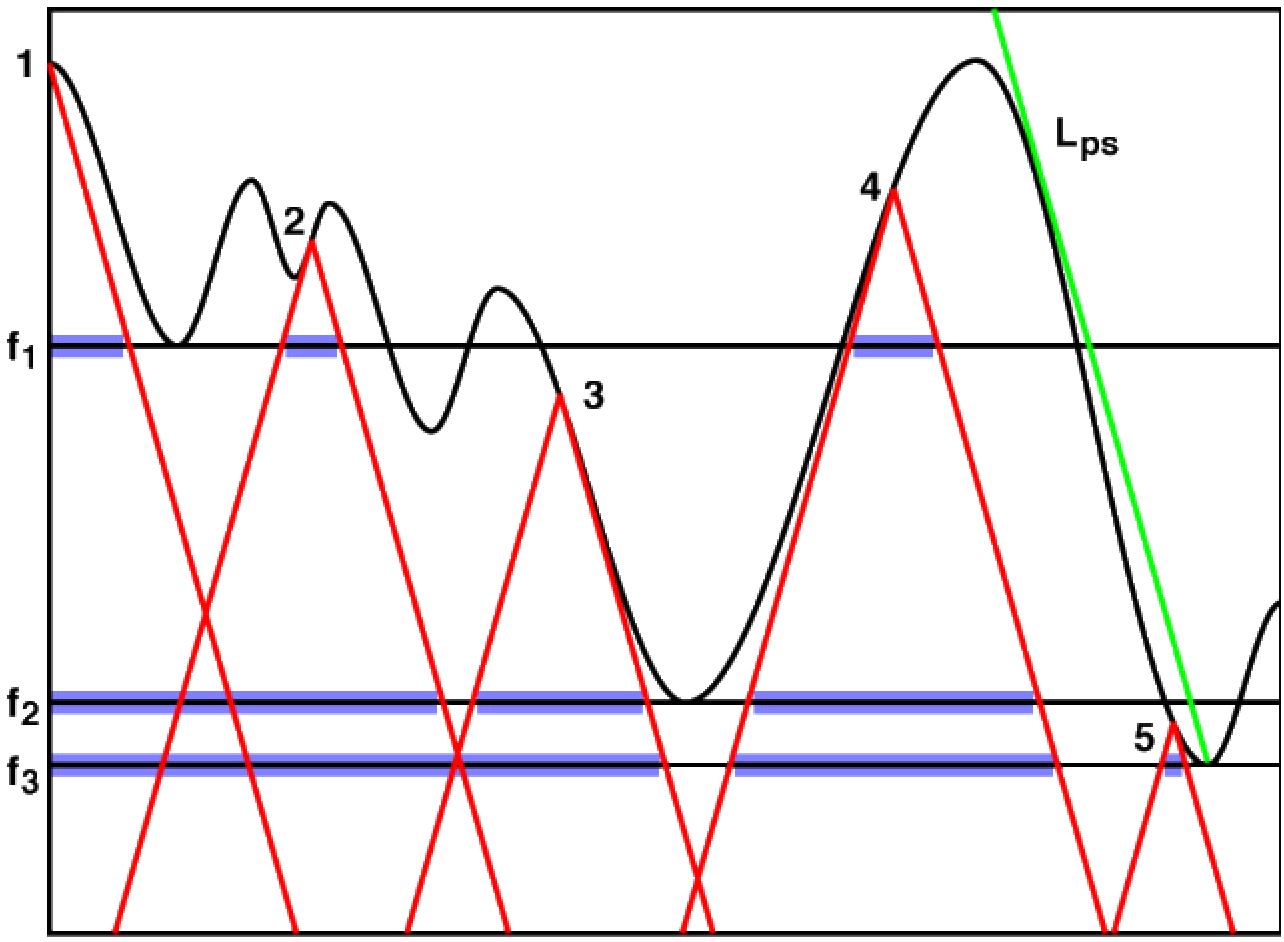
Key parameter in the SPT algorithm is a pseudo-Lipschitz constant (L_{ps}). Used to define unpromising regions in the search space. For the 1D case (where the parameter vector is a scalar), the Lipschitz constant (L) is an upper bound on the rate of change of the objective function [f(x)]:

$$\left|\frac{df}{dx}\right| \pounds L$$

The pseudo-Lipschitz constant (L_{ps}) is the largest slope of any line drawn from the global minimum that is tangent to the curve that defines the basin of attraction for the global minimum

In general, the pseudo-Lipschitz constant is less than the Lipschitz constant. For a square well, the Lipschitz constant is infinity while the pseudo-Lipschitz constant is finite (if we assume that the location of the global minimum is at the midpoint between a and b).

The SPT Algorithm - 1D Case



The SPT Algorithm - Notes for 1D Case

We will use Fig. 1 to discuss a simple version of the SPT algorithm for the 1D case. We evaluate the function at point 1 (x_1). From x_1, we descend to the local minimum (f_1). The local minimum is our best candidate for the global minimum: $f^* = f_1$. From x_1, we draw a line (half of a Pijavskij cone) that has the same slope as the pseudo-Lipschitz line. The intersection of the line from x_1 and the line ($y = f_1$) determines a small region of the x axis (the left shaded region on the f_1 line in Fig. 1) that cannot have a lower value than the current minimum value (f_1).

Rather than performing cycles of descent and tunneling, we randomly choose a second point in the available portion of the x axis (x_2) and evaluate the function. If $f(x)$ is greater than f^*, we are tunneling and if $f(x)$ is less than f^*, we descend to a local minimum. Since $f(x_2)$ is less than f^*, we are tunneling. From x_2, we draw a Pijavskij cone (the magnitudes of the slopes of the two lines are equal to the slope for the pseudo-Lipschitz line). The intersection of the cone from x_2 and the f_1 line excludes a small region of the x axis.

We randomly choose a third point in the available portion of the x axis (x_3) and evaluate the function. Since $f(x_3)$ is less than f^*, we are in the basin of attraction of a new local minimum. From x_3, we descend to the local minimum (f_2) and set $f^* = f_2$. From x_3, we draw a Pijavskij cone. Since the current local minimum is much lower than f_1 , the three Pijavskij cones now exclude much more of the x axis. We choose a fourth point in the available portion of the x axis (x_4) and evaluate the function. Since $f(x_4)$ is greater than f^*, we are tunneling. From x_4, we draw a Pijavskij cone that excludes a large region of the x axis.

We choose a fifth point in the available portion of the x axis (x_5) and evaluate the function. Since $f(x_5)$ is less than f^*, we are in the basin of attraction of a new local minimum. From x_5, we descend to the local minimum (f_3) and set $f^* = f_3$. We evaluate the function at a few more points and conclude that f_3 is the global minimum for the example in Fig. 1.

The SPT Algorithm - Lower Bound

The Pijavskij cones exclude more of the x axis as the current best local minimum (f^*) decreases.

Improve the efficiency of the SPT algorithm by adding a second parameter: a close lower bound for the global minimum (f^{LB}) and using the minimum of f^* and f^{LB} to define the cones.

In the early stages of the algorithm, f^{LB} will increase the region on the x axis that is excluded by the Pijavskij cones.
At the later stages of the algorithm, the current candidate for the global minimum (f^*) may become lower than f^{LB} and the parameter will have no impact.

After several attempts at finding the global minimum, we can use the best value found to date as an estimate of f^{LB}.

The SPT Algorithm - Resolution

The "resolution" as the width of the smallest basin of attraction that can be detected. We use 100,000 points in the parameter range.
SPT allows high resolution at low cost.

Worst case: Golf Course. Objective function is constant with randomly distributed square wells (w = resolution).
No exclusion by the Pijavskij cones.
Need 100,000 function 100,000 function evaluations.

The example problem is much better than the worst case, the Pijavskij cones quickly eliminate all of the x axis and we usually evaluate the function about ten times to achieve a resolution of 100,000 points in the parameter range.

No penalty for working at high resolution.

The SPT Algorithm - Parameter Connections

Two complementary methods to estimate L_{ps}:
measure derivatives and set resolution.
Can monitor the derivatives and determine the largest values which are lower bounds for L_{ps}.

In 1D, L_{ps} is the ratio of delta f(x) to delta x.
Delta f = f - f^{LB}. Delta x = one half the resolution.
Thus, f^{LB} and the resolution yield L_{ps} .

Since resolution is very small, second method gives much higher values than measuring derivatives.

The SPT Algorithm - N D Problem

1. Select a starting point (x).
2. Descend to a local minimum.
3. Begin the outer loop.
4. Begin a loop over the N components of x.
   For each loop, randomly choose n: 1 £ n £ N.
5. Evaluate K_r random points x_{nr} for x_n.
   If x_{nr} is not excluded evaluate the function.
   Save the x_{nr} that gives the lowest value for f(x).
6. If new basin of attraction, descend to minimum.
7. End of the loop that began in step 4.
8. End of the outer loop.

The Example Problem - Geophysics

Data are provided by trace $[t = 1, N_t]$.

Data are Fourier components $[f = 1, N_f]$.

The data $[D_{f\,t}]$ are complex numbers.

Trace: source $[s_t]$ to a receiver $[r_t]$ via a midpoint $[k_t]$.

For each midpoint (k), the data are stacked:

$$H_{k\,f} = \sum_t \exp[2\pi i f(S_{s_t} + R_{r_t})]\, D_{ft}$$

Define the energy for each CMP (E_k):

$$E_k = \sum_f |H_{kf}|^2$$

The statics corrections (S_s and R_r) are determined to maximize the total energy (E) in the stacked data:

$$E = \sum_k E_k$$

Example Problem - Parameters and Results

Synthetic data set. Added disrupting statics to measured seismic data.

316 parameters (100 shots and 216 receivers),
4776 traces, and 423 CMPs.
118 Fourier components.

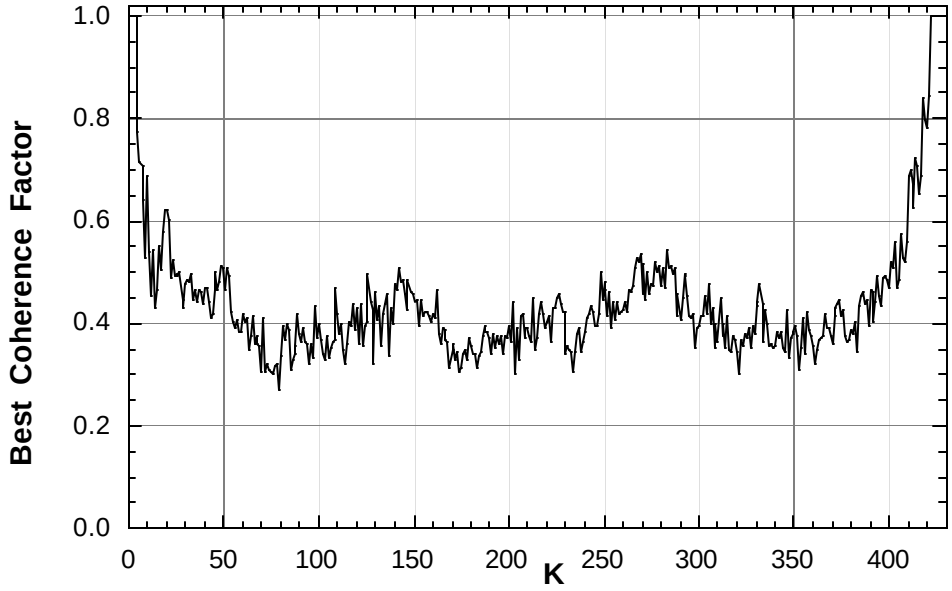
Base Energy (x = 0) = 882.
Upper Bound. $G = \sum_k G_k = 6589.$

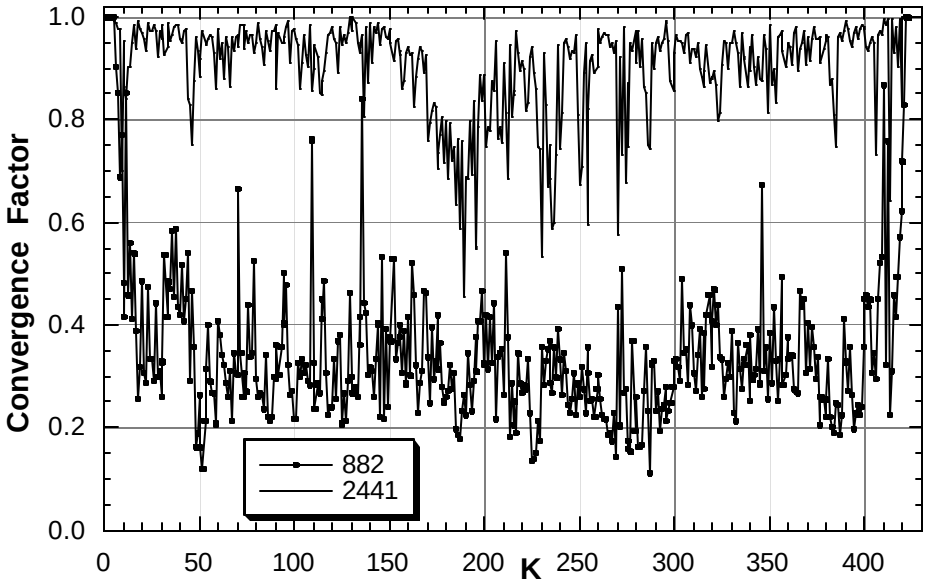
Close upper bound. $^{D}G = \sum_k {}^{D}G_k = 2706.$

Best solution = 2441.

Have 22 points with energy above 2365.

Results


Best Coherence Factor
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400
K


Convergence Factor
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400
K
882
2441