Comparison between Genetic Algorithms and Particle Swarm Optimization

Russell C. Eberhart and Yuhui Shi
Department of Electrical Engineering
Indiana University Purdue University Indianapolis
723 W. Michigan St., SL160
Indianapolis, IN 46202
eberhart,shi@engr.iupui.edu

Abstract

This paper compares two evolutionary computation paradigms: genetic algorithms and particle swarm optimization. The operators of each paradigm are reviewed, focusing on how each affects search behavior in the problem space. The goals of the paper are to provide additional insights into how each paradigm works, and to suggest ways in which performance might be improved by incorporating features from one paradigm into the other.

Introduction

Four well-known paradigms currently exist in evolutionary computation: genetic algorithms [5], evolutionary programming [4], evolution strategies [9], and genetic programming [8]. A new evolutionary computation technique, called particle swarm optimization (PSO), inspired by social behavior simulation, was originally designed and developed by Eberhart and Kennedy [2,3,6,7]. In PSO, instead of using more traditional genetic operators, each particle (individual) adjusts its "flying" according to its own flying experience and its companions' flying experience.

This paper compares genetic algorithms and particle swarm optimization. Operators that are used by each paradigm are reviewed. The focus is on how each operator affects the paradigm's behavior in the problem space.

There are, of course, many ways to implement a genetic algorithm (GA). Regardless of the specific implementation, it is generally agreed that GAs utilize one form or another of three operators: selection, crossover, and mutation. We will examine implementations of these operators, and compare them with PSO operators. In this paper, it can be assumed that, in most cases, a basic, binary version of a GA is being referred to, such as the "plain vanilla" GA in [2] or the elementary GA 1-1 in [1].

The authors readily concede that few applications use these basic GA configurations, and a number of modifications to these configurations are also examined. It should be noted that it is not the goal of this paper to compare GAs and PSO in order to declare one or the other as somehow better. Rather, the goals are to provide insights into how GAs and PSO work, and to suggest ways in which performance might be enhanced by incorporating features from one paradigm into the other.

Analysis

In PSO, a particle is analogous to a population member (chromosome) in a GA. Like a GA chromosome, a particle represents a candidate solution to the problem being addressed. Each particle is treated as a point in the D-dimensional problem space. The *i*th particle is represented as $X_I = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The best previous position (the position giving the best fitness value) of the *i*th particle is recorded and represented as $P_I = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol *g*. The rate of the position change (velocity) for particle *i* is represented as $V_I = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The particles are manipulated according to the following equations:

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

By adding the inertia weight *w* into PSO, a new version of PSO is introduced in [10]. The inertia weight *w* is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global (wide-ranging) and local (fine-grained) exploration abilities of the "flying points." A larger inertia weight facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight provides a balance between global and local exploration abilities and thus requires fewer iterations on average to find the optimum [10]. (In some ways, the name *damping weight* might be more descriptive, but the term *inertia weight* is used in this paper.)

In a GA, each of the three main classes of operations (selection, crossover, and mutation) can be implemented in a number of ways. PSO does not label its operations in the same way as GAs, but analogies exist. These analogies depend, of course, on the implementation of the GA operation. Complicating any comparisons is the fact that the effects of the various operations often vary over the course of a run. (A *run* is defined as the total number of generations of the GA prior to termination. Termination usually occurs either because a prescribed fitness level has been achieved by at least one member of the population, or because the maximum number of generations allowed has been reached.)

Effects of GA crossover, for example, usually vary significantly during a run. At the beginning, the population members are usually randomized, so that crossover can have significant effects, moving a chromosome a relatively large distance in problem space. Toward the end of a run, a population has often converged, meaning that many, if not most, chromosomes have similar structures. Crossover then usually has less effect, and the resulting movements are relatively smaller. Adding another layer of complexity is the fact that the probability of crossover is sometimes varied during a run, often starting out at with a relatively large probability, and ending with a smaller one.

PSO does not have a crossover operation. However, the concept of crossover is represented in PSO because each particle is stochastically accelerated toward its own previous best position, as well as toward the global best position (of the entire