Description of Genetic Algorithm

General

The genetic algorithm takes as input a classifier system and other parameters listed below. Some of the classifiers, those with high fitness or large frequency, in the system are chosen for mating and reproduction and some, those with low strength or little use, are chosen for elimination. Thus, stronger classifiers which yield a high return are the survivors.

Reproduction of classifiers involves selecting pairs of "parents" via a roulette wheel. Stronger or more fit classifiers occupy a good percentage of the wheel and thus have a greater chance of being selected. A pair produces two offspring that have a portion of each parent's genes. Also, there may be some mutation of genes.

Keeping a constant number of classifiers in a system then entails replacing classifiers with new offspring. Only a subset of the classifier systems are considered for elimination. Then, a certain proportion (possibly all) are analyzed to find the "worst" (*i.e.* in terms of strength) among them. A pair considered to be the worst after some sampling is then replaced by a pair of new offspring.

Specific

The inputs to the program are as follows:

- CS = the classifier system; CS_c is the consume classifier system, CS_e is the exchange classifier (in program ctype=1 implies "e" ctype=2 implies "c")
- n_{select} = the number of pairs of classifiers chosen for reproduction
- p_{cross} = the probability that crossover occurs
- $p_{mutation}$ = the probability that mutation occurs
- crowdingfactor = the number of times that a search for the worst (*i.e.* has lowest strength) classifier is conducted in "crowding.m"
- crowdingsubpop = the proportion of the classifier system used in a search for the worst classifier
- n_{class} = the number of classifiers in CS
- $\ell_{cond} = 6$ if $CS = CS_e$ and 3 if $CS = CS_c$

- last = a matrix giving indices for the consume classifier systems for rewarding bids in period $t-1$ (when the current period is t)
- iteration = the current period (*i.e.* t)
- $\underline{\mathrm{S}}$ = the first criterion for choosing the subset of candidates for killing (in program, this is uratio(1))
- f_u = the second criterion for choosing candidates for killing (uratio(2))

Only a subset of the classifier system is allowed to be "eliminated." Classifiers with strength less that $\underline{\mathrm{S}}$ or classifiers that have been used less than $f_u max$ times are those that are candidates for elimination. Here, max is the maximum number of times that any classifier has been used. If the number of classifiers to be replaced (*i.e.* $2n_{select}$) exceeds the number satisfying these constraints, then n_{select} is adjusted downward.

Given that n_{select} is positive, we then do the following steps n_{select} times:

- spin a roulette wheel $p_{used}n_{class}$ times without replacement choosing candidate classifiers for mating according to the number of times they have been used; thus, we choose a certain proportion of the most used classifiers as candidates to be parents
- spin a roulette wheel 2 times with replacement to get a pair of parents
- generate a random number from a uniform distribution; if the random number is less than p_{cross} then "crossover" in reproduction will occur
- if there is crossover, then choose a number between 1 and 6 if $CS = CS_e$ or between 1 and 3 if $CS = CS_c$; call this number j_{cross}
- let reproduction by our 2 parents of 2 new children occur as follows: child k has the same genes (binary bits) in bit positions 1 through j_{cross} as parent k and the same genes in bit positions $j_{cross}+1$ through 6 or 3 (depending on the classifier type) as parent ℓ, $k, \ell = 1, 2$, $k \neq \ell$; the action bit (7 in CS_e and 4 in CS_c) is the same in child k and parent k, $k = 1, 2$; for example, if the parents are:

  $$\begin{array}{ccccccc} \# & 0 & 1 & \# & 1 & 0 & 1 \\ 1 & \# & 0 & 0 & 1 & \# & 0 \end{array}$$

  and the crossover point is 2 then the children are:

  $$\begin{array}{ccccccc} \# & 0 & 0 & 0 & 1 & \# & 1 \\ 1 & \# & 1 & \# & 1 & 0 & 0 \end{array}$$

if crossover occurs, increase n_{cross} by 1

- generate 7 or 3 random numbers from a uniform distribution if $CS = CS_e$ or $CS = CS_c$, respectively; if the k^{th} random number is less than $p_{mutation}$ then "mutation" will occur in gene position k
- if there is mutation in a "condition" gene then randomly change a # to a 0 or 1, a 1 to a # or 0, or a 0 to a # or 1; generation of the alternatives occurs with equal probability; if mutation occurs, increase $n_{mutation}$ by 1
- choose a candidate (from those eligible) to be replaced by one of the children as follows: pick crowdingsubpop random numbers (from uniform distribution) between 1 and n_{class}; let these be the indices of the reduced classifier system (*i.e.* with only those eligible for killing included) over which we find the one with the worst strength; determine how many similar condition bits and dissimilar action bit(s) there are between this low strength classifier and the child who will be replacing; store the index and the number of matches for this "worst" classifier; repeat the procedure of choosing random numbers, etc. crowdingfactor times; the classifier to be killed is the one with the overall "worst" strength from the crowdingfactor samples

When we are done replacing bad classifiers with new (hopefully fitter) classifiers, we report statistics and rescale the strengths of the classifier system if any had been negative upon entering.