Defining Modularity, Hierarchy, and Repetition

Edwin D. de Jong[1], Dirk Thierens[1], and Richard A. Watson[2]

[1] Utrecht University, Decision Support Systems Group
PO Box 80.089, 3508 TB Utrecht, The Netherlands
dejong@cs.uu.nl, dirk.thierens@cs.uu.nl

[2] Harvard University
Dept. of Organismic and Evolutionary Biology
Cambridge, MA 02138
rwatson@oeb.harvard.edu

1 Introduction

In order to benefit from the presence of modularity, hierarchy, or repetition in a problem, a necessary first step is to identify precise notions of these problem features. Modularity, hierarchy, and repetition have often been used as terms to characterize the operation of a search method. However, if these features are to improve the efficiency of a search method, then there must exist corresponding problem features that permit such improvements. A primary question to be asked therefore is how these or other problem features may be defined in such a way that their identification by a search method guarantees potential efficiency improvements. The aim of this contribution is to provide such definitions, and to discuss how algorithms may exploit the corresponding efficiency improvements.

2 Modularity, Hierarchy, and Repetition

We first discuss the notion of modularity. The definition that will be proposed is based on the idea that a subset of the variables in a system may be optimized to some extent independent of the remaining variables in the system; see [5, 7]. It is related to the notions of building blocks and what we will call *additive partitions*, but differs in that it facilitates the definition of a hierarchy of modules.

The term *building block* was introduced by Holland [4] to refer to above-average fitness low-order schemata of short defining length. The notion of additive partitions is related, but has the advantage that it directly shows how the identification of an additive partition may be used to achieve efficiency improvements, as will be discussed with an example shortly. Let a partition be any subset of the variables in a problem. Then an *additive partition* is a subset of up to k variables whose fitness contribution is independent of the remaining variables. The presence of additive partitions is sometimes referred to as *separability*.

As an example, suppose a problem can be subdivided into two partitions, one having 3 and the other having 4 binary variables. Given the knowledge that these partitions are present in the problem, in the worst case we need to search through

$2^3 + 2^4$=24 individuals to find the optimum. If this potential for decomposition is not discovered, one would need to search $2^{3+4} = 128$ individuals. Thus, if additive partitions defined in the above sense can be identified, this substantially reduces the amount of computation that must be performed in order to identify an optimal solution. If the settings of a partition that occur in global optima all have above-average fitness, then finding the building blocks of the problem is sufficient. For deceptive problems where this is not the case, the building blocks may guide the search in the wrong direction, while identification of the additive partitions still leads to the global optimum. A limitation of additive partitions however is that they do not permit the exploitation of hierarchical structure, and we proceed therefore to define the concept of a module.

In order to present our notion of modularity, let us define some auxiliary terms. The *context* of a partition is its complement; this set contains all the variables that are not a member of the partition. A *setting* for a partition is a value assignment of all its variables; for example, a setting that sets the second bit to 1 and the fifth bit to zero might be written as $\{(2, 1), (5, 0)\}$. A setting for a partition is *optimal* for a given context setting if its fitness is maximal given the context setting; that is, the fitness resulting from the setting is close to the highest fitness that can be obtained by any setting of the variables in the partition given the context setting. A setting for a partition is ϵ-*optimal* for a given context setting if its fitness is within a small constant ϵ of the maximal setting. A setting for a partition is (ϵ-)*context-optimal* if there exists *some* context setting for which it is (ϵ-)optimal.

The concept of an additive partition can be made more general without affecting the efficiency improvements it permits. The definition of an additive partition implies a requirement that the fitness function can be written as a sum of fitness contributions over non-overlapping partitions of up k variables. The resulting efficiency improvement is achieved by dividing the problem into subproblems, viz. additive partitions, whose settings can be optimized independently. However, this subdivision only requires that the optimal settings for each partition be independent of the context; this more general version of the separability concept is provided in [8]. It is less strict than the condition that the fitness contribution of a partition must be independent of the remaining variables (indeed, the latter implies the former), and thereby provides a more general criterion to determine whether a problem can be divided into independent subproblems.

While separability can greatly reduce the number of potential solutions that must be considered, it is still a strict criterion, and many problems of interest are not strictly separable. In order to obtain a more generally applicable concept, the separability criterion can be further relaxed while retaining a potential for decomposition. Watson [8] provides an extensive discussion of this, and defines *decomposability* as the property that the number of optimal settings of a module is lower than its total number of settings. Based on this criterion, the following definition of modularity can be given.

We distinguish between two types of **modules**: primitive modules and composite modules. The *primitive modules* in a problem are simply the variables in

the problem. We define a *composite module* as a combination of modules with the property that its number of context-optimal settings is lower than its total number of settings, and no subset of the module combination establishes such a reduction. When modules are combined into module combinations, their only settings that need to be considered as possible settings for the module combination are their context-optimal settings, as no other settings can be part of a global optimum. A module combination may be usefully viewed as a module itself if and only if this results in a *further* reduction of the number of settings that must be considered and if no subset of the module set results in a reduction. Using the above definitions, the identification of a composite module always permits a further reduction of the fraction of the search space that must be considered in order to identify a global optimum, and this property motivates the definitions. A problem is said to feature **modularity** if it contains at least one composite module.

In earlier work, we have defined the notion of *functional modularity* [1]. Functional modularity provides a notion of modularity for *variable length* problems based on a relative positioning encoding. Apart from this distinction, functional modularity requires a partition setting to be optimal for some set of contexts and compared to some set of comparison settings. If we consider all possible comparison settings, then the existence of a small set of such settings, each of which is optimal for some set of contexts, corresponds to the modularity concept defined here.

An important feature of the modularity definition that has been given is that modules can be combined into composite modules. Whenever this occurs, the definition of modularity implies a dependency between the constituent modules; otherwise, their combination would not lead to a further reduction. If the same problem were to be viewed in terms of additive partitions, the constituent modules would therefore form a single large additive partition. A method looking for additive partitions would therefore need to search over the 2^{m+n} combinations of all $m+n$ variables, while a method that identifies modules will benefit from the reductions that have been made for the m and n variables of the constituent modules. This difference can already be substantial at a single level, but when modules are recursively combined into larger modules the resulting savings can be very large.

A problem is said to feature *hierarchy* if there is at least one composite module that contains another composite module. By identifying such modules, certain hierarchical problems that would be difficult to solve otherwise can be addressed in an efficient manner. We are aware of two constructions that permit the definition of hierarchical test problems [9, 6], but these constructions are not restricted to producing hierarchical problems, as all functions on the domain can be produced by the constructions. The definition of hierarchy given here provides a clear and simple criterion to delineate the class of hierarchical problems.

Finally, we define **repetition** as the presence of multiple modules whose optimal settings are identical, where it is assumed that modules consist of con-

secutive variables so that the mapping of the variable settings simply corresponds to translocation.

3 Utility of the Problem Features

In the following, we discuss how the three problem features as they have been defined may be used to achieve a potential performance improvement. Regarding modularity, the part of the search space that must be visited to guarantee finding the optimum is lower when composite modules have been identified. Thus, once identified, a module can offer a substantial efficiency improvement. Whether the net effect of exploiting modular structure is beneficial will therefore depend on the amount of computation required to identify the modules.

For hierarchy, the case is similar, but here the savings can be larger, since the number of variables involved grows as higher levels of modules are formed. Again, the net benefit will depend on the cost of identifying composite modules. Finally, repetition can be of advantage when the problem features repeated modules; after a first module and its optimal setting(s) have been identified, the same settings may be tried on other sequences of variables, or to sequences of variables that have already been found to be modules (i.e. some settings have been found never to be optimal at some degree of confidence) but for which no optimal module setting has yet been identified. Whether this confers benefit depends on the size and number of the modules and the occurrence frequency of the specified settings; see [3] for conditions under which a related notion of repetition confers benefit.

4 Exploiting Modularity, Hierarchy, and Repetition

While the purpose of this contribution is to define the problem features under study such that their identification is guaranteed to result in a potential performance benefit, we will now briefly discuss how a search algorithm might achieve this identification. Recall that a module is defined as a partition for which the number of context-optimal settings is lower than the total number of settings. This immediately suggests a test to determine whether a given partition is modular. If there are one or more settings that never maximize fitness, then the partition is guaranteed to be modular. To determine with certainty whether this is the case, one would have to consider all possible context settings, which would typically require a prohibitive amount of computational expense. However, if one can sample from the space of all possible context settings in such a manner that a context setting for which a given partition setting is optimal will be found with some reasonable probability, then a limited number of tests is sufficient to determine with high accuracy whether a partition is modular. A question therefore is whether sampling procedures can be identified that result in reasonable values for this probability in problems of interest; if this is the case, we will say the problem features *probabilistically detectable modularity*. The view of module detection as a sampling problem provides a general framework for the construction

of methods exploiting modularity and hierarchy, and indeed existing modular and hierarchical methods such as SEAM [10] and DevRep [2] may be fruitfully viewed in this way.

References

1. Edwin D. De Jong. Representation development from Pareto-coevolution. In E. Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-03*, pages 262–273, Berlin, 2003. Springer.
2. Edwin D. De Jong. Exploiting modularity, hierarchy, and repetition in variable-length problems. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, 2004.
3. Ivan Garibay, Ozlem Garibay, and Annie Wu. Effects of module encapsulation in repetitively modular genotypes on the search space. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04*, 2004.
4. John H. Holland. *Adaptation in Natural and Artifical Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
5. Hod Lipson, Jordan B. Pollack, and Nam P. Suh. On the origin of modular variation. *Evolution*, 56(8):1549–1556, 2002.
6. Martin Pelikan and David E. Goldberg. Hierarchical problem solving by the bayesian optimization algorithm. In Darrell Whitley et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 267–274, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.
7. Günter P. Wagner. Adaptation and the modular design of organims. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proceedings of the Third European Conference on Artificial Life : Advances in Artificial Life*, volume 929 of *LNAI*, pages 317–328, Berlin, June 1995. Springer Verlag.
8. Richard A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, 2002.
9. Richard A. Watson, Gregory S. Hornby, and Jordan B. Pollack. Modeling building-block interdependency. In A.E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-V.*, volume 1498 of *LNCS*, pages 97–106, Berlin, 1998. Springer.
10. Richard A. Watson and Jordan B. Pollack. A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, 69(2-3):187–209, May 2003. Special Issue on Evolvability, ed. Nehaniv.