No Free Lunch for Module Encapsulation

Ozlem O. Garibay[1], Ivan I. Garibay[1], and Annie S. Wu[1]

University of Central Florida, School of Computer Science,
P.O. Box 162362, Orlando, FL 32816-2362, USA,
{igaribay,ozlem,aswu}@cs.ucf.edu,
WWW home page: http://ivan.research.ucf.edu

The concept of modularity has been extensively studied in the evolutionary computation community. In contrast with a fitness-driven modularity approach, we present an analysis where modules are simply strings of genomic symbols. Our contribution is two fold: we provide a theorem on the impact of module encapsulation on the search space and, we offer a static mathematical analysis of the necessary conditions for the encapsulation to be beneficial.

Theorem 1. *Strictly-encapsulating lower-order modules into a complete set of higher-order modules does not change the search space size or structural bias.* Under certain assumptions, we prove that encapsulating and replacing all lower level modules or primitives with the higher level counterparts does not affect the size of the search space or bias the search. The proof for this theorem and additional experimental support can be found elsewhere (Garibay, Garibay & Wu 2004).

Arbitrarily encapsulating primitives into modules has two effects on the search space: increase in the size of the search space and bias the search. The size of the search space increases when a new module is encapsulated, since it introduces a new element into the search space alphabet. Encapsulating "good" modules bias the search towards the solution. We define "good" modules as modules that are present in the optimal solution and "bad" modules as modules that are not. Hence, there is a trade-off between the gains and losses of introducing a new module in terms of search space size and bias. In order for module encapsulation to benefit the search, the encapsulated module should enable the search algorithm to search a smaller space. The following closed form expression determines whether the creation of a module will be beneficial in terms of the size of the new search space, $C \geq l\frac{\ln(1+1/|\Sigma|)}{\ln(|\Sigma|+1)}$, where $|\Sigma|$ is the length of the alphabet, l is the length of the individual and C is a constant, $C = x(|w|-1)$, where x is the number of copies of the module defining string w in the optimal solution and $|w|$ is the length of this string. Using this expression, we can summarize the effects of encapsulating a module as follows: encapsulating bad modules is always detrimental; encapsulating good modules is advantageous only if the expression above is satisfied.

References

Garibay, I., Garibay, O. & Wu, A. (2004), Effects of module encapsulation in repetitively modular genotypes on the search space, *in* 'Proc. of GECCO'04', To Appear.