Ruffled by Ridges: How Evolutionary Algorithms Can Fail

Darrell Whitley, Monte Lunacek, and James Knight

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 and explored; these rotated representations can be based on Principal Components Analysis, or use the Gram-Schmidt orthogonalization method. Some algorithms, such as CMA-ES, already make use of similar rotated representations.

1 Introduction

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 minimization algorithms [1,2]. However, there is little discussion of this problem in the heuristic search literature. Our 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 [3]. Our work focuses on when Gray codes *create* new optima: this happens only along *ridges*.

Second, some algorithms have semi-random behaviors that don't seem to make any sense from either a theoretical or intuitive perspective, and yet they work relatively well on certain benchmarks. What we now believe is that these "weaker" algorithms are sometimes better able to avoid becoming trapped on ridges.

Third, we have been working on real world applications for computing inverses for prediction problems in weather and geophysics. 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.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 294–306, 2004.
© Springer-Verlag Berlin Heidelberg 2004

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. Gray codes are often used for bit representations because, by definition, adjacent integers are adjacent neighbors.

Let the objective function be defined on the unit interval $0 \leq x < 1$. 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 which results in the following 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.*

Under 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 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.

A simplified *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.

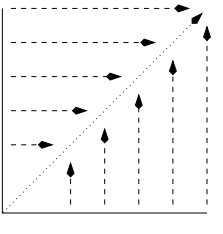

Fig. 1. Local search moves only in the horizontal and vertical directions. It therefore "finds" the diagonal, but becomes stuck there. Every point on the diagonal is locally optimal. Local search is blind to the fact that there is gradient information moving along the diagonal.

A genetic algorithm is not absolutely trapped by ridges. Early population sampling of schema can allow the search to avoid being trapped by "ridges." But 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 [5] points out that when using uniform crossover, 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.

Kazadi motivated a representation for genetic algorithms called a conjugate schema. Kazadi asserts that sometimes crossover disrupts the efficiency of a real-valued genetic algorithm by producing low fitness offspring. Kazadi proposed a new basis, called a conjugate schema, that minimizes the functional dependencies between the parameters of the objective. This basis creates local separability in the objective function and, hopefully, allows parents to cross with a higher likelihood of strong children. Kazadi uses the eigenvectors of the absolute Hessian matrix to find optimal basis. As noted by the author, "this adaptation is curiously successful on our test functions" [6]. One practical problem is that Hessians require twice differential functions that are difficult to compute.

Wyatt and Lipson [7] take a more "linkage" theoretic view of the crossover problem to arrive at similar conclusions. *Genetic linkage* is a measure of the correlation between functional dependency and gene location. Representations with high linkage will push separable parts of the problem together, and these parts will become useful building blocks for genetic search. In order to find an optimal gene ordering, Wyatt and Lipson also use the eigenvectors of the Hessian around the current best point, and use this ordering for the entire population.

Neither of these studies were concerned with mutation or the direction of the gradient. The proposed ideas may indeed be useful, but these studies take a very narrow view of how the representation is being changed. However, work starting with Salomon, and more recently work on the CMA-ES algorithm, focuses on rotations and concerns about search direction and step size.

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 ellipse is oriented with the x and y axis. But when the space is rotated 45 degrees (or my some random value), the bowl becomes a ridge and the search problem is more difficult for many search algorithms. Salomon 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].

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 et al. [10] show 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. 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.

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. F2 from the De Jong test suite [11] is relatively easy, while the "Rana" function is difficult. F2 was created specifically by Rosenbrock around 1960 to illustrate the weakness of methods which change only one variable at a time during search. Rosenbrock showed that even gradient methods move very slowly on this function because the direction of the gradient significantly changes at each time step.

Rosenbrock proposed a search method that uses the Gram-Schmidt orthogonalization algorithm to adapt the search coordinate system. Later the Nelder-Mead Simplex method was introduced [12], in part to deal with this problem. 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.

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., the Random Bit Climber (RBC) [13]) 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.

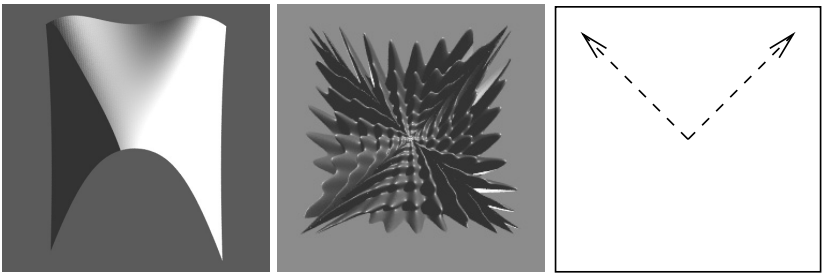

Fig. 2. The leftmost figure is F2 from the De Jong Test Suite, also known as Rosenbrock's banana function. The middle figure is F102, or Rana's function. The rightmost figure is a cartoon showing the "ridges" that lead to the global optimum as well as other competitive local optima.

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.

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 often used at 10 bits of precision. Genetic algorithms at 20 bits of precision can be 10 to 100 times slower to converge using 20 versus 10 bits of precision.

Table 1. Results of steepest ascent bit climbing with 100 restarts at 10 and 20 bit resolution. 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.

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

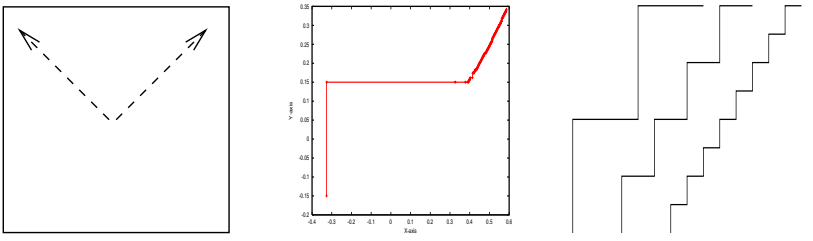

Fig. 3. The "arrows" figure shows the approximate ridge location. The middle figure tracks the movement of an actual run of local search on F2. After 2 moves the ridge is encountered. The rightmost figure: adding 1 bit of precision doubles the number of steps.

3 Ridges and Temperature Inversion

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

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

Figure 4 is a 2-D slice taken from the temperature inversion problem in the area around the global optimum. There is a distinct ridge. The companion image shows the location of "false" local optima under Gray code: no improving exists in the x or y dimension. There are a surprising number of false local optima that trap local search. Such ridges occur *throughout* the search space and respond to the nonlinear interaction between variables: changing a parameter in dimension x simultaneously changes the location of the ridge in many other dimensions.

Empirical results show that many evolutionary algorithms are trapped in local optima at about the same rate as local search, including a Steepest Ascent

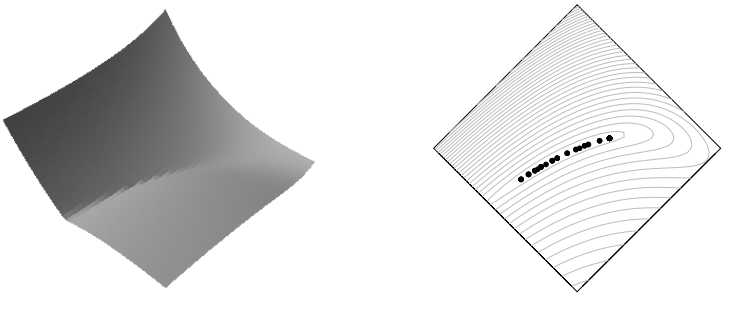

Fig. 4. On the left is a 2-D slice near the global optimum of the 43-D temperature inversion problem. The points in the rightmost figure shown where "false" local optima occur.

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

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
	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
	PCA SABC	-511	-480	24	23	6	1,262	341

Bit Climber, CHC, a (30,210)ES and a (30+210)ES with standard mutation adaptations and no rotations, PBIL and Differential Evolution.

4 Rotating Search Coordinates

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 use a heuristic rotation by adapting a set of rotation strategy parameters via evolution [14]. However, adapting rotation "strategy" parameters of the form used by Evolution Strategies is too imprecise and impractical for large problems.

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 performed. The eigenvectors are represented by a rotation matrix $\mathbf{R}$. Let Λ be the diagonal eigenvalue matrix. Let $\mathbf{X}$ represent a matrix of data vectors. Using PCA we find $\mathbf{R}$ and Λ such that

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

For a single search point represented by the vector $\mathbf{x}$ we compute $\mathbf{x}\mathbf{R}$, which is the projection of the point $\mathbf{x}$ into the space defined by $\mathbf{R}$. The rotation matrix is orthonormal, so a simple correction is also needed to translate and re-center the rotation during search.

To find a structure such as a ridge, PCA can be used to sample locally and isolate a subset of the better sample points. For example, sample 20 points and then apply PCA analysis to the 10 best solutions. 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 sloping plateau.

Another approach is to use the Gram-Schmidt (GS) orthogonalization algorithm to rotate the space. Often the Gram-Schmidt 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 heuristic way of determining a useful "rotation" for changing the problem representation.

In Table 2 Steepest Ascent Bit Climbing (SABC) with a Gray Code representation is compared with SABC using PCA to rotate the search space. For the PCA, 15 points were sampled, with PCA applied to the best 8. The speed-up is *dramatic* using rotated representations.

5 Constructing Higher Dimensional Test Problems

We have found that there are no good test functions that are difficult and which also scale up to higher dimensions. Researchers have rotated existing test functions to make them more difficult. But there still exists a single rotation that converts the problem back into a relatively easy problem; and this does nothing to address scalability.

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 linear 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 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:

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

However, the symmetry that exists in this function can make some problems easier to solve in higher dimensions. Figure 5 (topmost) shows the two dimensional slices taken from the three dimensional Rosenbrock function when generalized in this way. In general, the surfaces are not as difficult as the original Rosenbrock function, and the problem becomes *much* easier in higher dimensions. In order to retain the original ridge structure, the following expansion can be 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 creates a *non-separable*, higher dimension problem that preserves the ridge features of the original problem. Figure 5 (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 multimodal surface from which it is difficult for a coordinate strategy to escape. 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.

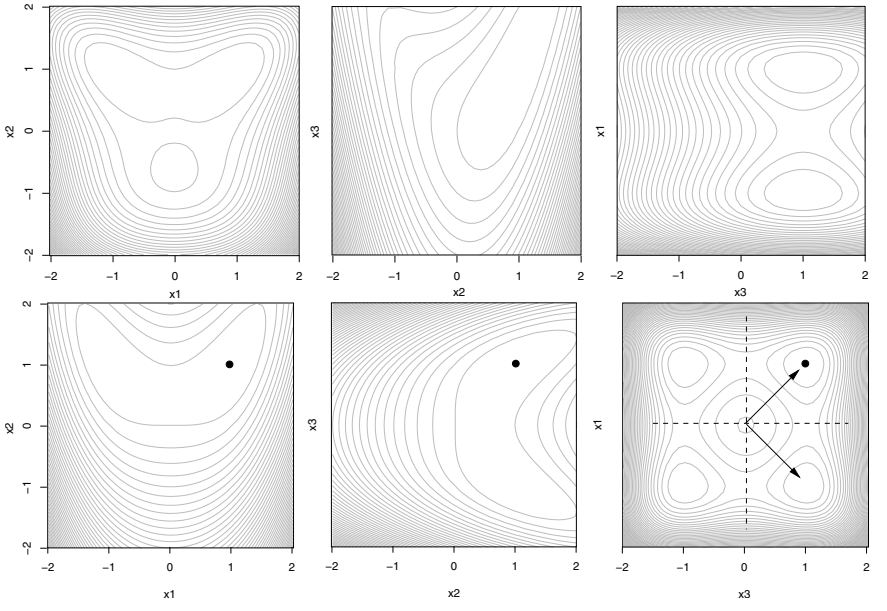

Fig. 5. Results of expanding Rosenbrock's function in three dimensions using Expansion Method 1 are shown in the top 3 slices of the space: with the possible exception of the first slice, the search space is simpler and easier than the original problem. Results of expanding Rosenbrock's function in three 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.

We 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 steps 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 that 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.

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.

Function	Search	Best	Mean	Std	Steps	Std	Evals	Std
F2, 5-D	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, 5-D	SABC	-386	-313	49	506	915	50,551	91,540
	PCA SABC	-399	-310	42	112	167	15,662	23,318

Function	Search	Best	Mean	Std	Steps	Std	Evals	Std
F2, 10-D	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, 10-D	SABC	-427	-354	40	758	940	151,628	187,959
	PCA SABC	-411	-308	47	525	632	146,917	176,958
Temp, 10-D	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

6 Discussion and Related Work

The goal of this paper is to highlight and explain the ridge problem and explore the use of rotated representations. Rotated representations could be used in conjunction with various types of evolutionary algorithms. And despite work in this direction, most of the evolutionary computation field has not looked seriously at rotated representations.

There is at least one algorithm that already makes extensive use of rotated representations and has mechanisms to address some of the key questions that arise when using rotated representations: *Covariance Matrix Adaptation*, or CMA, rotates and scales the mutation operators used by an Evolution Strategy. The key question is what kind of sample should be used when computing rotations. If a localized sample is taken and then the best points (e.g., the best half of the sample) are used to calculate the eigenvectors and eigenvalues, the direction of maximum variance can be in the direction of the gradient or it can be orthogonal to the gradient.

Recall that the Gram-Schmidt algorithm is often 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 kind of approach used less information, but emphasizes knowledge about gradient based on the most recent move

or moves. This is a more localized and heuristic way of determining a useful "rotation" for changing the problem representation.

In effect, PCA exploits information about variance, whereas Gram-Schmidt uses path information. The path information acts as a kind of "momentum" term that keeps the search moving in it's current direction. Simple empirical experiments show that path information is most useful when approaching a ridge, or when following a straight ridge. But path information is sometimes misleading, for example on a curved ridge. On a curved ridge, calculating the direction of maximum variance helps to track the changing gradient.

These ideas are already exploited by the CMA-ES algorithm.

6.1 CMA-ES

Traditional *evolution strategies* produce offspring based on adaptive strategy variables that attempt to improve the likelihood of producing better offspring. *Strategy parameters* are coded onto the chromosome along with the objective parameters and are adapted indirectly based on the assumption that highly fit individuals will carry better strategy parameters. The mutation strength of the strategy parameters must be high enough to create significant differences between offspring while minimizing adaption time [15]. Ostermeier et al. attempted to dampen the mutation strength without compromising adaption speed [16]. Hansen et al. offers a solution that completely removes the mutation strength when adapting the strategy parameters [15]. Unfortunately, this solution cannot be easily extended to other strategy parameters that control the angle of rotation between each dimension, as necessary in *correlated mutations*. Without this rotational adaptation, the algorithm's success is limited on ridge functions.

Covariance Matrix Adaptation, or CMA, uses a covariance matrix to rotate and scale the mutation distribution [15]. The covariance matrix is an estimate based on the evolution path and, when applicable, information extracted locally from strategies with large populations [17]. Hansen and Ostermeier define the reproduction phase from generation g to generation $g+1$ as:

$$x_k^{(g+1)} = \langle x \rangle_\mu^{(g)} + \sigma^{(g)} \mathbf{B}^{(g)} \mathbf{D}^{(g)} z_k^{(g+1)}$$

where $z_k^{(g+1)}$ are randomly generated from an $N(0, I)$ distribution. This creates a set of base points that are rotated and scaled by the eigenvectors ($\mathbf{B}^{(g)}$) and the square root of the eigenvalues ($\mathbf{D}^{(g)}$) of the covariance matrix C. The single global step size, $\sigma^{(g)}$, scales the distribution based on adaptation. Finally, the points are translated to center around $\langle x \rangle_\mu^{(g)}$, the mean of the μ best parents of the population.

Instead of only using a single generation to compute covariance, CMA-ES utilizes the entire evolution path, called *cumulation*. The evolution path updates at each generation using a weighted sum of the current evolution path, $p_c^{(g)}$, with the vector that points from the mean of the μ best points in generation g to the mean of the μ best points in generation $g+1$. When a larger population (λ) is

used, the best μ individuals may help describe the topology around the mean of the current generation. This is potentially useful information. Assuming $\mathbf{Z}^{(g+1)}$ is the covariance of the μ best individuals, and $\mathbf{P}^{(g+1)}$ is the covariance of the evolution path, the new covariance matrix is:

$$\mathbf{C}^{(g+1)} = (1 - c_{cov})\mathbf{C}^{(g)} + c_{cov}\left(\alpha_{cov}\mathbf{P}^{(g+1)} + (1 - \alpha_{cov})\mathbf{Z}^{(g+1)}\right)$$

Where c_{cov} and α_{cov} are constants that weight the importance of each input.

7 Conclusions

The CMA-ES algorithm has already raised interesting issues about how best to implement Evolution Strategies. Traditionally an ES encodes $\mathcal{O}(N^2)$ rotation or covariance parameters onto the chromosome to be evolved along with the N object parameters. Such rotations are simply not practical on large problems. Empirical tests of the CMA-ES algorithm are very positive. The use of rotated representations could produce a fundamental change in the theory and application of Evolution Strategies. The use of rotated representations also needs to be explored for a wider range of evolutionary algorithms.

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

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. Whitley, D., Barbulescu, L., Watson, J.: Local Search and High Precision Gray Codes. In: Foundations of Genetic Algorithms FOGA-6, Morgan Kaufmann (2001)
4. Rana, S., Whitley, D.: Representations, Search and Local Optima. 14th National Conf on Artificial Intelligence AAAI-97, (1997) 497–502
5. Syswerda, G.: Simulated Crossover in Genetic Algorithms. In Whitley, D., ed.: FOGA - 2, Morgan Kaufmann (1993) 239–255
6. Kazadi, S.T.: Conjugate schema and basis representation of crossover and mutation operators. Evolutionary Computation **6** (1998) 129–160
7. Wyatt, D., Lipson, H.: Finding building blocks through eigenstructure adaptation. In: GECCO, Morgan Kaufmann (2003) 1518–1529
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. ppsn5, Springer-Verlag (1998) 34–43
11. DeJong, K.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, Dept. of Computer and Communication Sciences, (1975)

12. Nelder, J., Mead, R.: A simplex method for function minimization. Computer Journal **7** (1965) 308–313
13. 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
14. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press (1996)
15. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9** (2001) 159–195
16. Ostermeier, A., Gawelczyk, A., Hansen, N.: A derandomized approach to self-adaptation of evolution strategies. Evolutionary Computation **2** (1994) 369–380
17. Hansen, N., Mãller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation. Evolutionary Computation **11** (2003) 1–18