Chapter 3

MDL-BASED EFFICIENT GENETIC PROGRAMMING FOR OBJECT DETECTION

3.1 Introduction

In chapter 2, the efficacy of genetic programming in learning composite features for object detection is demonstrated. The motivation for using GP is to overcome the human experts' limitation of considering only a very limited number of conventional combinations of primitive features. Chapter 2 shows that GP is an effective way of synthesizing composite features from primitive ones for object detection. However, genetic programming is computationally expensive. In the traditional GP with hard limit on the individual size (also called a *normal GP*), crossover and mutation locations are randomly selected, leading to the disruption of the effective components (subtree in this approach) of composite operators especially at the later stage of the GP search. This greatly reduces the efficiency of GP. It is very important for GP to identify and keep the effective components of composite operators to improve the efficiency. In this chapter, smart crossover and smart mutation are proposed to better choose crossover and mutation points to prevent effective components of a composite operator from being disrupted. Also, a public library is established to save the effective components of composite operators for later reuse. Finally, a fitness function based on the minimum description length (MDL) principle is designed to incorporate the size of a composite operator

into the fitness evaluation to address the well-known code bloat problem of GP without imposing severe restrictions on the GP search. The GP with smart crossover, smart mutation and MDL-based fitness function is called a *smart GP*.

3.2 Motivation and Related Research

Crossover and mutation are two major mechanisms employed by GP to search the composite operator space (also called feature combination space). As GP proceeds, effective components are generated. The power of crossover lies in the fact that by swapping sub-trees between two effective composite operators (parents), the effective components (sub-trees) in these two parents can be assembled together into child composite operators (offspring) and the new offspring may be better than both parents. However, although crossover can assemble good components to yield better offspring, it is also a destructive force in the sense that it can disrupt good components due to the random selection of crossover points. When the search begins, since the initial population is randomly generated, it is unlikely that a composite operator contains large good components and the probability of crossover breaking up a good component is small. At this time, crossover is a constructive force and the fitness of a composite operator is increased. As search proceeds, small good components are generated and assembled into larger and larger good components. When more and more composite operators contain large good components to achieve high fitness, the good component accounts for a large portion of a composite operator and the composite operator becomes more and more fragile because the good components are more prone to being broken up by subsequent crossover due to the random selection of crossover points. The crossover can damage the fitness of a composite operator in ways other than disrupting good components. Sometimes, a good component is moved into an inhospitable context, that is, the crossover inserts a good component into a composite operator that does not use the good component in any useful way or other nodes of the composite operator cancel out the effect of the good component. According to [82], crossover has an overwhelmingly negative effect on the fitness of the offspring from crossover, especially in the later stage of GP search.