Applying Search Algorithms to the Temperature Inversion problem.

Monte Lunacek, Darrell Whitley, Philip Gabriel and Graeme Stephens

Colorado State University
Fort Collins, Colorado 80523 USA

Abstract. Several inverse problems exist in the atmospheric sciences that are computationally costly when using traditional gradient based methods. Unfortunately, many standard evolutionary algorithms do not perform well on these problems. This paper investigates why the temperature inversion problem is so difficult for heuristic search. We show that algorithms imposing smoothness constraints find more competitive solutions. Additionally, a new algorithm is presented that addresses the difficulties of this problem and finds approximate solutions fast.

1 Introduction

There are a number of problems in the atmospheric sciences where forward models are used to map a set of atmospheric properties to a set of observations.

$$\textbf{MODEL(Atmospheric.properties)} \longrightarrow \textbf{Observations}$$

What is actually needed is the inverse: given the observed data, what atmospheric properties produced those observations? Typically, observations are noisy. In many cases, it is necessary solve these inverse problems in real-time. For example, in several satellite missions, it is necessary to solve these inverse problems several times a second in order to process a days' collection of data in a day.

Traditional gradient based methods can be used, but such methods are computationally extremely costly [**?**]. On the other hand, it would seem that these problems are perfect candidates for heuristic search methods. However, we have found that well known, well tested evolutionary algorithms and local search methods applied to inversion problems do not always yield acceptable solutions.

This paper describes the temperature inversion problem that is central to the retrieval of water vapor profiles. These profiles are used in global atmospheric circulation and weather prediction models. Every set of observations that is collected results in a new temperature inversion problem that must be solved. Results are formally presented for evolution strategies, the CHC algorithm, and a local search bit climber. A number of algorithms have been also been applied to the temperature inversion problem on a more limited basis, including Population-Based Incremental Learning, or PBIL [**?**], and Differential Evolution [**?**]. All of these algorithms fail, and do so in similar ways.

This paper also looks at why the temperature inversion problem is difficult for evolutionary algorithms and local search methods. While the problem is nonlinear, the 2-D slices are smooth and uniformly unimodal. However, these slices show there are ridges in the fitness landscape that can induce false local minima. There are also biases in the evaluation functions so that some parameters (i.e., estimated temperatures) exert a much larger effect on the evaluation function than others.

An algorithm based on ideas proposed by Salomon is constructed and tested that exploits known properties of temperature profiles and produces much more useful results [**?**]. Finally another new algorithm called "Tube Search" is developed and tested. It ignores bias in the evaluation function, and uses smoothness constraints to avoid ridge problems. It quickly produces good approximate solutions.

2 Background

Researchers working in the atmospheric sciences have created a *forward model* that relates vertical temperature profiles to observed measurements. The forward model, as described in this paper, generates 2000 radiance measurements (observations) given a 43 dimensional temperature profile. The k^{th} parameter in the profile is the estimated temperature at an altitude of approximately k kilometers in the atmosphere (in reality, the spacing is somewhat greater at higher altitudes). We actually want to solve the inverse problem: given a set of observations, what is the corresponding temperature profile? In practice, radiance measurements from a constellation of satellites are used in an inverse radiative transfer model. Examples of extant observing systems are: Operational Vertical Sounder (TOVS), the Special Sensor Microwave Imager (SSM/I), and the Advanced Microwave Sounder Unit (AMSU). The inverse solution must be accurate and *fast*; measurements are often collected at a high spatial resolution from satellites whose orbital period is about 90 minutes (or moving at about 8 km /sec).

In this paper the forward model is the simplified form of the equation of radiative transfer that does not account for the presence of clouds. The equation of transfer is solved for the radiances at different wavelengths observed at the top of the atmosphere. This radiative transfer model is "plane parallel" (e.g. one with no horizontal variations in its properties). The radiances are calculated at a certain viewing angle θ as:

$$I_{(\tau,\mu)} = B_\nu(T_s)e^{-\tau_s/\mu} + \int_0^\nu B_\nu(T)e^{-\tau/\mu}\mu^{-1}d\tau$$

$$\begin{aligned} \text{where }\; I_{(\tau,\mu)} &= radiance \\ \mu &= cos(\theta) \\ \tau &= optical\ depth \\ s &= surface \\ B_\nu(T) &= Planck\ radiance\ for\ temperature\ T. \end{aligned}$$

An analytical inversion of this model is impossible because radiances are nonlinearly related to the temperature profiles. Alternatively, the inverse temperature model can be formulated as an optimization problem, where the target temperature profile is the global optimum of the search space. Specifically, the objective function is the root mean squared error between the observable measurements, and the output of the forward model at any point in the search space.

First order derivatives can be calculated analytically for the temperature inversion problem [**?**]. In the more general case where clouds or aerosols are present, the analytical calculation of these derivatives is impossible. Currently, in the simple model, where only blue sky exist, success has been achieved using *Newtonian iteration* and a good starting guess. Although the solution can be accurate, achieving a quadratic convergence rate for solutions near the optimum is highly dependent on a good a priori guess of the temperature profile. In an effort to improve on computational efficiency we attempt to solve the inverse problem using non-derivative search methods.

3 The Search Algorithms: Evolutionary Algorithms and Local Search

Evolutionary algorithms are search methods that simulate evolution through a process of selection, recombination and mutation. The two main forms are genetic algorithms and evolution strategies.

One of the more successful variants of genetic algorithms is CHC [**?**]. CHC uses a bit representation. In this study, the standard binary reflected Gray code is used. CHC uses *cross generational* selection: newly created offspring must compete with the parent population for survival. Parents are not allowed to cross unless they are sufficiently different. CHC uses a modified version of uniform crossover, where half of the non-matching bits are exchanged. No mutation is used, but uniform crossover already randomly assigns non-matching bits, so that mutation would be meaningless. The children under this scheme are always the same maximal Hamming distance from both parents. CHC also includes a restart mechanism that reinitializes the population by randomly flipping 35% of the bits of the best individual.

Evolution strategies, on the other hand, emphasize mutations over recombination. Individuals are represented as real valued vectors. Each individual modifies its parameters to produce offspring. Depending on the implementation, there can be several parents in the population, and each can generate one or more offspring. If many offspring are generated, selection is used to keep each generation the same size. In a (μ, λ) selection strategy, the new population is chosen only from the offspring. A elitist strategy, on the other hand, selects the next generation from both the parents and the offspring. This is known as a $(\mu + \lambda)$ selection strategy. Mutation is usually performed based on a distribution around the individual undergoing mutation. A global distribution can be used for all individuals, or each individual may maintain its own distribution, $\boldsymbol{\sigma}$, often interpreted as a step size.

Other self-adaptive strategies allow the angle of mutation to change. Correlated mutations attempts to estimate the covariance for each pair of object parameters. In other words, an n dimensional problem requires $n(n-1)/2$ rotation parameters, in addition to the n object parameters, and n step size multipliers, $\boldsymbol{\sigma_i}$.

Local search encompasses a broad range of algorithms that search from a current state, moving only if new states improve objective fitness. This has proven to be a simple, yet often effective search method. In this paper, local search refers to a Gray

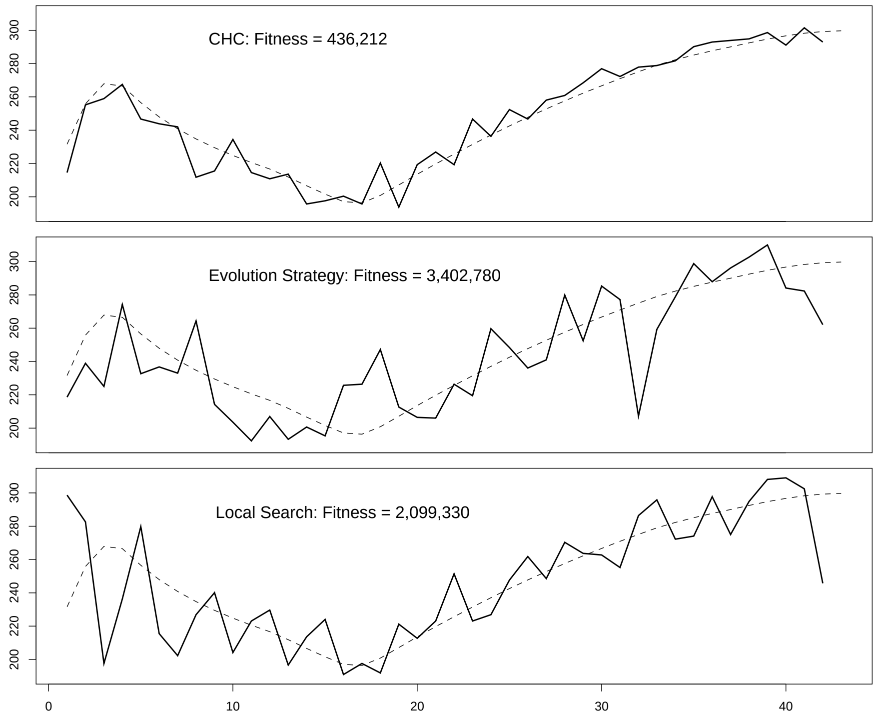

Fig. 1. The best solutions in 30 trials on a McClatchey *tropical* profile. The dashed line indicates the target *tropical* profile, and the solid black line is the best solution found by each algorithm. Even CHC's dominating performance finds a disappointing "zig-zag" solution. None of the solutions find a useful temperature profile.

coded *steepest ascent bit climber*. Each parameter is encoded as a Gray bit string and, by flipping one bit at a time, a neighborhood pattern forms around the current best solution. Local search evaluates all these neighborhood points before taking the best, or *steepest*, step. Because each neighbor differs from the current best by only one dimension, the neighborhood forms a coordinate pattern. Local search terminates when no improving move is found.

Empirical Results

In order to evaluate the various search algorithms on the temperature inversion problem, we used five, well known, McClatchey temperature profiles [**?**]. These profiles encompass a broad range of conditions, from sub-arctic profiles to tropical summer.

The range of the temperatures is $(190, 310)$ Kelvin, a difference of 120. In order to represent this with integer precision, seven bits are needed ($2^7 = 128$). CHC and local search used a Gray encoding scheme. A population size of 50 was used for CHC. For evolution strategies, the high dimension space means that using the correlated mutations model would require $43(42)/2 = 903$ rotation parameters. This much additional over-

Algorithm	Best	Mean	Std Dev
CHC	436,212	850,381	226,674
Evolution Strategies	3,402,780	6,344,321	1,891,605
Local Search	2,099,330	2,886,128	621,260

Table 1. Results of 30 runs of CHC, a (30,210)ES and a local search bit climber on a McClatchey *tropical* profile.

head is impractical; rotations were not used. Back and Schewfel recommend a (μ, λ) selection strategy and indicate that the ideal ratio of parents and offspring is $\mu/\lambda = 1/7$, [**?**]. The (30,210)ES we tested on the temperature problem outperformed a (30+210)ES, and is reported as the evolution strategy contribution in this paper. This implementation used the standard rules for adapting σ [**?**].

Each algorithm was run for 30 trials, each trial using exactly 10,000 evaluations. While 10,000 evaluation is small, we need to reduce the number of evaluations further to achieve real-time performance. Also, limited experiments using up to 100,000 evaluations did not improve the results. The best solutions, using a McClatchey *tropical* profile as the target, are shown in figure 1. The dashed line indicates the target temperature profile, and the solid back, zig-zagging line is the best solution found by each method. The best and average error along with standard deviation is given in Table 1. The amount of error in even the best solutions coupled with the high number of evaluations makes all of these methods impractical.

4 The Ridge Problem, Nonlinearity and the Bias Problem

What makes the temperature inversion problem hard? Without question, the nonlinearity of the problem plays a major role. Specifically, changing parameter k in the temperature profile changes the error surface almost everywhere in the space. An incorrect temperature at location k makes it impossible to correctly assign temperature at other locations. In future work, we may be able to modify the evaluation function to localize the nonlinear effects, since the atmosphere should display physical locality.

Additionally, there seems to be two other major factors. One problem is bias in the evaluation function. The other problem is ridges in the landscape.

Starting from a globally optimal solution, we varied each parameter by plus/minus 2.0. Every move increases the objective error, which is zero when no change is applied. Figure 2 shows the average of the two numbers over the range of the temperature problem. The upper dimensions have greater influence on the error value returned by the evaluation function. The parameters that offer the greatest opportunity to reduce the error will be in the upper dimensions. The bias can cause search algorithms to fit the upper dimensions of the temperature profile first–and to potentially assign incorrect values to the temperature parameters in the lower atmosphere.

Perhaps the most serious problem is that there are *ridges* in the search space. Figure 3 shows several representative 2-D slices of the search space. Although each slice is

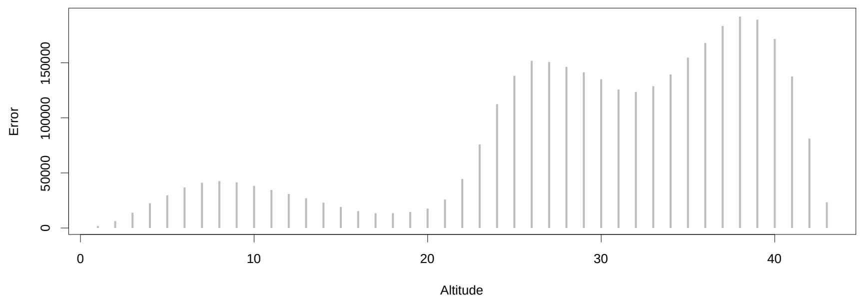

Fig. 2. The average error profile near the optimal solution. Higher dimension parameters contribute more to the error profile.

smooth and unimodal, the curved ridge that cuts through each slice can cause search to become stuck.

Rosenbrock was among the first to noticed that search methods, including derivative-based methods such as *steepest descent*, are crippled by ridge features [**?**]. Winston also notes that ridges cause problems for with simple hill climbers [**?**]. A ridge can cause a search algorithm to believe it has found a local optima, when, in reality, the algorithm is simply stuck on the ridge. Even with an algorithm is not stuck, convergence can be slowed down dramatically.

The ridge problem involves two factors: precision and search direction. If an algorithm looks for improving moves by changing only one dimension at a time (in a coordinate pattern), it will not see better points that fall between the neighborhood axis. This is the *direction* problem. Instead, the search will find improvements close to the current best solution that lie on or near the ridge. Precision dictates how close an algorithm looks for improving neighbors. If the ridge is very steep and narrow, higher precision will be needed to find an improving move.

Increasing the precision will generally decrease the number of false optima. A higher precision search algorithm will be able to move further on the ridge and arrive at better solutions than an algorithm searching at lower precision. The lower precision search will get stuck on the ridge, blindly assuming it has found a local optima. In-

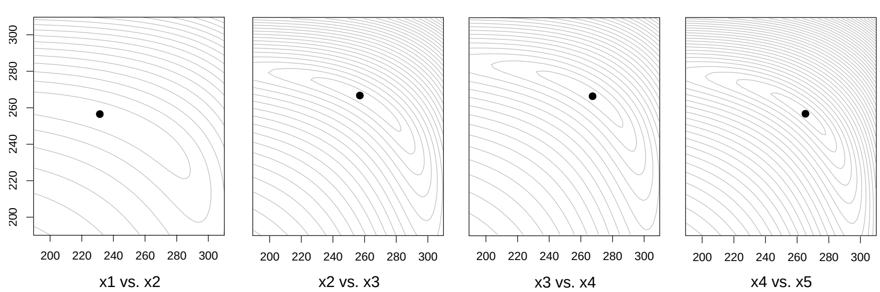

Fig. 3. Two dimensional contours of the first five parameters in the temperature problem. The black dots represent the optimal solution.

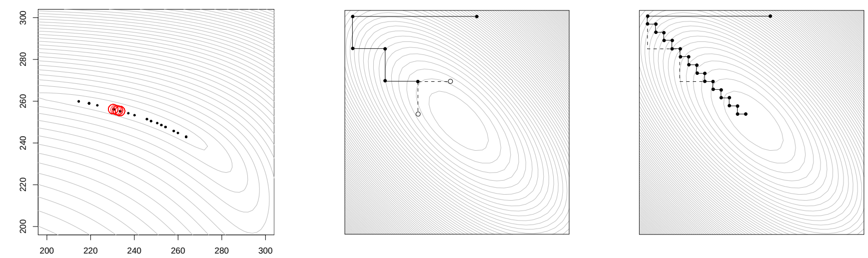

Fig. 4. In the leftmost graph, a high precision local search (large circles) finds the global optima, whereas the low precision search gets stuck in local optima (black dots). In the middle graph, a low precision search induces local optima on a simple parabolic ridge because all the neighbors (dashed lines) have poorer evaluation. The higher precision search (rightmost) is able to make more progress, but at the expense of significantly more evaluations.

creasing precision, however, forces search algorithms to move very slowly through the landscape. This causes an increase in evaluations and a much slower convergence. This phenomena is called *creeping*. Figure 4 graphically explains this problem on a simple parabolic ridge and also documents the existence of this problem on the first two dimensions of the temperature problem. The higher precision search is able to move along the ridge and find a better solution. Low precision induces false optima in the valley.

Local search uses a coordinate pattern to search for globally competitive solution. Therefore, it is not surprising that local search performs poorly in the presence of ridges.

Salomon [**?**] showed that ridges can be created by rotating common benchmark problems. Salomon also points out that the performance of evolution strategies are invariant with respect to a rotation of the coordinate systems. Mutations can move in any direction and multiple parameters normally change. This implies that offspring will not be reproduced on the coordinate axis.

Salomon contrasts this with the Breeder Genetic Algorithm (BGA). On common benchmarks, if the coordinate system is rotated in the $n-$dimensional space, the breeder genetic algorithm often fails. The reason for this failure is largely due to the low probability that a parameter is modified under mutation (commonly $1/l$, where l is the chromosome length). More specifically, the probability that two or more parameters change simultaneously is small. When a ridge runs through a space that is offset from the coordinate axis, it is necessary for all the parameters that align with the ridge to change. The conclusions drawn by Salomon indicate that "crossover's niche" is quite small, and not suitable for problems that have ridges.

The limitations of the breeder genetic algorithm do not extent to all genetic algorithms that use crossover. For example, CHC uses a variation of uniform crossover that changes many parameters at once. Nevertheless, CHC does use a fixed coordinate system. The results of the previous section seem to indicate that CHC performs better than the evolution strategy on the temperature inversion problem.

Salomon suggests that evolution strategies are impervious to the ridge problem because they are invariant to rotations of the search space. However, a 1998 paper by Oyman, Beyer, and Schwefel, presents conditions on a simple parabolic ridge where the elitist ES *limps*, or *creeps* [**?**]. The problem occurred using a $(1 + 10)$ES where a single parent produces ten offspring; the best offspring replaces the parent. The parent is retained if none of the offspring are better. This technique used the "1/5 rule" which means that the step size should produce an improving move one out of every five tries to converge most efficiently. When the parent encounters a ridge, the step size will naturally decrease because of this rule. After reaching the ridge, it is difficult for the evolution strategy to re-adapt its step size and follow the ridge. As a result the evolution strategy can also *creep*.

5 Optimize & Refine

In another paper, Salomon [**?**] also notes that some search algorithms produce results that "zig-zag" the actual solution when the desired solutions displays physical smoothness. To address this problem, Salomon suggests an *optimize and refine* evolution strategy.

The *optimize and refine* technique was inspired by manufacturing methods: many products start with a rough approximation that is refined to be more smooth. The smooth target profile of the temperature inversion problem may be tackled in the same way. The procedure starts by approximating the target with a linear fit. The endpoints, x_1, and x_{43}, are searched for the position where linear interpolation minimizes the objective error. Refinement reduces the regions by half, and the solution becomes a piecewise linear approximation. For example, the next iteration would increase the dimensionality from two to three by adding the point x_{20}. This two piece linear approximation is optimized before more points are added in the next refinement phase.

This method is efficient in several ways. First, a close approximation to the target is found by searching small landscapes. In the temperature inversion problem, a linear approximation reduces the dimensionality of the search space from 43 to only two. This gives higher dimensional searches a good place to start. Second, it forces a smoothness constraint on the problem. Neighboring points in the domain are forced to be relatively close in the range. Finally, assuming that the two dimensional search space is unimodal, the first optimization step establishes a common starting point for higher dimensional searches. This implies that each trial will be consistent in it's solution, making a single trial highly robust.

Salomon used a (1,6) evolution strategy, where a single parent produces six offspring, and uses a non-elitist selection strategy. Instead, in the optimize procedure, we implemented a simple *binary search* to locate the minimum at each inflection point of the piece-wise linear solution. The search started at the endpoints, x_1 and x_{43}. The *binary search* moved to the optimum in each dimension, for several iterations until no improvement could be found. Then, the point x_{20} was added to break the linear region in two, and the optimize procedure was repeated, this time with three points instead of two. At each step, the regions, defined by the current set of points, were cut in half after they had been fully optimized. Figure 5 shows this procedure for the McClatchey *sub-arctic summer* profile. Although this method shows promise, it is able to fit some data examples better than others. Sometimes the solution still "zig-zags" the target. This method also struggles to fit the ends of the profile.

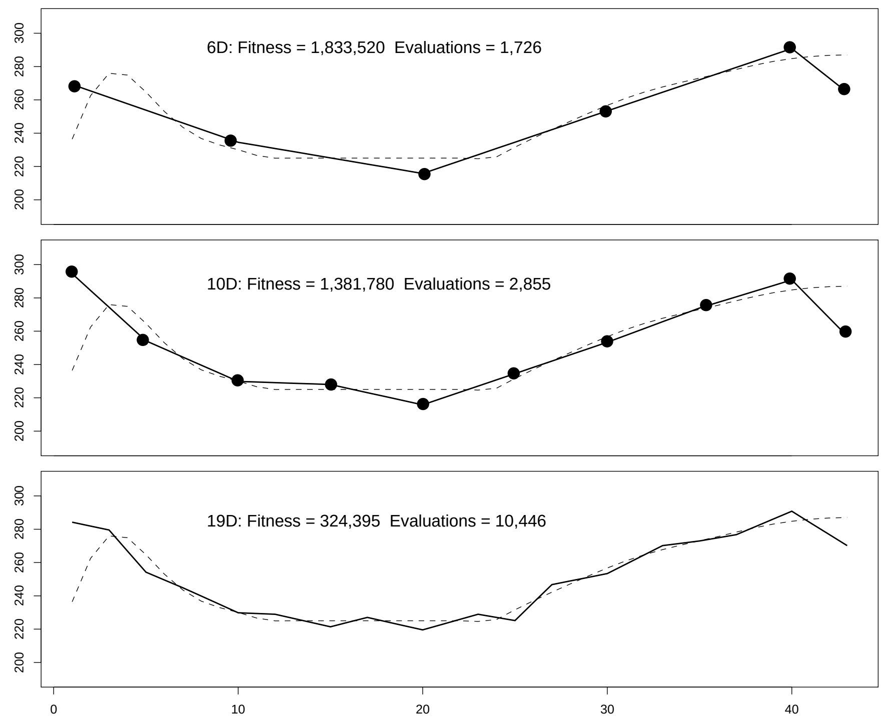

Fig. 5. Optimize and Refine. A fully convergent *sub-arctic summer* solution required an average of 10,446 evaluations. Although this solution fits the profile better than previous methods, it still wanders the target solution.

6 A New Algorithm: Tube Search

The temperature problem appears to have two barriers to a fast and accurate search: ridges structures and bias in the objective function. At the same time, we know that the target temperature profile we are trying to retrieve is relatively smooth, a constraint that is exploited by the *optimize and refine* algorithm.

We implemented a new optimization algorithm called *tube search*. Like optimize and refine, the algorithm starts with a linear fit. This provides a consistent starting point that is smooth, a quality we hope to retain throughout the search. Once the linear fit has been determined, tube search begins. A fixed step is taken on either side of the linear fit–in effect defining a tube about that solution–and the change in evaluation is recorded and stored in a vector. Some moves will offer improvement, while others will not. Once improving moves have been determined, a step of the same magnitude is taken in each improving dimension simultaneously. A three parameter moving average is run on the solution, every five iterations, to maintain a smoothness. More clearly, each parameter is replaced by the average of itself and its two neighbors.

$$temp[i] = \frac{temp[i-1] + temp[i] + temp[i+1]}{3}$$

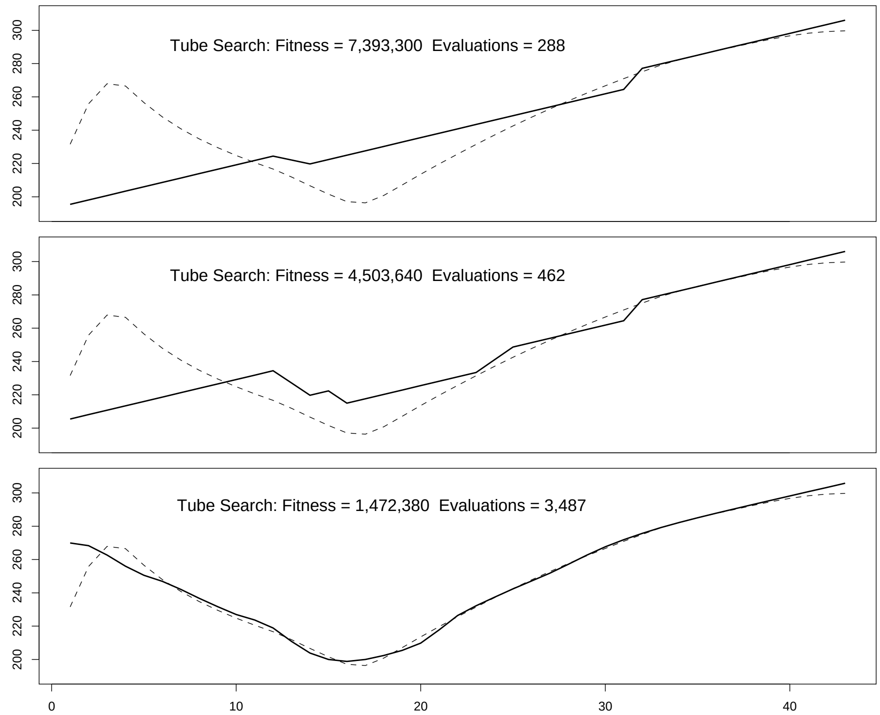

Fig. 6. Tube Search: The top two graphs show select iterations of the tube search. The bottom graph shows the final solution after 3,487 evaluations. Of all the profiles tested, this was the worst fit. The step distance for each parameter is exactly the same, so bias has no impact on tube search.

Figure 6 graphically explains the tube search and shows the final solution generated by searching the temperature problem. Note that, 43*2 evaluations are needed to evaluate the moves defined by the tube. Given the small number of moves used by the tube search, the total number of evaluations is less than half of that used by the the optimize and refine algorithm.

The error values associated with the move forming the *tube* around the current best solution will drive the search toward better points while maintaining smoothness. Because all of the parameters change at once, tube search is not a simple coordinate search scheme. Additionally, when each step is taken the magnitude of the step is the same, independent of the magnitude of the error. In this way, tube search ignores the bias in the evaluation function. Lower dimension can change just as much as higher dimension parameters, even when they have a smaller contribution to the error.

Tube search works surprisingly well on all temperature profiles we have optimized. Oddly enough, the errors associated with the tube search solutions are not particularly low–the errors are generally much lower for optimize and refine. Even CHC achieves lower errors. However, if we compute a sum-squared error (SSE) between the actual target temperature (which we don't have in the general case) and the tube search solution,

Profile	Tube Search		Optimize & Refine	
	Fitness	SSE	Fitness	SSE
Mid-latitude Summer	932,322	933	256,605	2,592
Mid-latitude Winter	743,194	738	298,684	3,342
Sub-arctic Summer	760,703	1,610	324,395	3,502
Sub-arctic Winter	1,092,430	348	314,486	1,383
Tropical Summer	1,664,570	1,189	399,106	1,423
Original Profile	1,472,380	1,950	314,486	1,370

Table 2. Sum-squared error (SSE) for the *optimize and refine* method and the *tube* search for all the McClatchey profiles we tested.

the fit between the tube search solutions and the actual profile is better, on average, than is achieved with other methods. Table 2 shows the *optimize and refine* method compared to the *tube* search method for all the McClatchey profiles we tested. The better objective fitness achieved in the *optimize and refine* algorithm do not imply a closer fit to the target solution. This may be because the other methods are more affected by bias in the evaluation function.

Tube search is also much faster than the other methods using fewer than 3,612 evaluations on all data sets. This is still not fast enough to allow for real-time evaluation. However, tube search has another attractive feature. Each of the 86 evaluations required to evaluate the moves defined by the tube around the current best solution are independent and can be done in parallel. This would allow us to use parallelism to speed up execution using a parallel implementation by a factor of 86. A parallel tube search can obtain a solution in the amount of time that is take to do $3,612/86 = 42$ sequence evaluations. This is a major advantage given the goal of doing real-time temperature inversion.

7 Conclusions

Temperature inversion is a practical example of an optimization problem that has not been efficiently solved using derivative-based search methods. Attempts to solve this problem using widely used evolutionary algorithms and local search methods produce poor results. Three algorithms were formally evaluated in this study, including CHC, a (30,210)ES and local search. We also applied PBIL and Differential Evolution to the temperature problem using approximately 100,000 evaluation and the results were similarly poor.

The temperature inversion problem highlights two problems that can cause a problem for optimization algorithms: bias and ridges. The *ridge problem* is relatively well documented in the mathematical literature on derivative free minimization algorithms [**?**] [**?**]. The benchmark test problem known as DeJong's F2 is also as Rosenbrock's "banana function" and was created by Rosenbrock around 1960 to illustrate the weakness of methods such as line search, which change only 1 variable at a time during search. Rosenbrock showed that even gradient methods *creep* or move very slowly on this function because the direction of the gradient significantly changes every time a small move is made in the search space. The ridge problem seems to be largely unexplored in the evolutionary algorithm community, except for the work of Salomon [**?**], Oyman et. al. [**?**], and Yuret et. al. [**?**]. It is a problem that needs far more attention.