

Constrained Optimization via Combinatorial Interpolation to the Feasible – Infeasible Boundary Along Gray Code Paths

Steven Orla Kimbrough
University of Pennsylvania
kimbrough@wharton.upenn.edu

David Harlan Wood
University of Delaware
wood@cis.udel.edu

Categories and Subject Descriptors

[Combinatorial Interpolation, Constrained Combinatorial Optimization, and Heuristic Search Algorithms]:

Keywords

combinatorial interpolation, constrained optimization

1. COMBINATORIAL GRAY CODE PATHS

For many sets of combinatorial objects, much attention has been given to techniques for sequentially enumerating these objects so that adjacent objects are minimally different in some sense [*The Art of Computer Programming, Volume 4, Fascicle 2*, D. E. Knuth, Addison-Wesley, 2005]. We refer to such enumerations as Gray code paths. Typically, the advantage sought is to be able to process the entire set of objects by a series of slight modifications. However, our motivation is different and will be explained below. The example we will use for illustration is the set of all binary strings of fixed length. There are many ways that these strings can be arranged into a sequence so that any two adjacent strings differ in only a single position. As is well known, these Gray code paths correspond to Hamiltonian paths on the n -cube.

We are interested in constrained optimization problems where we seek optimal combinatorial objects. That is, each object has a value, but each object can be either feasible or infeasible. We seek feasible objects that have maximal value. In problems of interest, it is impractical to process any list of all possible objects, regardless of how efficiently they are arranged. There are simply too many objects.

Here is our key heuristic: a feasible solution with maximal value can be slightly changed to become infeasible. We say that such an object is “on the boundary” of feasibility. Thus, we are motivated to concentrate attention on this boundary. We are therefore led to introduce a type of interpolation between an arbitrary feasible combinatorial object and any other, even very different, infeasible ob-

ject such that interpolation produces a feasible – infeasible pair *on the boundary*, that is, a pair having minimal difference.

Combinatorial interpolation is illustrated below for the case of binary strings of fixed length. But it can be generally be thought of in this way: instead of successively evaluating all intermediate objects on a segment of a Gray code path, we do a binary search on that portion of the path, evaluating only a sequence of midpoints on the segment. This rapidly produces a *adjacent* pair of objects that have to be on the boundary because of the Gray code property.

2. EVOLVING TWO POPULATIONS

One ongoing theme of our research [*e.g.* PPSN VIII, Yao, ed. LNCS 3242, pp. 292–301, Springer-Verlag, 2004.] is the value of infeasible solutions of constrained optimization problems. Infeasible solutions violate the constraints but they can nevertheless contain valuable genetic information. Particularly valuable ones might be only a single mutation away from feasibility, or even optimality. Consider also, that an optimal solution is likewise only a single mutation away from infeasibility provided that the constraints are relevant (which we assume). The guiding principle is that for *constrained* optimization problems, evolutionary computation methods should pay particular attention to the region near the feasible – infeasible boundary. A desire for boundary exploration led us to create the Feasible – Infeasible Two Population Genetic Algorithm (FI-2PopGA). This model alternates

1. Evolving feasible objects only to *improve* them by seeking better objective function values and

2. Evolving infeasible objects only to *repair* them by lessening their constraint violations.

Let us give a general description of the observed behavior of the FI-2PopGA (*op. cit.*). A first order effect is that the infeasible population seeks out the boundary indiscriminately (exploration), and the feasible population seeks out the optima which will be located on or near the boundary (exploitation). Upon reaching the region of the boundary, a population can produce mutant children that end up in the other population where they are subjected to a different competitive criterion. A very interesting second order effect is obtained when mutant grandchildren end up reentering the population of their grandparents. This means that their genetic material has survived competitions involving two criteria: both approaching optima and also diminishing constraint violations. This makes both populations gradually become genetically similar and cluster on opposite sides of the boundary near optima.

Of course, both boundary searching and dual population evolution have been studied before (see references in *op. cit.*). Our approach is one of the few that avoid attempting to heuristically evaluate candidate solutions by two unrelated criteria at the same time.

3. INTERPOLATION TO THE BOUNDARY

Conspicuously missing from the FI-2PopGA is any interbreeding *between* the two populations. After all, with n variables, why not just interpolate between a feasible parent and an infeasible in n -dimensional space to reach the boundary? This could produce two children with nearly identical genetic material, but with one feasible and the other one infeasible. This could replace some or most of the more leisurely and explorative evolution to the boundary and the waiting for chance mutants to cross the boundary. The highly exploitive nature of interpolation could be eased by interpolating along different paths rather than straight lines.

However, we are most concerned with binary variables, where it is less clear how interpolation might be carried out. This brings us, at last, to the heart of our subject, which we illustrate using binary strings of fixed length. We find ourselves in a Hamming space where the points are vectors of bits (zeros and ones), often referred to as bitstrings.

If we have a master list of all possible bitstrings of length n , then we can interpolate to the boundary in the following way. Suppose we have two arbitrary parent bitstrings, one feasible and the other one infeasible. Find where these two bitstrings are located on the master list. Probabilistically pick out a bitstring roughly halfway between the two parents. Test whether this bitstring is feasible or infeasible. In either case, we now have determined another feasible – infeasible pair of parent bitstrings that are closer together on the list than the original parents. Repeat this shrinking until we obtain a feasible – infeasible pair that are adjacent on the list. Since each feasibility test cuts the sublist approximately in half, about n tests, at worst, are required to obtain adjacency.

But two adjacent children may not be genetically similar. For example, if the master list is just the binary form of the numbers from 0 to 63, interpolation might produce the adjacent but genetically very different bitstrings [001111] and [010000]. So, we require the master list to have the following property: adjacent bitstrings must be on the boundary, i.e. differ by a single bit. That may seem like asking for a lot, but there are many such lists; they are known as binary Gray codes. Actually, nobody knows how many binary Gray codes there are. So we can ask for more. Balanced Gray codes, for example, may be useful to us; in these any one bit position has an equal (as possible) probability of changing as one goes through the list.

4. SOME EXAMPLE APPLICATIONS

Here are four examples where combinatorial interpolation combines with constrained optimization.

1. Initial populations might be generated by binary interpolation between a known feasible – infeasible pair such as the all-zero bit string and the all-ones bitstring.
2. Interpolating parents that are on the boundary but pairwise genetically dissimilar would generically explore *along* the boundary.
3. In problems where it is quite difficult to find solutions that are feasible, interpolation may be a valuable source of new feasible solutions.
4. If we interpolate between parents that are close on the Gray code list, and the feasible parent is near an optimum, the children will also be near this optimum.