Expert Systems with Applications 36 (2009) 11088–11098

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Potential offspring production strategies: An improved genetic algorithm for global numerical optimization

Sheng-Ta Hsieh [a], Tsung-Ying Sun [b,*], Chan-Cheng Liu [b]

[a] Department of Electrical Engineering, Oriental Institute of Technology, Taipei County 22042, Taiwan, ROC
[b] Department of Electrical Engineering, National Dong Hwa University No. 1, Sec. 2, Da-Hsueh Rd., Shou-Feng, Hualien 97401, Taiwan, ROC

ARTICLE INFO

Keywords:
Numerical optimization
Population manager
Sharing cross-over
Sharing evolution genetic algorithm (SEGA)
Sharing mutation
Survival rate

ABSTRACT

In this paper, a sharing evolution genetic algorithms (SEGA) is proposed to solve various global numerical optimization problems. The SEGA employs a proposed population manager to preserve chromosomes which are superior and to eliminate those which are worse. The population manager also incorporates additional potential chromosomes to assist the solution exploration, controlled by the current solution searching status. The SEGA also uses the proposed sharing concepts for cross-over and mutation to prevent populations from falling into the local minimal, and allows GA to easier find or approach the global optimal solution. All the three parts in SEGA, including population manager, sharing cross-over and sharing mutation, can effective increase new born offspring's solution searching ability. Experiments were conducted on CEC-05 benchmark problems which included unimodal, multi-modal, expanded, and hybrid composition functions. The results showed that the SEGA displayed better performance when solving these benchmark problems compared to recent variants of the genetic algorithms.

1. Introduction

As more and more real-world optimization problems become increasingly complex, algorithms with more capable optimizations are also increasing in demand. Unconstrained optimization problems can be formulated as a *N*-dimensional minimization problem as follows:

$$\text{Min } f(x), x = [x_1, x_2, \ldots, x_N]$$

where *N* is the number of the parameters to be optimized.

The genetic algorithm (GA) is use for moving from one population of *chromosomes* to a new population by employing a principle similar to Darwin's "natural selection" together with the genetics-inspired operators of selection, cross-over, mutation, and inversion. The basic principles of GA were first introduced by Holland (Holland, 1975; Booker, Gokdberg, & Holland, 1978). Holland's GA was the first evolutionary computation (EC) paradigm developed and applied.

The fitter chromosome is likely to be selected for reproduction. The objective of cross-over is to randomly choose loci and exchange the subparts of chromosomes to create offspring. Mutation randomly flips the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome (Mitchell, 1996). The term *chromosome* typically refers to a candidate solution to a problem. Through the genetic evolution progress, genetic algorithms can search for solutions efficiently without derivative information; an optimal solution will be represented by the final winning chromosome in the genetic competition.

Many researchers are devoted to developing new operations to enhance the optimal capacity of the original GA. There are several strategies for cross-over operation, such as two-point, multi-point, and uniform (Syswerda, 1989), etc., proposed for improving the efficiency of binary-coded GA. The extrapolation, interpolation (Michalewicz, Logan, & Swaminathan, 1994), and multi-cross-over (Chang, 2003) were proposed for enhancing the performance of real-coded GA. Mutation is one of the most significant and promising areas of investigation in evolutionary computation, since it is able to prevent the GA from falling into the local minimal. Hong and Wang have proposed a dynamical mutation to tune the rate about different types of mutation (Hong et al., 1996). Zhang et al. proposed a mutation GA with adaptive probability, and associated *k*-mean and fuzzy-based system to update that probability (Zhang, Chung, & Hu, 2004).

Leung and Wang proposed OGA/Q (Leung & Wang, 2001) which utilized the characteristics of the orthogonal matrix and quantized searching space as several subspaces for generating offspring. A hybrid Taguchi genetic algorithm (HTGA) (Tsai, Liu, & Chou, 2004) was proposed by Tsai et al. The HTGA can efficiently generate better offspring and performed with better results than OGA/Q.

* Corresponding author. Tel.: +886 3 8634078; fax: +886 3 8634060.
E-mail addresses: FO013@mail.oit.edu.tw (S.-T. Hsieh), sunty@mail.ndhu.edu.tw (T.-Y. Sun).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.02.095

Recently, Liu et al. proposed a fuzzy-based method to generate offspring (Liu, Xu, & Abraham, 2005).

Different from the other methods of computational intelligence, such as fuzzy theory, artificial neural network, etc., GA has the ability of avoiding the local search and can increase the probability of finding the global best. It has been successfully applied to the fields of machine learning (Booker et al., 1978; Gokdberg, 1989), numerical optimization (Jong, 1980; Muhlenbein, Schomisch, & Born, 1991; Tu & Lu, 2004), signal processing (Guo & Mu, 2002; Li & Mao, 2004; Yue et al., 2002; Zheng, Liu, Tian, & Cao, 2004), etc.

Although numerous variants of GA have been empirically shown to perform well on many optimization problems in the last three decades, issues with premature convergence when solving complex problems are still prominent in GA. In GA, solution exploration depends on new born chromosomes, which are generated by cross-over and mutation operations to have higher ability (valid information) of finding better solutions. When the solution searching comes to a standstill, superior (potential) chromosomes are required in order to break free from the local minimal, explore unsearched areas and speed up the solution searching progress. Thus, the quality of generated offspring will affect GA's solution exploration ability directly. If most of the new born offspring, which have poor information, are eliminated by selection, the solution searching progress may slow down and even halt at a standstill. It is an important issue for parents to generate better offspring as means of speeding up the solution searching process. In order to efficiently drive the population and improve GA's performance on complex multi-modal problems, the sharing evolution genetic algorithm (SEGA) is proposed. The SEGA is including population manager and sharing strategy, for enhancing the solution searching abilities and increasing offspring's survival rate in genetic algorithms.

The paper is organized as follows. Section 2 presents an overview of the genetic algorithm. Section 3 describes the sharing evolution genetic algorithm. Section 4 presents the test functions, experimental settings and results. Section 5 of the paper contains the conclusions.

2. Genetic algorithm

The traditional GA (TGA) had following features: (1) a bit string representation, (2) proportional selection (3) cross-over as the primary method to produce new individuals (4) mutation for disturbing evolution to avoid solutions falling into the local search and (5) elitism policies are used. In this section, a brief introduction of genetic algorithm will be described.

2.1. Chromosome representation

Considering that a problem is presented as $f(x_1, x_2, \ldots, x_N)$ which consists of N tunable parameters to be optimized. In GA, it can be encoded by vector representation (i.e., chromosome) as $\mathbf{C}^m = [x_1, x_2, \ldots, x_N]^m, m = 1, 2, \ldots p$, where p denotes the population size. For high-dimension or complex problems, GA will require a larger population for uniform distribution of population in the searching space; otherwise it may be unable to explore all possible solutions. The value of p is always given experimentally.

2.2. Initial population

For most optimal techniques, the final solutions are usually restricted by the initialization. However, GA is able to overcome this drawback with the cross-over and mutation operation. Therefore, chromosomes can be scatter on an area in first generation. The initial population will be used to generate p chromosomes which will be distributed over the searching space uniformly.

2.3. Cross-over

The cross-over operation is to produce new chromosomes (offspring) by choosing two random parent chromosomes, but it does not guarantee that all the offspring are better than the parent. However, after adopting "exploration" and "exploitation" for performing cross-over will obtain good results because the offspring will be generated around the better parent. The detail of "exploration" and "exploitation" are described in Section 4.3. The number of individuals which will be joined during cross-over is based on a pre-defined parameter r_c which is called *cross-over rate*. Thus, there will be $round(p \times r_c)$ individuals (parents) joined to perform cross-over.

2.4. Mutation

The mutation operator is to randomly change some subparts of a chromosome. When GA is learning, the chromosomes will move to the nearest optimal solution to itself, but that may be not a global optimization. Therefore, some disturbances to extend the searching range are quite important. In general, the offspring of mutation $\mathbf{O}_m$ is generated inside the searching space randomly as

$$\mathbf{O}_m = \alpha \tag{1}$$

where α denotes a mutation vector with random components of uniform distribution in searching space. The number of parents which joins mutation is based on a pre-defined parameter r_m which is called *mutation rate*. Thus, there are $round(p \times r_m)$ individuals (parents) that will be joined to perform mutation.

In general, the fitness of mutated offspring may be better or worse than their parents and/or any cross-over offspring. On the other hand, adopting the mutation operation will extend the searching range in order to explore unsearched area in the searching space for finding the potential optimal solution.

2.5. Selection

After cross-over and mutation operations, all chromosomes, including parents and offspring in a population, will be larger than the initialization. In order to produce better offspring, the elitism operation is adapted to select p better chromosomes which will survive in next generation.

The GA optimization is combined with operations mentioned above and repeats the evolution process until it reaches the predefined terminated conditions.

3. Sharing evolution genetic algorithm

In genetic algorithm, new offspring are produced by cross-over and mutation operations. After selection, the new population consists of selected better chromosome, which are called *parents* for next generation, will keep generating new offspring to perform the routine of solution searching. The quality of the generated offspring will greatly affect GA's solution exploration ability. If the new born offspring, which usually have inferior information, are eliminated by selection, the solution searching progress may slow down or even standstill. It's also an important issue in GA to give offspring generated by cross-over or mutation operation a higher survival rate. A higher offspring survival rate means more (superior) solutions will be found in each generation while a lower one will decrease GA's solution searching ability. Considering these concerns, in this paper, the sharing evolution genetic algorithms (SEGA) is proposed. The SEGA can be separated into three parts: population manager, sharing cross-over and sharing mutation. In the following sections, the detail of the three elements will be described.