A Tree-Based Genetic Algorithm for Building Rectilinear Steiner Arborescences

William A. Greene
Computer Science Department
University of New Orleans
New Orleans, LA 70118 USA
504-280-6755
bill@cs.uno.edu

ABSTRACT

A rectilinear Steiner arborescence (RSA) is a tree, whose nodes include a prescribed set of points, termed the vertices, in the first quadrant of the Cartesian plane, and whose tree edges from parent to child nodes must head either straight to the right or straight above. A minimal RSA (a MRSA) is one for which the total path length of the edges in the tree is minimal. RSAs have application in VLSI design. Curiously, although a RSA is a tree, to our knowledge, previous genetic attacks on the MRSA problem have not used tree-based approaches to representation, nor to the operations of crossover and mutation. We show why some care is needed in the choice of such genetic operators. Then we present tree-based operators for crossover and mutation, which are successful in creating true RSAs from source RSAs without the need of repair steps. We compare our results to two earlier researches, and find that our approach gives good results, but not results that are consistently better than those earlier approaches.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search – *heuristic methods*.

General Terms

Algorithms

Keywords

Genetic algorithms, rectilinear Steiner arborescences, tree-structured chromosomes, tree-based geneticism.

1. INTRODUCTION

Let a fixed set of points in the Cartesian plane be given, and be situated in the interior of the first quadrant. Call each such point a *vertex*. Add the origin into the set as an additional vertex. By definition, a *rectilinear Steiner arborescence* (RSA) is a rooted tree, rooted at the origin, that contains all the vertices as nodes in the tree, and subject to the constraint that each directed edge that goes from a parent tree node to a child node must head either straight to the right or straight above. In order to satisfy the constraint, typically the tree must contain many other nodes besides the vertices. These additional non-vertex nodes are termed the *auxiliary* nodes. Figure 1 shows a RSA on 12 vertices, with the vertex nodes shown as black dots and the auxiliary nodes shown as white dots.

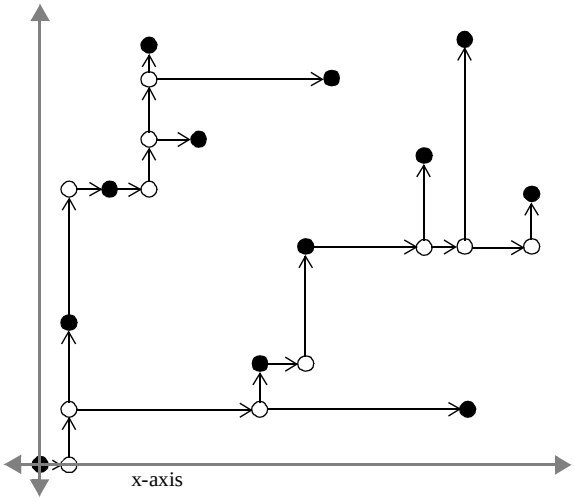

Figure 1. Example RSA

The *total path length* of an RSA is the sum of the lengths of all the edges in it. In general there are many RSAs for a given set of vertices, and two such can have different total path lengths. Figure 2 shows an alternative RSA for the twelve vertices seen in Figure 1. (In Figure 2, one vertex is labeled *A*, for future reference.) The only change is the path which reaches the vertex farthest to the northeast. Clearly the alternative RSA has a lower total path length than the original.

A *minimal rectilinear Steiner arborescence* (MRSA) for a given set of vertices is an RSA for them which has minimal total path length. The problem of finding a MRSA for a given set of vertices has applications in VLSI design; see Cong, Khang, & Leung [3]. In VLSI design, signals from a source must be delivered to terminals, as quickly as possible, by following rectilinear pathways. It has been shown by Shi & Su

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

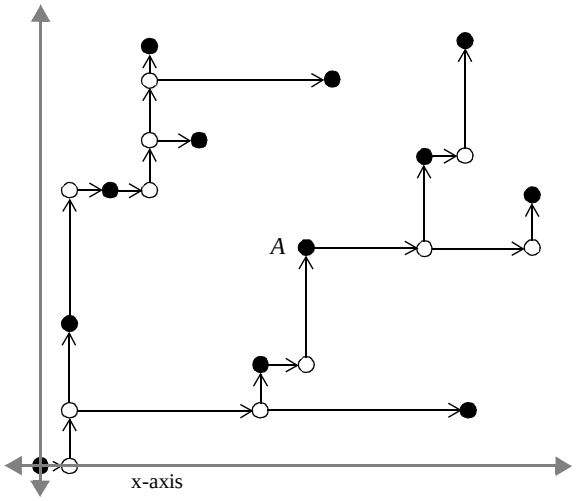

Figure 2. Alternative RSA to Figure 1

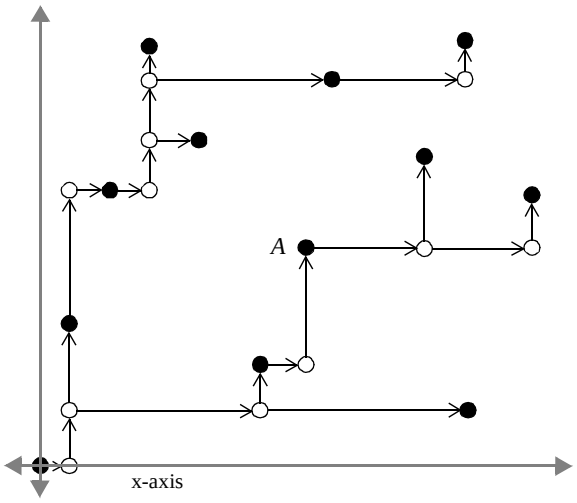

Figure 3. Second alternative RSA to Figure 1

[14] that finding a MRSA for a given set of vertices is NP-complete. Additional research on the MSRA problem has been contributed by Rao *et al.* [13], Córdova & Lee [4], Leung & Cong [11], Ramnath [12], and Julstrom & Antoniades [8], [9].

Of particular interest to us is the work of Rao *et al*. [13] and Julstrom & Antoniades [9]. Details of their work are given in subsequent sections; here we provide a gloss. Rao *et al*. give an elegant greedy algorithm for building a RSA for a given set of vertices; results appear to give often near optimal total path length. Julstrom & Antoniades have invented genetic algorithms that perturb the locations of the vertices, followed by an application of the heuristic of Rao *et al*. Their results often improve upon the already excellent results from Rao *et al*.

Our own interest in this area comes from the following question. Since RSAs are trees, why not employ a genetic algorithm that uses tree-based approaches to crossover and mutation? To our knowledge this has not been practiced in any past efforts on the MRSA problem.

Genetic algorithms that use tree structures as population members, and use tree-based forms of crossover and mutation, have certainly appeared in the field of evolutionary computing. Mostly this has come from practitioners of genetic programming, promulgated first by Koza; see [10]. In genetic programming, population members are functions, represented as expression trees. Interior tree nodes are functions, such as logarithm or addition. The degree of an interior node is the arity of the associated function. Leaves are variables and constants. Crossover, by way of example, can consist of snipping a subtree out of a parent and replacing it by a likewise snipped subtree from the other parent. There are many possibilities for mutation. One form is to replace a function labeling an interior node by another one having the same signature of parameters.

In the case of RSAs, it is not hard to discover what could discourage one from practicing tree-based approaches to the usual genetic operators. Consider Figure 3. Imagine a form of crossover between the RSAs shown in Figure 2 and Figure 3, performed as follows. In both parents, snip away the subtree rooted at the vertex labeled *A* and implant the subtree in the other parent. The two resulting children are unsuitable for our purposes. For instance, one child will lack a copy of the vertex located furthest to the northeast, and the other child will have two copies of this vertex and, moreover, two paths that lead to it. Both children are disasters, and repair operations seem formidable. Section 4 below describes a suitable crossover for such trees.

2. THE HEURISTIC OF RAO *ET AL*.

Given a point $p = (p_x, p_y)$ in the Cartesian plane, define its *rectilinear norm* $||p||$ (also called the *city-block* norm, or the L^1 norm) to be $| p_x | + | p_y |$. Given a second point $q = (q_x, q_y)$, define the rectilinear distance between p and q to be $| p_x - q_x | + | p_y - q_y |$. The rectilinear norm $||p||$ is the rectilinear distance from p to the origin.

Assuming p and q are in the first quadrant of the Cartesian plane, define $min(p, q)$ to be the point $(\min(p_x, q_x), \min(p_y, q_y))$. If p and q do not lie on the same vertical or horizontal line, then the two possibilities are illustrated in Figure 4.

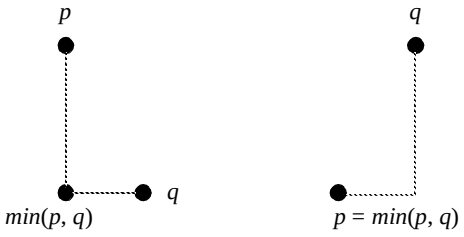

Figure 4. Illustrations of min(p, q)

The heuristic of Rao *et al.* [13] maintains a set of trees (i.e., a forest). Initially each element of the forest is a vertex point (in the set of vertices for which we seek a RSA), treated as a tree with just one node. On one iteration of the heuristic, three steps occur. Firstly, one identifies those forest elements t_1 and t_2 whose roots r_1 and r_2 satisfy: $||min(r_1, r_2)||$ is maximal (i.e., farthest from the origin). Secondly, a new tree t' is constructed by joining the roots of t_1 and t_2 by directed horizontal and

vertical edges from the point $min(r_1, r_2)$. Finally, t' replaces t_1 and t_2 in the forest. Iteration continues until the forest is reduced to just one RSA, rooted at the origin.

3. THE PERTURBATION-BASED GENETIC ALGORITHMS OF JULSTROM & ANTONIADES

Perturbation techniques have proven beneficial in certain geometric minimization problems. In this vein, Julstrom & Antoniades [9] tell us they were inspired by the results for the traveling salesman problem which were obtained by Valenzuela & Williams [15] and Cohoon *et al*. [2].

In [9], Julstrom & Antoniades describe two perturbation-based genetic algorithms for the MRSA problem. Here is our description of their first approach, which they name Long Perturbations. Let n be the number of vertices for which we seek a RSA of low total path length. For their genetic algorithm, a population member (a chromosome) is a list of $2n$ small real numbers, which are thought of as n pairs, one for each vertex.

Each vertex has its location perturbed slightly, by altering its x and y coordinates, respectively, by the first and second small real numbers in its associated pair from the population member of $2n$ small real numbers. Next, the heuristic of Rao *et al*. is applied, in the following interesting way: for *choosing* which pair of forest elements next to unite, it is the perturbed vertices which are used, but it is the unperturbed vertices which are actually used in *assembling* a RSA *à la* Rao *et al*. The total path length of the so constructed RSA is then taken as the fitness of the population member of $2n$ small real numbers.

The members of the initial population are manufactured by taking random real numbers from a normal distribution $N(0, \sigma_1)$. The authors provide these other details: Appropriate crossover operators are positional ones such as k-point crossover. Mutation modifies each value in a population member with values from another normal distribution $N(0, \sigma_2)$. Standard deviations σ_1 and σ_2 depend on the magnitudes and ranges of the set of vertices. Their genetic algorithm is a generational one. Population size is n, the same as the number of vertices. Evolution is allowed to run for $3n$ generations, and the best individual found therein is returned as the result. To form the next generation, 1-elitism is practiced: the best individual is automatically carried into the next generation. To fill the rest of the next generation, crossover is used 70% of the time, with parents chosen by tournament selection, and mutation of a previous population member is used 30% of the time.

The results from this perturbation-based genetic algorithm are very good, and will appear later in this paper. The authors have a second genetic algorithm, which is similar to the first one. For the second algorithm, it is the distance of a vertex to the origin which is perturbed, and a population member is thereby a list of (not $2n$ but rather) n small real numbers. To this approach the authors give the name Short Perturbations. The performance of this second algorithm is very similar to that of their first one, and so we will ignore the second algorithm. Actually, the authors introduce a third genetic algorithm that encodes an RSA as a permutation of the vertices, but that algorithm did not perform as well as the others, and it will not concern us here.

We took the research of Julstrom & Antoniades as our starting point. In particular, in our experiments we use the data sets used by them. The data sets originated with Beasley, and are online [1]. Each data set consists of points in the unit square $[0, 1]^2$ in the Cartesian plane. Sets are grouped; there are some data sets of 50 points, others of 70, 100, and 250 points (and yet others, not used by Julstrom & Antoniades nor us).

4. A TREE-BASED GENETIC ALGORITHM FOR THE MRSA PROBLEM

Introductory to discussing geneticism for the MRSA problem, we step back for a moment to view a bigger picture. Imagine there is some interesting problem at hand, for which there are numerous solutions, some of which are better than others. Assuming solutions differ one from another by exhibiting different property values, for some set of properties that can be used to describe solutions, then in classical genetic algorithms, a solution gets represented as its set of property values, lined up in a sequence, just like the genes strong linearly along a chromosome. Then genetic operators of crossover and mutation can be applied. All this is well illustrated by the traveling salesman problem. A salesman must visit k cities, and we seek the shortest itinerary. A solution can be represented as a permutation of the k cities, and the fitness of a solution is the total mileage driven in visiting the cities in the order dictated by the permutation.

But to structure a chromosome as a linear sequence of entities is not necessarily most natural to the problem at hand. For instance, tree structures are more natural in genetic programming. For traffic flow problems in a city, to structure a solution as a grid or graph may be most natural. Theoretical issues with these ideas have appeared in the literature; to cite but one author, there is Greene [5], [6].

For tree-structured chromosomes, appropriate forms of the genetic operators easily come to mind. Crossover can mean exchange of subtrees between parents. Mutation can mean change of a property value at a random node, or perhaps interchange of two randomly chosen nodes. Earlier, using the parents shown in Figure 2 and Figure 3, we showed there are pitfalls for RSAs, because two RSAs built for the same vertex set can have different tree structures, perhaps even radically different structures. Now we will describe a general approach to crossover that is both natural for trees and natural for RSAs.

Let a set of vertices be given. A RSA for this set includes the vertices as tree nodes, and also there are the auxiliary nodes that appear in the RSA. If we grant equal citizenship to the auxiliary nodes, then we see that a RSA is in fact a binary tree. Each node has at most two out-edges, one heading straight right and one heading straight up. Now we want to think of snipping off some subtrees in the RSA. We have in mind snipping off the subtrees contained in some well-chosen region, which for the moment we will take to be one side of a certain line. If the tree edge leading to a node is crossed by the line, then we wish to snip out the subtree rooted at that node. But ambiguities can arise. Consider the RSA and dotted line shown in Figure 5. (The RSA is the same one seen in

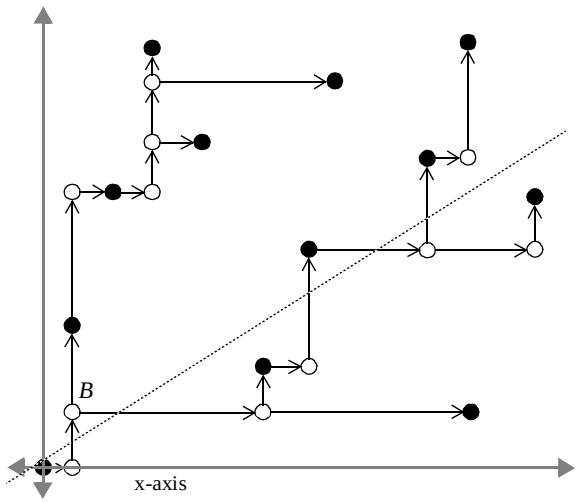

Figure 5. Ambiguous severance of subtrees

Figure 2.) Imagine we snip away subtrees that are rooted above the line. Closest to the origin, the line can snip away the subtree rooted at the auxiliary node labeled *B*. But the line also crosses other edges within that subtree. It is not obvious which subtrees are the ones that we should say are snipped away by virtue of being above the line shown.

So one must exert some care in choosing the regions and lines that snip away subtrees. Let us say a region H in the Cartesian plane is *northeast-inclusive* provided: if a point p is in H, then any point q that lies to the northeast of p is also in H. (We include the cases that q lies on a horizontal or vertical line with p.) Here are some examples of regions that are northeast-inclusive.

- the half-plane to the northeast of a straight line with negative slope;
- the half-plane to the right of a vertical line (this is a special case of the first case, with slope = $-\infty$);
- the half-plane above a horizontal line (this is a special case of the first case, with slope = 0);
- the set of all points that lie to the northeast of any point p;
- the set of points northeast of the graph of the hyperbolic function $y = 1/x$;
- the sets of points northeast of translations of the graph of the function $y = 1/x$;
- the set of points northeast of the graph of a differentiable (therefore continuous) function f: $[0,1] \rightarrow \mathrm{R}$ with negative derivative (this generalizes most of the above cases);
- the set of points northeast of the graph of a function that is monotone decreasing (this generalizes most of the above cases).

Here are the key insights. Since tree edges in a RSA can only head straight right or straight up, it follows that if a tree node lies in a northeast-inclusive region H, then the entire subtree rooted at that node also lies in H. Also, if T_1 and T_2 are two RSAs for the same vertex set, and H is a northeast-inclusive region, then the set of vertices of T_1 that lie in H is the same as the set of vertices of T_2 that lie in H. This is because whether a vertex lies in H is independent of the tree structures of T_1 and T_2. Next, consider the set S_1 (resp., S_2) of subtrees of T_1 (resp., T_2) whose roots are highest in the tree T_1 (which means the roots are closest to the origin in the Cartesian plane), among those subtrees which are rooted at points in H. Again, S_1 and S_2 contain the same set of vertices. Subtree sets S_1 and S_2 can be exchanged between T_1 and T_2. That is, for instance, if the subtrees in S_2 are attached to T_1, after the latter has been shorn of the subtrees in S_1, the result will be a RSA for the vertex set and will in particular contain exactly one copy of each vertex.

To attach a subtree in S_2 to the shorn T_1 of course means to attach its root r into the tree structure of the shorn T_1. There are three possibilities: attach the root r to a tree node z in T_1 that lies southwest of r, or attach r to a horizontal tree edge that lies below it, or to a vertical edge that lies to the left of r. See Figure 6. We attach r to the closest entity that is one of

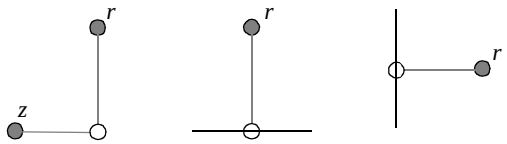

Figure 6. Possible attachments of node *r*

these three possibilities, with distance being rectilinear distance. Recall that our data sets lie in the unit square $[0, 1]^2$ in the Cartesian plane. As in Julstrom & Antoniades [9], in the case we attach r to a tree node z in the shorn T_1, then of the two possible auxiliary corner nodes (northwest or southeast) between r and z, we choose the one closer to the diagonal line $y = x$.

It is important to note that the order in which the elements of S_2 are attached to the shorn T_1 makes a difference. If subtree a_1 is attached before a_2, then possibly when a_2 is attached, it gets attached to an entity in a_1. Our experience was that the elements of S_2 should be sorted with some care. Our best result was, with 50-50 probability, to use one of two sorts, next described.

The first is a modified shell sort, that sorts subtrees so they satisfy the following property. If a subtree's root r_1 lies to the southwest of another subtree's root r_2, then r_1's subtree gets positioned so that it precedes that of r_2. The relation "point p lies to the southwest of q" does not give a total order on points in the Cartesian plane, so more mundane sorts such as bubble sort or insertion sort do not typically arrange subtrees so as to satisfy the above property. This motivates our use of a shell sort approach.

The second sort arranges points so those nearer the x- and y-axes come before points nearer the diagonal line $y = x$. Points $p = (p_x, p_y)$ are sorted by focusing on the slope $m_p = p_y / p_x$ of the line between p and the origin. Specifically, we sort points as m_p ranges $+\infty \rightarrow 1^+$ (nearer the y-axis, then approaching the diagonal line $y = x$, from above), but interleave them with the points for which m_p ranges $0 \rightarrow 1^-$ (nearer the x-axis, and approaching the diagonal line, from below). The motivation is that, for instance, the point with greatest slope m_p must be connected to the origin by employing a vertical edge that many other nodes may later connect to.

Details of the Geneticism

4.1 Crossover

The fundamental ideas of how we perform crossover have now been presented. Some additional details now follow.

For our northeast-inclusive regions, we use three. First a random point p in the unit square is concocted. Then we use the region above the horizontal line through p, the region to the right of the vertical line through p, or the region of points to the northeast of p, with respective probabilities 15%, 15%, 70%.

Experience proved it was better to do more than snip subtrees at the tree edges which cross the boundary into the northeast-inclusive region H. The parent node of such an edge, and the child node, could both be auxiliary nodes. With the thought that auxiliary nodes exist merely at the service of the verticial nodes, in fact our snipping does more. In the RSA, "above" the snip in the tree (which means towards the southwest part of the unit square in the Cartesian plane) and iterating by levels, leaves that are not vertices are discarded. Similarly, the child subtree (at the end of the tree edge that crosses the boundary into region H) is further pruned down to its set of descendant subtrees which are rooted at vertices. Thus the snip eliminates tree apparati in between one or more vertices inside H and the nearest verticial ancestor that is outside region H.

4.2 Generational Turnover

Our genetic algorithm is generational (versus steady-state). All of the population at generation $t+1$ is accumulated by applying the evolutionary operators of survival of the fittest, mating with crossover, and mutation, to the members of generation t. Elitism is practiced: the best 2 members of a generation are carried over into the next generation. Each generation is sorted into ascending order of total path length. For mating with crossover, parents are chosen by rank order selection: an individual is selected with a probability that is a linear function, of negative slope, of its position within the ascendingly sorted population (thus, better individuals are more likely to be selected). Two parents produce two children. A child enters the next generation only if it does not already appear there. Once the entire population is assembled, all non-elite individuals are subjected to mutation. Finally, the resulting new population is sorted.

On any given trial for a fixed vertex set, we let population size be $3n$, where n is the number of vertices, and we ran the evolution to $6n$ generations. The best individual that surfaced over those generations was then taken as the result of the trial.

4.3 Fitness

As the reader can now predict, for us the "fitness" of a RSA is its total path length, which we are trying to minimize. Once the two child RSAs have been constructed from our crossover operation, their total path lengths must be calculated. An economy is available for this, although it is not implemented in our present work. Namely, the total path length of a snipped subtree does not need to be recalculated. Instead, its total path length just needs to be incorporated into that of its ancestors in the tree it joins.

4.4 Mutation

One step in the mutation of a RSA is the following sequence of actions. Pick a random vertex z. Find its nearest verticial ancestor w. Remove the tree structure below w and meanwhile collect the vertices below w. In our present implementation, we apply two of these "steps" when we mutate. Then, using the same regimen as described when crossover appends subtrees into a shorn RSA (the beginning half of Section 4), append the liberated vertices back into the RSA. In particular, first the liberated vertices are sorted, employing, with equal probability, one of the two methods for sorting tree nodes (or points in the unit square) that were described earlier.

4.5 The Initial Population

Recall that the order in which tree nodes are appended into a growing RSA makes a difference. Each element of the initial population is formed as follows. First, randomize the order of a list of the vertices. Then, in that order, add the vertices to an initial RSA which consists of just the origin.

(Lastly, regarding the general topic of how we construct RSAs, there is another embellishment. An auxiliary node which is a right-child and which has a right-child but no above-child exemplifies an unnecessary auxiliary node. Similarly an above-child can be an unnecessary auxiliary node. We eliminate unnecessary auxiliary nodes. An unnecessary auxiliary node does not affect the total path length of an RSA, but does complicate how one defines equality.)

5. RESULTS

In [9], Julstrom & Antoniades compare algorithms which are run on 20 data sets. There are 4 groups, each containing 5 data sets. The data sets are grouped according to how many vertices there are to be incorporated in a RSA. There are 5 data sets consisting of, respectively, 50, 70, 100, and 250 vertices. The data sets are the first 5 instances found in Beasley's on-line OR library [1], under the heading "Euclidean Steiner problem".

As an illustration, we begin our results with an actual RSA. Figure 7 depicts our best RSA found for the second data set consisting of 50 vertices. In the figure, only the vertex nodes

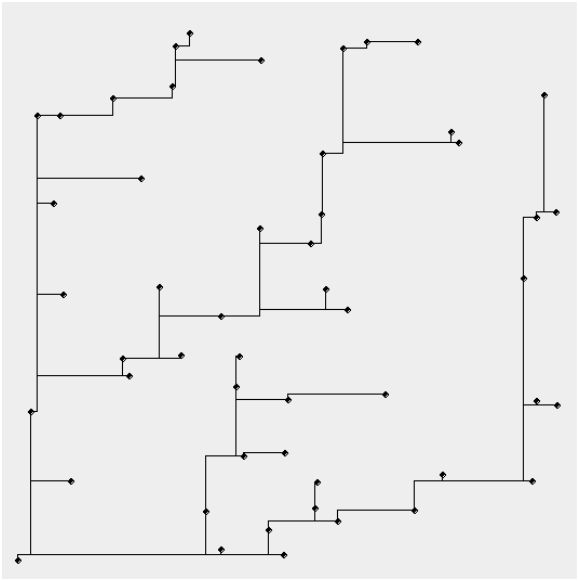

Figure 7. Best RSA for 50 vertices, Data Set # 2

Table 1. Results and comparisons

Num Points	Data Set #	Rao *et al.*	Long Perturbations	Percent Improvement	Tree-Structured	Percent Improvement
50	1	7.163	7.072	1.27%	7.073	1.25%
	2	6.576	6.524	0.79%	6.529	0.71%
	3	6.589	6.590	– 0.02%	6.703	– 1.74%
	4	6.509	6.272	3.64%	6.310	3.04%
	5	6.771	6.687	1.24%	6.755	0.23%
70	1	7.971	7.836	1.69%	7.921	0.63%
	2	7.594	7.501	1.22%	7.568	0.33%
	3	7.483	7.462	0.28%	7.533	– 0.67%
	4	7.835	7.643	2.45%	7.676	2.02%
	5	7.121	7.114	0.10%	7.279	– 2.23%
100	1	8.870	8.869	0.01%	9.030	– 1.81%
	2	9.161	9.003	1.72%	9.186	– 0.28%
	3	9.039	9.027	0.13%	9.350	– 3.45%
	4	9.408	9.046	3.85%	9.157	2.67%
	5	8.840	8.810	0.34%	9.027	– 2.12%
250	1	14.158	13.993	1.17%	14.889	– 5.17%

are depicted, as black dots. The auxiliary nodes would appear at the other branch points in the tree.

In Table 1 we show the performance of our tree-based genetic algorithm, in comparison to the heuristic of Rao *et al.* (Section 2) and in comparison to the Long Perturbations genetic algorithm of Julstrom & Antoniades (Section 3).The values for total path length that are given in the columns for Rao *et al.* and for Long Perturbations are taken from the paper [9] by Julstrom & Antoniades. The column headed as Tree-Structured gives the values for total path length from our own approach, and each value listed is the best one found over 20 trials, for the given data set. The columns headed as Percent Improvement mean the improvement of the associated genetic algorithm, over the result of Rao *et al.*, when run on the same data set. A positive improvement means real improvement; a negative improvement means worse performance than Rao *et al*. We show performance for only one data set of 250 vertices, since our own algorithm ran impractically long on it and we abandoned the remaining 4 instances.

Table 1 shows that our algorithm usually improves upon the RSAs created by Rao *et al.*, for the cases of 50 and 70 vertices, but then is beginning to lose its advantage for larger vertex sets. Our algorithm roughly parallels that of Julstrom & Antoniades, in that when their improvement is positive and on the larger side, then our performance is positive, and when their improvement is slight then ours is negative. Certainly, our algorithm is outperformed by the perturbation-based genetic algorithm of Julstrom & Antoniades. This comes as a surprise to us, since we are of the general sentiment that the most natural approach to a problem should give the best results. RSAs are trees, and so we entered our research effort with the faith that a tree-based approach to representation of individuals and to the genetic operators of crossover and mutation would produce better RSAs, usually, than our two competitors. We must tip our hat to the excellent results of Julstrom & Antoniades. Of course, the heuristic of Rao *et al*. is itself already an admirable performer, which builds its RSAs in one pass, without the greater cost of evolving over many generations.

Our faith that our tree-based approach would win out over earlier approaches was not rewarded in fact, and of course we must wonder why. Many trials suggested that our cuts by northeast-inclusive regions resulted in many subtrees that had to be appended to another shorn parent. That is, population members underwent some degree of churning. But if one is to avoid the problems indicated in the last paragraph of Section 1, it is necessary to migrate *all* the subtrees found in a northeast-inclusive region. Trouble-free alternative ways of cutting, so as to produce fewer migrating subtrees, did not occur to us in the midst of the research trials. An approach that is worth trying but has not yet been implemented is to let a majority of the regions be ones northeast of points which are close to the "north" and "east" sides of the unit square. Such regions are clipping off fewer subtrees, and those subtrees are on levels near the leaves.

An anonymous reviewer made an interesting observation: the approach of Julstrom & Antoniades uses the heuristic of Rao *et al* and so is always likely to do at least as well as that heuristic. By comparison, our tree-based approach has a harder task, since it must go the whole distance on its own. The suggestion offered is to include the RSA obtained *à la* Rao *et al* as a seed in the population.

6. CONCLUSION

We have presented a tree-based approach for a genetic algorithm for the minimal rectilinear Steiner arborescence problem. To our knowledge, it is the first that is really tree-based. The representation of a RSA as a tree is of course completely natural. It is our operator for crossover that is our most original contribution. The operator is natural for trees, and creates offspring that are true RSAs without the need for any repair work.

Our results are rather good, but are bettered by those of Julstrom & Antoniades. And it appears our approach would be bettered by the heuristic of Rao *et al*. as vertex sets get larger.

Ideas have occurred to us that conceivably could lead to improved performance of our general approach. We give them next.

Perhaps a local improvement operator is called for. Consider Figure 1 again. Regarding the vertex that is located most to the northeast, the path to it is plainly longer than necessary. An obvious improvement is afforded by the RSA in Figure 2, where the vertex is seen appended to another nearby vertex. To our observation, vertices that would be better attached to some nearby vertex were a not uncommon phenomenon. An idea for a local improvement operator is to re-optimize the sub-RSAs that are the lower-level subtrees of a population member.

We found that a surprisingly sensitive issue was how one sorts the subtrees snipped from one parent before entering them into the other shorn parent. Perhaps there are better sorting methods.

Similarly, the tree cuts which we made for crossover were rather simple ones. Perhaps better ones can be found by following the suggestions for northeast-inclusive regions which we listed in the text below Figure 5.

Again we will say our results are rather good (even if often bettered by other approaches). As a positive closing note, we would let our research stand as a good advertisement for tree-based approaches to geneticism, when trees are the natural structure for individuals in the population.

Finally, we express our thanks to an anonymous reviewer whose commentary was lengthy and perceptive.

7. REFERENCES

[1] Beasley, J. E.: OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society* 41(11) (1990) 1069-1072. The on-line collection of the data sets is at URL `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`. The data sets for the rectilinear Steiner arborescence problem are found through the link labeled *Euclidean Steiner problem*.

[2] Cohoon, J. P., Karro, J. E., Martin, W. N., Niebel, W. D.: Perturbation method for probabilistic search for the traveling salesperson problem. In *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation*, Vol 3455 of Proceedings of SPIE, SPIE Press (1998) 118-127.

[3] Cong, J., Khang, A. B., Leung, K. S.: Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17 (1998) 24-39.

[4] Córdova, J., Lee, Y. H.: A heuristic algorithm for the rectilinear Steiner arborescence problem. Technical Report TR-94-025, Department of Computer Science, University of Florida (1994).

[5] Greene, W. A.: A non-linear schema theorem for genetic algorithms. In Whitley, D. *et al*. [Eds.] *Proceedings of the Genetic and Evolutionary Computation Congress (GECCO 2000)*, Morgan Kaufmann Publishers (2000) 189-194.

[6] Greene, W. A.: Schema disruption in tree-structured chromosomes. In Beyer, H.-G. *et al.* [Eds.] *Proceedings of the 2005 Genetic and Evolutionary Computation Congress (GECCO 2005)*, ACM Press (2005) 1401-1408.

[7] Julstrom, B. A.: Encoding rectilinear Steiner trees as lists of edges. In Lamont, G. B., Yfantis, E. A., Haddad, H., Papadopoulos, G. A., Carroll, J. [Eds.], *Proceedings of the 16th ACM Symposium on Applied Computing*, New York, ACM Press (2001) 356-360.

[8] Julstrom, B. A., Antoniades, A.: Two hybrid evolutionary algorithms for the rectilinear Steiner arborescence problem. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, Nicosia, Cyprus (2004).

[9] Julstrom, B. A., Antoniades, A.: Three evolutionary codings of rectilinear Steiner arborescences. In *Genetic and Evolutionary Computation Conference (GECCO 2004)*, Springer Verlag, LNCS 3102 (2004) 1282-1291.

[10] Koza, J.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[11] Leung, K. S., Cong, J.: Fast optimal algorithms for the minimum rectilinear Steiner arborescence problem. *Proceedings of the International Symposium on Circuits and Systems* (1997) 1568-1571.

[12] Ramnath, S.: New approximations for the rectilinear Steiner arborescence problem. *IEEE Transactions on Computer-Aided Design* 22 (2003) 859-869.

[13] Rao, S. K., Sadayappan, P., Hwang, F. K., Shor, P.W.: The rectilinear Steiner arborescence problem. *Algorithmica* 7 (1992) 277-288.

[14] Shi, W., Su, C.: The rectilinear Steiner arborescence problem is NP-complete. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms* (2000) 780-786.

[15] Valenzuela, C. L., Williams, L. P.: Improving simple heuristic algorithms for the traveling salesman problem using a genetic algorithm. In Bäck, T. [Ed.] *Proceedings of the Seventh International Conference on Genetic Algorithms*, San Francisco, CA, Morgan Kaufmann Publishers (1997) 458-464.