Ruffled by Ridges: How Evolutionary Algorithms can fail

Darrell Whitley and Monte Lunacek

Computer Science, Colorado State University, Fort Collins, CO 80523

Abstract. The representations currently used by local search and some evolutionary algorithms have the disadvantage that these algorithms are partially blind to "ridges" in the search space. Both heuristics search and gradient search algorithms can exhibit extremely slow convergence on functions that display ridge structures. A class of rotated representations are proposed; these rotated representations can be based on Principal Components Analysis, or use the Gram-Schmidt orthogonalization method. Preliminary experiments show that local search using a rotated representation is able to align the coordinates of the search with ridge structures. Search using a rotated representation can convergence are as much as 1000 times faster than local search using a fixed coordinate representation.

1 Introduction

Various experiments reported here and by others show that various genetic algorithms and local search methods are more or less blind to "ridges" in the search space of parameter optimization problems. In two dimensions, the ridge problem is essentially this: a method that searches north, south, east and west will not see improving moves that are oriented at a 45 degree angle in the search space.

The *ridge problem* is relatively well documented in the mathematical literature on derivative free mimization algorithms [1, 2]. However, there is little discussion of this problem in the heuristic search literature. One reason may be because many applications of heuristic search have focused on combinatorial optimization problems such as scheduling, the TSP, graph partitioning and routing problems: it is difficult to say what it would mean for there to be "ridges" in these search spaces. On the other hand, genetic algorithms have a strong connection with parameter optimization. Holland's original "schema theory" makes much more sense when viewed in terms of parameter optimization problems using bit encodings [3]. Our current exploration of the "ridge problem" was motivated by three concerns.

First, over the last few years experiments have shown that genetic algorithms are more sensitive to local optima induced by different bit representations than was previously believed. Until recently, much of this work has focused on how representations such as Gray codes destroy local optima [4]. Our works focuses on when Gray codes *create* new optima: we can prove this happens only along *ridges*.

Second, why do some seemingly unintelligent algorithms seem to do well on certain benchmarks? Some algorithms have semi-random behaviors that don't seem to make any sense from either a theoretical or intuitive perspective. What we now believe is that these "weaker" algorithms are sometimes better able to avoid becoming trapped on ridges. If so, this says more about the heuristic search communities' lack of understanding of ridge problems than about intelligent algorithm design.

Third, we have been working on real world applications for computing inverses for prediction problems in weather and geophysics. These problems seem to be a good application for heuristic search. Traditional gradient based methods can be used, but the gradients are extremely costly to compute and the search is extremely slow (e.g. hours). However, we have found that genetic algorithms and evolution strategies do not work on these inverse problems. We now know that there are ridges in these search spaces that induce "false" optima in the representation spaces, or ridges otherwise slow down the progress of search.

2 Local Optima and Ridges under Gray Encoding

Let $\Omega = \{0, 1, \ldots, 2^\ell - 1\}$ be the search space which can be mapped onto a hypercube. Elements $x, y \in \Omega$ are *neighbors* when (x, y) is an edge in the hypercube. Bit climbing search algorithms terminate at a *local optimum*, denoted by $x \in \Omega$, such that none of the points in the neighborhood $N(x)$ improve upon x when evaluated by some objective function. Of course, the neighborhood structure of a problem depends upon the coding scheme used. Gray codes are often used for bit representations because, by definition, adjacent integers are adjacent neighbors.

Suppose the objective function is defined on the unit interval $0 \leq x < 1$. To optimize this function we discretize the interval by selecting n points. The *natural encoding* is then a map from Ω to the graph that has edges between points x and $x + 1$ for all $x = 0, 1, \ldots, n - 2$.

Under a Gray encoding adjacent integers have bit representations that are Hamming distance 1 neighbors (e.g. 3 = 010, 4 = 110). Thus a Gray encoding has the following nice property.

Theorem 1. *A function $f : \Omega \to \mathbb{R}$ cannot have more local optima under Gray encoding than it does under the natural encoding.*

A proof first appears in Whitley and Rana [5]. This theorem states that when using a Gray code, local optima of the objective function considered as a function on the unit interval can be destroyed, but no new local optima can be created. In particular:

Corollary: *If a function is unimodal under the natural encoding, it is unimodal under Gray code.*

But are there unimodal functions where the natural encoding is multimodal? If the function is 1-dimensional, the answer is no. But if the function is not 1-dimensional, the answer is yes. "False" local optima are induced on ridges.

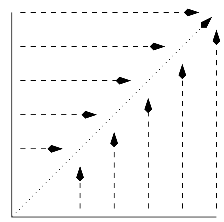

Fig. 1. Local search moves only in the horizontal and vertical directions. It therefore "finds" the diagonal, but becomes stuck there. Every point on the diagonal is locally optimal. Local search is blind to the fact that there is gradient information moving along the diagonal.

A simplified representation of a *ridge* problem appears in Figure 1. Changing one variable at a time will move local search to the diagonal. However, looking in either the x-dimension or the y-dimension, every point along the diagonal appears to be a local optimum. There is actually gradient information if one looks *along* the diagonal; however, this requires either 1) changing both variables at once, or 2) transforming the coordinate system of the search space so as to "expose" the gradient information.

This limitation is not unique to local search, and it is not absolute for genetic algorithms. Any method that searches 1-dimension at a time has the same limitation, including local search as well as simple "line search" methods. Any fixed neighborhood scheme has a similar problem. For the Traveling Salesman Problem, for example, 2-opt can be stuck in a local optimum which can then be escaped using 3-opt [6].

A genetic algorithm is not absolutely trapped by ridges. Early population sampling of schema can potentially allow the search to avoid being trapped by "ridges". But it is also well known that genetic algorithms quickly lose diversity and then the search must use mutation or otherwise random jumps to move along the ridge. For example, simple 1-point crossover inherits "fixed but mixed" parameters from parents for the most part. That is, the inherited parameters come directly from the parents without changes except for one parameter that is broken by crossover. Uniform crossover would seem to have the ability to move along ridges: every bit is independently inherited from the 2 parent structures. But Syswerda points out [7] that uniform crossover has the following behavior: bits that are common between the two parents are inherited, and all non-common bits are *randomly reset* because 0 or 1 is randomly inherited. So the ability of uniform crossover to move along ridges may be no better than that of random mutation.

Salomon [8] showed that most benchmarks become much more difficult when the problems are *rotated*. Searching a simple 2-D eliptical bowl is optimally solved by one iteration of line search when the elipse is oriented with the x and y axis. But when the space is rotated 45 degrees, the bowl become a ridge and the search problem is more difficult for many search algorithms. Salomon also showed that some genetic algorithms, such as the Breeder Genetic Algorithm [9] had been tuned to behave much like line search, largely moving in one dimension at a time; this allows $\mathcal{O}(N\ ln\ N)$ convergence proofs using the assumption that the search problem is decomposable into N subproblems and solved in a piecewise fashion [9].

And of course, hybrid genetic algorithms often rely on local search to improve the populations–and most local search methods are very prone to being trapped on ridges.

Salomon points out that *Evolution Strategies* are invariant under rotation. But being invariant under rotation and being able to exploit ridge structures is not quite the same. Oyman, Beyer and Schwefel [10] showed that Evolution Strategies also "creep" on ridge functions. The problem occurred when a (1+10)ES was used for a simple parabolic ridge problem with a 1/5 rule to adjust the step size. Oyman et al. analytically show that longer jumps in the search space result in poorer evaluation near the ridge. Thus, the adaptive mutation mechanism reduces the step size, until finally the algorithm also creeps along the ridge. Similar problems with self-adaptive mutations have been noted by Mathias et al. [11].

For such widely used technologies as evolutionary algorithms and local search, this is a serious limitation.

2.1 Benchmarks, Ridges and Direct Search Methods

Common benchmarks contain a couple of problems with "ridges" features. Figure 2 shows 2-D illustrations of 2 benchmarks, one of which is the common F2 from the De Jong test suite [12]. F2 is relatively easy, while the "Rana" function is difficult. F2 is also known as Rosenbrock's "banana function." F2 was created specifically 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.

Rosenbrock proposed a search method that uses the Gram-Schmidt orthogonalization algorithm to adapt the search coordinate system. Later Powell introduced the method of conjugate directions for search methods without derivatives [2]. Ultimately, the Nelder-Mead Simplex method was introduced [13]; it is probably the best known direct search method that does not use derivatives. All of these methods use simple heuristics to allow search to move in arbitrary directions in the search space. These methods often compute a direction of improvement based on a sample of points; then, line-search type methods are often used to look for improving moves. In theory, these methods should be able to follow ridge structure *if* they select the correct direction. The potential disadvantage of these methods is that they heuristically compute a direction based on very little information.

Even the best of these direct search methods, such as Nelder-Mead [13], have been shown to be inferior to methods such as Evolution Strategies on several test problems. Theory suggests that population based forms of search can filter out regions that are less likely to be productive places for search.

One of the fundamental problems that is encountered when trying to compare direct search methods, local search methods, and even different evolutionary algorithms, is the representation and precision used for constructing the search space.

Genetic algorithms and local search (e.g.: RBC: the Random Bit Climber [14]) tend to use low precision bit encodings. Evolution Strategies and direct search methods such as Nelder-Mead use high precision, real-valued representations. The search community has long struggled with the debate over which is better, bit representations or real-valued representations? Unfortunately, different experiments seem to support different conclusions, even when compared on the same test functions. This seems odd and confusing.

Recent work suggests that the choice of real-valued versus bit encodings may not be nearly as important as the level of precision. Precision can dramatically change the rate of convergence. One of the potential advantages of using bit encoding is that they can use lower precision for many applications and achieve faster convergence compared to real-valued encodings. This also makes comparing real-valued representations and bit encoded representations difficult. However, forcing the bit encodings and real-valued encoding to use 32 bit precision just to make them the same is probably not a reasonable solution: precision matters.

2.2 Local Search, Ridges and Precision

We ran a Steepest Ascent Bit Climber on the F2 and Rana test problems at 10 and 20 bits. Results are shown in table 1. The number of steps required to reach a local optimum jumps from about 200 steps under local search at 10 bits of precision to 200,000 or more steps at 20 bits of precision. These numbers have been confirmed by hand-traces of the search.

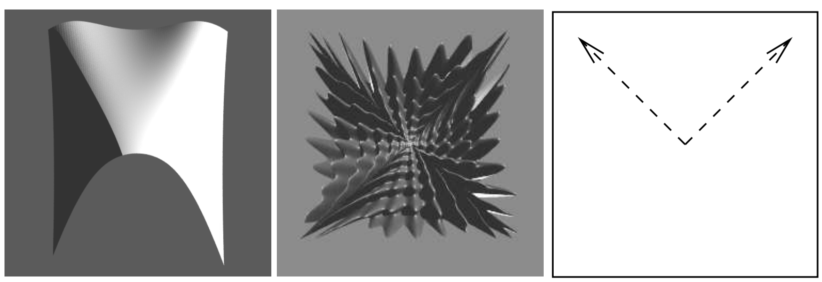

Fig. 2. The leftmost figure is F2 from the De Jong Test Suite, also known as Rosenbrock's banana function. The middle figure is F102, or Rana's function. The rightmost figure is a cartoon showing the "ridges" that lead to the global optimum as well as other competitive local optima.

Function	Precision	Mean	Std	Steps	Std
F2	10-bits	0.001	0.002	235	30
F2	20-bits	4×10^{-7}	1×10^{-7}	2×10^{5}	4×10^{3}
Rana	10-bits	-501.9	06.0	225	22
Rana	20-bits	-503.0	04.8	3×10^{6}	8×10^{3}

Table 1. Results of steepest ascent bit climbing with 100 restarts at 10 and 20 bit resolution on 2-D problems. Results are averaged over 30 runs. Mean is calculated using the best of the 100 restarts. Steps is the average number of steps needed to reach a local optimum.

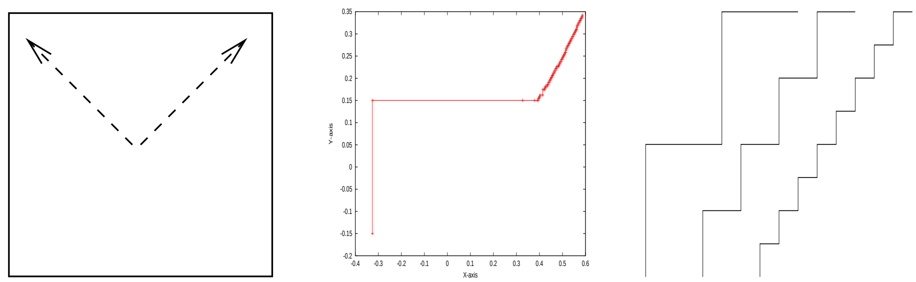

Fig. 3. The “arrows” figure shows the approximate location of the ridges. The line in the middle figure tracks the movement of an actual run of local search on F2. After 2 moves the ridge is encountered. The rightmost figure: adding 1 bit of precision doubles the number of steps.

As shown in Figure 2, both of these functions have ridge like properties. Search must move toward the upper corners to find the global optimum, as well as the better local optima. The behavior of local search on F2 is shown in Figure 3. The first south to north move occurs in 1 step. The second west to east move occurs in 1 step. The ridge is then encountered after 2 moves. Because the ridge is not exactly at 45 degrees, local search is not completely blind and does not stop. Instead, the ridge becomes a “staircase”. Local search makes the smallest move possible and therefore “creeps.” The problem is exponentially worse at high precision because the steps of the staircase are exponentially smaller.

Genetic algorithms are very often used at 10 bits of precision. We have tested various genetic algorithms at 20 bits of precision across a number of common benchmarks. Genetic algorithms can be 10 to 100 times slower to converge using 20 versus 10 bits of precision. This may in part be due to the failure of genetic algorithms to exploit ridges in the search space.

3 Ridges and Temperature Inversion

The temperature inversion problem is a real world application in atmospheric science that involves searching for 43 temperatures which produce a set of radiance observations using a forward model.

$$\textbf{MODEL(temperature.vector)} \longrightarrow \textbf{radiance.observations}$$

The k^{th} variable is the temperature at an approximate altitude of k kilometers (this spacing is somewhat greater at higher altitudes).

Figure 4 is a 2-D slice taken from the 43-dimensional temperature inversion problem in the area around the global optimum. While the space is relatively smooth, there is a distinct ridge. The companion image shows the location of “false” local optima: there is no improving move in the x or y dimension of the space. As can be seen, there are a surprising number of false local optima that will trap local search and which make search difficult. We have shown that such ridges occur *throughout* the search space.

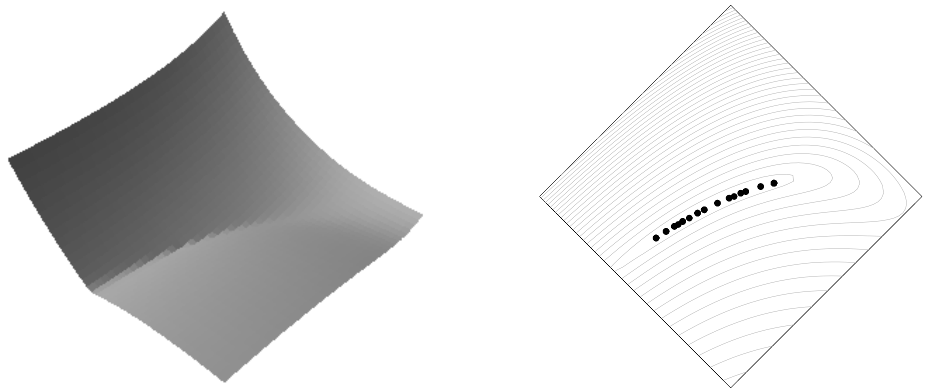

Fig. 4. On the left is a 2-D slice near the global optimum of the 43-D temperature inversion problem. The points in the rightmost figure shown where "false" local optima occur under Gray code due to the effect of the ridge in the search space.

These ridge structures represent the nonlinear interaction between variables: changing a parameter in dimension x simultaneously changes the location of the ridge in many other dimensions. In carefully studying the temperature inversion problem, we have also found that different parameters (i.e., potential temperatures at the different altitudes) have very different effects on the error functions: some changes in temperature affect the error slightly, while others have an impact 100 times larger. These factors all work together to produce a difficult nonlinear evaluation function with "false" local optima.

Our results show that evolutionary algorithms are trapped in these local optima at about the same rate as local search. Evolutionary algorithms will easily escape local optima that are not competitive with the best members of a population; but evolutionary algorithms are not guaranteed to escape all local optima. We have applied a Steepest Ascent Bit Climber, the CHC algorithm, a (30,210)ES and a (30+210)ES with standard mutation adaptations and no rotations, PBIL and Differential Evolution. All of these algorithms failed to solve the temperature inversion problem and failed in similar ways.

4 Rotating Search Coordinates with PCA and Gram-Schmidt

One way to deal with ridges in the search space is to change the search coordinates. One way to do this is to use a rotated representation. Evolution Strategies which use a heuristic rotation of the search space [15] by adapting a set of rotation strategy parameters via evolution. However, adapting rotation "strategy" parameters of the form used by Evolution Strategies is too imprecise and impractical for large problems: for the temperature problem, there would be 43 temperatures as object variables, 43 step size strategy parameters and 903 rotation strategy parameters.

We have results using two different forms of local search that includes a more formal rotation mechanism for changing the search coordinates.

A standard way of computing a rotation is to use Principal Component Analysis. Given a data set of sample points, an eigenvalue/eigenvector decomposition is per-

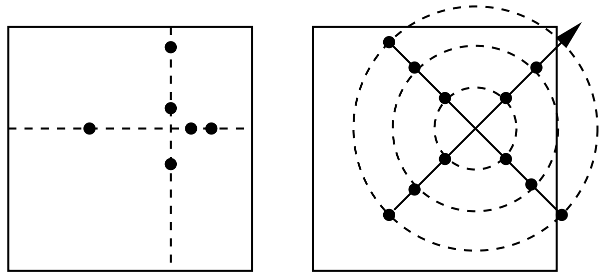

Fig. 5. Under Gray code, search can only move north-south east-west, the neighborhood is not symmetric (leftmost figure) and sampling is highly localized. "GS Ring Search" rotates the search coordinates; the neighborhood is symmetric and neighbors are more uniformly spaced.

formed. The eigenvectors are represented by a rotation matrix **R**. Let Λ be the diagonal eigenvalue matrix. Let **X** represent a matrix of data vectors. Using PCA we find **R** and Λ such that

$$\mathbf{R} \cdot \mathbf{X}\mathbf{X}^T = \Lambda \mathbf{R}$$

For a single search point represented by the vector **x** we compute **xR**, which is the projection of the point **x** into the space defined by **R**. The rotation matrix is orthonormal, so a simple correction is also needed to translate and re-center the rotation during search. The rotation is general purpose, and we can define a neighbor as a layered hypersphere rotated about the centroid, or we can rotate the neighbors of a reflected Gray neighborhood (e.g. see Figure 5).

If we want to find a structure such as a ridge, one way to do this with PCA is to sample locally, then isolate a subset of the better sample points. For example, sample 20 points and then apply PCA analysis to the 10 best solutions. Or we can use all 20 points, but weight them by solution quality. While this can give a correct rotation, the direction of maximal variance might be in the direction of the gradient if the samples are on a ridge, or the maximal variance may be orthogonal to the gradient if the sample is drawn from a large gently sloping plateau. Still, it is one of the best formal methods to determine rotation.

Another approach is to use the Gram-Schmidt (GS) orthogonalization algorithm to rotate the space. Often the Gram-Schidt algorithm is used to construct a representation that is orthogonal with respect to two points in the search space–such as the best two points seen so far. This is a more localized and heuristic way of determining a useful "rotation" for changing the problem representation.

We implemented a 2-D version of Gram-Schmidt to create an algorithm we call "GS Ring Search." The space is rotated to align with the best two solutions seen so far. A symmetric neighborhood was used instead of a Hamming neighborhood. This is illustrated in the cartoon shown in Figure 5. We tested "GS Ring Search" on F2 and Rana's functions in two dimensions. The results in Table 2 are compared to Steepest Ascent Bit Climbing (SABC) with a Gray Code representation. We also searched these functions using SABC after using PCA to rotate the search space. For the PCA, 15

Functions	Search	Best	Mean	Std	Steps	Std	Evals	Std
F2	SABC	+4.5E-07	+5.4E-07	+1.2E-07	6,193	814	247,710	32,541
	GS Ring Search	+1.1E-08	+5.8E-04	+7.3E-04	259	355	15,606	21,287
	PCA SABC	+3.1E-10	+2.5E-07	+2.5E-07	138	58	7,603	3,189
Rana	SABC	-510	-417	87	208	321	8,305	12,822
	GS Ring Search	-512	-476	28	18	24	1,148	1,443
	PCA SABC	-511	-480	24	23	6	1,262	341

Table 2. Results of steepest ascent bit climbing (SABC) and two rotated local search methods. No restarts were used in these experiments. **Mean** is the mean best over 30 experiments, and best is the best of the 30 experiments.

points were sampled, with PCA applied to the best 8. The speed-up is *dramatic* using rotated representations.

Both GS Ring Search and SABC using a PCA rotated representation dramatically speeds up search in two dimensions. The "ring" search neighborhood we constructed uses 15 concentric rings that are uniformly spaced. This provides more neighbors (30 instead of 20 per dimension) but less precision at the smallest step size. We tried 10 and 20 rings, with 10 rings providing somewhat poorer solutions but being very fast, and with 20 rings providing very good solutions, but requiring more steps and evaluations. This was done in part to see if a different neighborhood structure would impact the results. Hamming neighborhoods are such that some points in the search space have far more neighbors in one direction compared to another–it all depends on how the bits code to integers. Ring search more uniformly samples the search space. Future work needs to explore these questions. Given there is no clear advantage to the ring neighborhood we used the Gray code neighbhor in subsequent experiments.

5 Constructing Higher Dimensional Test Problems

Functions with two variables are often scaled to higher dimensions using an *expansion* function. Different expansion functions have different properties. Sometimes subfunctions are added together with no interactions between subfunctions. The problem here is the linearly combination of subfunctions results in a test problem that is *separable* when there are no interfunction interactions between the parameters. Specifically, it is possible to optimize the each component (e.g., $f(x_1, x_2)$) independently of the other parameter values.

Expansion functions that create *non-separable* problems are often generalized in the following way:

$$\text{Expansion Method 1:} \quad f(x) = \sum_{i=1}^{n-1} f(x_i, x_{i+1})$$

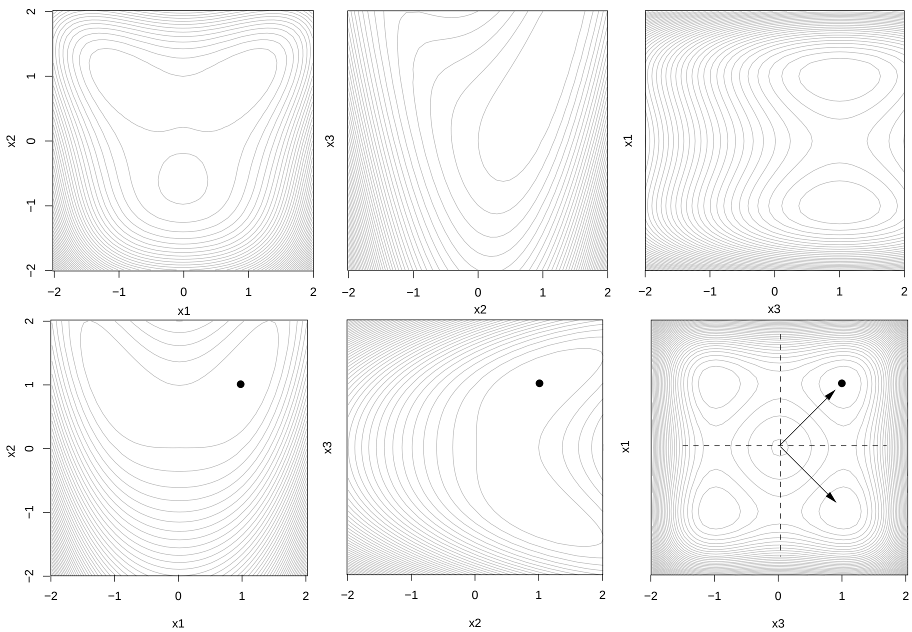

Fig. 6. Results of expanding Rosenbrock's function in three dimensions using Expansion Method 1 are shown in the top 3 slices of the space: with the possible exception of the first slice, the search space is simpler and easier than the original problem. Results of expanding Rosenbrock's function in three dimensions using Expansion Method 2 are shown in the bottom 3 slices of the space: the first two slices retain the long narrow ridge and the third slice is a multimodal surface.

However, the symmetry that exists in this function can make some problems easier to solve in higher dimensions. Figure 6 (topmost) shows the two dimensional slices taken from the three dimensional Rosenbrock function when generalized in this way. With the exception of the first slice, the other surfaces aren't as difficult as the original landscape proposed by Rosenbrock. In fact, local search finds the optimum rather quickly under this transformation.

In order to retain the original ridge structure, the following expansion function was used:

$$\text{Expansion Method 2:} \quad f(x) = \sum_{i=1}^{\lfloor n/2 \rfloor} f(x_{2i-1}, x_{2i}) + \sum_{i=1}^{\lfloor (n-1)/2 \rfloor} f(x_{2i+1}, x_{2i})$$

This effectively creates a *non-separable*, higher dimension problem that preserves the ridge features of the original problem. Figure 6 (bottom) illustrates using Expansion Method 2 to create the three dimensional Rosenbrock function. The first two slices retain the long narrow ridge that characterizes the Rosenbrock function. The interaction between parameters one and three creates a multi-modal surface from which it is difficult for a coordinate strategy to escape.

Function	Search	Best	Mean	Std	Steps	Std	Evals	Std
F2	SABC	2.4E-06	2.4E-06	8.1E-08	11,663	1,415	1,166,268	141,503
	PCA SABC	5.3E-07	2.4E-06	1.3E-06	1,057	177	148,042	24,800
Rana	SABC	-386	-313	49	506	915	50,551	91,540
	PCA SABC	-399	-310	42	112	167	15,662	23,318

(a) Five Dimension Results

Function	Search	Best	Mean	Std	Steps	Std	Evals	Std
F2	SABC	5.9E-06	6.1E-06	1.3E-07	29,437	1,865	5,887,321	372,921
	PCA SABC	3.8E-06	5.9E-06	2.2E-06	8,915	406	2,496,201	113,735
Rana	SABC	-427	-354	40	758	940	151,628	187,959
	PCA SABC	-411	-308	47	525	632	146,917	176,958
Temp	SABC	11,607	16,026	2,497	373	64	41,064	7,092
	PCA SABC	5,353	10,121	2,910	109	36	20,100	6,722

(b) Ten Dimension Results

Table 3. The results of applying local search SABC with and without PCA rotated representations on 5-D and 10-D versions of the Rana and F2 functions using Expansion Method 2. At 5-D, PCA used the best half of 40 samples; at 10-D PCA used the best half of either 80 or 74 samples to compute rotations.

This same pattern extends to slices of 5 and 10 dimensional functions contructed using Expansion Method 2: more of the features that make the primitive 2-D function difficult are preserved in the expanded functions.

We next applied the local search SABC algorithm with and without a PCA rotated representation to 5-D and 10-D versions of Rosenbrock's banana function (F2) and the Rana function. We also ran a 10-D temperature inversion problem looking for temperature for the first 10 kilometers of the atmosphere. PCA was applied after every step, which adds to the number of evaluations. The number of steps taken during search is 5 to 10 times less under SABC with PCA compared to the non-rotated representations on the 5-D problems. The number of step is 2 to 3 times less under SABC with PCA compared to the non-rotated representations on the 10-D problems. Using a rotated representation is more effective on F2 than Rana. This may be because the ridge is the only thing than makes F2 difficult, while Rana is also extremely multimodal. There is a clear advantage using PCA on the 5-D problems; the advantage is less clear at 10-D for Rana. For the 10-D temperature problem, using PCA significantly reduces both the error and the number of evaluations.

6 Discussion and Future Directions

Our results and the related literature [10] [8] indicate that rotated ridge structures can be a serious problem for both local search and evolutionary algorithms. Using rotated representations can potentially help. The relative advantages of using a localized Gram-Schmidt orthogonalization algorithms versus PCA for computing rotations need to be explored. It is likely that PCA only needs to be run occasionally, not after every step.

Another key question raised by this work concerns the use of strategy parameters in Evolution Strategies for computing heuristic rotations. To use rotations, an ES encodes $\mathcal{O}(N^2)$ rotation or covariance parameters onto the chromosome to be evolved along with the N object parameters. How does this compare to using PCA derived rotation parameters? Using a $(\lambda + \mu)$ES or (λ, μ)ES, a population of μ parents generates λ offspring. Typically, $\lambda > \mu$. The set of λ offspring represents a natural unit for applying PCA, so that no additional sampling is needed. After a set of offspring are generated and evaluated, rotation and (potentially) variance information can be computed and used to generate the next set of offspring. Strategy parameters for rotations are no longer needed. This could produce a fundamental change in the theory and application of Evolution Strategies.

References

1. Rosenbrock, H.: An automatic method for finding the greatest or least value of a function. Computer Journal **3** (1960) 175–184
2. Brent, R.: Algorithms for Minization with Derivatives. Dover (1973)
3. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
4. Whitley, D., Barbulescu, L., Watson, J.: Local Search and High Precision Gray Codes. In: Foundations of Genetic Algorithms FOGA-6, Morgan Kaufmann (2001)
5. Rana, S., Whitley, D.: Representations, Search and Local Optima. In: Proceedings of the 14th National Conference on Artificial Intelligence AAAI-97, MIT Press (1997) 497–502
6. Lin, S., Kernighan, B.: An Efficient Heuristic Procedure for the Traveling Salesman Problems. Operations Research **21** (1973) 498–516
7. Syswerda, G.: Simulated Crossover in Genetic Algorithms. In Whitley, D., ed.: FOGA - 2, Morgan Kaufmann (1993) 239–255
8. Salomon, R.: Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions. Biosystems **39(3)** (1960) 263–278
9. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm. Journal of Evolutionary Computation **1** (1993) 25–49
10. Oyman, A., Schwefel, H.B.H.: Where elitists start limping: Evolution strategies at ridge functions. In Eiben, G., Bäck, T., Schoenauer, M., Schwefel, H.P., eds.: ppsn5, Springer-Verlag (1998) 34–43
11. Mathias, K., Schaffer, J., Eshelman, L., Mani, M.: The effects of control parameters and restarts on search stagnation in evolutionary programming. In Eiben, G., Bäck, T., Schoenauer, M., Schwefel, H.P., eds.: ppsn5, Springer-Verlag (1998) 398–407
12. DeJong, K.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, Department of Computer and Communication Sciences, Ann Arbor, Michigan (1975)

13. Nelder, J., Mead, R.: A simplex method for function minimization. Computer Journal **7** (1965) 308–313
14. Davis, L.: Bit-Climbing, Representational Bias, and Test Suite Design. In Booker, L., Belew, R., eds.: Proc. of the 4th Int'l. Conf. on GAs, Morgan Kaufmann (1991) 18–23
15. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press (1996)