A Forest Representation for Evolutionary Algorithms Applied to Network Design

A.C.B. Delbem[1] and Andre de Carvalho[1]

University of Sao Paulo – ICMC – USP, Sao Carlos – SP, Brazil,
{acbd,andre}@icmc.usp.br

Abstract. Network design involves several areas of engineering and science. Computer networks, electrical circuits, transportation problems, and phylogenetic trees are some examples. In general, these problems are *NP-Hard*. In order to deal with the complexity of these problems, several strategies have been proposed. Among them, approaches using evolutionary algorithms have achieved relevant results. However, the graph encoding is critical for the performance of such approaches in network design problems. Aiming to overcome this drawback, alternative representations of spanning trees have been developed. This article proposes an encoding for generation of spanning forests by evolutionary algorithms.

1 The Proposed Representation

The proposed forest representation basically consists of linear lists (which may be an array T) containing the tree **nodes** and their **depths**. The order the pairs *(node,depth)* are disposed in the list is important and it must follow a preorder traversal. The forest representation is composed by the union of the encodings of all trees of a forest.

Two operators are proposed (named **operator 1** and **operator 2**) to generate new spanning forests using the node-depth encoding. Both operators generate a spanning forest F' of a graph G when they are applied to another spanning forest F of G. The results produced by the application of the operators are similar. The application of the operator 1 (or 2) to a forest is equivalent to transfer a subtree from a tree T_{from} to another tree T_{to} of the same forest. Applying operator 1, the root of the pruned subtree will be also the root of this subtree in its new tree (T_{to}). On the other hand, the transferred subtree will have a new root when applying operator 2.

In the description of the operator 1, we consider that two nodes were previously chosen: the prune node p, which indicates the root of the subtree of T_{from} to be transferred; and the adjacent node a, which is a node of a tree different from T_{from}. This node is also adjacent to p in G. An efficient procedure to determine such nodes are proposed in [1]. Besides, we assume that the node-depth representation was implemented using arrays and that the indices of p (i_p) and a (i_a), respectively, in the arrays T_{from} and T_{to} are also known. The operator 1 can be described by the following steps:

1. Determine the range (i_p-i_l) of indices in T_{from} corresponding to the subtree rooted at node p. Since we know i_p, we only need to find i_l;
2. Copy the data in the range i_p-i_l from T_{from} into a temporary array T_{tmp} (corresponding to the subtree being transferred) and update the node depths using the depth of a.
3. Create an array T'_{to} (new tree) copying T_{to} and inserting T_{tmp} (pruned subtree) after the node a in T_{to}.
4. Construct an array T'_{from} copying T_{from} without the nodes of T_{tmp}.
5. Copy the forest F to F' exchanging the pointers to T_{from} and T_{to} for pointers to T'_{from} and T'_{to}, respectively.

The operator 2 requires a new root node r, besides the nodes p and a. The copy of the pruned subtree for the operator 2 can be divided in two steps: The first step corresponds to the step 2 for the operator 1 exchanging the range i_p-i_l by i_r-i_l. The array returned by this procedure is named T_{tmp1}. The second step considers the nodes in the path from r to p (i.e. r_0, r_1, $r_2, \ldots$, r_n, where $r_0 = r$ and $r_n = p$) as roots of subtrees. The subtree rooted at r_1 contains the subtree rooted at r_0. The subtree rooted at r_2 contains the subtree rooted at r_1, and so on. The algorithm for the second step copies the subtrees rooted at r_j ($j = 1, \ldots, n$) without the subtree rooted at r_{j-1}, updates the depths [1], and store the resultant subtrees in a temporary array T_{tmp2}. The step 3 of the operator 2 is equivalent to the same step of the operator 1, exchanging T_{tmp} for the concatenation of T_{tmp1} and T_{tmp2} ($T_{tmp} = [T_{tmp1}|T_{tmp2}]$). The steps 4 and 5 are equal in both operators.

2 Final Considerations

This proposal focuses on the production of spanning forests instead of trees (usually found in the literature). As consequence, the operator complexity depends on, for example, the size of the modified trees from F to F', while the complexity of the operators found in the literature are usually functions of the number of nodes and/or edges in the underlying graph. The proposed operators do not require a graph G to be complete in order to produce only feasible spanning forests of G. Many practical problems do not involve complete graphs (in fact, several networks correspond to sparse graphs).

References

[1] A. C. B. Delbem and Andre de Carvalho. New data structure for spanning forest operators for evolutionary algorithms. *Centro LatinoAmericano de Estudios en Informatica – CLEI 2002*, CD-ROM, 2002.

[1] The updated depth of node x is given by $T_{from}[i_x].depth - T_{from}[i_{r_i}].depth + T_{from}[i_r].depth - T_{from}[i_{r_j}].depth + depth\_of\_a + 1$.