AN INTRODUCTION TO GENETIC ALGORITHMS FOR NUMERICAL OPTIMIZATION

Solutions for Exercises of Section 5

Exercise 1:

So what we're doing is running `PIKAIA` without crossover, by setting `ctrl(4)=0`. Call this modified algorithm GA2-c, for "GA2 minus crossover". The Table below lists global performances for a series of GA2-c runs, together with equivalent GA2 runs (from Table 2 in Tutorial):

Table 5.1

Global Performance on test problems

Test Problem	Generation	GA2	GA2-c
P1	100	0.914	0.886
P2	100	0.883	0.855
P3	1000	0.191	0.138
P4	1000	0.845	0.903

the global performance of GA2 is slightly better than GA2-c on P1, very slightly better on P2, markedly better on P3 and markedly worse on P4. Meet once again the Relativity of Optimality Rule: like everything else in global optimization, the effect of crossover is problem dependent.

To complicate things ever further, the beneficial effects of crossover manifest themselves most strongly in the early phases of the evolutionary runs. Figure 5.1 below shows convergence curves (averages of $1 - f(x, y)$ over 1000 runs) for the P1 and P2 runs listed in the above Table. Clearly crossover greatly enhances the exploratory capabilities of the algorithm, as long as sufficient diversity is present in the population; if the population has converged, then crossover interchange indentical fragments of solutions, and so has no net effect. This leaves only mutation as a source of variability. In fact, if GA2 and GA2-c are run for 200 generations instead of 100, their global performances are essentially identical on P1, and GA2-c does very slightly better than GA2 on P2.

Exercise 2:

I have ideas on how to do this one, but haven't yet found the time to try them. The following are just a few hints. You are encouraged to try different things also.

- The key thing to realize is that strong selection pressure is needed when the population is widely distributed through parameter space, and lower pressure when

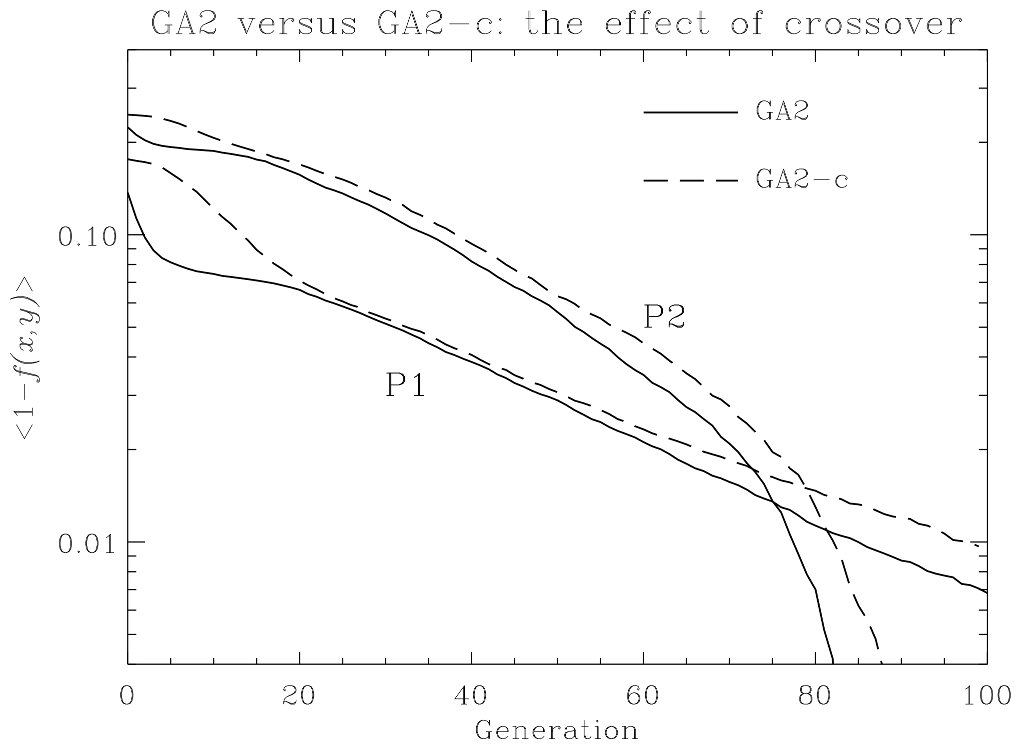

Figure 5.1: Convergence curves for GA2 and GA2-c on P1 and P2. Each curve is an average over 1000 separate runs.

the population has converged; this is the opposite requirement from adjustable mutation rate.

- You might just want to try writing a subroutine similar to `adjmut`, which adjusts the selection pressure parameter `fdif` in a similar manner. I would stick to `fdif`= 1 as an upper bound, and would not let `fdif` fall below 0.25. The increment in `fdif` should be small enough to ensure a smooth variation between these bounds, but still large enough to ensure that `fdif` responds rapidly (i.e., within 5 generations or so) to population convergence/divergence. I guess I would start with an *additive/subtractive* increment of 0.1, and see what happens.
- Selection pressure is one of the more complex genetic algorithm parameter, in terms of its interactions with other algorithmic components. If you come up with a "recipe" for adjusting `fdif` that seems to work with `PIKAIA`'s default settings, I would encourage you to try it with the other two basic reproduction plans provided with `PIKAIA`, namely steady-state-replace-random and steady-state-replace-worst. These plans are activated by setting the control vector element `ctrl(10)` to 2 and 3, respectively.

Paul Charbonneau, HAO/NCAR, August 1998.

E-mail questions or queries to: `paulchar@ncar.ucar.edu`