(Summary of the course:)
Introduction to Evolutionary Computations

Akira Imada
Brest State Technical University
e-mail: akira@bstu.by

1 Introduction: What are Evolutionary Computations?

I start the lecture by explaining

- What on earth are Evolutionary Computations, what for, and how?

using a simple example of

- Evolution of weight configurations of a Feed-forward Neural Networks,

expecting audiences to understand, and more importantly, become interested in, the principal idea of Evolutionary Computations (ECs). EC is a category of algorithms analogous to, or inspired by, biological Darwinian evolution. That is, it employs the survival-of-the-fittest principle. In this section, I will explain how a set of somehow already familiar terms, such as *chromosome (genome), gene, allele, phenotype, genotype, recombination, crossover, mutation, fitness, population, generation* and so forth, do mean in the algorithm, and what kind of roles they play. The explanation here will be instinctive rather than being theoretically rigid. A little familiarity about the concept of NNs, especially the one that solves some simple Boolian function like AND, OR and XOR is preferable, but not necessarily. Although these are just toy examples, I hope it's interesting enough to trigger the audiences' curiosity hereafter.

2 What are ECs? — A little more in detail.

To show ECs are different from a simple random search, I'll compare ECs with Random-Mutation-Hill-climbing (RMHC), one of the random searches. Then I'll show a fitness distribution of a typical problem that can be easily solved by ECs but very difficult by RMHC. Some of such problems are like searching *"needles in a haystack"* and it is these problems for ECs to be worth applied. Implementation of ECs are rather easy. All we should design is (1) How we represent the problem (or equivalently, candidate solutions to the problem) by chromosomes? and (2) How do we evaluate fitness (the degree to how good each chromosome performs)? That's almost all there is to it, which will be emphasized here. Various schemes of selection: *Roulette-Wheel (fitness-proportionate), Truncate, Tournament Selection*, etc, and recombinations: *One-point, Two-point, and Uniform Crossover* will be given here too.

3 And Beyond — Why do ECs work?

When we recall our childhood, we used to play with *building blocks*, don't we? To make a castle, for instance, we combined small building blocks into larger building blocks. Here,

in the context of EC, building blocks are shorter pieces of an overall solutions. And a metaphor of *features* that all beings tend to inherit from parents, such as good features of *wildcat's sharp teeth*. By combining features from two good parents we can expect crossover to produce even better children. Sometimes, crossover may recombine the worst features, but if so, children will be less likely to survive. By iteratively combining, most likely to survive will be good building blocks.

In this section we will study very familiar two concepts

- Building Block Hypothesis
- Schema Theorem

which were originally given by Holland who proposed the GA in 1975.

Schema is a string which includes a symbol ♯ implying *don't-care* whatever symbol be the position. For example (11♯♯♯♯) is a schema that instantiates (110011), (110100) $\cdots$ etc. What this implies is that the important genes to specify some specific feature like *sharp teeth* is the first two one's. If a particular schema gives high fitness values to its instances, then the population is likely to converge on this schema, and once it so converges, all offspring will be instances of this schema. Thus, *crossover* scatters the building blocks throughout the population. and, as the population converges, the search becomes more and more focused on smaller and smaller subspaces of the entire search space. The concepts are more formally explained in the lecture.

4 Neuronal Darwinism

Neuronal Darwinism is the term proposed by Edelman, Nobel laureate, ascertaining that there is yet another *evolution in our brain at the neurons' level* besides the usual Darwinian evolution at species' level. This section is from this aspect. However, since this lecture is not regarding neither NN nor brain science, we just overlook this topics only as an application of ECs.

In the previous example of feed-forward NN, our chromosomes were made up of continuous value as *alleles* (possible value of genes). Here we use *binary genes*, which is of rather typical case. Here, we study *Associative memory* which is sort of like a model of human memory, in the sense that it recalls stored patterns from imperfect stimuli. Associative memory has been realized by fully-connected *(Hopfield-type)* neural network model. It learns patterns usually by *Hebbian Learning Algorithm* to memorize. However, one of the drawbacks are its small storage capacity. Once we studied and reported that the storage capacity is enhanced by pruning some of the synapses in which an EC was used to determine which synapses are to be pruned. For the purpose, binary cromosomes in which "zero" indicates to prune the corresponding synapse and "one" indicates to intact it were used. I hope this is a good example to understand how usual binary chromosomes are exploited, besides topics per se is very interesting. Though, in the previous section, I mentioned that basic knowledge of feed-forward NN should be required, the concept of fully connected NN will be given here, and no need to study it in advance.

5 NP-hard Combinatorial Optimization Problem

Hereafter, for the time being, we learn how *combinatorial optimization problems* could be attacked by using ECs. As in the previous section, we use binary chromosomes in most

cases to solve this category of problems. We will learn here (1) *Knapsack Problem*, and (2) *Traveling Salesman Problem (TSP).* Usually in most real-world problems, it would be enough to obtain a near optimum solution instead of exact one. We, using an EC, search for such near optimum solutions to large scale NP-hard problems which are actually impossible to be approached by analytical methods.

6 Exploitation of Diploidy Chromosomes

As an example of more biologically plausible evolutions, we will try to exploit *diploidy chromosomes* (a pair of chromosomes) instead of so-far-explained *haploidy chromosomes* (single string of chromosome). The target problem here is

- Sorting Network Problem.

Sorting is a problem familiar for everyone who learns computer programming. When we are to sort a number of items, for example, to sort N integers in descending order, we compare two items one by one, and swap them each time if necessary. Then the question is what will be the minimum number of comparisons to sort all items out. Let's take an example of 16 items. In 1962, Bose and Nelson declared the minimum number to be needed was 65. But in 1964, different algorithms with minimum comparisons of 63 were found by Batcher, and independently by Floyd and Knuth, It had been the minimum number of comparisons until 1969 when 62 was claimed to be the minimum by Shapiro. And in the same year, 60 was claimed by Green. However it has not been proved that this is the minimum up until now. In 1992, Hillis explored the problem using an EC in which diploidy chromosomes were elegantly employed. Here we will learn his excellent method.

7 Evolutionary Game Theory

In this section the problem of

- Prisoner's Dilemma

will be studied. The problem is as follows. Two arrested prisoner A and B are offered a deal: If A confesses and B does not, A will be forbidden and B will get 5 years in jail, and vice versa. If both confess, then both will get 4 years in jail. If both do not they will each get 2 years. So, this is dilemma isn't it? An iterated version of this problem has been studied by ECs. That is to say, when the prisoner's dilemma is iterated what is the best strategy to obtain the maximum reward? It is known that the strategy called "Tit-for-Tat", namely, always respond with the same action as the opponent is the optimum strategy. And what EC's found is...

8 A Visualization of High-D space — Summon Mapping by EC

Dimension reduction is an important technique for visualization of high dimensional space. The *Sammon Mapping* is one of these techniques. EC can make it by mapping a set of N points in n-dimensional space to 2-dimensional location data so that the distance information is preserved as much as possible, or we might paraphrase this as, *so that the n-dimensional distances are approximated by 2-dimensional distances with a minimal error.* This problem is somewhat of an old optimization problem and nowadays this could be easily solved by using ECs, As a matter of of course, in the sense of near-optimum solutions.

9 NN Revisited — Can we evolve not only the weights but also its architecture?

Again an application of ECs to NNs. When we are to evolve NNs, A question would arise: Can we also evolve the structure of NN? That is, can we evolve its architecture as well as weight values? The answer is "yes". Here I'm going to show one of the methods out of many so far proposed. Usually chromosomes for the purpose are tricky more or less, and that's why we have prolonged this very interesting topic up until this moment. We now are ready for that, aren't we?

10 Lamarckian Inheritance & Baldwin Effect — Not Biologically Plausible but ...

Once Lamarck believed that acquired characteristics during individual's lifetime could be passed to its offspring. And Baldwin thought that although results of learning of individuals during their lifetime do not change their chromosomes, learning affects the selection after fitness evaluation. Nowadays they are called Lamarckian Inheritance and Baldwin Effect, respectively. As subtitle of this section suggests, modern biologists do not believe that both of the Lamarck and Baldwin's idea occur in real biological evolution. But in an artificial evolution inside computer, these ideas sometimes give a great efficiency. Here we will study how each of these ideas is implemented in ECs.

11 Search for Multiple Peaks Simultaneously

Sometimes multiple optimal solutions exist and what are of interest is not one of them but all of them. Typically, ECs converge one of these solutions, although the solution obtained in each run might be different from run to run. I will introduce some of the techniques to locate multiple solutions simultaneously at a run which is called *multi-modal optimization*.

In biological environment species tend to live in their own *niche* sharing resources there and restrict mating within them. The algorithms we will learn here were proposed by borrowing this analogy of the natural environment. Three categories of such algorithms we will learn here are called

- Niching Method
- Crowding Method
- Speciation Method

12 Multi-objectives Optimization

On the other hand, sometimes we have multiple criteria in evaluating which individuals are better than others, and usually some of the criteria are trade-off. That is, we sometimes have multiple fitness functions some of which conflict others. Assume we are looking for the optimal point $\mathbf{x}$ in the search space. When a new point increases all these fitness functions than the old point, then the new point is said to *dominate* the old point. If there's no such new point anymore, the point is called *non-dominated* or *Parate optimum*. This section will show how EC searches for these Parate optimal points.

13 Variations of EC's

So far explained ECs are mainly the ones that are referred to as GAs. That is, those evolve a population of binary, sometimes continuous though, haploid chromosomes under roulette-wheel/truncate/tournament selection with using both one/two/uniform crossover and bit-flip mutation in the case of binary chromosomes, and random replacement in the case of continuous chromosome. Evolution Strategy (ES) and Evolutionary Programming (EP), on the other hand, basically evolve continuous chromosomes. In addition, mutation is by adding a small Gaussian random number to each of the genes, and the other more important difference is that the amount of the mutation is adaptive. To be more specific, standard deviations of the random Gaussian numbers to be added are modified from generation to generation adaptively. In most cases they become smaller and smaller as generation proceeds and as individuals approach the solution. EP employs this mutation alone without crossover, while ES uses crossover besides this adaptive mutation. Genetic Programming (GP) evolves a population of tree structures, typically LISP programs. So, GP might be said to directly evolve programs themselves.

14 Commonly used Test Functions — to learn more about EC's

To learn how an EC converges to the optimum, how it avoid local optima, why it cannot converge to the global optimum, how individuals remain each niche in multi-modal EC and so on, we have a couple of commonly used test functions each of which has specific characteristics as for its optimum. In this section, we learn these test functions defined both on a domain of continuous and binary of arbitrary multiple dimensional domain.

15 There's no free lunch

We have a theorem given somewhat of a peculiar name. This No Free Lunch Theorem states:

> *All algorithms that do not resample points from the search space perform exactly the same when applied to all possible problems and averaged the performance.*

So, we have to be careful when we want to assert "this algorithm performs better than that algorithm."

16 Summary and Conclusions

17 Related Web-pages