Applying Adaptive Algorithms to Epistatic Domains

Lawrence Davis

Knowledge-Based Systems Group
Texas Instruments Inc.

ABSTRACT

John Holland has shown that when adaptive algorithms are used to search certain kinds of extremely large problem spaces, they will converge on a "good" solution fairly quickly. Such problem spaces are characterized by a low degree of epistasis. A host of classical search problems, however, are epistatic in nature. The present paper describes some new techniques for applying adaptive algorithms to epistatic domains, while retaining some of the strength of Holland's convergence proof. These techniques are described for two-dimensional bin-packing problems, and summarized for graph coloring problems. What makes these problems amenable to an adaptive approach is a two-stage evaluation procedure. Encodings of solutions are mutated and reproduced as they are in non-epistatic domains, but their evaluation is carried out after a decoding process. Using the techniques described, convergence is promoted in two ways: one of the natural mutation operators is a weaker version of Holland's crossover, and domain knowledge may be built into decoding processes so that the size of the search space is radically cut down.

The Holland Approach to Non-epistatic Domains

In [Holland 1975], John Holland showed how the metaphor of evolution through adaptation may be profitably applied to searching non-epistatic domains. The problem of designing humans is such a domain, for nose shapes and leg shapes may be combined and recombined without drastically affecting the survival abilities of their recipients. Holland notes that epistatic domains are not fertile domains for the techniques he proposes. A central result of Holland's book is his proof that, in non-epistatic domains, a good solution may be arrived at relatively efficiently by use of mutations like crossover, that splice together parts of two different chromosomes.

Holland's work in these domains is exciting and provocative, and the sorts of adaptive algorithms he describes have a number of attractive features. They can be run in parallel, since individual members of a population may be created, mutated, and evaluated without interaction with other members of their generation. A user has near-total flexibility in allocating resources to them, since the population size and the number of generations to be produced are under the user's control. Since they are non-deterministic, they can be re-run in hopes of improving on results already found. Finally, they are not difficult to implement; given a utility to drive the evolution process, one needs only define one's data structures, evaluation procedure, and mutations in order to set the search procedure in motion.

In fact, for the researcher or programmer concerned with hard search problems, perhaps the greatest problem with adaptive algorithms is that they do not seem usable in epistatic domains. In the case of packing a bin with rectangles, for example, a solution would consist of a specification of the placement of each rectangle packed into the bin. It seems that attempts to combine part of one such solution with parts of another will nearly always result in a worse solution, because the result will be a packing that contains overlaps or large gaps.

In what follows, a technique for applying adaptive algorithms to the bin packing domain will be explained, and similar approaches to some other epistatic domains will be outlined. The strategy used in each case is a twofold one. First, the adaptive algorithm is given coded solutions to mutate and propagate. Second, in order to evaluate these codings, they are decoded with techniques that always produce legal solutions.

Mutatable Encodings: A Matter of Representation

Several representation techniques were tried out on the bin packing problem; the one that worked best was a simple list of rectangles to be packed. The decoding algorithm proceeded by placing the first member of the list into the first place it would fit in the bin, the second in the second, and so forth, omitting from the packing any rectangle that would not fit. (Two types of decoding algorithms were tried, but both had this effect; in the next section, they are discussed in more detail.)

The resultant system has the following features:

1. Every list represents a legal packing, by virtue of the operation of the decoding algorithm. Thus, the search procedure never departs from the domain of legal solutions.

2. The position of a rectangle on its list is meaningful in a relative, rather than absolute, way. That is, a rectangle that appears fourth on a list might be packed in any of a number of places, depending on what rectangles precede it.
3. By virtue of 2, a rectangle's appearance early on in a list receiving a good evaluation frequently signals that it is "easily packable." This fact was exploited in REWRITE, a mutation that turned out to be extremely useful. REWRITE takes a list and re-orders it so that those rectangles that were not packed are placed at the end. The result of this mutation is to move "packable" rectangles to the front of the list, and float less useful ones to the end.
4. By virtue of 3, another useful mutation is MODIFIED-CROSSOVER, which takes the first part of a solution, broken at random, and orders the rest of its members in accordance with their order in another solution. The list (3 1 2 6 4 5), for example, broken after its second member and crossed over with the list (4 1 6 5 2 3), yields the list (3 1 4 6 5 2). MODIFIED-CROSSOVER has the effect of moving forward rectangles that have been found to be useful in other lists. Since the part of the list preserved in the crossover yields a partial packing, the MODIFIED-CROSSOVER mutation often re-orders the rectangles occurring next in a beneficial way.
5. The other useful mutations discovered so far are all varieties of SCRAMBLE. Initial experimentation with a single mutation that scrambled a segment of the list led to its replacement by three other mutations. One scrambles a segment of the list within the packed part. Another scrambles the unpacked part. A third scrambles across the boundary.
6. A useful combination of probability levels for each mutation may be found by running an adaptive algorithm on lists of values. (The domain is non-epistatic, and classical techniques work.) Using a population of size thirty, an adaptive routine converged on a blend of values that substantially out-performed a human-produced blend by the sixtieth generation.

Variations on the Decoding Theme

Several strains of decoding algorithms were tested. The most successful fell into two classes. One packed rectangles on the list by entering them at the lower right corner of the packing area and sliding them alternately left and up until they could move no further. Associated with this class of decoder was an array of rectangle orientations. The principal virtue of this class of decoder was that it was fast. Sliding algorithms spend very little time decoding lists of rectangles–given an orientation, sliding a rectangle to its proper position is a straightforward matter. The second class of decoders positioned rectangles by moving them along the "skyline" of rectangles already packed, searching for a position that maximized the amount of perimeter in contact with previously-placed rectangles. The left and upper walls of the packing area were used in computing perimeter contact; the decoding process produced a packing that grew from the upper left corner outward.

On a test domain of 30 rectangles being packed in a bin that would hold approximately 90 per cent of their surface area, skyline decoders were roughly six times slower than sliding decoders, but on average, they tended to produce much denser packings (see figures 1 and 2). When the two decoding routines were run for equivalent amounts of time over populations of members evolving as described above, the best packings found by the sliding decoder were, on average, uniformly worse than those found by the skyline decoder.

This result is very interesting. Practical bin-packing problems rarely involve pure bin-packing. Among the sorts of constraints that real-world examples may include are: varying minimum separations between types of objects packed, constraints on packing types of objects adjacently, constraints on orientation, and so on. Our work on the two classes of decoder suggests that there is a good deal to be learned about the best way to put such constraints into an adaptive search program. It suggests, too, that blind mutation combined with an evaluation function discouraging constrained occurrences will be less efficient than blind mutation in combination with a decoding routine that intelligently avoids producing constrained results. A good deal more needs to be done in this area, but the initial results support the received AI wisdom that "in the knowledge lies the power". Evolutionary adaptation is a good way to guide one's search of a large space. Intelligent decoders are good ways to move to the points in that space that are likely to be worth investigating.

Adaption in Other Epistatic Domains

The technique of using mutatable structures for the search and encoding them for the evaluation procedure seems to generalize to a number of other epistatic domains. Research is currently under way in another of them; the nature of the approach being used is sketched here in order to indicate the generalizability of the technique.

Consider the problem of coloring the nodes of a graph with n colors so that no two connected nodes share the same color, while the total weight of the colored nodes is maximized. We approach n-coloring in a way similar to that described above: encodings are represented as lists of nodes, and the decoding algorithm proceeds by giving each node on the list, in order, the first legal coloring it can have. The REWRITE and MODIFIED-CROSSOVER mutations have the effect of moving good nodes to color to the front of the list. The scrambling mutations described above perform well in this domain as well.

Conclusion

Epistatic domains can be profitably searched using adaptive algorithms. Holland's proof of the power of adaptive algorithms does not hold in epistatic domains, but two ways to regain that power have been described: a modified crossover mutation that passes information from population member to population member, and the insertion of domain knowledge into the decoding routines.

---

*This paper describes the ongoing work of a group of affiliated researchers in the Knowledge-Based Systems branch of Texas Instruments' Computer Science Laboratory. The decoding strategy was devised by Vibhu Kalyan and myself, and the bin-packing decoders are the result of work by Derek Smith. Jeff Eisen provided invaluable technical support, and comments by Paul Kline and Nichael Cramer have improved our results considerably.

REFERENCES

Bethke, A.D., *Genetic Algorithms as Function Optimizers*, University of Michigan doctoral dissertation, 1981.

Dejong, Ken A., *Analysis of the Behavior of a Class of Genetic Adaptive Systems*, University of Michigan doctoral dissertation, 1975.

Holland, John H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

Mauldin, Michael L., "Maintaining Diversity in Genetic Search," AAAI-84 Proceedings.

Smith, Derek, "Bin-Packing With Adaptive Search," submitted to 1985 International Conference on Genetic Algorithms, Carnegie-Mellon University.