Chapter 15
LZW Encoding in Genetic Algorithm

Worasait Suwannik
Kasetsart University, Thailand

ABSTRACT

To solve a problem using Genetic Algorithms (GAs), a solution must be encoded into a binary string. The length of the binary string represents the size of the problem. As the length of the binary string increases, the size of the search space also increases at an exponential rate. To reduce the search space, one approach is to use a compressed encoding chromosome. This paper presents a genetic algorithm, called LZWGA, that uses compressed chromosomes. An LZWGA chromosome must be decompressed using an LZW decompression algorithm before its fitness can be evaluated. By using compressed encoding, the search space is reduced dramatically. For one-million-bit problem, the search space of the original problem is $2^{1000000}$ or about $9.90x10^{301029}$ points. When using a compressed encoding, the search space was reduced to $8.37x10^{166717}$ points. LZWGA can solve one-million-bit OneMax, RoyalRoad, and Trap functions.

INTRODUCTION

A Genetic Algorithm (GA) is an algorithm inspired by natural evolution (Mitchell, 1998). To solve a problem using GA, a candidate solution is encoded as a chromosome. Normally, a chromosome is encoded as a fixed length binary string. GA searches in the space of this representation. The length of a chromosome is related to the size of the search space, for l-bit chromosomes the search space is 2^l points. When l is large, the computational time becomes very long.

DOI: 10.4018/978-1-4666-3628-6.ch015

There are some approaches to reduce a search space. One approach is to apply a heuristic in an evolution process. For instance, the specific type of crossover is introduced to preserve some constraints can beneficially reduce the search space (Chen & Smith, 1999). The result shows that the proposed crossover can find better solution for a flow shop scheduling problem.

Another approach to reduce the search space is by using compressed encoding. Compressed GA employed compressed encoding chromosome using a format similar to run-length encoding (Suwannik, Kunasol, & Chongstitvatana, 2005). The result shows that Compressed GA uses 805

times less fitness evaluations than Simple GA when solving 128-bit OneMax problem. In c^2ga, the compressed encoding was combined with compact genetic algorithm (Watchanupaporn, Soonthornphisaj, & Suwannik, 2006). The performance of the c^2ga is better than cGA (Harik, Lobo, & Goldberg, 1999) in OneMax and RoyalRoad problems.

To use Compressed GA, an appropriate number of bits of the repetition times (the run length) has to be specified. If the number of bits is too low or too high the effectiveness of compression is suffered. To overcome this problem, Kunasol, Suwannik, and Chongstitvatana (2006) proposed LZWGA. LZWGA uses a compressed encoding that can be decompressed using Lempel-Ziv-Welch (LZW) decompression algorithm. The result shows that LZWGA outperforms Compressed GA for 2048-bit OneMax problem. LZWGA is used to solve one-million-bit OneMax, Royal Road, and Trap problems. The one-million bit problem has an enormous search space. The search space of this problem is $2^{1000000}$ or 9.90×10^{301029} points. Solving the problem of this size using any canonical GA is not practical. Using LZWGA, the search space is reduced dramatically. LZWGA can solve one-million-bit OneMax problem in 18 minutes.

This paper summarizes recent researches on LZWGA, which cover various aspects of the algorithm such as selection, crossover, and mutation. This paper is organized as follows. The next section describes LZWGA. The test problems are then explained. The results are reported on selection, crossover, and mutation respectively and a new genetic operator called Shift is described. The final sections provide discussion and conclusions.

LZWGA

The main difference between LZWGA and Simple GA is that a chromosome is in a compressed format. The LZWGA chromosome has to be decompressed before its fitness can be evaluated. The pseudo code of LZWGA is shown in Figure 1. The algorithm begins by creating the first generation of compressed chromosomes. Before evaluating the fitness of a chromosome, the compressed chromosome is decompressed using LZW Decompression algorithm. The fitness evaluation is performed on the uncompressed chromosome. After that, the new population is created to replace the old population. The algorithm repeats the process of decompression, fitness evaluation, and creating a new population until the termination criterion is met. The algorithm terminates when a solution is found or a maximum generation is reached.

A. Creating the First Generation

Unlike a canonical GA, a chromosome in LZWGA is encoded as integers. The chromosome in LZWGA is in a compressed format. Each integer is a code for an index of an entry in the dictionary.

Figure 1. LZWGA pseudo code

```
Algorithm LZWGA
Z←create_first_generation ()
repeat
        P←decompress(Z)
        evaluate(P)
        Z←create_next_generation(Z)
until is_terminate()

A variable Z is the population of compressed chromosome.
A variable P is the population of uncompressed binary chromosomes.
```