The Microbial Genetic Algorithm

Inman Harvey

School of Cognitive and Computing Sciences

University of Sussex

Brighton BN1 9QH, UK

email: `inmanh@cogs.susx.ac.uk`

tel: +44 1273 678754

fax: +44 1273 671320

Submitted as a Letter to *Evolutionary Computation*

19th January 1996

Abstract

The genetic algorithm is simplified to a minimal form which retains selection, recombination and mutation. This introduces the idea of bacterial recombination, or infection, as a substitute for inheritance from parents.

The Microbial Genetic Algorithm

Abstract

The genetic algorithm is simplified to a minimal form which retains selection, recombination and mutation. This introduces the idea of bacterial recombination, or infection, as a substitute for inheritance from parents.

1 Background

Genetic Algorithms (GAs) come in many shapes and flavours, and it is of interest to work out the minimalist algorithm which still can be considered a GA.

I shall take the minimum requirements to be the use of a population, with **selection** such that later members of the population tend to have inherited genetic material from fitter ancestors; this genetic material to be subject to **recombination** and **mutation**. Evolutionary algorithms can use mutation only (without recombination, as in some Evolution Strategies) or recombination only (without mutation as in some Genetic Programming); but I am assuming that a GA uses both.

Classically this has been achieved through the use of a generational system. From one generation a parental pool is selected with probabilities based on their fitnesses, or scaled fitnesses, or ranking. Offspring are generated from pairs of parents, using 1-point, 2-point, uniform or some other variety of crossover. The offspring undergo mutations and then form the next generation.

Steady-state GAs (Whitley, 1989) relax the generational requirement, and produce just one (or two) offspring at a time from a pair of parents selected according to their fitnesses; the possibilities for recombination and mutation remain as before. Tournament selection then becomes an attractive option for those seeking minimalist systems. Each parent can be chosen by taking a random pair from the population, and selecting the fitter one. The probability of being selected as a parent is then linearly related to ranking, although there is typically much more variance in the chances of being selected than there are with conventional ranking algorithms (De Jong & Sharma, 1995)[1].

There are several variations on simple binary tournament selection (Harvey, 1993); the choice of who is to die is just as relevant as the choice of who is to be a parent. One interesting trick is to choose two parents *at random*, and use their recombined offspring to replace the *loser* of a tournament; this provides elitism for free if the two are different — the current fittest member can

[1] In that paper the variance is demonstrated experimentally, but it is also simple to show analytically that as tournament selection is a type of Poisson process, the variance in probability of selection is equal to the probability itself.

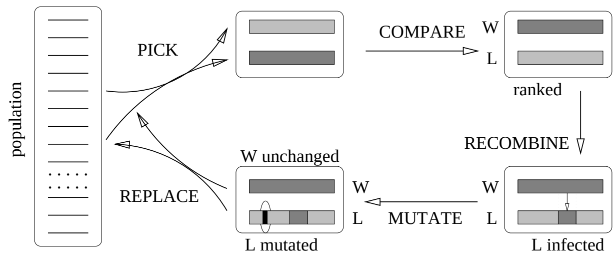

Figure 1: *The basic Microbial tournament: the winner of a randomly picked pair is unchanged, whilst the loser is infected by a proportion of the winner's genotype, and further mutated.*

never be replaced or mutated. It also avoids the need to have two tournaments, each providing one parent, and hence is a further step towards minimalism.

2 A Minimalist Tournament

However this method still requires the choice (at random) of a pair of parents, plus the further choice (at random) of a pair of tournament members so as to select the loser. Whilst trying on aesthetic grounds to simplify yet further, I realised that it was possible to collapse all this into one operation, using the *same* pair *twice*.

To produce and insert one new offspring, two members should be chosen at random from the population. From these parents an offspring is generated, using recombination and mutation; the less fit of the pair is then eliminated and replaced by the offspring. An alternative description of the same process is to say that a segment of the fitter member of the pair (one 'parent') is picked out by the recombination operator and copied into the corresponding segment of the less fit member (in its dual rôle as 'parent' and 'offspring'); the altered individual is then subject to mutation.

With this radical new method of collapsing several processes into a single tournament, the normal constraint of standard GA recombination (that the offspring should be equally likely to inherit genetic material from each of two parents) can be relaxed. Instead of insisting on an expected 50%, any proportion between 0% and 100% can be transferred from the winner of the tournament to the loser[2]. Mutation may then be applied to the loser; as the winner is unchanged we have elitism for free (unless winner and loser can be the same individual).

The metaphor of parenthood is difficult to sustain with this version of a GA; it makes more sense to say that the loser is infected with genetic material from the winner. It can be seen that nevertheless the Schema Theorem still applies.

[2] A distinction should be made between the rate of recombination used in conventional GAs (the probability of using recombination in reproduction), and the rate discussed here (the probability of recombinatorial transfer from a particular individual at any locus).

3 The Microbial GA

In the following C code we assume that the population is held in a 2-D array of binary characters, `gene[POP][LEN]`, indexed by position in population and locus on genotype. The function `evaluate(n)` returns the fitness of the n^{th} member of the population; `drand48()` returns a pseudo-random number between 0.0 and 1.0.

```
void microbial_tournament(void)
{
    int a,b,c,W,L,i;
    a=POP*drand48();                            /* TOURNAMENT PAIR         */
    do {b=POP*drand48();} while (a==b);         /* 2 different at random   */

    if (evaluate(a)>evaluate(b)) {W=a;L=b;}     /* SELECTION:              */
    else {W=b;L=a;}                             /* also ELITISM for free   */

    for (c=LEN*drand48(),i=0;i<SEG;i++)         /* RECOMBINATION(1)        */
        gene[L][(c+i)%LEN]=gene[W][(c+i)%LEN];

    for (i=0;i<LEN;i++)                         /* MUTATION:               */
        if (drand48()<MUT) gene[L][i]^=1;       /* bit-flip                */
}
```

Mutation here flips a bit with a probability `MUT` assessed independently at each locus. In the above version recombination copies a segment of length `SEG` from winner to loser, with toroidal boundary conditions, comparable to 2-point crossover. Alternatively, the probability of gene transfer from winner to loser can be made independent at each locus, comparable to uniform crossover but with a probability `REC` between 0.0 and 1.0.

```
    for (i=0;i<LEN;i++)                         /* RECOMBINATION(2)        */
        if (drand48()<REC) gene[L][i]=gene[W][i];
```

In this form, the parallels between the recombination and mutation operators are striking. This allows further simplification, particularly if elitism is sacrificed by not insisting that `a` and `b` are different. At the risk of crossing a boundary between elegance and freak-show, we have the ultimate Microbial Genetic Algorithm: a succession of tournaments of randomly picked pairs, with selection, recombination and mutation, in a single line of C code, here indented to fit on the page.

```
for (t=0;t<NUM_TOURNAMENTS;t++)
    for (W=(evaluate(a=POP*drand48())>evaluate(b=POP*drand48())?a:b),
        L=(W==a?b:a),i=0;i<LEN;i++)
            if ((r=drand48())<REC+MUT)
                gene[L][i]=(r<REC ? gene[W][i] : gene[L][i]^1);
```

This code is written for compactness rather than efficiency. As it stands, an individual is reevaluated every time it is picked for a tournament. This may be appropriate when using the GA for adaptive improvement in a changing environment, or in the presence of noise, but for deterministic optimisation it should be more efficient to evaluate an individual just once, when it first arises. Multiple calls to a pseudo-random number generator are expensive, and for small rates of the mutation and uniform-style crossover operators economies can be made by using the Poisson distribution to first stochastically set *how many* loci are to be affected, and only then to fix the *positions* of these few (usually a small number, often zero).

4 Bacterial Recombination

This form of recombination is related to bacterial conjugation, where segments of DNA are transferred between two members of a population. Lessons have been taken from the GA community into that of molecular biotechnology, in the context of DNA shuffling (Stemmer, 1994), and ideas going in the other direction may be fruitful. I am not aware of previous use of bacterial-style recombination for GAs, except that during the course of this work two other independent suggestions on related lines came simultaneously to my attention (Smith, 1996; Xxxx, 1996).

Such bacterial infection need not be directly associated with the selection process. I suggest a distinction between a 'Bacterial GA' where in any context there is genetic transfer from one individual to another that is pre-existing rather than 'newly-born'; and a 'Microbial GA' as a special case of Bacterial GA where such transfer is from the winner to the loser of a tournament — as in the minimalist version presented here.

The rate of recombination $\rho =$ `SEG/LEN` or `REC` also regulates the selection pressure σ (Harvey, 1993), which equals $e^{2\rho}$. This implies that a single new mutation conferring superior fitness in a large population can expect to have $e^{2\rho}$ copies after a number of tournaments equal to the population size. Hence different selective pressures can be chosen, and there is scope for ρ to change in value during the course of a run, or to have differing values within the population. There are interesting theoretical properties in the context of setting selection pressures for SAGA (Harvey, 1993), which will be pursued in a further paper.

5 Discussion

The Microbial GA conforms to the fundamental requirements of a GA, and indeed versions have been tested successfully with standard test problems. No claims are yet made as to what circumstances are most appropriate for its use, but rather it is presented here as an exercise in minimalism which provides new insights into the fundamentals of a GA. It is hoped that this will act as a spur for further investigation into infection in Bacterial and Microbial GAs.

Acknowledgments

Funding for this work has been provided by the EPSRC.

References

De Jong, K., & Sharma, J. (1995). On decentralizing selection algorithms. In Eshelman, L. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 17–23 San Francisco CA. Morgan Kaufmann.

Harvey, I. (1993). Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In Langton, C. (Ed.), *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI*, pp. 299–326. Addison Wesley.

Xxxx, X. (1996). Permission awaited to cite currently unpublished material. Under Submission.

Smith, P. (1996). Finding hard satisfiability problems using bacterial conjugation. In *Proc. of 1996 AISB workshop on Evolutionary Computation*. Under Submission.

Stemmer, W. (1994). Rapid evolution of a protein *in vitro* by DNA shuffling. *Nature*, *370*, 389–391.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116–121 San Mateo CA. Morgan Kaufmann.