Comparing Search Algorithms for 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 rapidly finds approximate solutions.

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 to 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 keep up with data collection.

Traditional gradient based methods can be used, but such methods are computationally costly [1]. 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 been applied to the temperature inversion problem on a more limited basis, including Population-Based Incremental Learning, or PBIL [2], and Differential Evolution [3]. All of these algorithms fail in similar ways.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 1340–1351, 2004.

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, 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 larger effect on the evaluation function than others.

An algorithm proposed by Salomon is tested that exploits known properties of temperature profiles and produces useful results [4]. Finally a 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

Atmospheric sciences researchers use 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 parameter indexed by $(44-k)$ in the profile is the estimated temperature at an altitude of approximately k kilometers in the atmosphere: the parameters are enumerated in reverse order, and 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).

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 model is "plane parallel" (e.g., with no horizontal variations in its properties). Radiances are calculated at a viewing angle θ as:

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

where $I_{(\tau,\mu)}$ = radiance

$\mu = cos(\theta)$

τ = optical depth

s = surface

$B_\nu(T)$ = Planck radiance for temperature T.

An analytical inversion of this model is impossible because radiances are non-linearly 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 [1]. When clouds or aerosols are present, the analytical calculation of these derivatives is impossible. In the simple model where only blue sky exists, success has been achieved using *Newtonian iteration* and a good starting guess. However, achieving a quadratic convergence rate for solutions near the optimum is highly dependent on a good a priori guess of the temperature profile, and search is still very costly. We attempt to improve computational efficiency by solving the inverse problem using non-derivative search methods.

3 Evolutionary Algorithms and Local Search

One of the more successful variants of genetic algorithms is CHC [5]. 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 (note that uniform crossover already randomly assigns non-matching bits). CHC also includes a restart mechanism that reinitializes the population by randomly flipping 35% of the bits of the best individual.

Evolution strategies 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. An 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. Self-adaptive strategies allow the angle of mutation to change. Correlated mutations attempt 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 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

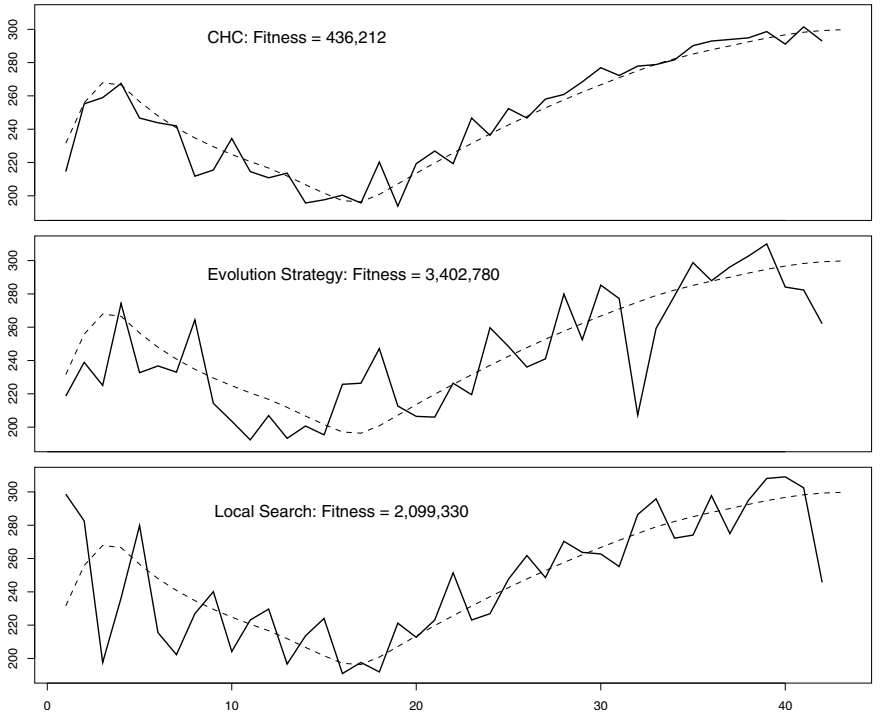

Fig. 1. The best solutions in 30 trials on a McClatchey *tropical* profile. The dashed line indicates the target *tropical* profile, and the solid black line is the best solution found by each algorithm. Even CHC's dominating performance finds a disappointing "zig-zag" solution. None of the solutions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

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 [6], which represent conditions ranging from subarctic winter 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 overhead is impractical; rotations were not used. Bäck and Schwefel [7] 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 the (30+210)ES and is reported as

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

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

the evolution strategy contribution in this paper. This implementation used the standard rules for adapting σ [7].

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

4 The Ridge Problem, Nonlinearity, and Bias

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

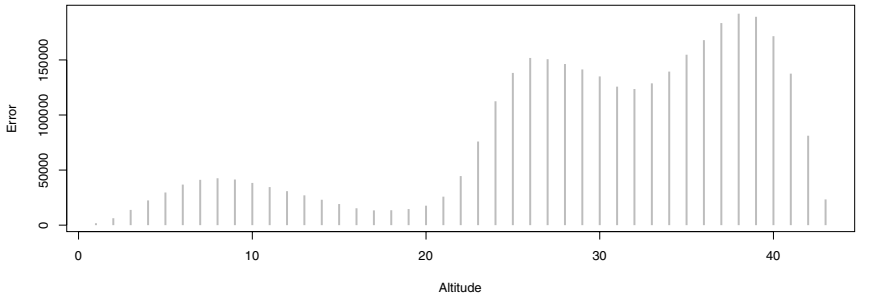

Fig. 2. The average error profile near the optimal solution. Higher dimension parameters contribute more to the error profile.

Rosenbrock was among the first to notice that search methods, including derivative-based methods such as *steepest descent*, are crippled by ridge features [8]. Winston also notes that ridges cause problems for simple hill climbers [9]. 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 when an algorithm is not stuck, convergence can be slowed 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 generally decreases the number of false optima. A lower precision search will get stuck on a ridge, blindly assuming it has found a local optima. Increasing precision allows more improving moves to be found, but it forces search algorithms to take smaller steps and move very slowly through the landscape. This causes an increase in evaluations and a much slower convergence.

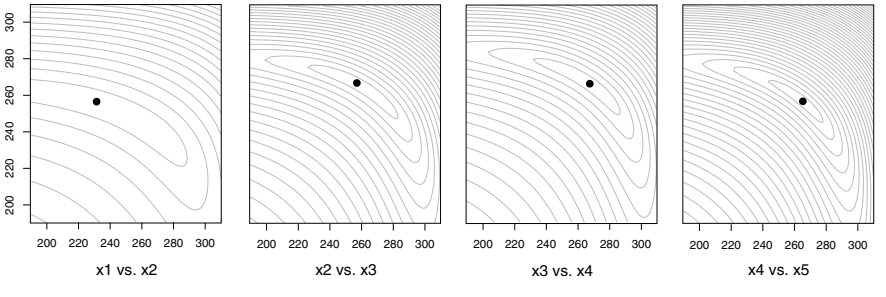

Fig. 3. Two dimensional contours of the first five parameters in the temperature problem. The black dots represent the optimal solution.

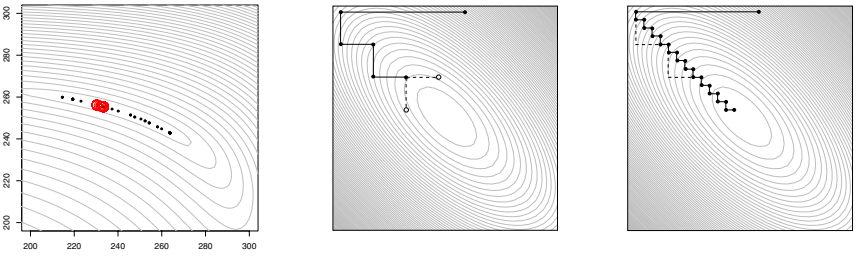

Fig. 4. In the leftmost graph, a high precision local search (large circles) finds the global optima, whereas the low precision search gets stuck in local optima (black dots). In the middle graph, a low precision search induces local optima on a simple parabolic ridge because all the neighbors (dashed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

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.

Local search uses a coordinate pattern to search for a globally competitive solution. Therefore, local search performs poorly in the presence of ridges.

Salomon [10] 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 axes.

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 extend to all genetic algorithms that use crossover. CHC uses a variation of uniform crossover that changes many parameters at once. Nevertheless, CHC does use a fixed coordinate system. Our results indicate that, in fact, CHC performs better than evolution strategies 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, Oy-

man et al. [11] define 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 if an improvement is found. The "1/5 rule" was used, which means that the step size is adjusted to produce an improving move one out of every five tries. When the parent encounters a ridge, the step size will 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. Thus, evolution strategies can also *creep*.

5 Optimize and Refine

Salomon [4] also notes that some search algorithms produce results that "zig-zag" the actual solution when the desired solutions displays physical smoothness. 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.

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 piecewise 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 *subarctic 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.

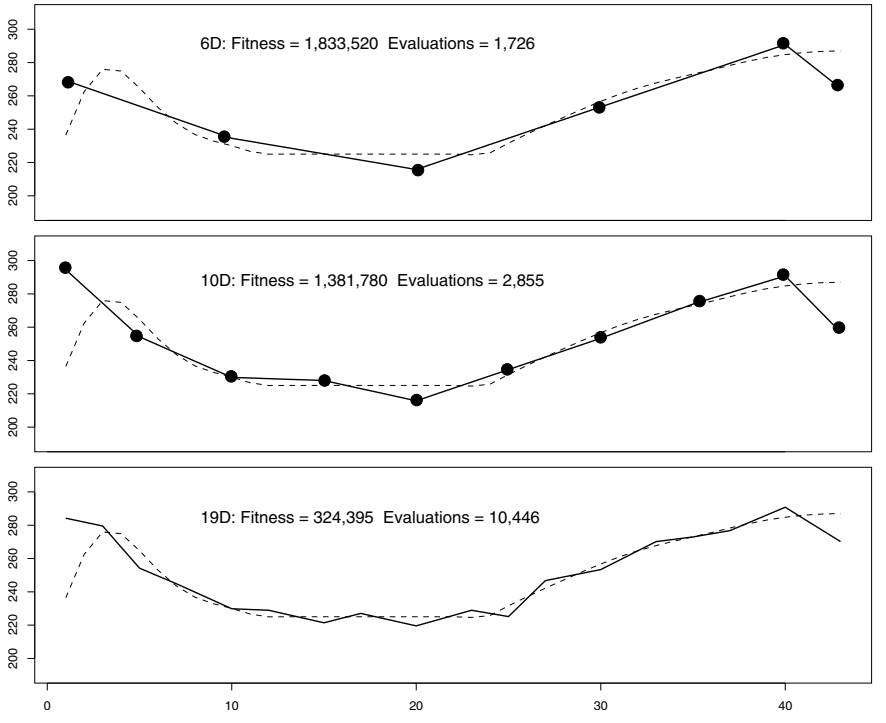

Fig. 5. Optimize and Refine. A fully convergent *subarctic summer* solution required an average of 10,446 evaluations. Although this solution fits the profile better than previous methods, it still wanders the target solution.

6 A New Algorithm: Tube Search

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 algorithm called *tube search*. Like optimize and refine, tube search 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. Each parameter, except the first and last two end points, is replaced by the average of itself and its two neighbors.

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

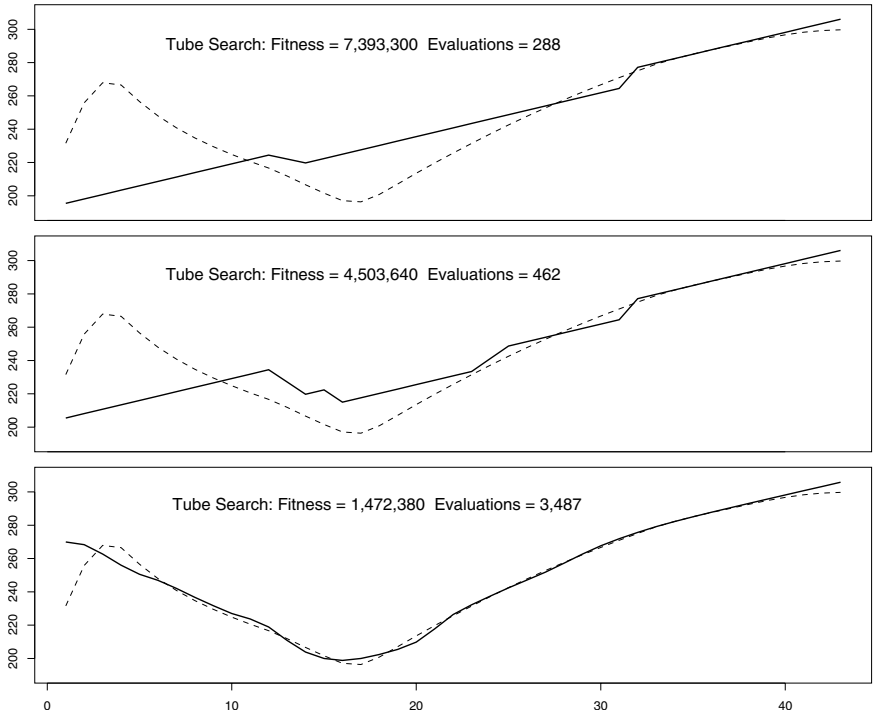

Fig. 6. Tube Search: The top two graphs show select iterations of the tube search. The bottom graph shows the final solution after 3,487 evaluations. Of all the profiles tested, this was the worst fit. The step distance for each parameter is exactly the same, so bias has no impact on tube search.

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 parameters 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

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

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

error (SSE) between the actual target temperature (which we don't have in the general case) and the tube search solution, 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 does 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 Tube Search by a factor of 86. Parallel tube search could obtain a solution in the amount of time taken 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 100,000 evaluations and the results were similarly poor. Methods that exploit the smoothness of the temperature profile are more effective and, in the case of the tube search, more efficient. Other types of smoothing, such as splines, may be a useful addition to the *tube search*, as well as other evolutionary algorithms.

The temperature inversion application highlights two difficulties 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 [8] [12]. The ridge problem seems to be largely unexplored in the genetic algorithm community, but has received attention in the evolution strategies community [10] [11] [13]. Recently, we have begun looking at the Covariance Matrix Adaptation method [14] [15] for rotating the representation space; on test functions it is highly effective, but it has not been tested on the temperature inversion problem.

Acknowledgments. This work was supported by National Science Foundation grant IIS-0117209.

References

1. R.J. Englen, A.S. Denning, K. Gurney, and G.L. Stephens. Global observations of the carbon budget: I. expected satellite capabilities for emission spectroscopy in the eos and npoess eras. *J. of Geophysical Research*, 106:20,055–20,068, 2001.
2. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. *The Int. Conf. on Machine Learning*, 38–46, 1995. Morgan Kaufmann.
3. R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341 – 359, 1997.
4. R. Salomon. Applying evolutionary algorithms to real-world-inspired problems with physical smoothness constraints. *Proceedings Congress on Evolutionary Computation*, 2:921–928, 1999. IEEE Press.
5. L. J. Eshelman. The CHC adaptive search algorithm. *Foundations of Genetic Algorithms*, 265–283. Morgan Kaufmann, 1991.
6. R.A. McClatchey, R.W. Senn, J.E.A. Feldy, S.E. Voltz, and J.S. Garing. Optical properties of the atmosphere. Technical Report TR-354, AFCRL, 1971.
7. T. Back and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.
8. H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3:175–184, 1960.
9. P. Winston. *Artifical Intelligence (2nd ed.)*. Addison-Wesley, 1984.
10. R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. *BioSystems*, 39:263–278, 1996.
11. A. I. Oyman, H. Beyer, and H. Schwefel. Where elitists start limping evolution strategies at ridge functions. *Parallel Problem Solving from Nature – PPSN V*, 34–43, 1998. Springer.
12. R. P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, New Jersey, 2002.
13. D. Yuret and M. Maza. Dynamic hillclimbing. *Second Turkish Symposium on Artificial Intelligence and Neural Networks*, 208–212, 1993.
14. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
15. N. Hansen, S. Müller and P. Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.