Quad Search and Hybrid Genetic Algorithms

Darrell Whitley, Deon Garrett, and Jean-Paul Watson
{whitley,garrett,watsonj}@cs.colostate.edu

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

Abstract. A bit climber using a Gray encoding is guaranteed to converge to a global optimum in fewer than $2(L^2)$ evaluations on unimodal 1-D functions and on multi-dimensional sphere functions, where L bits are used to encode the function domain. Exploiting these ideas, we have constructed an algorithm we call **Quad Search.** Quad Search converges to a local optimum on unimodal 1-D functions in not more than $2L + 2$ function evaluations. For unimodal 1-D and separable multi-dimensional functions, the result is the global optimum. We empirically assess the performance of steepest ascent local search, next ascent local search, and Quad Search. These algorithms are also compared with Evolutionary Strategies. Because of its rapid convergence time, we also use Quad Search to construct a hybrid genetic algorithm. The resulting algorithm is more effective than hybrid genetic algorithms using steepest ascent local search or the RBC next ascent local search algorithm.

1 Background and Motivation

There are several advantages to using a reflected Gray code binary representation in conjunction with genetic algorithms and local search bit climbers. For both unimodal 1-dimensional functions and separable multi-dimensional unimodal functions, proofs show that steepest ascent local search is guaranteed to converge to a global optimum after $2L$ steps when executed on an L-bit local search neighborhood. The proofs assume that the functions are bijections, so that search cannot become stuck on plateaus of equally good solutions. Furthermore, the proofs demonstrate that there are only 4 neighbors per step (or move) that are critical to global convergence [1].

Using a reflected Gray code representation, it can be proven that a reduction from L to $L - 1$ dimensions is always possible after at most 2 moves (even if the search repeatedly "steps across" the optimum). Hence the global optimum of any 1-dimensional unimodal real-valued function can be reached in at most $2L$ steps. Under steepest ascent local search, each step requires $\mathcal{O}(L)$ evaluations. Thus, it follows that steepest ascent local search actually requires up to $\mathcal{O}(L^2)$ evaluations to converge to the global optimum. In practice, next ascent local search methods are often much faster than steepest ascent. However, the worst case order of complexity of next ascent algorithms is exponential. This is due to the fact that next ascent algorithms in the worst case can take very small steps at each iteration.

The current paper exploits the ideas behind these convergence proofs to construct a new algorithm which we call **Quad Search**. Quad Search is a specialized form of steepest ascent that operates on a reduced neighborhood. The algorithm cuts the search space into four quadrant and then systematically eliminates quadrants from further consideration. On unimodal functions Quad Search converges to the global optimum in at most $\mathcal{O}(L)$ evaluations, as opposed to $\mathcal{O}(L^2)$ evaluations for regular steepest ascent. For multi-dimensional functions, Quad Search converges to a point that is locally optimal in each dimension.

The new algorithm is tested on different types of unimodal functions. Quad Search is compared to steepest ascent and Davis' Random Bit Climber, RBC, which is a next ascent local search method [2]. Quad Search using a Gray code representation converges after at most $2L+2$ evaluations on classes of functions such as sphere functions where convergence proofs have also been developed for Evolution Strategies and Evolutionary Programming [1]. Given the convergence results on unimodal functions, we compare Quad Search with Evolution Strategies. The representations used by Evolution Strategies are continuous while Quad Search uses a discrete representation, but in practice, both encodings can be high-precision.

Finally, one of the most exciting uses of Quad Search is in combination with genetic algorithms. We combine Quad Search with the "Genitor" steady-state genetic algorithm. We also look at hybrid forms of Genitor that use steepest ascent and RBC. Genitor in combination with Quad Search proved to be the most effective hybrid genetic algorithm.

2 The Gray code representation

Gray codes have two important properties. First, for any 1-D function and in each dimension of any multi-dimensional problem, adjacent neighbors in the real space are also adjacent neighbors in the Gray code hypercube graph [3]. Second, the standard Binary reflected Gray code folds the search space in each dimension.

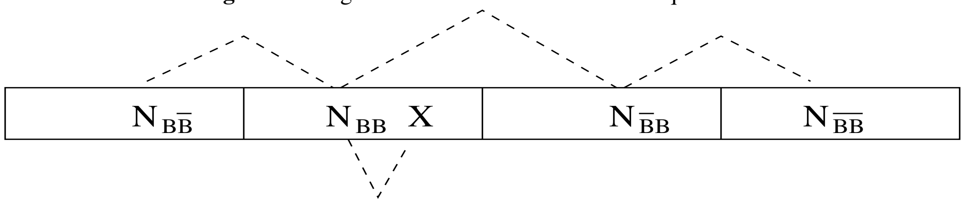
For each reference point R, there is exactly one neighbor in the opposite half of the search space. For example, in a 1-D search space of points indexed from 0 to $2^L - 1$, the points i and $(2^L - 1 - i)$ are neighbors. In effect these neighbors are found by folding or reflecting the 1-D search space about the mid-point between 2^{L-1} and $2^{L-1}+1$.

There are also reflections between each quartile of the search space. Thus, starting at some point i in the first quartile of the search space we can define a set of four points that will be critical to Quad Search. These points are found at locations given by the integer indices i, $(2^{L-1} - 1 - i)$, $(2^{L-1} + i)$ and $(2^L - 1 - i)$.

An inspection of these integer values shows that each of the points is in a different quadrant of the search space. This follows from the following observations. Point i is in the first quadrant and $i < 2^{L-2}$. Furthermore, $2^{L-1} - 1$ is the last point in the second quadrant and therefore $(2^{L-1} - 1 - i)$ must also be in the second quadrant. Similarly 2^{L-1} is the first point in the third quadrant and therefore $(2^{L-1} + i)$ must be in the third quadrant. Finally $2^L - 1$ is the last point in the fourth quadrant and therefore $(2^L - 1 - i)$ is in the fourth quadrant.

If we interpret each of these integers as Gray encoded bit strings, we can find each of these points using exclusive-or (denoted by $\oplus$) as follows:

Fig. 1. The neighbor structure across and within quadrants.



$$
\begin{aligned}
i &= i \\
(2^{L-1}-1) - i &= i \oplus (2^{L-1}-1) \\
(2^{L}-1) - i) &= i \oplus (2^{L}-1) \\
(2^{L-1}+i) &= i \oplus (2^{L}-1) \oplus (2^{L-1}-1)
\end{aligned}
\tag{1}
$$

Note that under a Gray code $2^L - 1$ is a string of all zeros except for a single 1 bit in the first position and $2^{L-1} - 1$ is a string of all zeros except for a single 1 bit in the second position. Therefore, moving from one of these points to the other involves flipping the first bit, the second bit, or both. (Also note that $i \oplus (2^L - 1) \oplus (2^{L-1} - 1)$ is a Hamming distance 1 neighbor of i under a Binary encoding [1].)

Also, since the exclusive-or operator really just applies a mask that flips the first or second bit, it works regardless of the quadrant in which the point i actually appears.

On average, flipping these bits moves the search the greatest distance. Recently published proofs show that these are the only moves that are critical in order to achieve convergence in a linear number of steps under a steepest local search algorithm algorithm [1]. We next construct the Quad Search algorithm to exploit these same moves.

3 Quad Search

Consider a unimodal function encoded using an L bit Gray code. We can break the space into four quadrants, defined by the two high order bits of the current individual. Note that any given point has two neighbors in quadrants outside of its own quadrant; they are the points generated by flipping the two high order bits. The highest order unfixed bit is called the *major* bit. The second highest order unfixed bit is called the *minor* bit. Flipping any other bit must generate a neighbor inside the current quadrant.

Consider a point N_{BB} located inside Quadrant 2, as illustrated in Figure 1. In general, BB denotes the first two bits that map to an arbitrary quadrant. We can flip the major bit to generate the point at $N_{\overline{B}B}$ and the minor bit to generate the point at $N_{B\overline{B}}$. We flip both bits to reach the point at $N_{\overline{BB}}$. We will also refer to the quadrants as Quadrant BB, each with its respective index.

In Figure 1 the point located at X is a neighbor of point N_{BB} that is located in Quadrant BB. X can be left or right of N_{BB}. We will express the problem in terms of minimization. One of the 4 quadrant neighbors is the current point from which the search is being carried out.

Theorem: *Quad Search converges to the global optimum on bijective unimodal 1-D functions after at most $2L + 2$ evaluations from an arbitrary starting point.*

Proof: The proof is recursive in nature. Assume the search space contains at least 8 points. means there are at least 2 points in each quadrant. After initializing the search, we show the search space can be cut in half after at most 2 new evaluations.

Sample a point N_{BB} and its reflected neighbors $N_{\overline{B}B}$, $N_{B\overline{B}}$ and $N_{\overline{BB}}$.

Let Z represent a string of zeros when interpreted as a string and the number of zeros when interpreted as an integer. Without loss of generality, we can relabel the points $\{N_{BB}, N_{\overline{B}B}, N_{B\overline{B}}, N_{\overline{BB}}\}$ using exclusive-or so that the minimal neighbor is point N_{000Z}. Then $\{000Z, 100Z, 010Z, 110Z\}$ denotes the same strings where the first 3 bits are as shown, and the remaining bits are all zero. We add one additional sample at point $001Z$.

Because the function is unimodal, the global minimum cannot be in Quadrant 11. Note that samples $000Z$ and $001Z$ are from Quadrant 00 which contains the current best known solution. Without loss of generality, assume that $100Z + c < 000Z + c < 001Z + c < 010Z + c$ when the strings are interpreted as integers and the integers are shifted by constant c using addition mod 2^L. (The function can be shifted by a constant, reversed, or relabeled using exclusive-or without changing the neighborhood structure: all that is really required for the proof is that there is 1 out of quadrant point to the left and the right and 2 samples in the middle quadrant.)

1. If $f(000Z) > f(001Z) < f(010Z)$, then the global optimum must reside in Quadrant 00 or 01. The major bit is fixed to 0. The points $000Z, 001Z, 010Z$, becomes the points $00Z, 01Z, 10Z$ in the reduced space. Only 2 additional points are needed to continue the search. First we evaluate $11Z$. Then the space is remapped again so that the minimum of these 4 points is at 00Z. Then we evaluate $001(Z-1)$.
2. If $f(100Z) > f(000Z) < f(001Z)$, then the global optimum must reside in Quadrant 00 or 10. The minor bit is fixed to 0. The points $000Z, 001Z, 100Z$, becomes the points $00Z, 01Z, 10Z$ in the reduced space. Only 2 additional points are needed to continue the search. First we evaluate $11Z$. Then the space is remapped again so that the minimum of these 4 points is at 00Z. Then we evaluate $001(Z-1)$.

After the first 5 evaluations, the search space contains $2^{(L-1)}$ points. At each iteration, the search space is cut in half after at most 2 new evaluations. After (L-3) iterations the search space is reduced to at most 4 points, since $2^(L-1)/2^(L-3) = 4$. However, at this point, we have already evaluated $00, 01$ and 10 (remapped so that 00 is the best solution so far). Only 11 remains to be evaluated. The total number of evaluation is 5 + 2(L-3) + 1 = 2(L) + 2. □

We assume the function is a bijection. Obviously, this is not true for many unimodal functions. While Quad Search cannot make progress over functions that are largely flat, it can step across small to moderate size flat regions. There are a number of ways ties might be handled. The current implementation does not always guarantee global convergence if there are flat regions, but its behavior is similar to that of next ascent. In general, convergence times that are less than exponential cannot be guaranteed if the space is largely flat (even if the space is unimodal).

At the beginning of the algorithm, the ordering of the quadrants is completely determined by the two high order bits. However, as bits are fixed, the high order bits used in the computation refer to the two highest order bits *which remain unfixed*. After a "1" bit has been fixed in a Gray code, this ordering is destroyed. The major bit is always the leftmost bit that has not been fixed. The minor bit is always the second bit from the left that has not been fixed. The major and minor bits need not be adjacent, but all bits to the right of the minor bit are unfixed. The only bit to the left of the minor bit that is unfixed in the major bit. Thus, "quadrants" are not always physically adjacent. However, the search space can always be remapped to a new physical space such that the major and minor bits can be used to move between adjacent quadrants in the wrapped space.

4 Performance of Quad Search versus Local Search

We examined the following variants of local search in this study: next-ascent local search, steepest-ascent local search, and Quad Search. We used Davis' Random Bit Climbing algorithm (RBC) [2] as our implementation of next-ascent. RBC is a simple next-ascent hill climber that randomizes the order in which bits are flipped. A random permutation is generated and used to determine the order in which to test bits. All L bits flips are tested before any bit flips are retested; after all bits have been flipped, a new permutation is generated and the process repeated. RBC also accepts bit flips resulting in equal fitness. The steepest-ascent algorithm, which we call steepest-ascent bit climbing, or SABC, differs in that it tests all bit flips, and accepts only the one that results in the largest improvement in fitness; ties are broken randomly.

Our implementation of Quad Search is based on an earlier proof that showed that not more than 8L evaluations were required [1]. It evaluates all of the potential Hamming 1 neighbors before evaluating any Hamming distance 2 neighbors. Also, the current implementation does not "reuse" samples inside of the middle quadrant. Obviously, this results in unnecessary computation; empirically, the current implementation uses just under $3L$ evaluations. This will be corrected in a future implementation.

We executed each local search algorithm on the set of three common test functions shown in Table 1. Each parameter was encoded using a 32-bit reflected Gray representation, and we considered 10, 20, and 30 dimensional variations of each test function. Each trial was executed until the global optimum was located.

We compare local search against Evolution Strategies because Evolution Strategies are proven to converge on unimodal functions and outperform other genetic algorithms in unimodal spaces. We did not use correlated mutation for 20 and 30 dimensional test functions due to the explosion of the number of correlation-strategy variables (for k object variables there are $\mathcal{O}(k^2)$ correlation-strategy variables). In addition, a restart mechanism was included in the ES to allow them to escape the condition noted by Mathias et. al., [4] whereby the strategy variables become so small as to halt the progress of the search. The specific restart mechanism employed is drawn from the CHC genetic algorithm [5]. Upon detection of convergence (defined to be all strategy variables in the population falling below some critical threshold) the best individual is copied

Table 1. Test functions used in this study. We consider $n = 10$, $n = 20$, and $n = 30$.

Function	Definition	Domain
Sphere	$\sum_{i=1}^{n} (x_i + 5)^2$	$-20 \leq x_i \leq 20$
Parabolic Ridge	$-10x_1 - \sum_{x=2}^{n} (x_i)^2 - 1000$	$-100 \leq x_i \leq 100$
Sharp Ridge	$-10x_1 - \sqrt{\sum_{x=2}^{n} x_i^2} - 1000$	$-100 \leq x_i \leq 100$

into the new population. The remainder of the population is filled with mutated copies of the best individual, and all strategy variables are reset to new values in the range $[-1.0, 1.0]$. In this study, the threshold for restarting was chosen as 1.0×10^{-6}, and the mutations were performed on 35% of the variables in each individual. In our implementation, restarting does not modify the object variables in the population, it simply resets the strategy variables. To facilitate comparisons with local search, we used a 32-bit floating-point representation to encode each parameter. All trials were terminated after 1,000,000 fitness evaluations or when the fitness of the best individual reached or exceeded -1.0×10^{-10}.

Table 2 shows the performance of each algorithm on the Sphere and Parabolic Ridge functions; the results for the Sharpe Ridge function are similar to the Parabolic Ridge function and are not shown. All local search methods are guaranteed to find the optimum. RBC and Quad Search empirically display linear time search behavior. The number of evaluations needed for the SABC steepest ascent algorithm to find the optimum is, as expected, $\mathcal{O}(L^2)$. The performance of the Evolution Strategies is roughly comparable to SABC.

The reader should be aware that this is probably not the most efficient form of Evolution Strategies for these test problems. Evolution Strategies using a 1/5 rule to control the mutation rate should lead to linear time convergence on the sphere function.

One problem with the Evolution Strategies we have implemented is that they are often unable to make the fine adjustments necessary to reach optimality to the desired accuracy. Often all but one (or a small number) of the parameters are set to the optimal values, while the remaining parameters are near-optimal. Sometimes the ES is not able to optimize the near-optimal parameters without disturbing the values of the optimal parameters; the values of the strategy variables become too small to be effective and the search stagnates.

Restarts rarely help on these particular functions – restarting resets several strategy variables, causing future mutations to disturb the parameters that are already optimal. It may also be the case that the Evolution Strategies used here would have displayed much better performance if we had used a less conservative convergence criteria.

5 A Quad Search/Genetic Algorithm Hybrid

We now evaluate the performance of Quad Search on a suite of multimodal test functions which are relatively difficult for most optimization algorithms [6]. The specific functions we consider are shown in Table 3. Powell's function is 4-dimensional, and we encode each parameter using 20 bits under a reflected Gray representation. For the

Table 2. Number of evaluations to find the optimum of the Sphere and Parabolic Ridge functions. The Evolution Strategies used in this study used a restart mechanism. **# Solved** refers to the number of times the optimal solution was found out of 30 independent trials. **# Evaluations** refers to the number of evaluations that were required.

Algorithm	Sphere 10-D		Sphere 20-D		Sphere 30-D	
	# Solved	# Evaluations	# Solved	# Evaluations	# Solved	# Evaluations
Quad Search	30	932.7	30	1801.0	30	2659.7
RBC	30	5875.3	30	12115.9	30	18420.2
SABC	30	51478.7	30	208198.0	30	458078.0
(1+1) ES	5	563967.0	0	-	0	-
(1+5) ES	8	412074.3	0	-	0	-
(10+10) ES	30	13206.2	30	151866.0	6	574046.0
(10+50) ES	30	17957.7	30	146173.0	17	477820.0
(50+50) ES	30	30707.3	30	130571.0	28	500807.0
(50+250) ES	30	50106.3	30	153628.0	28	418223.0
(100+100) ES	30	50868.3	30	191504.1	28	595862.3
(100+500) ES	30	89413.7	30	242475.6	28	544634.4

Algorithm	Parabolic Ridge 10-D		Parabolic Ridge 20-D		Parabolic Ridge 30-D	
	# Solved	# Evaluations	# Solved	# Evaluations	# Solved	# Evaluations
Quad Search	30	951.9	30	1825.6	30	2670.3
RBC	30	5918.1	30	13013.3	30	19605.4
SABC	30	50151.9	30	201809.0	30	446225.5
(1+1) ES	9	457005.1	0	-	0	-
(1+5) ES	9	298370.4	0	-	0	-
(10+10) ES	30	15438.8	30	228059.1	4	531212.2
(10+50) ES	30	21234.3	30	122101.0	11	568849.0
(50+50) ES	30	36132.1	30	120264.7	28	539147.3
(50+250) ES	30	60439.7	30	169586.2	29	413286.7
(100+100) ES	30	61381.3	30	236389.5	28	665666.7
(100+500) ES	30	108480.2	30	284724.0	23	564163.3

Table 3. Test functions: Griewangk, EF101, and EF102 are 10-Dimensional and Powell is 4-Dimensional. EF101 and EF102 were also tested in 20-Dimensional variants. Powell's function was encoded with 20 bits per parameter. All others used 10-bit encodings.

Name	Function	Global Optimum
Griewangk	$1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} (cos(\frac{x_i}{\sqrt{i}}))$	0.0
Powell	$(x_1 + 10x_2)^2 + (\sqrt{5}(x_3 - x_4))^2 +$ $((x_2 - 2x_3)^2)^2 + (\sqrt{10}(x_1 - x_4)^2)^2$	0.0
EF101	$-x sin(\sqrt{\lvert x - \frac{y+47}{2} \rvert}) - (y + 47) sin(\sqrt{\lvert y + 47 + \frac{x}{2} \rvert})$	-939.8
EF102	$x sin(\sqrt{\lvert y + 1 - x \rvert}) cos(\sqrt{\lvert x + y + 1 \rvert}) +$ $(y + 1) cos(\sqrt{\lvert y + 1 - x \rvert}) sin(\sqrt{\lvert x + y + 1 \rvert})$	-511.7

Table 4. Performance of local search and genetic algorithms on multimodal test functions. **Mean** refers to the mean value of the best solution averaged over 30 runs. **# Evaluations** refers to the mean number of evaluations averaged over 30 runs. **# Solved** refers to the number of trials (out of 30) in which the optimum was located. If **# Solved**=0, then 500,000 evaluations were consumed in all trials. σ denotes the standard deviation of the quantity in the adjacent left column.

Function	Algorithm	Mean	σ	# Solved	# Evaluations	σ
Griewangk	QuadSearch	0.073	0.032	2	233222.0	92175.1
	RBC	0.009	0.014	21	181605.3	129114.3
	SABC	0.0	0.0	30	85053.8	82216.0
	Genitor	0.108	0.058	2	13761.5	570.3
	CHC	0.001	0.005	29	62179.0	80013.5
Powell	QuadSearch	1624.330	1519.975	0	—	—
	RBC	2.62e-4	5.56e-5	0	—	—
	SABC	2.06e-4	9.94e-5	0	—	—
	Genitor	2.92e-4	7.24e-5	0	—	—
	CHC	0	0.0	30	200998.1	94196.4
EF101	QuadSearch	-724.880	25.322	0	—	—
	RBC	-759.683	28.087	0	—	—
	SABC	-756.576	34.588	0	—	—
	Genitor	-817.61	69.70	5	83151.2	117792.3
	CHC	-939.82	0.02	30	17171.3	11660.1
EF102	QuadSearch	-434.873	8.744	0	—	—
	RBC	-446.421	9.905	0	—	—
	SABC	-442.754	11.301	0	—	—
	Genitor	-443.45	17.87	0	—	—
	CHC	-495.56	5.53	0	—	—

other three functions, we use 10-dimensional variants, encoding each parameter using 10 bits under a reflected Gray representation. EF101 and EF102 are both based on the 2-dimensional functions shown in Table 3; the 10-dimensional versions are constructed using a wrapped weighting scheme [6].

For each local search algorithm (QuadSearch, RBC, and SABC), we execute 30 independent trials on each test function for a maximum of 500,000 evaluations. Search is terminated if a global optimum is encountered, and is re-initiated from a randomly generated starting point when descent terminates at local optima. For comparative purposes, we also consider two standard GAs: Genitor and CHC. For Genitor, we used a population size of 300. CHC is a state-of-the-art GA with its own form of diversity management [5]. We execute each GA for 500,000 evaluations or until a globally optimal solution is located. Such a methodology favors algorithms that produce the best results possible given a large number of evaluations; an algorithm might give good quick approximate results but not perform well in the current competition.

We show the results in Table 4. The data indicate that Quad Search performs poorly on multimodal, multi-dimensional test functions. This is not surprising. Quad Search behaves much like a greedy search algorithm: after finding the best move among 4 neighbors, it eliminates half the space from further consideration at each step. In contrast, RBC and SABC continue to sample the entire search space. But this also means that Quad Search often does much less work than RBC or SABC.

The advantage of Quad Search seems to be that it quickly locates the first local optimum it can find while evaluating only a few points in each dimension of the search space. It is interesting to note that RBC is usually the best of the local search methods when used in isolation, except on Griewangk, where SABC finds the optimal solution every time. Additionally, Quad Search can work within the normal bit representations often used by genetic algorithms. It is therefore natural to combine Quad Search with a genetic algorithm. Such algorithms are called hybrid genetic algorithms [7]. There has also been a recent trend to refer to such algorithms as memetic algorithms [8].

We consider hybrid genetic algorithms based on the Genitor algorithm [9]. Genitor is an elitist steady state GA which utilizes rank-based selection. It has proven to be amenable to hybridization, due to a number of factors. Selection is based solely on rank, which helps to maintain diversity; this can be especially important for hybrid genetic algorithms since local search often produces highly fit individuals. Under proportional selection, such individuals can quickly come to dominate the population.

All of the hybrid genetic algorithms we examine use uniform crossover and bitwise mutation with probability 0.001 per bit. The population size was 100. In addition, two-point reduced surrogate [10] and HUX crossover (a variant of uniform crossover in which exactly half of the bits are exchanged) [5] were tested and yielded negligible differences in performance. We investigate hybrid genetic algorithms based on the following local search algorithms: Quad Search, RBC, and SABC. Each algorithm can be executed in the following modes:

1. **Full Local Search:** Use a local search method to improve each member of the population until a local optimum is reached.

Table 5. Performance of Hybrid GAs on 4 multimodal test functions. For RBC and SABC, results for the Stochastic and Restricted forms of local search are reported, depending on which worked best. **Mean** is the mean value of the best solution averaged over 30 runs. **# Evaluations** refers to the mean number of evaluations averaged over 30 runs. **# Solved** is the number of trials (out of 30) that found the optimum. If **# Solved**=0, then 500,000 evaluations were consumed in all trials. σ denotes the standard deviation of the quantity in the adjacent left column.

Function	Local Search	Mode	Mean	σ	# Solved	# Evaluations	σ
Griewangk	Quad Search	Full	0.016	0.014	13	100090.3	45138.8
Griewangk	RBC	Full	0	0	30	144024.0	45507.5
Griewangk	SABC	Full	0	0	30	418697.8	14614.9
Griewangk	RBC	Restricted	0.011	0.016	20	123474.5	41413.3
Griewangk	SABC	Stochastic	0.013	0.018	19	76160.2	37452.1
Powell	Quad Search	Full	3.46e-9	9.07e-9	22	262931.2	57478.2
Powell	RBC	Full	2.14e-4	8.44e-5	0	—	—
Powell	SABC	Full	9.87e-5	8.34e-5	0	—	—
Powell	RBC	Stochastic	1.94e-7	7.83e-7	5	351407.2	44262.3
Powell	SABC	Restricted	5.55e-5	6.28e-5	0	—	—
EF101	Quad Search	Full	-939.84	0.01	30	216299.3	38496.7
EF101	RBC	Full	-924.63	34.95	25	395994.2	52569.5
EF101	SABC	Full	-772.59	45.55	1	448305.0	—
EF101	RBC	Stochastic	-906.79	55.21	22	237970.1	87683.9
EF101	SABC	Stochastic	-892.53	70.75	20	283078.8	80614.8
EF102	Quad Search	Full	-510.29	2.53	26	267792.7	42256.3
EF102	RBC	Full	-471.45	7.09	0	—	—
EF102	SABC	Full	-447.55	13.14	0	—	—
EF102	RBC	Stochastic	-484.08	7.66	0	—	—
EF102	SABC	Stochastic	-463.64	9.56	0	—	—

2. **Stochastic Local Search:** Use local search to improve each member of the population with probability 0.05; again local search is run until a local optimum is reached.

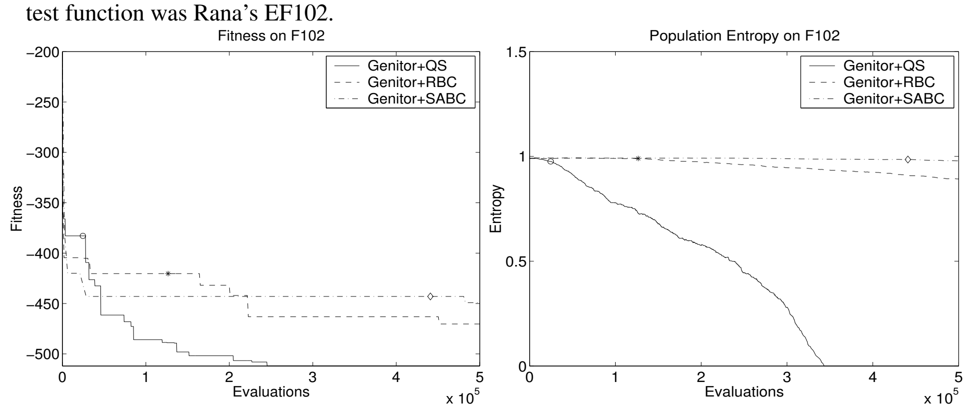
3. **Restricted Local Search:** Apply local search to every member of the population until a single improving move is found.

For the Restricted and Stochastic forms of local search, only the best results are given for SABC and RBC. The local search methods were not hybridized with CHC because we were not able to develop a hybrid form of CHC that improved on the original form of CHC.

Overall, the hybrid genetic algorithm combining Genitor and Quad Search is the best hybrid genetic algorithm; it was best on Powell, EF101 and Rana's function (EF102). The RBC and SABC algorithms were best on Griewangk, but Griewangk is also solved by using SABC alone.

Both CHC and the Genitor/Quad Search Hybrid solved EF101 every time, but CHC did so faster. CHC was somewhat better than the Genitor/Quad Search Hybrid on Powell's functions, but the Genitor/Quad Search algorithm was dramatically better than

Fig. 2. Performance of a Hybrid Genetic Algorithm utilizing Quad Search, RBC, and SABC as local optimizers. Fitness of the best individual versus number of evaluations is shown in the left figure. Entropy of the population versus number of evaluations is shown in the right figure. The test function was Rana's EF102.



CHC on Rana's function (EF102). To the best of our knowledge, this is the first time any search algorithm has solved a 10-D version of Rana's function to optimality. Overall, the Genitor/Quad Search Hybrid is better than any of the other hybrid genetic algorithms we tested and is competitive with and sometimes superior to CHC.

What make the Genitor/Quad Search combination so effective? We explored two hypotheses in an attempt to explain its behavior.

1. **Hypothesis 1**: Quad search improves strings quickly with minimal effort.
2. **Hypothesis 2**: Quad search works because it sustains genetic diversity.

The two hypotheses are not mutually exclusive, but experiments were performed to try to determine if one of these factors might be more important.

Figure 2 shows two graphs. The test function in this case was Rana's EF102. On the left, we show the speed with which improving moves are found using Genitor in combination with Quad Search (QS), RBC and SABC. On the right, we measured the entropy in the population.

$$Entropy(Population) = -B1 \, log \, B1 \quad - B0 \, log \, B0$$

where $B1$ counts the number of 1 bits in the population and $B0$ counts the number of 0 bits in the population. The figure shows that the Hybrid GA combining Genitor and Quad Search does not preserve diversity. Instead, it drives diversity out of the population faster than the other two local search methods when used as part of a hybrid genetic algorithm. It also shows that Genitor using Quad Search finds improvements more rapidly that Genitor used in combination with RBC or SABC.

6 Conclusions

The theoretical analysis of Quad Search presented in the paper, along with the empirical data, shows that Quad Search can be both extremely fast and effective on a special subclass of problems.

Because Quad Search is so fast, it may also have important practical uses in the construction of hybrid genetic and evolutionary algorithms. Empirical studies show that a hybrid genetic algorithm combining Genitor and Quad Search is generally superior to a hybrid genetic algorithm based on RBC or SABC.

Acknowledgments

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

References

1. Whitley, D., Barbulescu, L., Watson, J.: Local Search and High Precision Gray Codes. In: Foundations of Genetic Algorithms FOGA-6, Morgan Kaufmann (2001)
2. 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
3. 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
4. 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
5. Eshelman, L.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In Rawlins, G., ed.: FOGA -1, Morgan Kaufmann (1991) 265–283
6. Whitley, D., Mathias, K., Rana, S., Dzubera, J.: Evaluating Evolutionary Algorithms. Artificial Intelligence Journal **85** (1996) 1–32
7. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
8. Moscato, P.: Memetic Algorithms: A Short Introduction. In D. Corne, F. Glover, M.D., ed.: New Ideas in Optimization, McGraw-Hill (1999)
9. Whitley, L.D.: The GENITOR Algorithm and Selective Pressure: Why Rank Based Allocation of Reproductive Trials is Best. In Schaffer, J.D., ed.: Proc. of the 3rd Int'l. Conf. on GAs, Morgan Kaufmann (1989) 116–121
10. Booker, L.: Improving Search in Genetic Algorithms. In Davis, L., ed.: Genetic Algorithms and Simulated Annealing. Morgan Kaufmann (1987) 61–73