Tree Network Design with Genetic Algorithms- An Investigation in the Locality of the Pruefernumber Encoding

Franz Rothlauf and **David Goldberg**

Working Paper 6/1999
April 1999

Working Papers in Information Systems
Editor: Prof. Dr. Armin Heinzl

University of Bayreuth
Chair of Information Systems
Universitaetsstrasse 30
D-95440 Bayreuth, Germany
Phone +49 921 552807, Fax +49 921 552216
E-Mail: wi@uni-bayreuth.de
Internet: `http://wi.oec.uni-bayreuth.de`

Bad Codings and the Utility of Well-Designed Genetic Algorithms

Franz Rothlauf
Chair of Information Systems
University of Bayreuth
Universitaetsstr. 30
D-95440 Bayreuth/Germany
rothlauf@ieee.org

David Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Av. Urbana, IL 61801
deg@illigal.ge.uiuc.edu

April 30, 1999

Abstract

When handling tree networks, a number of researchers have tried using the pruefernumber representation for encoding the network, but GAs often degraded or broke down when used on this encoding. This paper investigates the locality of the pruefernumber, which can be described as the relatedness of the phenotype (the tree) and the genotype (the pruefernumber) of trees. It is shown that the locality is highly irregular on the entire solution space. We demonstrate that for star and list networks small changes of the pruefernumber lead to small changes in the tree, whereas for all other networks the locality of the pruefernumber is low. Even worse, all areas of high locality are separated from each other by large areas of low locality. Using a GA with the pruefernumber can be useful when the good solutions tend to be a "star" or a "list". As soon as the GA should find good solutions, which are more general trees, a degradation or even complete failure of the GA is inescapable.

1 Introduction

When applying GAs on tree networks sometimes the pruefernumber is used for encoding the trees (Julstrom, 1993). But often this type of encoding results in poor GA performance, which some have attributed to be due to the bad locality of the encoding.

In this paper we investigate the locality of the pruefernumber encoding. We want deeper insights into why this encoding fails and try to give hints to other researchers how they could use this representation more effectively.

The remainder of the paper is organised as follows. We start with a short overview how the pruefernumber is constructed. Section 3 then describes some properties of the pruefernumber encoding. A description about the computer experiments is given in Section 4. The results are presented in Section 5 and the paper ends with a short conclusion.

2 From the phenotype to the genotype of a tree and vice versa

This section gives a short review on how to get a pruefernumber (the genotype) from a tree (the phenotype) and how to get the tree back from the pruefernumber.

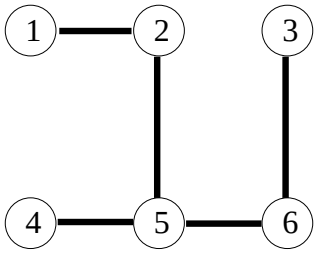

Figure 1: A tree network and the corresponding pruefernumber $P = 2565$

Tree networks are generally defined as an undirected and connected graph with no cycles. For a tree with n nodes are exactly $n-1$ links. For a graph with n nodes there are exactly $n^{(n-2)}$ trees. The pruefernumber for a n-node tree is a number with $n-2$ digits. Each digit is of base n.

To understand the way a pruefernumber works a little bit better, let's have a short look how to construct the pruefernumber from the tree and vice versa.

2.1 The construction of the pruefernumber

Every tree has at least two nodes, which have only one connecting edge (the degree of the node is 1). All nodes are labelled with numbers from 1 to n.

The pruefernumber can be constructed by the following algorithm:

1. Let i be the lowest numbered node of degree 1 in the tree.
2. Let j be the one node, which is connected to i (there is exactly one!). The number of the jth node is the rightmost digit of the pruefernumber.
3. Remove node i and the edge (i, j) from the tree and from further consideration.
4. Go to 1 until only two nodes (that means one link) are left.

After termination you have a pruefernumber with $n-2$ digits which represents your tree.

Let us demonstrate the construction of the pruefernumber with a short example. The network in figure 1 has 6 nodes. Therefore the pruefernumber consists of 4 digits.

The lowest numbered node with degree 1 is node 1. This node is connected to node 2 so the pruefernumber starts with a 2. We remove node 1 from further consideration and look for the lowest numbered node with degree 1.
We find the node 2, which is connected to node 5. The pruefernumber becomes 25. After removing node 2, we see that node 3 is now the lowest numbered node, which is eligible. Node 3 is connected to 6 so we get 256. The node 4 is the lowest eligible node and we add 5 to the pruefernumber. Now only two nodes remain in the tree. We have to stop and the resulting pruefernumber is 2565.

2.2 The construction of the tree from the pruefernumber

The construction of the tree from the pruefernumber follows the construction of the pruefernumber.

1. Let P be a pruefernumber with $n-2$ digits. All node numbers which are not in P can be used for the construction of the network (are eligible).
2. Let i the lowest numbered eligible node. Let j be the leftmost digit of P.

3. Add the edge (i, j) to the tree.

4. Designate i as no longer eligible and remove the leftmost digit j from the pruefernumber.

5. If j does not occur anywhere else in the remaining pruefernumber, designate j as eligible.

6. Go to 2 until there remain no digits in the pruefernumber. If no digits left then there are exactly two numbers r and s eligible. Add at last the edge (r, s) to the tree.

3 Properties of the pruefernumber encoding

In this section we give a short overview on the properties of the pruefernumber in relation to genetic and evolutionary algorithms (Palmer and Kershenbaum (1994); Gen, Zhou, and Takayama (1998)).

3.1 Benefits

The use of the pruefernumber has some remarkable benefits:

- every tree can be represented by a pruefernumber
- only trees are represented by pruefernumbers
- every pruefernumber represents exactly one tree
- all trees are equally represented

A look at the construction rule of the pruefernumber shows that the pruefernumber is able to represent all possible trees. Each tree has at least two nodes with degree 1 so the construction rule can be applied to all possible trees.

It was shown by Pruefer (1918) that the pruefernumber represents only trees. This means you can create random numbers and they represent always a tree. In contrast to many other representations no repairing of a randomly chosen representation is necessary. It is also not necessary to repair the individuals, that are generated in each generation. This normally promises better results than an encoding where the offspring must be repaired.

In addition, all trees are represented equally. Each tree is represented by exactly one specific pruefernumber. The number of different trees for a graph with n nodes is n^{n-2}. There are also exact n^{n-2} different pruefernumbers for a n node tree. So if a GA is applied to a network problem that is encoded with the pruefernumber, you have not to worry with biased populations.

3.2 Drawbacks

The pruefernumber has also some disadvantages:

- complex calculation
- bad locality

The calculation of the pruefernumber is a little bit complex, but it can be done in $O(n \log n)$. This seems to be not too bad. The real disadvantage of the pruefernumber is the little locality of the representation. Small changes in the representation can lead to big changes in the represented network. This means the mapping from the phenotype to the genotype is not homogeneous. A

little change in the tree (e.g. you change one edge in the tree) could lead to a big change in the pruefernumber and vice versa.

This means that the basic mutation operator that searches the local solution space around the individuals, does a poor job. In such a case, mutation works not as a local search, but more as a random search over the solution space.

The same problem happens to the crossover operator. If there is little locality, a combination of two (or more) individuals produces offsprings, which have little in common phenotypically with their parents.

With a poor locality of the pruefernumber representation, normal GAs should generally perform poorly and may even fail completely on this encoding.

4 Experiments

This section presents computer simulations on evaluating the differences between the locality of the phenotype and genotype of tree networks. The topology of the network is encoded as a pruefernumber. Each pruefernumber is represented as a bitstring.

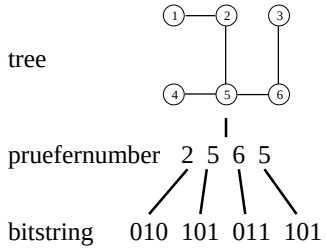

Figure 2: Encoding of a tree

The number of nodes n is set to 2^i with i a whole number. Results are presented for 8, 16 and 32 node tree networks.

There are three different types of networks:

- Star: One node is of degree $n-1$ and the rest of the nodes have degree 1.
- List: Two nodes are of degree 1 and the rest of the nodes have degree 2.
- Tree: Every other tree network

Looking at the locality means that you can either have a random walk through the search space or you explore the complete neighbourhood of one individual. We investigate the locality in six different ways:

- random walk through the genotype (each step means a one bit change of the bitstring)
- random walk through the pruefernumber (change of one digit in the pruefernumber)
- random walk through the phenotype (change of one edge in the tree)
- neighbourhood of a bitstring (all neighbours that are different in one bit from the examined individual)
- neighbourhood of a pruefernumber (all neighbours that are different in one bit)

- neighbourhood of a tree network (all neighbours are different in one edge)

During a random walk through the bitstring representation of a tree one bit in the bitstring is randomly changed and the change of edges in the tree is examined. The results are presented in figure 3. When walking through the pruefernumber encoding one digit of the pruefernumber is changed and the resulting edge changes are investigated (fig. 5). A random walk through a tree means, that one edge of the tree is replaced by a randomly chosen edge. Then either the difference of bits in the bitstring (fig. 4) or digits in the pruefernumber (fig. 6) between the actual and the last individual is examined. The start-individual (bitstring, pruefernumber or tree) is chosen randomly. Because of independence from the start solution and to get statistical significance 400 steps were carried out in each run.

Exploring the neighbourhood of the pruefernumber means that you have a look at all individuals, which are different in one bit or one digit in comparison to the examined individual. We have to distinguish between the neighbourhood of the pruefernumber itself and the neighbourhood of the bitstring that represents the pruefernumber. Each pruefernumber with k digits of base $k+2$ is represented by a bitstring of length $k * \lceil ld(k+2) \rceil$. A change of one digit in the pruefernumber can result in up to $\lceil ld(k+2) \rceil$ different bits. The number of neighbours for a bitstring is equal to its length $k * \lceil ld(k+2) \rceil$. It is independent of different network types. A look at the neighbourhood of the pruefernumber itself shows that each pruefernumber with $n-2$ digits of base $n = k+2$ has $(n-1)(n-2)$ neighbours. The number of neighbours is independent of the network type. The resulting numbers of neighbours are presented in table 2.

The neighbourhood of a bitstring can be seen in figure 3 and of the pruefernumber in figure 5. The neighbourhood of the phenotype is illustrated in figure 4 and 6. It is shown how many bits of the bitstring (fig. 4) or digits of the pruefernumber (fig. 6) are different, if one edge of the tree is changed.

All plots on the top of figure 3, 4, 5 and 6 show the results for 8 node trees. In the middle the results for 16 and at the bottom the results for 32 nodes tree networks are shown.

When looking at the phenotype, you must consider, that the change of one digit of the pruefernumber means that 3 (8 node network), 4 (16 node network) or 5 (32 node network) bits are affected.

For each problem, 20 independent runs were performed in order to get results with statistical significance.

5 Results

This section presents computer simulations on the examination of the locality of the pruefernumber. We start with investigating how many neighbours a bitstring, a pruefernumber or a tree has. This is followed by an investigation of the locality by doing a random walk through the pheno- and genotypes of trees. The section ends with a look at the neighbourhood of individuals.

5.1 Number of neighbours

Different star networks have only one edge in common[1]. Each star network has exactly $(n-1)(n-2)$ neighbours. For a list network each tree has $\sum_{i=1}^{n-1} (n-i-1) * i - (n-1)$ neighbours. All digits of a pruefernumber that encodes a list, are a permutation of $n-2$ different digits of base n (e.g. P =124 or P =312 as examples of a 5 node list). The number of neighbours for tree networks lies in between the number for star and list networks and must be calculated for each network separately.

[1]For a star network all digits of the pruefernumber have the same value (e.g. P =222 for a star with center 2).

It depends on the different degree of the nodes in the tree. The number of neighbours is shown in table 1 for different tree types.

number of nodes	network type	number of neighbours
8	star	42
	tree	62.45 (avg.)
	list	77
16	star	210
	tree	447.2 (avg.)
	list	665
32	star	930
	tree	2595 (avg.)
	list	5425

Table 1: number of neighbours for trees (different in one edge)

Each pruefernumber with $k = n - 2$ digits has exactly $(n-2) * (n-1)$ neighbours. Comparing the number of neighbours of a pruefernumber (table 2) to the numbers of neighbours of a tree (table 1) shows a one to one correspondence for star networks. A star has as many neighbours as a pruefernumber. It can be seen in subsection 5.3 that for star networks not only the same number of neighbours exists, but also the locality is the same. When modifying the tree toward a list network the number of neighbours for the pruefernumber stays constant, whereas the number of neighbours for the tree is increasing. The result is different locality in different areas of the solution space.

number of nodes n	number of neighbours (bitstring)	number of neighbours (digit)
	$(n-2) * \lceil ld(n) \rceil$	$(n-2)(n-1)$
8	18	42
16	56	210
32	150	930

Table 2: number of neighbours for a bitstring (individuals with one bit difference) and the pruefernumber itself (individuals with one digit difference)

5.2 Random walks

We present the results for random walks through the solution space.

5.2.1 Random walks through the genotype

A look at the results for a random walk through the bitstring (fig. 3) and pruefernumber space (fig. 5) shows that most of the one bit or one digit changes lead to a one edge change in the tree. But it can be seen that a one bit change can also lead to an almost complete different tree. Figure 3 shows that for 8 node networks almost 35%, for 16 and 32 nodes networks about 55% of all one

bit changes lead to changes in the phenotype of at least three different edges. The situation is the same for the pruefernumber. The locality of the genotype (bitstring or pruefernumber) is low.

5.2.2 Random walks through the phenotype

When walking through the solution space of the phenotype (tree networks) the plots in figure 4 show an accumulation between 1 and 5 bit changes for the pruefernumber, but also a remarkable high possibility for high order changes, when changing one edge in the tree. For the pruefernumber (fig. 6) about 75% of the neighbouring individuals are different in more than one digit. The locality of the phenotype seems to be quite bad.

5.3 A look at the neighbourhood

When looking at the neighbourhood, it must be considered that the pheno- and genotype has different numbers of neighbours[2].

5.3.1 Neighbourhood of the genotype

The neighbourhood of a bitstring (fig. 3) and a pruefernumber (fig. 5) is different for different network types. The neighbourhood of a bitstring or a pruefernumber that encodes a star network shows that a change of one bit in the bitstring or one digit in the pruefernumber results in a change of one edge in the encoded star network[3]. The locality of the pruefernumber for star networks represents a one to one projection from the genotype to the phenotype[4].

A look at the neighbours of a bitstring that encodes a list network shows that the change of one bit results in a maximum change of 4 edges (fig. 3) in the tree independent from the number of nodes. The pruefernumber (fig. 5) shows the same behaviour. A change of one digit leads to a maximum change of 4 edges. The locality of the pruefernumber for list networks is a little worse than for stars, but much better as for tree networks.

Bitstrings and pruefernumbers that encode tree networks show very low locality. All neighbours that can be created by an one bit change of the bitstring (fig. 3) or a one digit change of the pruefernumber (fig. 5) have phenotypically not much in common with the considered network.

These results seems quite interesting for the use of a GA on this encoding. The locality of the pruefernumber is not the same on the whole solution space. When a GA explores the neighbourhood of a star network, it is really a search around one individual. Also a search around a list network is quite local. When it comes to searching the solution space around a common tree network however, the search is not more than a blind and random search. A GA will not be able to find a nearby good solution. If a good solution is near a star (or list) network the probability that a individual could find its way from an almost star (or list) network to this solution should be quite high. It can be assumed if using a GA for optimisation problems which tend to have optimal solutions that are more like a star or list, the pruefernumber representation sometimes can do a good job. The more the optimal solutions tend to tree networks, the worse the locality is and the higher is the probability that a GA will fail when using the pruefernumber encoding.

[2] compare subsection 5.1

[3] Because of better clarity this trivial case is missing in figure 3 and 5.

[4] The reader should not be confused because of the change of more than one bit when investigating the neighbourhood of star networks (figure 4). The change of a n based digit in the pruefernumber results in the change of up to $\lceil ld(n) \rceil$ bits

5.3.2 neighbourhood of the phenotype

If the neighbourhood of a star network with n nodes is investigated (fig. 4 and fig. 6), the change of one edge results in the change of one digit in the pruefernumber[5], but up to $\lceil ld(n-2) \rceil$ bits in the bitstring that represents the pruefernumber. The locality of the phenotype is very high. A small change in the encoded star leads to a small change in the bitstring or pruefernumber.

The locality for tree and list networks shows a different behaviour. The change of one edge results most of the time in a complete different bitstring or pruefernumber. For 16 and 32 nodes networks the change of one edge leads in about 80% to a bitstring that is different in more than 5 bits[6]. For the pruefernumber (fig. 6) the results are even worse. It must be considered, that the change of one digit of the pruefernumber can result in a maximum change of $\lceil ld(n) \rceil$ bits in the bitstring. But nevertheless the locality of the phenotype is quite poor for tree and list networks.

6 Extensions

The performance of Simple GAs in different parts of the solution space has never been tested before. We want to present results in later work.

7 Summary and Conclusions

This paper presented an investigation in the locality of the pruefernumber, which can be used for encoding tree networks.

We illustrated that the locality of the phenotype (the tree graph) and the genotype (the pruefernumber) is not homogeneous. When looking at star networks the pruefernumber encoding shows a perfect locality. A change of one edge leads to the change of one digit of the pruefernumber and vice versa. Also for list networks the locality of the pruefernumber is pretty nice. Small changes in the pruefernumber lead to small changes in the encoded tree. For tree networks the picture is completely different. Here the pruefernumber encoding shows its ugly face. Every small change of the pruefernumber leads to a big change in the encoded tree and vice versa. Unfortunately the list and star networks are on the borders of the solution space. Each star has only one edge in common with another star and only two edges with a list network. So the areas of high locality are separated from each other.

A GA using the pruefernumber encoding can sometimes do a good job when looking for good solutions which are "star" or "list"-like. When it comes to finding other solutions the probability of failure is quite high.

References

Gen, M., Zhou, G., & Takayama, M. (1998). A comparative study of tree encodings on spanning tree problems. In *Proceedings of the Fifth International Conference on Evolutionary Computation* (pp. 33–38). IEEE Service Center.

Julstrom, B. A. (1993). A genetic algorithm for the rectilinear steiner problem. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 474–480). San Mateo, CA: Morgan Kaufmann.

[5] case missing because of better clarity

[6] fig. 4

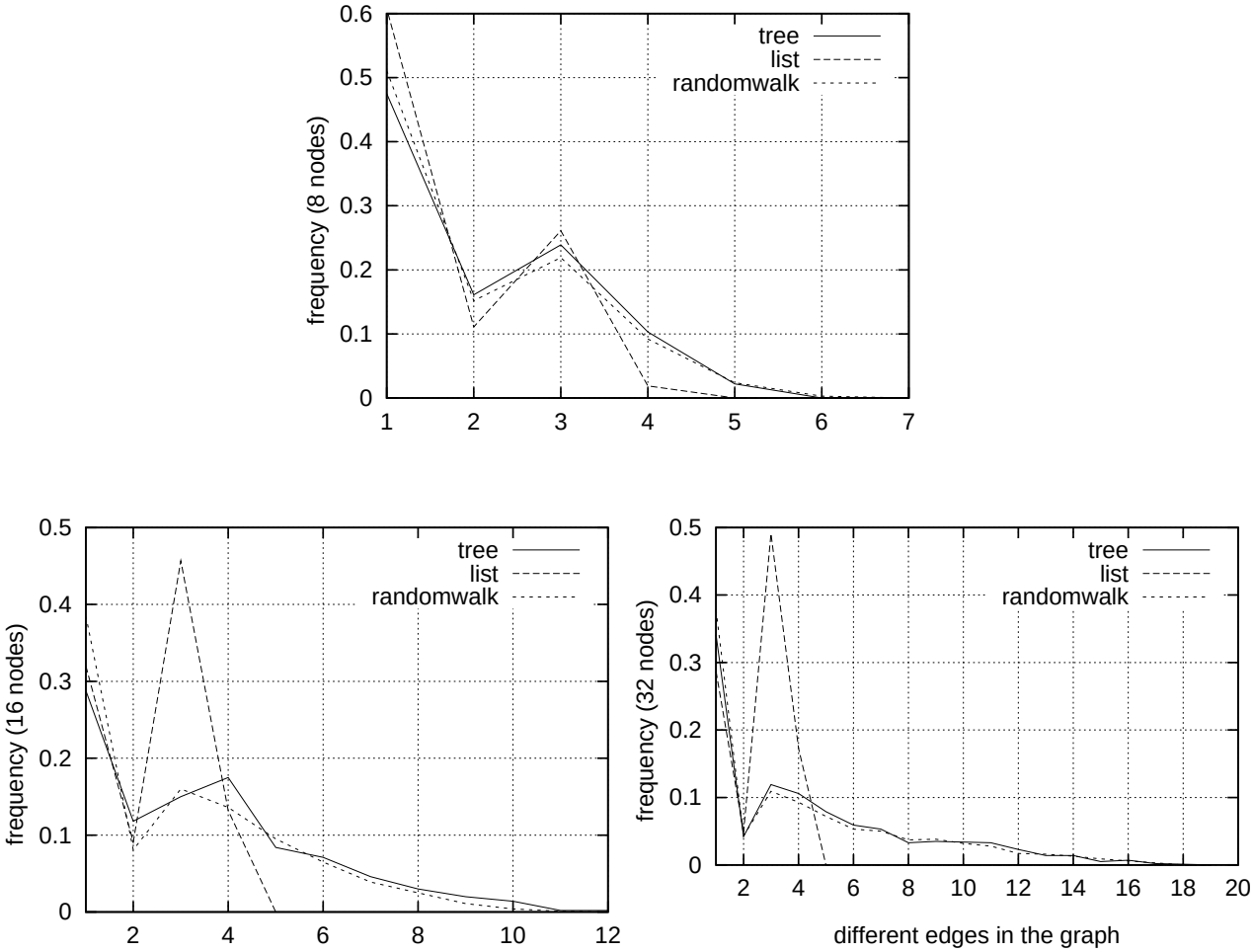

Figure 3: These plots show the locality of the genotype (bitstring) for 8 (top), 16 (middle) and 32 (bottom) node networks. It is illustrated how many edges of a tree are different, if one bit of the bitstring is changed. It can be seen that bitstrings that encode a list network have quite a nice locality, whereas bitstrings that encode a tree show very low locality.

Palmer, C. C., & Kershenbaum, A. (1994). Representing trees in genetic algorithms. In *Proceedings of the First IEEE International Conference on Evolutionary Computation* (pp. 379–384). IEEE Service Center.

Pruefer, H. (1918). Neuer beweis eines satzes ueber permutationen. *Arch. Math. Phys.*, *27*, 742–744.

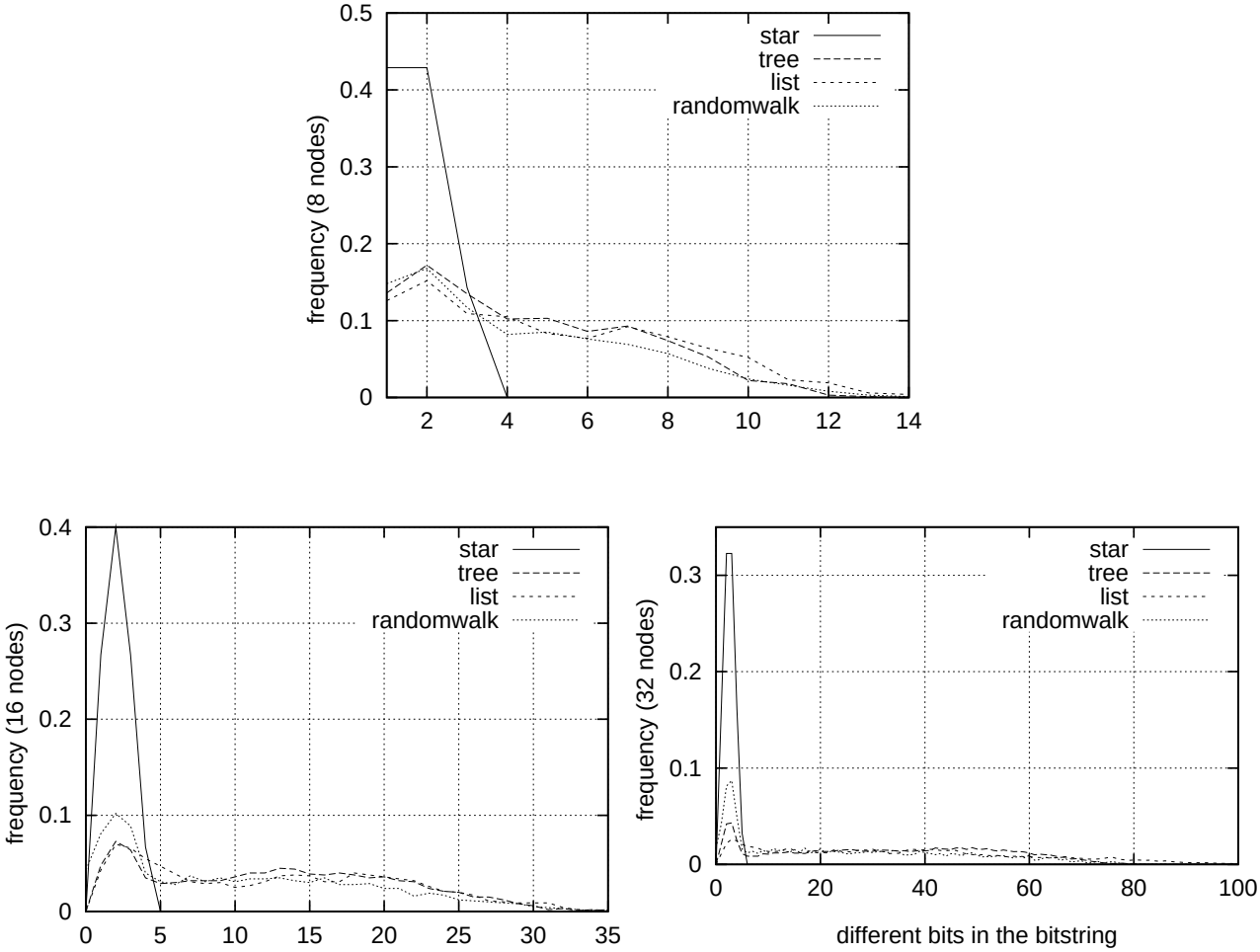

Figure 4: These plots show the locality of the phenotype (tree) for 8 (top), 16 (middle) and 32 (bottom) node networks. The plots show the number of different bits in the bitstring, if one edge of the tree is changed. It can be seen that list and tree networks have only little locality, whereas star networks show perfect neighbourhood.

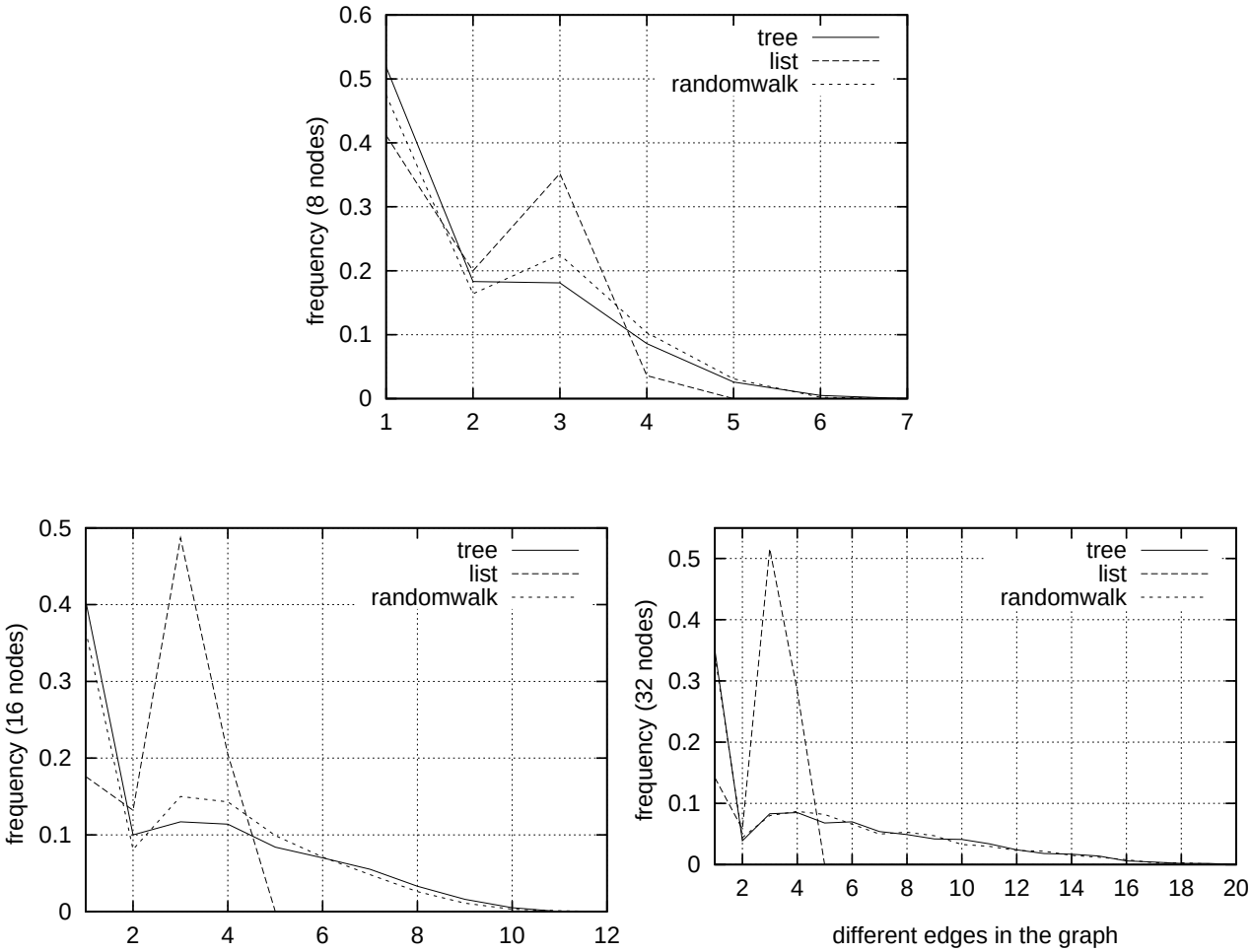

Figure 5: These plots show the locality of the genotype (pruefernumber) for 8 (top), 16 (middle) and 32 (bottom) node networks. It is illustrated how many edges of a tree are different, if one digit of the pruefernumber is changed. It can be seen that pruefernumbers that encode a list network have quite a nice locality, whereas numbers that encode a tree show very low locality.

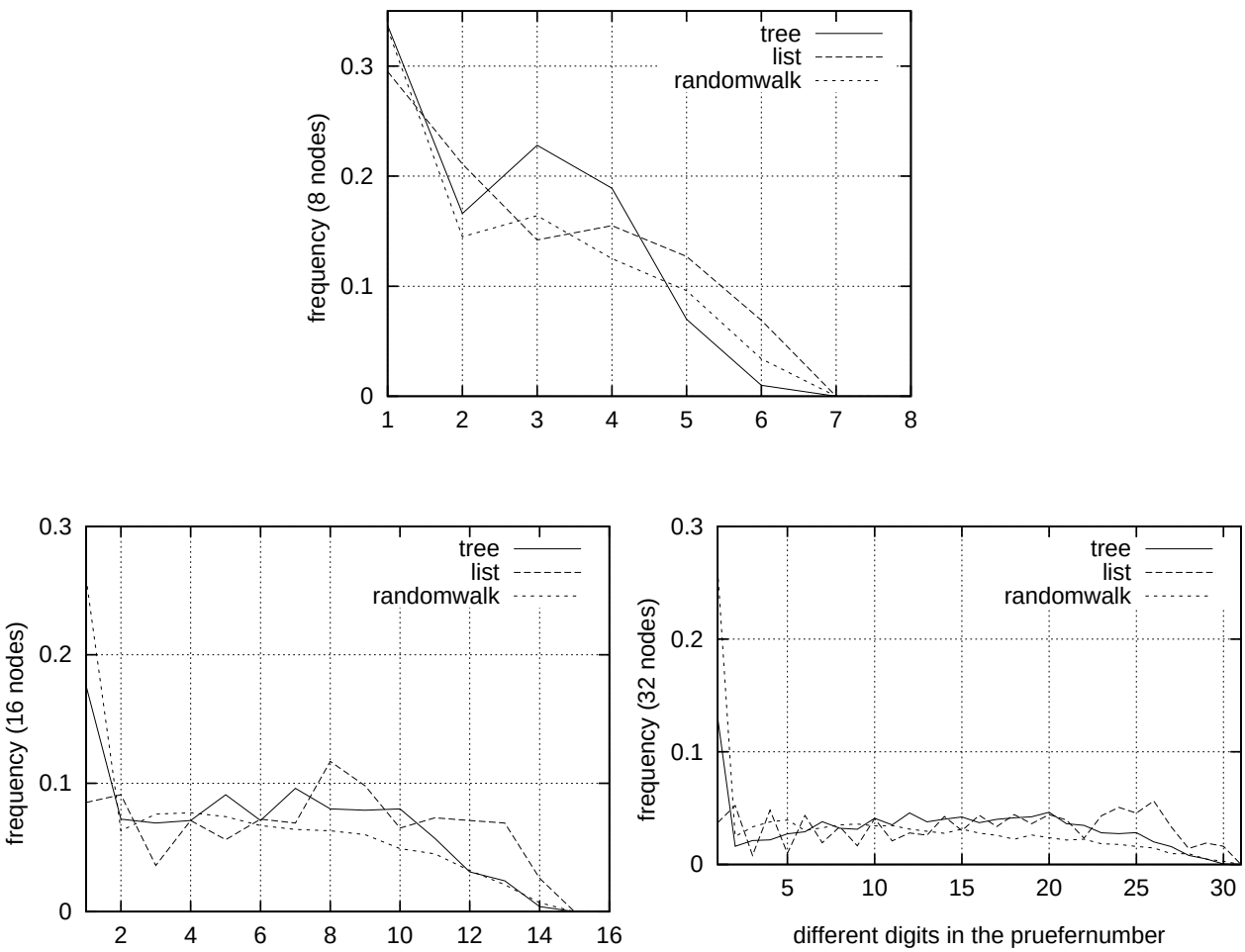

Figure 6: These plots show the locality of the phenotype (tree) for 8 (top), 16 (middle) and 32 (bottom) node networks. The plots show the number of different digits in the pruefernumber, if one edge of the tree is changed. It can be seen that list and tree networks have only little locality.