Measuring Mobility and the Performance of Global Search Algorithms

Monte Lunacek
lunacek@cs.colostate.edu

Darrell Whitley
whitley@cs.colostate.edu

James N. Knight
nate@cs.colostate.edu

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

ABSTRACT

The global search properties of heuristic search algorithms are not well understood. In this paper, we introduce a new metric, *mobility*, that quantifies the dispersion of local optima visited during a search. This allows us to explore two questions: How disperse are the local optima visited during a search? How does mobility relate to algorithm performance? We compare local search with two evolutionary algorithms, CHC and CMA-ES, on a set of non-separable, non-symmetric, multi-modal test functions. Given our mobility metric, we show that algorithms visiting more disperse local optima tend to be better optimizers.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: [heuristic methods]; G.1.6 [**Optimization**]: [global optimization]

General Terms

Algorithms, Measurement, and Performance

Keywords

Evolutionary algorithms, global search, mobility

1. INTRODUCTION

We define a new metric, *mobility*, that quantifies the dispersion of local optima visited during a search. Intuitively, an effective global search will spend most of its time comparing the best local optima. We pose two questions that we believe will help quantify this intuition. First, how disperse are the local optima visited during a search? Second, do effective global search algorithms tend to be more mobile? We show that on two difficult multi-modal functions, algorithms that visit more disperse local optima tend to find better overall solutions.

We believe mobility is a useful metric for understanding global search. Often, algorithms are classified as global optimizers based on empirical studies that, in reality, may say little about how the algorithm will perform on real world multi-modal problems. Several multi-modal test functions routinely used in these empirical studies can easily be solved by local search methods. This is problematic when the only criteria for evaluation is the fitness of the final solution and not the actual global search properties.

Evolutionary algorithms are population-based search methods that employ selection, recombination, and mutation as a heuristic to find competitive solutions. These search methods work well in practice and have been successful on multi-modal problems where more elaborate gradient-based methods become trapped. Despite their success on these rugged landscapes, the global search properties of evolutionary algorithms are not well understood.

After a detailed description of the algorithms, we identify some critical test function properties that make search more difficult. We describe a method for quantitatively answering questions about global search. Then, we compare local search with two advanced evolutionary algorithms on two difficult benchmark functions. Specifically, in addition to local search, we look at CHC, a variation of the traditional genetic algorithm [1], and an Evolution Strategy with Covariance Matrix Adaption, or CMA-ES [3]. We conclude the paper with an analysis of our findings.

2. SEARCH ALGORITHMS

In this section, we describe the three algorithms evaluated in this paper: CHC, CMA-ES, and local search.

CHC

One of the more successful variants of the traditional genetic algorithm is CHC [1]. Genetic algorithms represent individuals in the search space as finite bit strings, called *genotypes*, which undergo mutation and recombination. In most genetic algorithms, the *crossover rate* for recombination is set much higher than the *mutation rate*, making the *crossover* operator responsible for more of the transformation. CHC uses a modified version of uniform crossover, where half of the non-matching bits are exchanged. Further steps are taken to ensure that parents are not allowed to cross unless they are sufficiently different. Eshelman refers to this as *incest prevention* [1]. No mutation is used to alter one generation to the next.

CHC diverges from genetic algorithms in that it uses *truncation* selection: newly created offspring must compete with the parent population for survival. This replaces the more traditional selection process with a "survival of the fittest" philosophy. The children under this scheme are always the same maximal Hamming distance from both parents. CHC also includes a restart mechanism, called *cataclysmic mutation*, that reinitializes the entire population by randomly flipping 35% of the bits of the best individual.

CMA-ES

Evolution strategies emphasize mutations over recombination. The traditional *evolution strategy* is an iterative process where μ parents produce λ offspring based on distributions around the parents. This distribution is defined by *strategy parameters* that, over time, adapt to create distributions that have a higher likelihood of producing more effective offspring. Strategy parameters are coded onto the chromosome along with the objective parameters and are indirectly adapted 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 [3].

De-randomized Evolution Strategy with Covariance Matrix Adaptation, or CMA-ES, uses a covariance matrix to explicitly rotate and scale the mutation distribution [3]. The orientation and shape of the distribution are not indirectly adapted, but calculated based on the evolution path and, when applicable, local information extracted from large populations. 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 uses the entire evolution path, called *cumulation*. The evolution path updates after each generation using a weighted sum of the current path, $p_c^{(g)}$, and a vector that points from the mean of the mu best points in generation g to the mean of the mu 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. In order to exploit this information, CMA-ES uses a rank-μ-update, that calculates the covariance of the μ best individuals

$$\mathbf{Z}^{(g+1)} = \frac{1}{\mu} \sum \mathbf{B}^{(g)} \mathbf{D}^{(g)} z_i^{(g+1)} (\mathbf{B}^{(g)} \mathbf{D}^{(g)} z_i^{(g+1)})^T$$

This information, along with the evolution path, $p_c^{(g)}$, is used to update the covariance matrix. Assuming $\mathbf{Z}^{(g+1)}$ is the covariance of the μ individuals, and $\mathbf{P}^{(g+1)}$ is the covariance of the evolution path, the new covariance matrix, $\mathbf{C}^{(g+1)}$, is

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

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

Hansen et al. assert that the initial step size, σ, for CMA-ES should be chosen "comparatively large" on multi-modal test functions [3]. Furthermore, they note that the local optimum found by using "small" step sizes is almost completely determined by the initial starting points [3]. We used an initial step size of 25% of each function's domain, which seems consistent with other research [2].

Local Search

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 differs from the current best by only one dimension, the neighborhood forms a coordinate pattern. Local search terminates when no improving move is found.

3. OBJECTIVE FUNCTIONS

Ideally, algorithms that are empirically effective on a set of test functions would also be effective on real-world applications. It is unclear if this is ever the case. Whitley et al. noticed that many of test functions used to evaluate evolutionary algorithms are *separable* [8]. Separable problems contain no non-linear interactions between the parameters of the objective function[1]. Separability implies functions can be easily solved by searching for the optimal solution separately in each dimension [8].

Salomon also found separability to be problematic [4]. Salomon rotated some of the common separable benchmark functions to create non-separable problems without changing the structure of the original problem. He found the breeder genetic algorithm (BGA) was significantly less effective when the coordinate system is rotated in $n-$dimensional space. 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.

The limitations of the breeder genetic algorithm do not extend to all genetic algorithms that use crossover. As mentioned previously, CHC uses a variation of uniform crossover that changes many parameters at once. Nevertheless, CHC does use a fixed coordinate system and it is unclear if CHC will suffer from the same limitations as BGA.

In addition to avoiding separable functions, we also wish to avoid highly symmetric functions [8]. One drawback of using symmetric functions for evaluation is their potential bias toward search neighborhoods that use a bit-encoding. For example, standard reflective Gray code always creates a symmetric neighbor reflected about the origin. On func-

[1]Some problems can be separable and still have some degree of non-linearity (e.g. $f(\vec{x}) = x_1 \cdot x_2 + x_3$). The problems noticed by Whitely *et al.* contained no non-linear interactions between parameters [8].

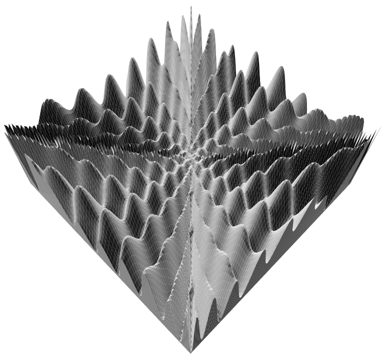


Figure 1: The two dimensional Rana surface.

tions like the two dimensional Rosenbrock, finding the local optima at (1,-1) implies one of the neighbors for the next search will be the global optima located at (1,1).

Since we are interested in global search properties, we are limiting our experiment to multi-modal test functions. The most challenging benchmark function for our experiment is the *Rana* function, whose surface is dotted with local minima, only to be interrupted by dominating ridges that run along the 45 degree lines (figure 1) [8]. We also used the highly multi-modal Schwefel function[2][6]. These base functions are summarized in Table 1.

Name	Function
Rana	$f(x, y) = x \sin\alpha \cos\beta + (y + 1) \sin\alpha \cos\beta$ with $\alpha = \sqrt{-x + y + 1}$ and $\beta = \sqrt{x + y + 1}$
Schwefel	$f(\vec{x}) = \sum_{i=1}^{N} \left(-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$

Table 1: The Rana and Schwefel base functions. Notice the Rana function is non-separable.

We used the following function to expand the base Rana function into a higher dimensions.

$$f(\vec{x}) = \sum_{i=1}^{n-1} f\mathrm{rana}(x_i, x_{i+1})$$

Clearly, the Schwefel function is separable and symmetric. The Rana function is symmetric. As previously mentioned, these properties create undesirable biases. In order to make our evaluation more robust, we rotated each function 22.5 degrees in all dimensions. This is similar to Salomon's method but differs in that the rotation is not random. Random rotations would create a different test surface for each trial and potentially a different number of local optima. Since we are interested in counting and comparing local optima, a static rotation seems more reasonable. We chose 22.5 degrees to ensure that the Rana problem did not become more predictable. Each function was translated by five percent of the domain, or 5.12 units, to reduce the potential bias caused by symmetry. Additionally, each dimension was scaled by five percent to create slightly more local optima. For the rest of the paper, Rana and Schwefel refer to the rotated, translated and scaled versions of these functions.

[2]This function is actually one of several described by Hans-Paul Schwefel and with which his name is unofficially connected. This particular function is based on problem 2.3, page 328 in the 1995 edition of "Evolution and Optimum Seeking" [6].

4. LOCAL OPTIMA AND MOBILITY

As mentioned previously, our goal is to quantify global search by answering the question: How disperse are the local optima visited during search? In order to measure how many of the local optima were sampled during search,we need quantify what it means to be "close" to a local optima. Once we know which local optima were sampled, we need to define a metric to measure their dispersion.

We ran local search, CHC, and CMA-ES on the two benchmark functions discussed previously. Each function was considered in five and ten dimensions. Since parameter settings affect performance, we ran two versions of each algorithm. Local search and CHC were run using 10-bits and 20-bits of precision. CHC used the time-tested population size of 50. CMA-ES heuristically chooses a population size. Hansen et. al. found that CMA-ES did surprisingly well on multi-modal functions when the population size was drastically increased and the algorithm used rank-μ-updates [2]. As mentioned, the rank-μ-update exploits the topological information contained in larger populations when updating the covariance matrix. In our tests, we ran CMA-ES with rank-μ-updates and a population size of 200 and 500. We distinguish each algorithm based on its parameters; CHC-10, CHC-20, LS-10, LS-20, CMA-200, and CMA-500. Each algorithm was run for 30 trials and each trial ran for exactly 25,000 and 50,000 evaluations in five and ten dimensions respectively.

We used *restarts* to ensure that each algorithm used all the alloted evaluations. That is, if an algorithm finished before all the evaluations were used, the algorithm was given a random restart during the same trial and allowed to run until all the remaining evaluations were used.

Threshold - defining local optima

Two problems arise with respect to measuring the number local optima sampled during search. First, how do we define boundaries for the basins that surround local optima Second, once we have defined this boundary, how do we cluster points that fall within a given basin?

Ideally, we would look at the basins of attraction that contain the best points of the search space. We can do this by choosing a fitness *threshold*, and only considering points that fall below this value. This defines a measure of closeness to the local optima that is based on fitness and only includes the best local optima of the search space – the local optima we are interested in counting. Figure 2 shows an example on the two dimensional Schwefel function. Notice that the *threshold* leaves only the best local optima.

What is the appropriate *threshold* value for each function? In low dimensions, we can approximate this value by enumerating a fine mesh grid in the search space, sorting the points by fitness, and retrieving the maximum fitness of the best percentage of points. For example, we may want to find the threshold that defines the best 10% of the search space. This estimation is only an approximation due to finite precision effects. In higher dimensions (more than three), the enumeration and evaluation is intractable so another method must be used.

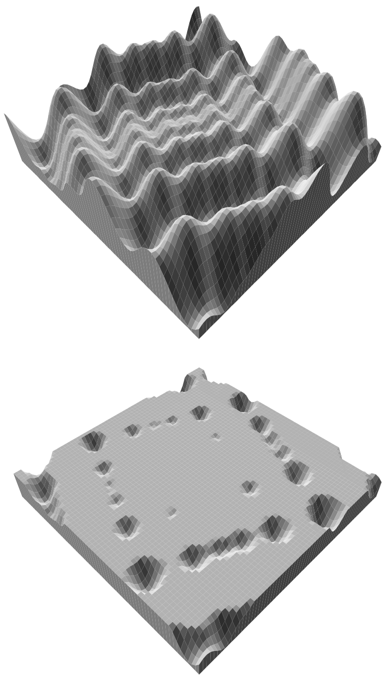


Figure 2: The two dimensional Schwefel surface (left) and the remaining local optima (right) after thresholding the surface.

We estimated a threshold for each function by combining all the points, from all 30 trials of each algorithm, and choose the threshold to be the fitness of the best 20% of the points. In other words, we looked at every point evaluated during 30 trials of CHC-10, CHC-20, LS-10, LS-20, CMA-200 and CMA-500, sorted the points and picked the threshold value to be the maximum fitness of the best 20% of all the points. We use this aggregate number to evaluate the global search properties of each trial.

Once a threshold has been calculated, we consider only the unique points that are below threshold. We assume that these points fall into the basins of attraction for the local optima we are interested in counting.

Mobility - a measure of dispersion

What we would like to measure is the dispersion of local optima visited during search. Often in real world search problems and problems in high dimension, little is known about the shape of the fitness landscape. An algorithm that samples many, disperse local optima increases confidence that its solutions are globally competitive.

Schuurmans and Southey defined three metrics that characterized local search performance on satisfiability problems [5]. The first metric, *depth*, measures the average depth local search travels into the fitness function. More specifically, *depth*, refers to the number of unsatisfied clauses at any point during search. *Mobility* measures how rapidly local search moves away from recently explored areas. Finally, *coverage* captures how well local search is exploring new regions that have not been previously visited. Schuurmans and Southey show that effective local search algorithms tend to have low depth and high mobility and coverage.

We are defining and exploring similar properties in the parameter optimization domain. Schuurmans' depth measurement is similar to our threshold. Our mobility metric is similar to both Schuurmans' mobility and coverage. Schuurman and Southey used *Hamming* distance to quantify local search. In the parameter optimization domain, *Euclidean* distance is more meaningful. Like Schuurmans and Southey, we relate mobility to algorithm effectiveness.

In the next section, we describe our method for assigning each sub-threshold point to a unique local optima. Once we have determined these points, we constructed a completely connected graph of the local optima with edges weighted by Euclidean distance. From this, we created a minimum spanning tree of the local optima. Figure 3 shows an example of the minimum spanning tree for local optima found by CHC on the Rana function in two dimensions. We call the sum of the weights on the minimum spanning tree *mobility.*

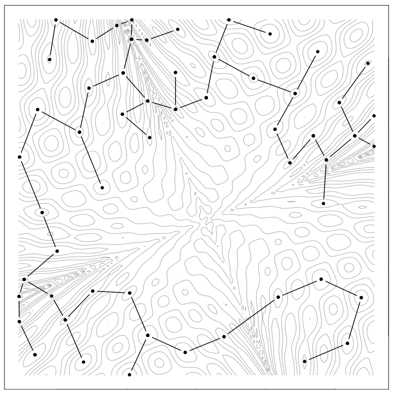


Figure 3: A minimal spanning tree of local optima on the Rana function.

Mobility measures the minimum connecting distances between a set of points, and does not distinguish, for example, between four points connected on a line and four points on connected on a box with equal edge lengths (Figure 4). In this example the *mobility* metric has a value of 4 for both configurations. We believe mobility gives a rough estimate of the dispersiveness of the visited local optima.

Bit-encoded algorithms can explore a large portion of the search space by only changing a few bits. One concern is: does measuring Euclidean mobility bias our results to bit-encoded algorithms? Although a small change in Hamming distance *can* correspond to a large change in Euclidean space, mobility is only measuring the distance to those sub-threshold neighbors. In other words, local search visiting a distant neighbor, as guided by its heuristic, does not imply that the point will be included in the mobility measurement. As discussed earlier, this point is considerably stronger when the test functions are non-symmetric and non-separable. We also observe that Hamming-space mobility would mean something completely different, which renders a direct comparison in Euclidean space confusing and potentially impossible.

5. MOBILITY IN LOW DIMENSIONS

In low dimensions, we use a modified form of local search to assign each sub-threshold point to a unique local optima. The neighborhood pattern of our modified local search was created by modifying the current best solution by ± 0.001

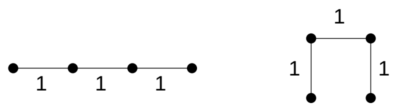


Figure 4: The mobility metric is invariant under certain sets of patterns.

in each dimension. This creates a compact neighborhood that includes two points in each dimension that surround the current best point. Keeping the neighbors close to the sub-threshold point is important for two reasons. First, it ensures that each point is assigned to the correct local optima. That is, local search can't discover better basins farther away. Second, a high precision local search also decreases the number of false local optima generated by ridges in the search space.

Ridges induce false optima when search algorithms use a coordinate pattern to find improving moves. If an algorithm looks for improving moves by changing only one dimension at a time, it will not see better points that fall between the neighborhood axis. In this case, a lower precision search will get stuck on the ridge, blindly assuming it has found a local optima. Increasing the precision will generally decrease the number of false optima. That is, a higher precision search algorithm will move further down the ridge by taking smaller steps along the ridge direction [7].

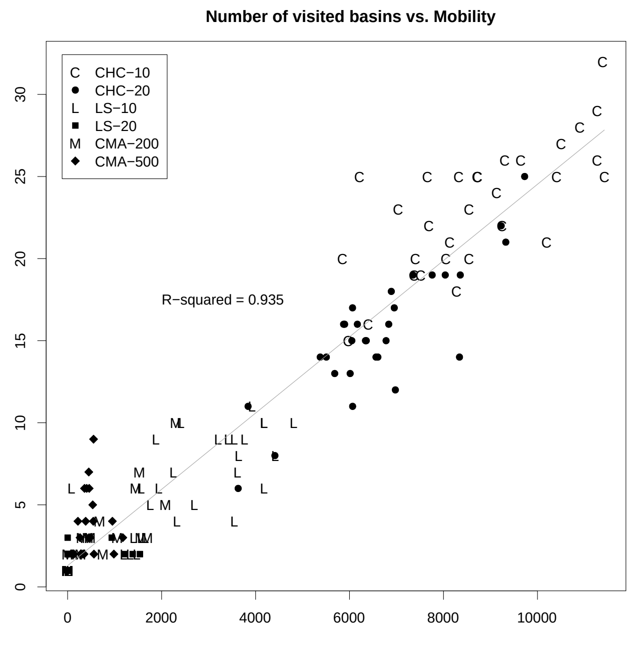
Various measurements for both the 5-D Rana and Schwefel functions are listed in Table 2. For each function, CHC-10 and CHC-20 visit significantly more basins of attraction and have significantly higher mobility than either local search (LS-10 and LS-20) or CMA-ES (CMA-200 and CMA-500). LS-10 had significantly higher mobility and visited more basins than the 20-bit local search. On the Rana Function, CMA-500's higher mobility was significant when compared with CMA-200. There were no other statistical significant patterns found between the algorithms.

	Rana		Schwefel	
	Mobility	Basins	Mobility	Basins
CHC-10	6,667.6	52.2	8,702.9	23.1
CHC-20	5,423.1	48.4	6,632.8	15.5
LS-10	995.3	6.47	2,530.9	6.17
LS-20	4.72	1.87	169.0	1.4
CMA-200	277.6	10.9	492.9	2.43
CMA-500	712.3	8.43	348.8	3.1

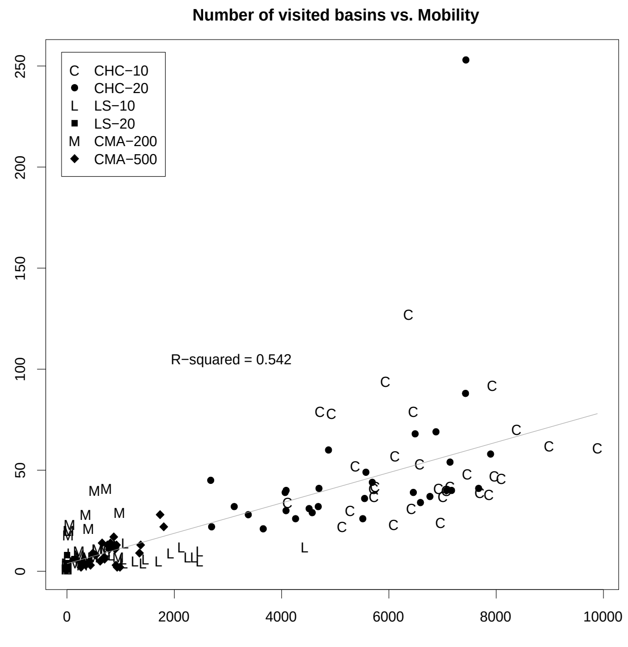
Table 2: Average *mobility* and number of *basins* visited per trial after 25,000 evaluations on the Rana and Schwefel 5D functions.

	Rana Fitness	Schwefel Fitness
Best known	-2,651	-2,371
CHC-10	-2,411.9	-2,334.4
CHC-20	-2,370.9	-2,274.9
LS-10	-2,010.6	-2,015.0
LS-20	-1,614.1	-1,790.4
CMA-200	-1,991.0	-1,906.2
CMA-500	-2,085.5	-1,884.0

Table 3: Average *fitness* per trial after 25,000 evaluations on the Rana and Schwefel 5D functions.



(a) Schwefel



(b) Rana

Figure 5: The number of basins visited (local optima) vs. Mobility on the Schwefel and Rana 5D test. Algorithms that visit more local optima tend to have a higher mobility and vice versa.

In five dimensions we noticed a high correlation between the number of basins visited and the mobility. Intuitively, this makes sense. Algorithms that visit more local optima will tend to have higher mobility and vice versa. Figure 5 shows the number of basins vs. the mobility on the Schwefel five dimension test. The Schwefel test has a strong correlation of 0.94. The correlation on the Rana five dimension test was 0.68.

Mobility in Higher Dimensions

In higher dimensions, performing modified local search from each of the threshold points becomes extremely computationally intensive. The primary problem here is that the modified local search simply takes too long to converge from the sub-threshold points to the local optima we wish to count.

This requires us to calculate the mobility metric discussed in the previous section without explicitly knowing to which basin of attraction each sub-threshold point belongs. We did this by constructing a minimal spanning tree of all the sub-threshold points. In other words, all the sub-threshold points will now be included as nodes of the minimal spanning tree. The mobility metric is different from the previous mobility metric in several ways. Most importantly, the nodes of the tree are no longer local optima, but rather sub-threshold points. As a result, the edges that join them will be included in the mobility measurement. That is, this mobility metric is still the sum of the edge lengths in the tree, but will now include the edges between points that lie in the same basin. This means the mobility measure will be slightly inflated because points within the same local optima will now add to the mobility metric. Comparing Figure 3 with Figure 6 graphically explains this difference.

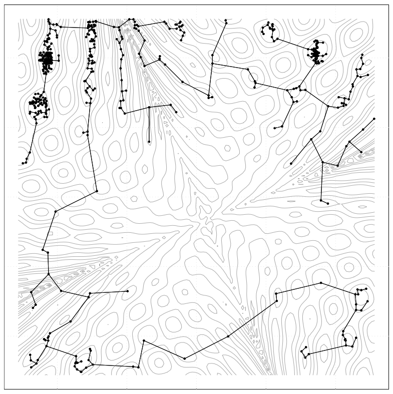


Figure 6: A minimal spanning tree of all the sub-threshold points on the Rana function. The minimal spanning tree in Figure 3 was constructed using the same sub-threshold points as a staring position for local search.

In ten dimensions, we calculated each algorithms mobility after 10,000 and 50,000 evaluations. The ten dimensional results for both the Rana and Schwefel functions are listed in Table 4. After 10,000 evaluations, CHC-10 and CHC-20 had significantly higher mobility than the other algorithms on both the Rana and Schwefel functions. Similarly, local search (LS-10 and LS-20) had higher mobility than CMA-ES (CMA-200 and CMA-500). After 50,000 evaluations, CHC-10 still exhibits significantly higher mobility than all the other algorithms on the Schwefel function. However, the difference between CHC (CHC-10 and CHC-20) and CMA-ES (CMA-200 and CMA-500) on Rana after 50,000 evaluations was not significant. All four algorithms outperformed local search (LS-10 and LS-20) on both functions after 50,000 evaluations.

	Rana		Schwefel	
	10K	50K	10K	50K
CHC-10	23,793.1	21,316.8	19,179.7	65,009.2
CHC-20	21,068.3	24,951.0	19,433.4	49,907.7
LS-10	9,819.6	835.5	11,512.6	2,137.4
LS-20	3,709.0	0.0	3,330.5	1,609.4
CMA-200	255.7	28,249.0	45.8	20,973.0
CMA-500	48.1	32,475.9	0.0	43,249.8

Table 4: Average *mobility* after 10,000 and 50,000 evaluations on the Rana and Schwefel 10D functions.

	Rana		Schwefel	
	10K	50K	10K	50K
Best Known	-4,933.0	-4,933.0	-4,556.9	-4,556.9
CHC-10	-3,184.2	-4,177.8	-3,411.6	-4,038.7
CHC-20	-3,319.0	-4,313.3	-3,523.5	-4,013.1
LS-10	-3,011.8	-3,382.5	-3,143.2	-3,464.0
LS-20	-2,822.2	-2,958.3	-2,749.3	-3,083.7
CMA-200	-2,253.4	-3,574.0	-2,159.3	-3,285.3
CMA-500	-2,124.5	-3,757.2	-1,989.3	-3,386.3

Table 5: Average *fitness* after 10,000 and 50,000 evaluations on the Rana and Schwefel 10D functions.

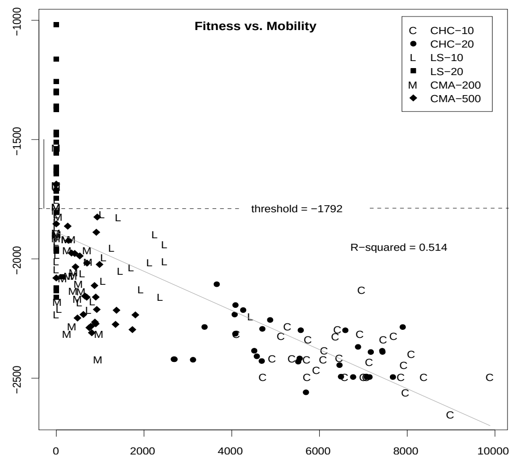
Mobility and Performance

Is mobility a good measure of global search performance? To answer this question we looked at the correlation between average fitness and mobility for each trial. Figure 7 shows the general trend for all six tests. Notice that algorithms with higher mobility tend to have more effective solutions.

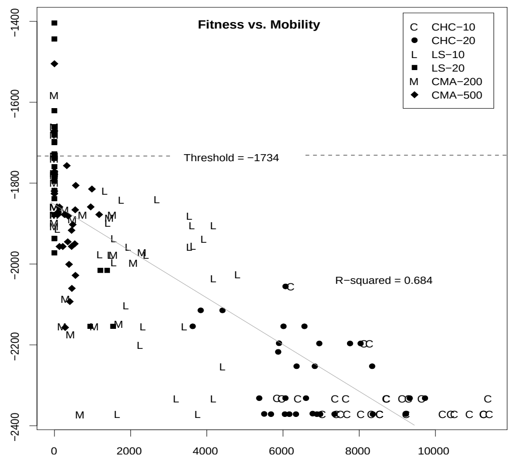
There is one exception to this observation. Some trials have no mobility, but vary in solution quality. In other words, each graph in figure 7 has trials that have a large range of fitness values, yet have zero mobility. These are the "stacked" points in the leftmost position of the graphs.

There are two reasons why trials could have low mobility and a large range of fitness values. Some algorithms may have no mobility and a poor range of fitness values. Recall that both mobility metrics defined in this paper only include points below a given threshold value, which is computed based on competing fitness values among all trials, across all algorithms. In order for a trial to have mobility, it must have some points below the threshold. So, trials that have no mobility and poor fitness simply did not visit below threshold basins of attraction. Notice that many of the low mobility points lie above the fitness threshold.

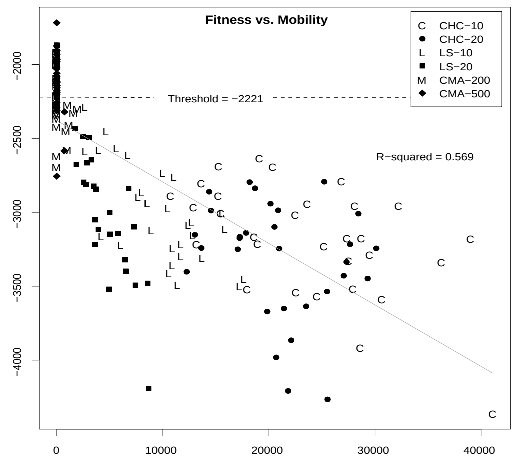
On the other hand, some algorithms tend to have low mobility and rather effective solutions. This is most pronounced in Local search (LS-10 and LS-20) after 50,000 evaluation in 10 dimensions (figures 7(e) and 7(f)). One explanation for this is that local search visited a few good basins of attraction that were probably relatively close together and failed to explore other below threshold areas of



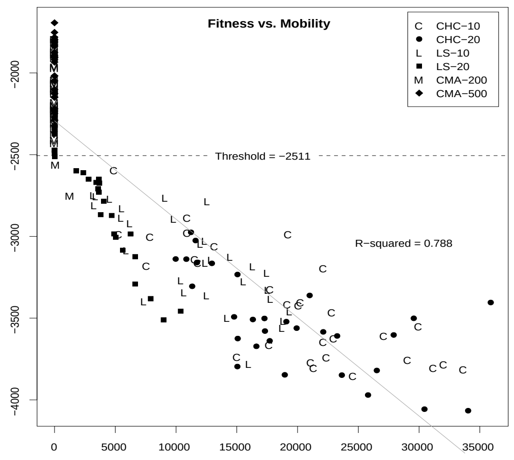
(a) Rana 5D 25K



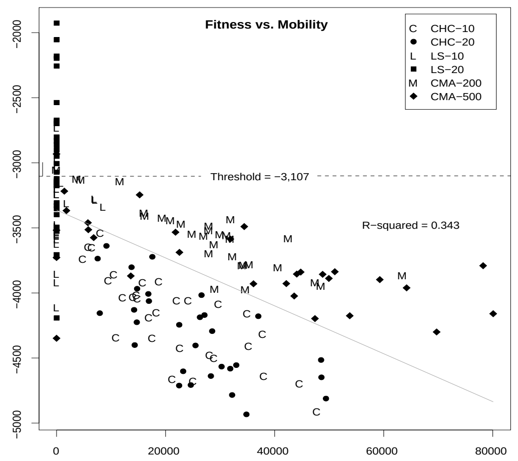
(b) Schwefel 5D 25K



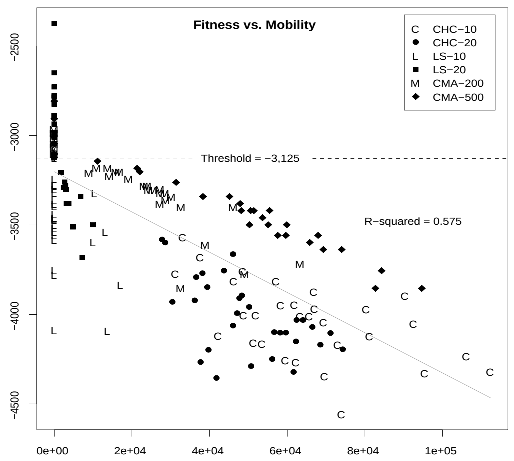
(c) Rana 10D 10K



(d) Schwefel 10D 10K



(e) Rana 10D 50K



(f) Schwefel 10D 50K

Figure 7: Mean Fitness vs. Mobility. In each case, there is some correlation that indicates more fit trials had high mobility. Intuitively, we would expect algorithms that explore more to find better solutions.

the search space. This could happen, for example, on a long ridge where local search creeps toward good solutions. In this case, local search uses many evaluations exploring only a small part of the space.

6. DISCUSSION

In ten dimensions, local search is significantly more mobile during the 10,000 evaluation measurement, but falls off the map entirely after 50,000 evaluations. At the same time, CMA-ES has more mobility after 50,000 evaluations. Longer running snapshots have higher thresholds. As algorithms begin to converge, densely populated sub-threshold distributions lower the threshold. So, early on in the search, the distribution of CMA-ES is still quite large and unfocused. Local search, on the other hand, has already focused in on several local optima. Eventually, CMA-ES starts to form a compact (and densely populated) distribution that focuses in on one or more local optima. This effectively lowers the threshold to a point that nearly excludes local search. Notice that CHC remains fairly consistent in during the 10,000 and 50,000 evaluation mobility snapshots.

In all but the Rana ten dimensional, 50,000 evaluations, test, CHC had significantly higher mobility. Why does CHC, on average, tend to visit more of the search space? Perhaps some of the answer lies in CHC's distribution. CHC's crossover operator HUX, or Half Uniform Crossover, exchanges half the non-matching bits. One consequence of this, according to Eschelman, is that children are always maximum Hamming distance from their two parents [1]. This highly disruptive operator tends to spread search out until good schemata start to form. It is possible that the distributed nature of CHC's child distribution, at least to some degree, accounts for its high mobility.

Restarts also help algorithms with stale distributions explore new parts of the search space. *Cataclysmic mutation*, for example, forces CHC to look in other parts of the search space once the population begins to converge. In the five dimensional tests, the lower precision search was significantly more mobile than its high precision counterpart in every case. This held in ten dimensions for the 10,000 evaluations tests, where local search actually had some mobility. The high precision local search has twice as many neighbors in each step, which quickly burns up valuable evaluation calls. And all of the neighbors are within the low precision neighborhood, which, in terms of exploration, adds nothing to the search. Furthermore, the small step size allows the high precision search to creep on ridges where the low precision search gets stuck and is forced to restart.

As previously mentioned, the initial step size for CMA-ES must be relatively large for the algorithm to be effective on multi-modal functions. Unfortunately, even with a large step size, the volume covered by each restart is less significant because the search space expands exponentially. This means CMA-ES must also rely on random restart to explore other parts of the search space. To make matters worse, unlike traditional evolution strategies, CMA-ES's population stays together. It is not possible for each member of the population to explore different parts of the search space at the same time unless the step size is large. Even then, the new children are based on a distribution around the mean and not the exploring parent. Several of the two dimensional trials we looked at would get trapped spanning several good local optima. The distribution of CMA-ES would continue to span all the local optima unable to explore each one individually.

7. CONCLUSION

In this paper, we have explored the global search properties of evolutionary algorithms. In particular, we compared local search with CHC, an aggressive genetic algorithm, to CMA-ES, which represents the state-of-the-art in evolution strategies. Each algorithm was tested on two rotated, translated and scaled multi-modal functions to ensure that the tests included non-separable and non-symmetric functions. Defining thresholds creates basins of attraction that are close to local optima in terms of fitness. Additionally, it focuses our study on only the best local optima. High precision local search provides a reliable way of clustering points in the same basin of attraction in low dimensions. The minimal spanning tree metric, *mobility*, was shown to be highly correlated with the number of optima found in a search. In high dimensions, estimating the mobility gives a reliable measure of the dispersion of local optima visited during search. In either case, algorithms with higher mobility tend to have more effective solutions.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0117209.

9. REFERENCES

[1] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In R. G. J. E., editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[2] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *PPSN*. Springer, 2004.

[3] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[4] R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39:263–278, 1996.

[5] D. Schuurmans and F. Southey. Local search characteristics of incomplete SAT procedures. *Artificial Intelligence*, 132(2):121–150, 2001.

[6] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.

[7] D. Whitley, M. Lunacek, and J. Knight. Ruffled by ridges: how evolutionary algorithms can fail. In *GECCO*, Berlin, 2004. Springer.

[8] D. Whitley, S. B. Rana, J. Dzubera, and K. E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.