Genetic Algorithm, Prisoner's Dilemma and Parallel Genetic Algorithm

Wenchong Chen

1. Trap of Structure Usage

It is allowed to assign a structure variable A to another structure variable B of the same type using the statement "MyStruct B = A;". It will copy the values of A's members to B respectively. Even when one of the members is an array, this operation will be successful. However, be very careful when there is a pointer as a member, "pGenes" for example. It will simply copy the address to which A.pGenes points to B.pGenes. If you only want to copy the value that A.pGenes points to instead of the address, you should allocate memory to B.pGenes first, then copy the value.

The mistake I made was in the select_parent() function, when I wanted to make a copy of the old generation so that new generation can be selected based on the old generation. But I didn't allocate memory to the pointer member and I copied the structure directly. Since the original variable was overwritten all the time, the copy was changing accordingly. This made the total fitness going up and down, which could mislead people to check mistakes in the crossover process.

2. Representation of Gene, Chromosome and Group

We can see from the following figure that a group contains a set of chromosomes, and each chromosome has a set of gene segments. Each gene segment is an unsigned char type, which can represent 8 genes. This representation saves more storage space than that of using int. It can also represent chromosomes containing any number of genes by using mod operation. For simplicity, my code only handles the situations where the number of genes is a multiple of 8 (the size of the unsigned char type).

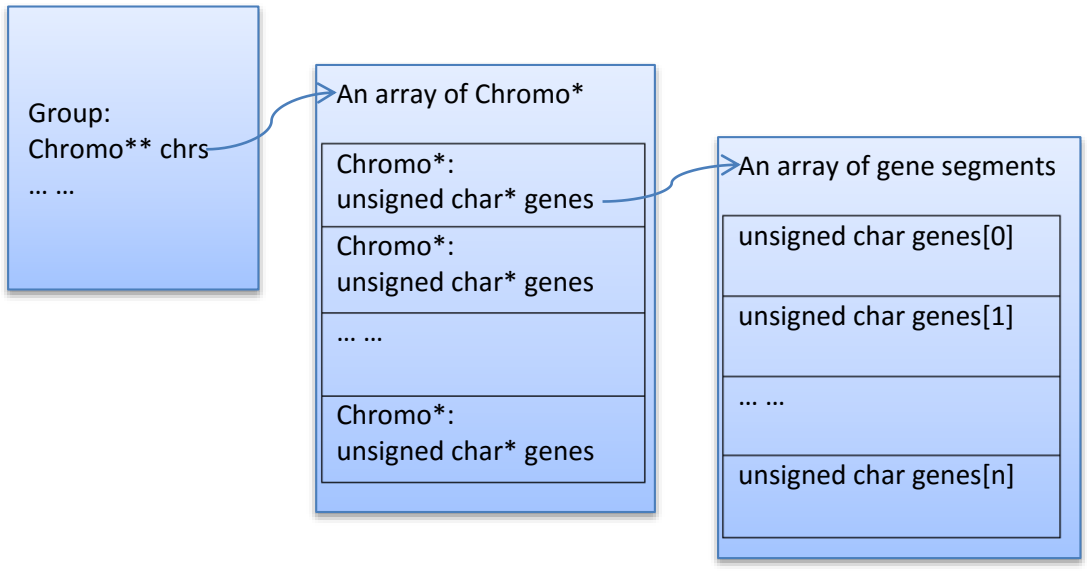

Figure 1 Data Representation of Gene, Chromosome and Group

3. Test Results of Genetic Algorithm

The following graphs show the total fitness of the whole group over time, with 8 chromosomes of 2 gene segments (16 genes). The crossover rate was set to 0.95 and the mutation rate was 0.001. Both graphs have small peaks and troughs, which is an effect of random selection, crossover and mutation. But the trend of total fitness is going up.

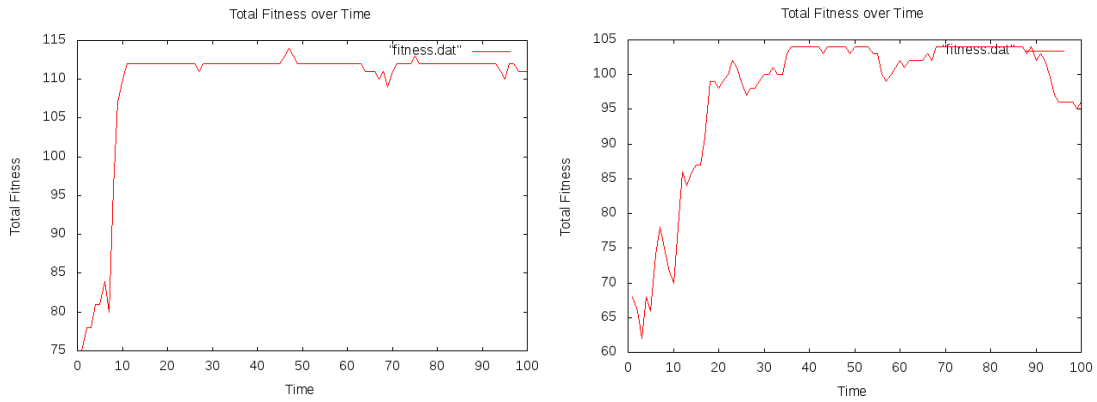

Figure 2 Total Fitness over Time

4. Prisoner's Dilemma using GA

It's important how we represent and store the rules and history tactics.

The game has four rules, and each rule has two scores for player A and B respectively. In my implementation, the two scores for A and B in pair are stored adjacent to each other. For example, for combination "DD", the score for A is rule[0]=1, for B is rule[1]=1. For combination "DC", the score for A is rule[2]=5, for B is rule[3]=0, etc.

The history tactics for A against B or B against A are stored only once, with history[index][0] for A and history[index][1] for B. This index can be calculated by

$$index = \frac{(2 \times num\_players - A - 1) \times A}{2} + B - (A + 1)$$

where A and B are the indexes that they are stored in the chromosome array.

5. Parallel GA

The key of the parallel genetic algorithm using Master-Slave method is to allocate fitness calculation tasks. Since the genes are represented by unsigned char type, it cannot send negative numbers to slaves to tell them no more jobs, I send genes with all-zero-value gene segments to represent no more jobs instead.