On Fitness Distributions and Expected Fitness Gain of Mutation Rates in Parallel Evolutionary Algorithms

David W. Corne[1], Martin J. Oates[2], Douglas B. Kell[3,4]

[1]Department of Computer Science, University of Reading, UK
d.w.corne@reading.ac.uk
[2]Evosolve Ltd, Stowmarket, Suffolk, UK
moates@btinternet.com
[3]Institute of Biological Sciences, University of Wales, Aberystwyth, UK
dbk@aber.ac.uk

Abstract. Setting the mutation rate for an evolutionary algorithm (EA) is confounded by many issues. Here we investigate mutation rates mainly in the context of large-population-parallelism. We justify the notion that high rates achieve better results, using underlying theory which notices that parallelization favourably alters the fitness distribution of a mutation operator. We derive an expression which sets out how this is changed in terms of the level of parallelization, and derive further expressions that allow us to adapt the mutation rate in a principled way by exploiting online-sampled landscape information. The adaptation technique (called RAGE - Rate Adaptation with Gain Expectation) shows promising preliminary results. Our motivation is the field of Directed Evolution (DE), which uses large-scale parallel EAs for limited numbers of generations to evolve novel proteins. RAGE is highly suitable for DE, and is applicable to large-scale parallel EAs in general.

1 Introduction

Setting the mutation rate for an evolutionary algorithm (EA) is complicated by the fact that much depends on various details of the EA and the application. Nevertheless, much published work provides generally accepted guidelines. An overall consensus, justified by theory [3,4,9] is that a rate of $1/L$ (where L is chromosome length) is often near-optimal (this can also be said of experimental biology [5]). When they have engaged in comprehensive parametric investigations in specific cases, researchers (e.g. [4]) have sometimes found that higher rates are better. Bäck also notes [4] that the optimal rate seems to increase with λ in a $(1 + \lambda)$ evolution strategy, but that no useful analytical results are known. However, the general suitability of $1/L$ in standard (i.e. serial implementation) settings has been more often confirmed than challenged. Thus, Oates et al [10] find a wide range of optimal mutation rates, with this range tending to *include* $1/L$. Meanwhile, recent theoretical work of note has studied the competent

[4] Present address: Dept Chemistry, UMIST, PO Box 88, MANCHESTER M60 1QD

design of parallel EAs [7], but the issue of mutation rate in this context has been little explored.

When we consider mutation rate setting in parallel EAs, there is a straightforward intuitive argument for high rates. These generally make higher-fitness mutants available, but with low probabilities; however, the larger the population, the better the chances of a high-rate yielding such mutants. In particular, parallelization means that this benefit is not at the expense of time. If we can evaluate P mutants in parallel, then we can regard the mutation operation as having effectively changed in nature. That is, we can evaluate P mutants in unit time, and can take the 'result' of the parallelized operation to be the fitness of the *best* of them. The mathematics of this follow.

1.2 Notes on Relevance and Applicability

Improved and cheaper hardware, and the wider availability and use of cluster-based computation, now makes the use of parallel implementations of EAs more feasible, and indeed such is now increasingly widespread. This heralds a need for better understanding of parallel EA design. Cantú-Paz [7], among others, is paving the way regarding several aspects of parallel EA design. Here we focus on mutation rate setting.

One relevant parallel EA application (which the authors are working on) is in the protein engineering/biotechnology community, and is called 'Directed Evolution' (DE). This refers to (what amounts to) the application of EAs to the discovery of novel proteins [1,2,12]. Consider, for example, the task of finding a protein which can bind to a particular area of a virus, and remain thermostable at body temperature. To address this in DE, a population of proteins is artificially evolved. They undergo selection, mutation and other operations in the normal way. There are, of course, many problems with this as regards its *in silico* implementation, since we know far too little about protein folding to implement the representation correctly, let alone estimate a novel protein's fitness. The trick, however, is that DE works entirely biologically. The 'representation' of a protein is actually via a gene encoding its overexpression (i.e. many copies are present) within a suitable cell (typically the bacterium *E. coli*). The cell's metabolism naturally produces the protein itself, and fitness comes from direct measurement of the protein's efficacy regarding the target activities of interest. Mutation is typically done by using so-called 'sloppy' or 'error-prone' versions of the Polymerase Chain reaction (PCR) in an *in vitro* step. Many details are of course omitted here, but none which alters the fact that the range of potentially applicable DE strategies essentially matches much of the space of EA designs. In particular, the biotechnology (with 'high-throughput screening') currently allows up to some 1,000,000 mutants to be evaluated per generation, and mutation rate is fully controllable (essentially by varying the conditions of the PCR reaction). DE is a technology with immense potential for novel and highly beneficial protein products, but the 'search spaces' are massive, and much depends on appropriate design and parameterization of DE strategies.

1.3 A Note on Related Work

Seminal work by Fogel and co-authors (e.g. [8] and subsequent papers) relates closely to that described here, but there are subtle differences in approach and applicability which are worth noting. In [8] and related work, Fogel *et al.*, like us, essentially recognize that repeated applications of genetic operators yield valuable information which can be employed to choose and parameterize the operator(s) more wisely. In [8] however, this is done essentially using extensive offline prior sampling (consuming significant numbers of fitness evaluations). Apart from our focus on the way that applicable mutation rates increase with parallelization, the main difference in this work is that we derive an approach which can be applied online, exploiting theory (relying on our restricted attention to standard mutation of *k*-ary chromosomes) which enables us to choose from among many appropriate rates having only sampled (in principle) a single rate previously This work is hence more applicable in cases where time saving in highly-parallelized EAs) is a premium concern.

2 Fitness Distributions and Expected Fitness Gain

Imagine a mutation operator which, with respect to a given parent, has n possible fitness outcomes $(f_1, f_2, ..., f_n)$. The *fitness distribution* gives, for each f_1, the chance the mutant will have that fitness. E.g. consider a mutation operator which flips a single randomly chosen bit in a binary string of length L, and where the fitness function is MAX-ONES, in which we seek to maximize the number of 1s in the string. If L=100 and the parent has fitness 80, then the fitness distribution can be written as ((79, 0.8),(81, 0.2)); i.e. the chance of the mutant having fitness 79 is 0.8, and the chance of it having fitness 81 is 0.2 (and fitness 82 zero). In contrast, an operator which flips two randomly chosen (but distinct) genes gives approximately: ((78, 0.64), (80, 0.32), (82, 0.038)).

'Expected fitness gain' essentially means what we expect the result of the mutation to yield in terms of a fitness improvement, keeping in mind that mutants less fit or equally fit as the parent will yield zero gain. Throughout, we assume a simple (but powerful) EA model which corresponds to a (1+ λ)-ES where λ is essentially the population size, which (by default) we assume is fully parallelized. Thus raising λ, up to the limit of parallelization, does not increase the elapsed time between generations. Consequently, an operator's fitness distribution is changed (since it now returns the 'best of λ' mutants), and so therefore is the expected fitness gain per generation. We argue that this expected gain increases faster for high-rate operators than for low-rate operators, and consequently there is a point or threshold (in terms of λ, which we will hereon simply call P) at which the higher rate becomes preferable.

2.1 Exploiting Parallelism Leads to Improved Fitness Distributions

From hereon we will work with the *gain* distribution, which only considers non-worse mutants. In the first case above, this is ((80, 0.8), (81, 0.2)), showing, for example, that the result of a mutation will be no change in fitness 80% of the time. For two-gene mutation it is ((80, 0.962), (82, 0.038)). It is natural to ask, what is the expected gain in one application of the operator? This is the sum of outcomes weighted by their probabilities. In the one-gene case this is 80.2, and in the two-gene case it is 80.076.

However, if we can suitably exploit parallelism, the gain distributions of higher rate operators become more favourable, and 'overtake' those with conservative distributions. Some simple intuition for this can follow from our running example. Imagine that we are able to evaluate 20 fitnesses in parallel. A *single* operator application (i.e. in unit time) now yields 20 mutants, from which we take the best. Now, the gain distribution of the single-gene operator is approximately ((80, 0.0115), (81, 0.9885)), in which the chance of achieving 81 is precisely the chance that not all of the 20 mutants were neutral (i.e. $1 - (1 - 0.2)^{20}$). But the gain distribution of the two-gene operator is now ((80, 0.461), (82, 0.539)). The expected fitness achieved after one time unit is therefore 80.9885 in the one-gene case and 81.078 in the two-gene case, so the higher rate operator has the more favourable distribution.

More generally, the 'P-parallelization of μ' is an operator which applies μ to the same parent P times in parallel, and the returned result is the best of these P (or the parent). P-parallelized μ has a gain distribution which we can state as follows, using Blickle and Thiele's analysis of tournament selection [6]. We will assume r distinct fitness outcomes, and give the gain distribution of μ in the form $((f_1,p_1),(f_2,p_2),\ldots,(f_r,p_r))$ where the terms' meanings are obvious from the examples above. An intermediate step is to note the 'cumulative gain distribution' $((f_1,\pi_1),(f_2,\pi_2),\ldots,(f_r,\pi_{rr}))$ where π_i is the probability of the mutant's fitness being either f_1 or worse, hence:

$$\pi_i = \sum_{j=1}^{r} f_j \tag{1}$$

and we can also note of course that $\pi_1 = f_1$ and $\pi_r = 1$.

We can now write, following [6], the gain distribution of the P-parallelised operator as $((f_1,q_1),(f_2,q_2),\ldots(f_r,q_r))$ where $q_i = \pi_i^P - \pi_{i-1}^P$, in which it is implicit that $\pi_0 = 0$. What is of particular interest is the expected fitness per application of the operator. P-parallelisation changes this, from:

$$f_E = \sum_{i=1}^{r} f_i p_i \qquad \text{to:} \qquad f_E = \sum_{i=1}^{r} f_i q_i = \sum_{i=1}^{r} f_i (\pi_i{}^P - \pi_{i-1}{}^P) \tag{2}$$

A key point is that the P-parallelisation of a 'high-rate' operator μ_H will often achieve a better expected gain than the P-parallelisation of its 'low-rate' counterpart μ_L. By simple calculations and approximations (which we shall omit), we can show, for example, that in a case with just three fitness outcomes and distributions as follows:

$$((f_1,L_1),(f_2,L_2),(f_3,L_3)) , \; ((f_1,H_1),(f_2,H_2),(f_3,H_3))$$

where $H_3 > L_3$ and $L_3 \approx 0$ then P-parallelized μ_H exceeds P-parallelised μ_L in expected fitness gain when $P > (H_1^P - L_1^P)/H_3$, from which two observations are apparent: first, if $L_1 > H_1$ then any value of P will lead to a better expected gain for the 'high-rate' operator. At first this seems odd, but notice that L_1 and H_1 denote the probabilities in the respective cases that the gain will be zero. Hence, since $L_1 > H_1$ the chance of at least *some* gain is necessarily higher for the high-rate operator. Otherwise, in the more normal situation $H_1 > L_1$, the expression reflects arguably modest needs in population size for parallelized μ_H to outperform parallelized μ_L.

To express the more general case for two operators μ_1 and μ_2, where (without loss of generality) μ_1 has a better expected fitness gain than μ_2 when both are P-parallelised, first, we can rearrange equation (2) to become:

$$f_E = (f_1 - f_2)\pi_1{}^P + (f_2 - f_3)\pi_2{}^P + \ldots + (f_{r-1} - f_r)\pi_{r-1}{}^P + f_r$$

and we can simplify this by considering only cases where fitness levels increase in units (hence $f_k - f_{k+1}$ always equals –1), and obtain: $f_E = f_r - \sum_{i=1}^{r-1} \pi_i{}^P$ Now, given two operators μ_1 and μ_2, we can simply derive:

$$f_E(\boldsymbol{\mu_1}) - f_E(\boldsymbol{\mu}_2) = \sum_{i=1}^{r-1} \pi_i(\boldsymbol{\mu}_2)^P - \sum_{i=1}^{r-1} \pi_i(\boldsymbol{\mu_1})^P$$

When this exceeds zero, the P-parallelization of μ_1 will have a better expected gain than that of μ_2. One general observation can now be made. In the limit as P becomes very large, the dominant terms are those involving the cumulative probabilities with the highest indices, and we can write:

$$f_E(\boldsymbol{\mu_1}) - f_E(\boldsymbol{\mu}_2) \approx \pi_{r-1}(\boldsymbol{\mu}_2)^P - \pi_{r-1}(\boldsymbol{\mu}_1)^P \approx P(p_r(\boldsymbol{\mu}_1) - p_r(\boldsymbol{\mu}_2))$$

Hence, in the limit, the superiority of μ_1 over μ_2 after P-parallelization is guaranteed as long as μ_1 has a better chance than μ_2 of finding the highest fitness.

Finally, we mention some illustrative calculations in the context of MAX-ONES with L=100. Space issues preclude a fuller display, but we note for example that for a parent with fitness 50, P-parallelized mutation at 9/L starts to outperform (in terms of expected gain) a similar parallelization of 1/L at P = 3. When fitness of parent is 80, the expected gain of P-parallelised 1/L is outperformed by that of 10/L at P = 362.

2.3 Adapting Rates in Parallel EAs Based on Expected Fitness Gain

A wider applicability emerges from recasting entries in the gain distribution in terms of numbers of point mutations and the fitness/distance correlation. That is:

$$p_{1+i}(m,x) = \sum_{j=1}^{L} d_j(m) \cdot c_{j,i}(x) \qquad (3)$$

in which we assume the operator under consideration is per-bit flip mutation on binary strings with rate m, and x stands for a specific parent, rather than a fitness. Meanwhile, $p_{1+i}(m,x)$ stands for the chance of a mutant of x having the i^{th} fitness better than that of *x*, while $d_j(m)$ gives the chance of the operator yielding a mutant *j* Hamming units distant from *x*, and $c_{j,i}(x)$ gives the proportion of mutants *j* Hamming units away from the parent which have the fitness indexed by *i*. The summation goes up to *L*, which is the highest Hamming distance attainable. Notice that:

$$d_j(m) = m^j(1-m)^{L-j}\binom{L}{j} \tag{4}$$

In particular, it does not depend on the landscape under consideration, while $c_{j,i}(x)$ expresses the detailed fitness/distance correlation map in the region of *x*, and does not depend on the mutation rate. Also, we reserve p_1 to stand for the following

$$p_1 = 1 - \sum_{i=2}^{H} p_i \tag{5}$$

Where *H* is the highest fitness attainable. Now, imagine the requirement to set suitable parameters for a parallel (1+*P*)-EA. By substituting equations (3) and (5) into (2) (via (1) and (4)), we can find the expected fitness gain per generation for any parent and any mutation rate, and we will suppose that a good rate to set per generation is one which maximizes this expected gain. However we need data for equation (3). The term $d_j(m)$ is analytically accessible, but $c_{j,i}(x)$ will generally be unknown. Data pertinent to it will normally be available, however, and we now propose a method for approximating $c_{j,i}(x)$ from online sampled fitnesses. This leads to a principled technique for adaptively resetting the mutation rate in such EAs after each generation. The dependence on straightforward bit-flip (in general, *k*-ary) mutation is partly a restriction on applicability, but also the key enabling factor, since this makes equation (4) available, which in conjunction with sampled data allows us to estimate gains for arbitrary rates, even though we may have sampled at only one rate.

We outline the approach first, and then set it out in detail. The essential idea is that mutation rate will be reset between generations based on expected gain. We assume, in the present work, a (1+*P*)-EA. In generation 0, *P* mutants of a randomly generated initial solution are generated; by the time we complete generation *g*, we have generated *gP* mutants, and have thus obtained *P* items of data from which to build an approximation of $c_{j,i}(x)$ for each of *g* parents *x*. This is used, together with equations (1–5), to find a good rate to use for generation *g*+1. A rather necessary further approximation stems from the fact that our sample model of $c_{j,i}(x)$ is silent with regard to individuals which are fitter than the current best – but the current best is (in a (1+*P*)-EA) the parent from which we will be generating mutants. To get around this, we set the mutation rate one generation 'out of phase'. Another difficulty is that we do not yet have a direct analytical route to find the *m* with maximal expected gain; how we deal with this and other issues is set out in the pseudocode description which follows. Before that, some further notation will serve to clarify the way that we handle approximations to $c_{j,i}(x)$.

Given an arbitrary problem, but assuming a binary encoding, and per-bit flip mutation, let $c_{fj,g}$ stand for the proportion of individuals which have fitness *g* among those which are *j* Hamming distant from an individual with fitness *f*. Notice that $c_{j,i}(x)$ is generally an approximation to $c_{j,g}(x)$ for any given *x* with fitness *f*. In cases such as

MAX-ONES (and many others, including some classes of rugged landscapes) the approximation is exact, but in general note that, where $X = \{x \mid f(x) = f\}$:

$$c_{f,j,g} = \frac{1}{|X|}\sum_{x\in X} c_{j,g}(x)$$

i.e. it is an average over all x with the same fitness. Intuitively, we might expect the approximation to improve with j. Next we define: $n_{f,j,g}$ to be the number of points *sampled* by a search algorithm which have fitness g, are mutants of a point with fitness f, and are j Hamming distant from their parent. By also defining $s_{f,j}$ as the total number of samples found so far which are j units distant from a parent with fitness f, we can now note that the operational approximation to $c_{f,j,g}$ is: $n_{f,j,g}/s_{f,j}$. We simplify matters by assuming a modestly-sized integer range.of fitnesses. In some cases in practice, however, it may be pragmatic (at least for the purposes of the calculations for setting the mutation rate), to map the range of fitnesses found so far onto a limited number of 'fitness levels', each standing for a fixed range, e.g. J_3 may capture all fitnesses between 0.7 and 0.8.

Now we can describe our routine for adaptively setting mutation rates in a fully-parallel (1+P)-EA. We assume that the time between generations (fitness evaluation) is significant enough for us to ignore the modest overhead in this routine.

1. Initialise: start with a randomly generated initial solution (our 'best-so-far' b), and set an initial rate m . Initialize z values $s_{i,j}$, for i from 1 to z, and reserve space for z^2L values $n_{f,j,g}$, initialized to 0.
2. Generate: Produce a set $M = \{m_1,m_2,...,m_p\}$containing P mutants of the best-so-far solution, and evaluate the fitness $f(m_i)$ of each m_i.
3. Calibrate: We now have P items of data with which we can improve (or initially construct) an approximation to $c_{f(b),j,g}$. For each individual m_i:
   Where h is the Hamming distance between b and m_i, increment $s_{f(b),h}$ by 1.
   If $f(m_i) > f(b)$, increment $n_{f(b),h,f(mi)}$ by 1.
4. Adapt: We now reset the mutation rate as follows, essentially by calculating what the best rate 'should' have been in the current generation, and setting that for the next generation.
   With reference to equation (5), approximate $p_{1+i}(m,b)$, for i >1, by setting $c_{j,g}(b) = n_{f(b),j,g}/s_{f(b),j}$ for all $g > f(b)$ and all j up to and including the most distant mutants of b which have been sampled. Then, assigning $i \in \{1,2,3\ldots\text{H}\}$ for convenience, such that fitness i is the ith in a ranked list of fitnesses in improving order starting with $f(b)$, calculate:

$$p_{1+i}(m,b) = \sum_{j=1}^{L} d_j(m)\cdot n_{1,j,i}/s_{1,j} \quad \text{for } i > 1 \text{ and then} \quad p_1 = 1 - \sum_{i=2}^{H} p_i$$

for each of a range of values of m from $1/L$ to $10/L$.

Using the results, and equations (1–6), we can then calculate $f_E(m)$ for the P-parallelised version of each rate m. Although requiring precision, the calculations are essentially straightforward and speedy, and arbitrarily many rates may be tried (e.g. 100), within an arbitrary range which perhaps goes beyond $10/L$. Finally, set m to be that rate which returned the best value of $f_E(m)$.

5. Book-keeping: At this stage we reset the best-so-far solution b to be the fittest individual from the set $M\cup\{b\}$.
6. Iterate: If a termination condition is reached, stop. Else, Return to step 2.

We will call this technique RAGE (rate-adaptation with gain-expectation)

3 Experiments

Here we report on preliminary testing of RAGE to establish proof-of-principle. We see its primary niche as being large-scale-parallel EAs, with limited numbers of generations. In these experiments we test the straightforward hypothesis that the theoretical basis of RAGE, and hence the justification behind each renewed rate setting per generation, should improve results over elementary methods. Later work will compare RAGE against other suitable mutation-rate adaptation techniques.

We used RAGE on four test problems: MAX-ONES with L = 100, and three simple deceptive problems with block sizes 3, 4, 5, with L = 90, 100, 100 respectively. For each, we experiment with two-scenarios: a (1+100)-EA run for 20 iterations, and a (1+1000)-EA run for 10 iterations. For each of these 8 cases, we try 10 versions of RAGE, differing only in the initial mutation rate in the first iteration (after which RAGE 'kicks in'), which ranged from $1/L$ to $10/L$ in steps of $1/L$. Our comparative technique is a straightforward fixed mutation rate throughout, again trialled for each of the 10 rates between $1/L$ and $10/L$. Each experiment was repeated for 50 trials.

Table 1. Comparison between RAGE and fixed-rates on MAX-ONES and Deceptive (block sizes 3, 4 and 5) using (1+100) and (1+1,000)-EAs

Problem	Method (population size)	RAGE vs Fixed	Best RAGE/ Best fixed	Best fixed rate
MAX-	(1+100), 20 gens	8 / 2 / 0	96.98 / 95.88	$2/L$
ONES	(1+1000), 10 gens	7 / 3 / 0	94.74 / 94.48	$4/L$
Deceptive,	(1+100), 20 gens	9 / 1 / 0	105.98/105.82	$2/L$
block size 3	(1+1000), 10 gens	7 / 3 / 0	104.54 / 104.02	$4/L$
Deceptive,	(1+100), 20 gens	6 / 4 / 0	105.14 / 105.06	$2/L$
Block size 4	(1+1000), 10 gens	7 / 3 / 0	106.65 / 105.84	$4/L$
Deceptive,	(1+100), 20 gens	9 / 1 / 0	103.6 / 100.66	$2/L$
Block size 5	(1+1000), 10 gens	7 / 3 / 0	101.52 / 102.26	$4/L$

Table 1 summarises the results, in the following way. Taking for example the row for the deceptive problem, block size 3, using a (1+100)-EA, column 3 summarises the results of 10 pairwise statistical comparisons, one for each mutation rate in the set $\{1/L, 2/L, \ldots, 10/L\}$. In the comparison for $3/L$, for example, a standard statistical test was performed comparing 50 trials of RAGE using $3/L$ as the initial rate, with 50 trials using $3/L$ as the fixed rate in each generation. We score 1 for a 'win' if RAGE was found superior with confidence at least 99%, 1 for a ‘loss’ if the fixed rate was superior, and 1 for a tie if the comparison was not conclusive. Column 3 adds these scores for each of the 10 rates. In column 4, the best RAGE mean result is shown (best of the 10 RAGE experiments with different initial rates) and is compared with the best fixed-rate mean result (best of the 10 fixed-rate experiments with different fixed rates). Column 5 indicates which rate gave the ‘best fixed-rate’ result in column 4.

Clearly, RAGE consistently outperforms fixed-rates, whatever the fixed rates are set at. The prospects for RAGE are therefore quite promising, Also, as generally expected, significantly higher rates than $1/L$ work best, increasing with population size, although there is too little data here on that topic to allow any further discussion Finally, though there is evidence that RAGE is a worthwhile technique, insufficient tests have been performed so far to establish it as a generally useful rate adaptation scheme. We discuss this point further below.

4. Concluding Discussion

By considering the concept of the ‘gain’ distribution of the per-bit flip mutation operator, we have been able to derive expressions which allow us to see how the gain distribution varies with mutation rate m and how it changes when the mutation operation is parallelised in the context of a $(1+P)$-ES. These investigations are particularly applicable to rate setting in parallel EA implementations, insofar as expected fitness gain is a good measure of the quality of an operator. By using online sample approximations to the exact expressions, we have proposed a routine called RAGE (Rate Adaptation with Gain Expectation), which is suitable for setting mutation rates on a per-generation basis in parallel EAs. Preliminary results show fairly convincingly that RAGE outperforms fixed-rate schemes for a wide range of fixed rates.

The background to the theory included here, and the subsequent proposed RAGE method, is the authors’ interest in finding a principled way to control mutation in very-large-scale-parallel evolutionary algorithms. The application of chief interest is Directed Evolution (as discussed in section 1.3), and the RAGE method can be used in that context. The DE application brings with it certain constraints and preferences which affect the choice of rate adaptation technique used (and EA technique in general) One major point is that very large populations are possible in DE, and consequently a considerable amount of appropriate landscape (fitness) data are available at each generation, so it would seem to be sensible to employ a technique which exploited this data as far as possible (rather than use, for example, deterministic fixed rates). A further issue is that using distinct and adaptive rates per individual (such as employed in modern evolution strategies) or distinct and adaptive rates per gene, are both (although ultimately possible) currently infeasible in the context of large-scale DE. The choice of adaptive mutation strategies in DE is thus practically limited to *per-generation* adaptation.

The essential point about the theory in section 2 is the ability to estimate the gain distribution of any mutation rate based on online-sampled landscape information seen to date (section 2.3). Calculating expected fitness gain is one way to exploit this estimated gain distribution, but ongoing work is exploring other methods, since expected gain *per se*, which is essentially an average, is likely to be unduly influenced by high probabilities for modest gains, hence perhaps unwisely favouring lower rates.

Finally, although we focus on a (1+*P*)-ES, the ideas underlying RAGE are certainly not restricted to this specific selection method. By building a model of landscape structure information as time progresses, RAGE-like adaptation can in theory be used to infer a promising rate with which to mutate any selected parent, via appealing to landscape information pertaining to the fitness of that parent. Developing similar techniques for *recombination* operators is less straightforward, however this is possible and is the topic of ongoing work.

Acknowledgements

We thank the BBSRC (the UK Biotechnology and Biological Sciences Research Council) for financial support, and Evosolve (U.K. registered charity number 1086384) for additional support during this work.

References

1 Arnold, F. M. (ed). Evolutionary protein design. *Advances in Protein Chemistry*, vol. 55. Academic Press, San Diego, 2001.

2 Arnold F. Combinatorial and computational challenges for biocatalyst design. *Nature* 2001;409:253-7.

3 Bäck T, Optimal Mutation Rates in Genetic Search, *Proc. 5th ICGA*, pp 2 – 9, 1993.

4 Bäck T, Evolutionary Algorithms in Theory and Practice, OUP, 1996.

5 Baltz, RH. Mutation in *Streptomyces*. In: Day L, Queener S, editors. The Bacteria, Vol 9, Antibiotic-producing *Streptomyces*. Academic Press, 1986:61-94.

6 Blickle, T., Thiele, L. (1995). A Mathematical Analysis of Tournament Selection, in L.J. Eshelman (ed.) *Proc. 6th International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 9–16.

7 Cantú-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers.

8 Fogel, D.B. and Ghozeil, A. (1996). Using Fitness Distributions to Design More Efficient Evolutionary Computations, in *Proceedings of the 3rd International Conference on Evolutionary Computation*, IEEE, pp. 11-19.

9 Mühlenbein, H. How genetic algorithms really work: I. Mutation and Hillclimbing, in R.Manner, B. Manderick (eds), Proc. 2nd Int'l Conf. on Parallel Problem Solving from Nature, Elsevier, pp 15-25.

10 Oates, M. and Corne, D. Overcoming Fitness Barriers in Multi-Modal Search Spaces, in Foundations of Genetic Algorithms 6 (2000), Morgan Kaufmann.

11 Rechenberg I, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart,1973

12 Voigt CA, Kauffman S & Wang ZG. Rational evolutionary design: The theory of in vitro protein evolution. In: Arnold FM, editor. Advances in Protein Chemistry, Vol 55, 2001:79-160.