Chapter 6: Introduction to Genetic Algorithms

- Download CD Content

In this chapter, we'll look at simulated evolution as a way to solve computational problems. The genetic algorithm, developed by John Holland [Holland 1962], is a search algorithm that operates over a population of encoded candidate solutions to solve a given problem. To illustrate the use of the algorithm, we'll use evolutionary computation to evolve sequences of instructions that represent a simple algorithm. This is otherwise known as genetic programming as defined by John Koza [Koza 1992].

Biological Inspiration

The genetic algorithm is an optimization technique that simulates the phenomenon of natural evolution (as first observed by Charles Darwin). In natural evolution, species search for increasingly beneficial adaptations for survival within their complex environments. The search takes place in the species' chromosomes where changes, and their effects, are graded by the survival and reproduction of the species. This is the basis for survival of the fittest—survival and the passing on of these characteristics to future generations. Survival in nature is the ultimate utility function.

a. 문제 정의 및 인코딩
b. 모집단 구성
c. 적합도 계산
d. 선택 연산
e. 유전 연산
종료 또는 b로 이동

이 중에서 사람이 관여하는 단계는 a

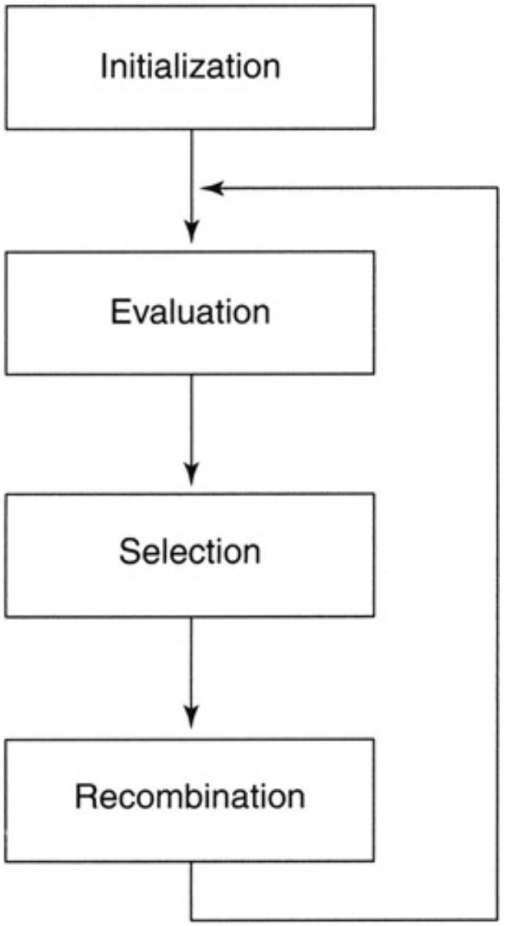

Figure 6.2: Genetic algorithm high-level flow.

6장 유전 알고리즘; 모집단 생성 population

1

2

3

4

6장 유전 알고리즘; 선택 selection

7/31=0.23

10/31=0.32

9/31=0.29

5/31=0.16

6장 유전 알고리즘; 재생산 recombination(교차 crossover , 돌연변이 mutation)

1′

2″

3′

4′

문제 정의 및 인코딩

유전 알고리즘을 적용할 수 있는 형태로 문제를 재구성하는 단계다. 최적화는 문제의 답을 찾는 일이고, 유전 알고리즘은 잠재적 후보 해를 생성한 후, 진화를 통해 최적의 해결책을 생성한다. 진화는 선택과 유전으로 작동한다. 생명은 DNA로 자신을 표현하고 자연은 그 DNA로 만들어진 생명의 적합도를 평가한다. 그렇다면 잠재적인 문제의 답 즉 후보 해를 DNA 형태로 표현하는(인코딩) 방법과 그 후보자의 적합도를 측정할 수 있는 수단(적합도 함수)을 설계해야 한다.

문제 정의: 神 라면 개발

자 다시 "神 라면"이다. 앞에서는 원리를 설명하기 위해 그림을 사용했지만, 컴퓨터를 이용해서 유전 알고리즘을 적용하려면 컴퓨터가 다룰 수 있게 표현해야 한다. 설명의 편의를 위해 라면에 첨가할 수 있는 재료의 종류를 MSG, 계란, 파, 김 4가지로 제한하자. 그렇다면 라면이라는 종을 DNA로 나타내면 다음과 같이 나타낼 수 있다. 예를 들어, 라면 5는 달걀과 김이 들어간

Sample Problem

Now that we have a basic understanding of the genetic algorithm, let's use it to solve what will be an unconventional optimization problem. Rather than focus on the traditional numerical optimization problems, let's try to optimize a more symbolic problem of sequences of instructions (under the genre of evolutionary programming).

문제 정의: Consider a simple instruction set for a zero-address architecture (a stack architecture 스택 머신). Our virtual computer has no registers, only a stack for which instructions can manipulate the values on the stack. Our virtual computer recognizes only a handful of instructions; these are shown in Table 6.1.

DUP 복사	Duplicate the top of the stack (stack: A => A A)
SWAP 교환	Swap the top two elements of the stack (stack: A B => B A)
MUL 곱	Multiply the top two elements of the stack (stack: 2 3 => 6)
ADD 합	Add the top two elements of the stack (stack: 2 3 => 5)
OVER 밑 복사	Duplicate the second item on the stack (stack: A B => B A B)
NOP -	No-operation (filler)

DUP
MUL

Solution Encoding 인코딩 Recall that we encode the solution of the problem, not the problem itself (since the problem is used as the fitness measure for our chromosomes). This problem is quite easy, since our chromosome simply represents a string of instructions in a contiguous block. We assign numerical values to each of the instructions (DUP = 0, through NOP = 5). Our encoding is then a contiguous string of bytes representing the stream of instructions.

Fitness Evaluation 적합도 평가

Evaluating the fitness of a given chromosome is a simple process of executing the string of instructions that are contained within the chromosome. We prepare a simple stack of some depth, load the initial values onto the stack, and then execute each instruction sequentially until the program reaches the end, or the END instruction is reached.

The fitness value is then calculated as the difference between the resulting value on the stack to what was expected per our objective function (predefined for the test).

Recombination 재생산 For this problem, we'll use both crossover 교차 and mutation 돌연변이. The crossover operator simply breaks an instruction stream at a single point, and swaps the tails of each parent. Mutation is a random reassignment of the gene with a new instruction (since the chromosome represents the program, each gene is a single instruction).

Source Discussion

The source code for the genetic algorithm, as well as the virtual computer that evaluates the chromosomes, is very simple. Let's first look at the virtual computer implementation (otherwise known as a virtual machine).

Virtual Machine Implementation

The virtual machine (or VM) is a very simple architecture that supports a single integer stack and six instructions that use operands on the stack. Listing 6.1 shows the VM symbolics (from common.h) as well as the virtual machine (stm.c).