A Study of the Performance Effect of Genetic Operators

Pi-Sheng Deng
California State University at Stanislaus, USA

INTRODUCTION

Performance of genetic algorithms (GAs) is mainly determined by several factors. Not only the genetic operators affect the performance of a GA with varying degrees, but also the parameter settings for genetic operators interact in a complicated manner with each other in influencing a GA's performance. Though many studies have been conducted for this cause, they failed to converge to consistent conclusions regarding the importance of different genetic operators and their parameter settings on the performance of GAs. Actually, optimizing the combinations of different strategies and parameters for different problem types is an *NP*-complete problem in itself, and is still an open research problem for GAs (Mitchell, 1996).

Recognizing the intrinsic difficulties in finding universally optimal parameter configurations for different classes of problems, we advocate the experience-based approach to discovering generalized guiding rules for different problem domains. To this end, it is necessary for us to gain a better understanding about how different genetic operators and their parameter combinations affect a GA's behavior. In this research, we systematically investigate, through a series of experiments, the effect of GA operators and the interaction among GA operators on the performance of the GA-based batch selection system as proposed in Deng (2007). This paper intends to serve as an initial inquiry into the research of useful design guidelines for configuring GA-based systems.

PARAMETER CONFIGURATION FOR GENETIC OPERATORS

It is commonly believed that crossover is the major operator of GAs, with mutation preventing the population from early convergence to a certain solution before an extensive exploration of other candidate solutions is made (Holland, 1992a). Crossover enables GAs to focus on the most promising regions in a solution space; however, mutation alone does not advance the search for a solution. Crossover is also a more robust constructor of new candidate solutions than mutation (Spears, 1993).

However, Muhlenbein (1992) argues that the power of mutation has been underestimated in traditional GAs. According to Mitchell (1996), it is not a choice between crossover or mutation but rather the balance among crossover, mutation, and other factors, such as selection, that is all important. The correct balance also depends upon the details of the fitness function and the encoding. Furthermore, crossover and mutation vary in relative usefulness over the course of a run. Actually, the theretical analysis of crossover is still to a large extent an open problem (Back, *et al*., 1997).

In addition to the GA operators, the population size also affects the performance of GAs. The specification of the population size affects the diversification of the population body and the implicit parallelism of a GA, and will thus affect the quality of the generated solutions and the performance of the solution-generating process. Choosing an appropriate population size for a GA is a necessary but difficult task for GA users. Usually, the parameter settings for most GA applications are based on De Jong's recommendations (De Jong, 1975). According to De Jong's experiments with five problems in function minimization, the best population size was 50~100, the best crossover rate was about 0.6, and the best mutation rate was 0.001. In a later study, Spears & De Jong (1991) suggested a wider range for the crossover rate as 0.5~0.8. Mitchell (1996) also observed that it was common in GA applications to set crossover rate at 0.7~0.8.

However, Schaffer *et al*. (1991) asserted that the best settings for population size, crossover rate, and mutation rate were independent of the problems. In their study of a small set of numerical optimization problems, a very small population of size 20~30 with

a large crossover rate ranging from 0.75 to 0.95, and with a very small mutation rate ranging from 0.005 to 0.01 would produce the best perofrmance. Grefenstette (1993) also reached similar conclusions in his study of parameter optimization for GAs, and suggetsed the following settings: population size 30, crossover rate 0.95, and mutation rate 0.01. While Schaffer *et al*. (1991) and Grefenstette (1993) advocated a very small population size, Goldberg (1989) and Liao & Sun (2001) argued for a much larger population size.

From the above discussion, the diverse recommendations on population size seem to indicate that population size interacts with some other factors not included in the previous research. In this paper, we investigated the effect of interaction among different parameters on a GA's performance. However, the choice of mutation rate needs to take into account, at least, the task complexity of an application. According to Mitchell (1996), it is impossible to specify an optimal setting for parameters in all different applications.

EXPERIMENTAL DESIGN

We focus mainly on investigating the effects of different combinations of parameter settings for genetic operators on our GA's performance, and compare our results with the claims made by previous research. We discuss the factors and parameter settings experimented in this study as below:

- **Task complexity:** Since the length of the solution string is usually a function of the complexity of the problem, we experiment with different parameter settings for two batch selection tasks of different complexity levels. One task has 30 products to be manufactured, 10 available tools, and 8 available machines. The other less complex task has 12 products, 6 available tools, and 4 available machines.
- **Representation scheme:** In this paper, we consider a common situation in FMSs in which if a product is selected in a batch for manufacturing, the entire quantity specified in the production table must be produced in the shift. Under this assumption, our batch selection task becomes a pseudo-Boolean optimization problem. This enables us to use a single binary bit to represent a component in a candidate solution. Therefore, each candidate solution to the batch selection task can be encoded as a binary string of fixed length P, where P is the cardinality of the entire set of products under consideration.
- **Population size:** If the population size is too small, the GA will converge too quickly to find the optimal solution; however, if the population size is too large, the computation cost will be prohibitive. In this research, we investigate the effect of population sizes 10, 100, and 200, representing Small, Medium, and Large, on generating solutions for our batch selection problem.
- **Selection strategy:** We adopted the elitism strategy so that the best candidate solutions at each generation could be retained for the next generation. Though elitism is used to prevent the elite solution strings in a population from being altered by crossover or mutation, retaining too many elite individuals might cause the domination of the entire population by suboptimal, though highly fit, solution strings. This might lead to degeneration for the population eventually. The usual practice is to retain a small number of elite candidate solutions (Goldberg, 1989). In this research our system preserves two fittest candidate solutions on each iteration of forming new population.
- **Crossover parameter:** We adopt the standard crossover operator, i.e., the one-point crossover. The crossover rate is the probability that the crossover operator will be applied to a pair of candidate solutions selected for reproduction. In order to re-examine the different claims by previous research on the importance of different crossover rates, we experiemnt with three different crossover rates: 0.1, 0.5, and 0.9, representing three levels: High, Medium, and Low.
- **Mutation parameter:** The parameter *mutation rate* is used to control the rate of diversification via probabilistic conversion of each bit value in a candidate solution. However, a mutation rate approaching 1 will theoretically lead to a completely stochastic search with no succession from generation to generation. The usual practice is applying an occasional mutation to make a random change in the elements of a solution string. There are also various conclusions from previous research regarding the mutation rate. In this research we experiment with three different mutation rates: 0.001, 0.01, and 0.5, representing three levels: Low, Medium, and High.