On the Debate Concerning Evolutionary Search using Prüfer Numbers

Franz Rothlauf[*]
Dept. of Information Systems
University of Bayreuth
Universitaetsstr. 30
D-95440 Bayreuth/Germany
rothlauf@uni-bayreuth.de

David E. 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

Armin Heinzl
Dept. of Information Systems
University of Bayreuth
Universitaetsstr. 30
D-95440 Bayreuth/Germany
heinzl@uni-bayreuth.de

Abstract

This paper sheds some light on the debate concerning evolutionary search using Prüfer numbers, and explains some of the controversial results. Previous work has shown that Prüfer numbers have low locality. Furthermore, it has been shown elsewhere that the locality of the Prüfer number depends on the structure of the encoded tree. The locality of a Prüfer number is high if it encodes a star network, and low elsewhere. The paper illustrates that when applying mutation- as well as recombination-based genetic search to the one-max tree problem, which allows to choose the structure of the optimal solution a priori, that both types of evolutionary search fail if the optimal solution is not star-like.

Therefore, the performance of evolutionary search depends on the structure of the optimal solution. If the high quality solutions are star-like, researchers see a good performance, whereas for other types of networks a failure is inescapable. Therefore, researchers are advised not to use Prüfer numbers on problems of unknown complexity because the encoding is not robust and its performance depends on the structure of the optimal solution.

1 Introduction

Prüfer numbers are a widely used representation for trees. However, the performance of genetic and evolutionary algorithms using Prüfer numbers is strongly disputed.

This paper gives an explanation for the controversial statements about the performance of Prüfer numbers and explains why some researchers report high, and others report low, performance. We know from earlier work (Rothlauf & Goldberg, 1999) that the locality of Prüfer numbers is high only when encoding star networks. Therefore, mutation-, as well as recombination-based genetic search using Prüfer numbers, only performs well if the optimal solutions are star-like. For other types of networks a failure is inescapable. As a result, researchers who use Prüfer numbers for problems where the optimal solution is star-like see a well performing GA, whereas others who use Prüfer numbers for finding non-star-like trees report a failure.

The paper starts with a historical review of the controversial debate about the use of the Prüfer number encoding in the context of genetic and evolutionary search. In section 3 we briefly review the results about the low locality of Prüfer numbers from Rothlauf and Goldberg (1999). Finally, we present empirical results for the performance of a mutation-based simulated annealing, and a recombination-based GA, when solving the one-max tree problem. The paper ends with concluding remarks.

2 Historical Review

The following section presents a review about the controversial debate concerning the performance of Prüfer numbers.

Cayley (1889), identified the number of distinct spanning trees on a complete graph with n nodes as n^{n-2} (Even, 1973, pp. 103-104). Later, this theorem was proven very elegantly by Prüfer (1918) using the introduction of a one-to-one correspondence between spanning trees and a string of length $n-2$ over an alphabet of n symbols. This string is denoted as the Prüfer number, and the genotype-phenotype mapping

[*]Also with Illinois Laboratory of Genetic Algorithms, University of Illinois at Urbana-Champaign, USA.

is the Prüfer number encoding. It is possible to derive a unique tree with n nodes from the Prüfer number of length $n-2$ and vice versa (Even, 1973, pp. 104-106).

Later, in the context of genetic and evolutionary algorithms, several researchers used the encoding for the representation of trees. Palmer used the encoding in his thesis at the beginning of the nineties (Palmer, 1994; Palmer & Kershenbaum, 1994a; Palmer & Kershenbaum, 1994b), and compared the performance of the Prüfer numbers with some other representations for the optimal communication spanning tree problem. However, he noted that the Prüfer number encoding has a low locality and is therefore not a good choice for encoding trees. The low performance of the encoding was confirmed by Julstrom (1993) who used Prüfer numbers for the rectilinear steiner problem, and also observed a low GA performance.

About the same time, Abuali et al. (1994) used Prüfer numbers for the optimization of probabilistic minimum spanning trees (PMST) with genetic algorithms. The investigation focused more on the influence of different operators than on the performance of Prüfer numbers. However, at the end of the work, the conclusion was drawn that in contrast to Palmer and Julstrom, Prüfer numbers "lead to a natural GA encoding of the PMST problem" (Abuali et al., 1994, p. 245). Some years later similar results were reported by Zhou and Gen (1997) who successfully used the Prüfer encoding for a degree constraint minimum spanning tree problem. The degree constraint was considered by repairing invalid solutions that violate the degree constraints. Krishnamoorthy et al. (1999) compared two other versions of genetic algorithms that use Prüfer numbers. They reported a few good results for a GA completely removing invalid solutions from the population. Furthermore, Prüfer numbers were used for spanning tree problems (Gen, Zhou, & Takayama, 1998; Gen, Ida, & Kim, 1998), the time-dependent minimum spanning tree problem (Gargano, Edelson, & Koval, 1998), the fixed-charge transportation problem (Li, Gen, & Ida, 1998) and a bicriteria version of it (Gen & Li, 1999), and a multi-objective network design problem (Kim & Gen, 1999). Most of this work reported good results when using Prüfer numbers, and label the encoding to be (very) suitable for encoding spanning trees. As an example of positive results we want to cite Kim and Gen (1999) who wrote:

> The Prüfer number is very suitable for encoding a spanning tree, especially in some research fields, such as transportation problems, minimum spanning problems, and so on.[1]

However, other relevant work by Krishnamoorthy et al. (1999), who used Prüfer numbers for the degree constraint spanning tree problem, from Julstrom (2000) who compared a list of edges encoding with Prüfer numbers, or from Gottlieb and Eckert (2000) who used Prüfer numbers for the fixed charge transportation problem showed that Prüfer numbers result in a low GA performance. A summarizing study by Gottlieb, Julstrom, Raidl, and Rothlauf (2001) compared the performance of Prüfer numbers for four different network problems and concluded that Prüfer numbers often perform worse than other encodings, and are not suitable for encoding trees.

To explain the differences between the good and bad results obtained by GAs using Prüfer numbers, Rothlauf and Goldberg (1999) investigated the locality of the encoding more closely. It was shown that Prüfer numbers have high locality if they encode star networks. For all other types of trees, the locality is low which leads to a degradation of a GA (see also Rothlauf and Goldberg (2000)). Therefore, the differences in performance could be well explained when assuming that the performance of the GA depends on the structure of the optimal solution. Researchers who report good solutions often used very small problems, or problems where the optimal solution is more star-like and therefore easy to find for GAs. However, when using Prüfer numbers for more general problems, a strong decrease in GA performance is inescapable. These results were confirmed by Gottlieb and Raidl (2000) who investigated the effects of locality on the dynamics of evolutionary search.

We have seen that the performance of genetic algorithms using Prüfer numbers is a strongly discussed topic. Some researchers report good results and favor the use of Prüfer numbers. Other researchers, however, point to the low locality of the encoding, report worse results and advise us not to use Prüfer numbers. A closer investigation into how locality depends on the structure of the tree could solve these contradictory results. As the work from Rothlauf and Goldberg (2000) indicates that the locality of Prüfer numbers strongly depends on the structure of the tree, we expect GAs to show good results if the good solutions are star-like, and worse results for all other types.

3 The Low Locality of the Prüfer Number Encoding

In this section we briefly review the results concerning the locality of Prüfer numbers as illustrated in Roth-

[1] Special thanks to Bryant A. Julstrom for his help with finding this statement.

lauf and Goldberg (1999) and Rothlauf and Goldberg (2000).

This work analyzed the locality of Prüfer numbers by performing either random walks through the Prüfer number space, or examining the complete neighborhood of an individual. The random walks showed that the locality of the Prüfer number encoding is very low. Most of the small changes of the genotype result in a completely different phenotype.

The analysis of the neighborhood of Prüfer numbers answered the question of whether the locality of the encoding is low everywhere in the search space. The investigation revealed that the locality of Prüfer numbers that represent star networks is perfect, whereas Prüfer numbers representing other network structures have low locality. Genotypic neighbors of a Prüfer number representing a list or an arbitrary tree have on average not much in common with each other.

To answer the questions of why exactly Prüfer numbers encoding stars have high locality, and how large the areas of high locality are, an investigation into the number of neighbors a Prüfer number individual was performed. The analysis showed that for Prüfer numbers the number of neighbors is independent from the represented tree. For phenotypes, however, the number of neighbors varies with the structure of the tree. Star networks have as many neighbors as the corresponding Prüfer numbers, and therefore, the locality around star networks is high. When modifying stars towards lists, the number of phenotypic neighbors increases which makes it impossible to obtain high locality for problems other than star networks. Furthermore, Gottlieb, Julstrom, Raidl, and Rothlauf (2001) showed that the areas of high locality are only of order $O(n^{const})$, whereas the whole search space grows with $O(n^{n-2})$. Thus, the regions of high locality become very small with increasing problem size which reduces the performance of a GA on larger problems.

Previous work has shown the problems of Prüfer numbers with low locality. In the following we illustrate the impact of locality on the performance of mutation- as well as recombination-based evolutionary search.

4 Performance of the Prüfer Number Encoding

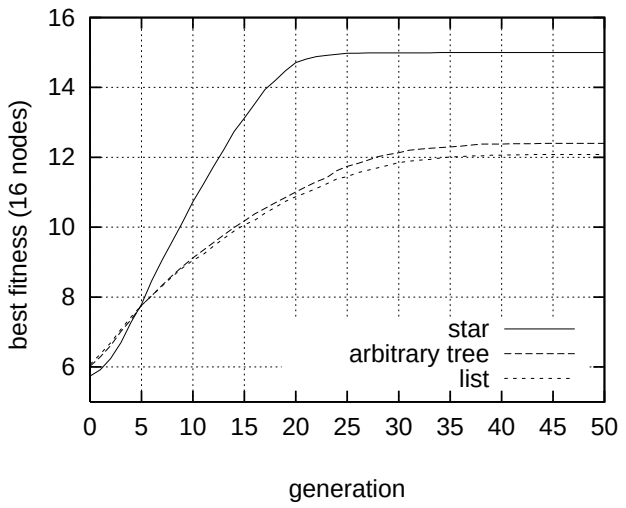
In the following section we verify empirically that evolutionary search algorithms using the Prüfer number encoding fail when searching for good solutions in areas where the locality is low. We present results for a GA using only one-point crossover, and for simulated annealing only using mutation. Both algorithms are applied to the fully easy one-max tree problem.

In the one-max tree problem (Rothlauf, Goldberg, & Heinzl, 2000), an optimum spanning tree is specified a priori, and the fitness of any tree is the number of edges that it shares with this target. Therefore, we can easily investigate how the performance of evolutionary search depends on the structure of the target. Similarly to the one-max problem, the problem should be easy for mutation-based, and slightly more difficult for recombination-based, search.

Simulated annealing (SA) can be modeled as a GA with population size 1 and Boltzmann selection (Goldberg, 1990; Mahfoud & Goldberg, 1995). In each generation a child is created by applying one mutation step to the parent. Therefore, the new individual has distance 1 to its parent. If the child has higher fitness than its parent it replaces the parent. If it has lower fitness it replaces the parent with probability $P(T) = e^{-\frac{f_{child} - f_{parent}}{T}}$, where f denotes the fitness of an individual. The acceptance probability P depends on the actual temperature T which is reduced during the run according to a cooling schedule. With lowering temperature T, the probability of accepting worse solutions decreases. Because the search algorithm uses only mutation, and can in contrast to, for example a $(1 + 1)$ evolution strategy, solve difficult multi-modal problems more easily, we use it as a representative of mutation based evolutionary search algorithms. For further information about simulated annealing the reader is referred to other work (van Laarhoven & Aarts, 1988; Davis, 1987).

In figure 1 we present results for a GA with $\mu+\lambda$ selection using one-point crossover and no mutation on 16 and 32 node one-max tree problems. The structure of the optimal solution is determined to be either a star, random list, ordered list, or an arbitrary tree. For the 16 node problems we chose $\mu = \lambda = 400$, and for the 32 node problems $\mu = \lambda = 1500$. We performed 250 runs and each run was stopped after the population was fully converged. Figure 2 presents results for using simulated annealing. The start temperature $T_{start} = 100$ is reduced in every step by the factor 0.99. Therefore, $T_{t+1} = 0.99 * T_t$. Mutation is defined to randomly change one digit of the Prüfer number. We performed 250 runs and each run was stopped after 5000 mutation steps.

The results in figure 1 and 2 show that if the optimal solution is a randomly chosen star network, both search algorithms, the recombination-based GA and the mutation-based SA are able to find the optimal star easily. A search near star networks is really a



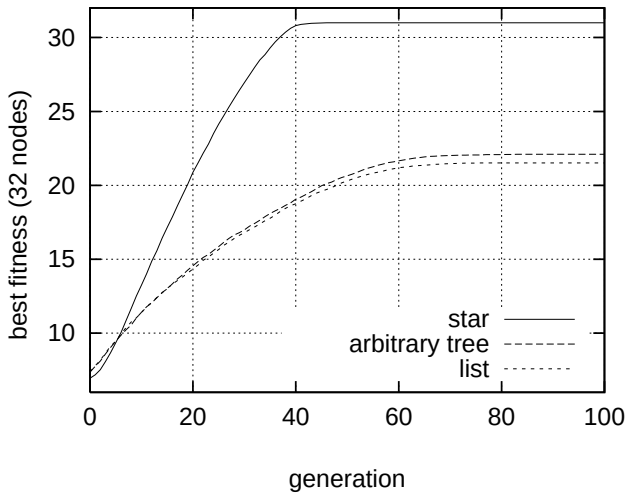
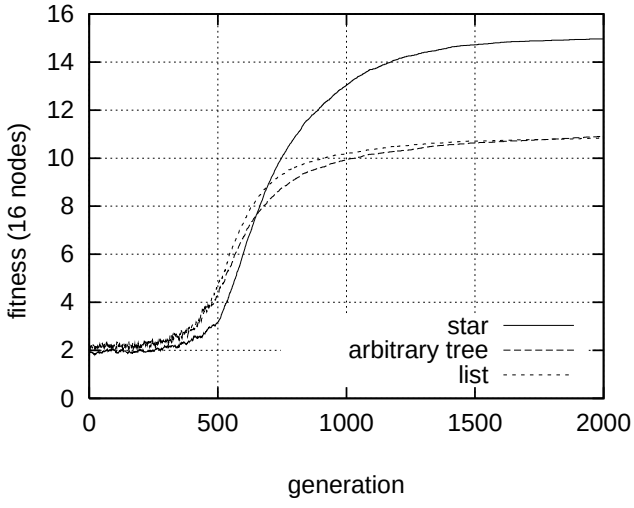


Figure 1: The performance of a genetic algorithm for a 16 (top) and 32 (bottom) node one-max tree problem. The plots show the fitness of the best individual over the run. The structure of the best solutions has a large influence on the performance of the GA. If the best solution is a star, the GA performs well. If the GA has to find a best solution that is a list or a tree, it degrades and cannot solve the easy one-max problem.



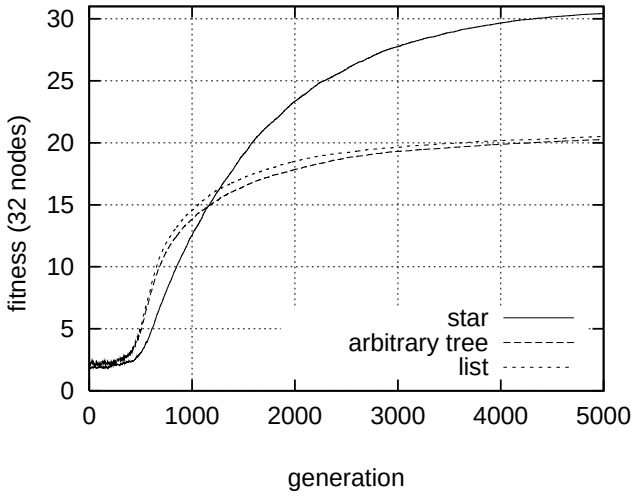


Figure 2: The performance of simulated annealing for a 16 (top) and 32 (bottom) node one-max tree problem. The plots show the fitness over the run. As for recombination-based approaches, the mutation-based simulated annealing fails if the best solution is not a star.

guided search and both algorithms are able to find their way to the optimum. However, if the optimal solution is a random list, an ordered list, or an arbitrary tree, the GA can never find the optimal solution and is completely misled. Exploring the neighborhood around an individual in an area of low locality results in a blind and random search. Individuals that are created by mutation of one individual, or by the recombination of two individuals, have nothing in common with their parent(s).

The results show that good solutions can not be found if they lie in areas of low locality. A degradation of the evolutionary search process is unavoidable. Evolutionary search using the Prüfer number encoding could only work properly if the good solutions are stars. Near stars the locality is high and a guided search is possible. Furthermore, the empirical results illustrate nicely that high locality is a necessary condition for mutation and recombination based evolutionary search. If the locality of an encoding is low, even very simple problems like the one-max tree problem become very difficult and can not be solved any more.

The presented empirical results shine some light on the controversial statements about the performance of Prüfer numbers in the context of evolutionary search. Researchers who investigate problems in which good solutions are star-like see acceptable results and favor the use of Prüfer numbers. Other researchers with non-star-like optimal solutions, however, observe a low performance and advise not to use the encoding.

5 Summary and Conclusions

This paper gave an explanation for the controversial statements about the performance of evolutionary search when using Prüfer numbers for encoding trees. We started with a historical review of the controversial statements about Prüfer number performance. This was followed by reviewing earlier work about the low locality of the Prüfer number encoding. Finally, we presented empirical results about the performance of a mutation-based simulated annealing and a recombination based GA optimizing the one-max tree problem.

The historical review showed that there has been a large increase in interest in the Prüfer number encoding over the last two years. However, the suitability of Prüfer numbers for encoding network problems is strong disputed as some researchers report good results, whereas others report failure. Previous work reported that the locality of the Prüfer number encoding depends on the structure of the represented network. The locality of the Prüfer number is high as long as a star-like structure is represented, and is low everywhere else. Interpreting these results we can shed light on the controversial statements about the performance of GAs using Prüfer numbers. Researchers who investigate problems in which good solution are star-like see acceptable results and favor the use of Prüfer numbers. Other researchers with non-star like optimal solutions, however, observe low performance and advise not to use the encoding.

Furthermore, the belief that low locality only hurts mutation, but not recombination-based genetic search, does not hold true. The empirical results show that locality is necessary for mutation- as well as recombination-based evolutionary search.

A robust, widely usable encoding should allow a GA to perform independently of the structure of the represented network. Our investigation showed that Prüfer numbers are not robust. Only if the good solutions are star-like can evolutionary search using Prüfer numbers perform well. Therefore, researchers should be careful when using Prüfer numbers on problems of unknown complexity. In general, genetic and evolutionary algorithms fail.

References

Abuali, F. N., Schoenefeld, D. A., & Wainwright, R. L. (1994). Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees. In Deaton, E., Oppenheim, D., Urban, J., & Berghel, H. (Eds.), *Proceedings of the 1994 ACM Symposium on Applied Computing* (pp. 242–246). ACM Press.

Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., & Porto, W. (Eds.) (1999). *Proceedings of the 1999 iEEE congress on evolutionary computation*. IEEE Press.

Cayley, A. (1889). A theorem on trees. *Quarterly Journal of Mathematics*, *23*, 376–378.

Davis, L. (1987). *Genetic algorithms and simulated annealing*. San Mateo, CA: Morgan Kaufmann.

Even, S. (1973). *Algorithmic combinatorics*. New York: The Macmillan Company.

Gargano, M. L., Edelson, W., & Koval, O. (1998). A genetic algorithm with feasible search space for minimal spanning trees with time-dependent edge costs. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Iba, H., & Riolo, R. L. (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference* (pp. 495). Morgan Kaufmann.

Gen, M., Ida, K., & Kim, J. (1998). A spanning tree-based genetic algorithm for bicriteria topological network design. See Institute of Electrical and Electronics Engineers (1998), pp. 15–20.

Gen, M., & Li, Y. (1999). Spanning tree-based genetic algorithms for the bicriteria fixed charge transportation problem. See Angeline, Michalewicz, Schoenauer, Yao, Zalzala, and Porto (1999), pp. 2265–2271.

Gen, M., Zhou, G., & Takayama, M. (1998). A comparative study of tree encodings on spanning tree problems. See Institute of Electrical and Electronics Engineers (1998), pp. 33–38.

Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, *4*(4), 445–460. (Also TCGA Report No. 90003).

Gottlieb, J., & Eckert, C. (2000). A comparison of two representations for the fixed charge transportation problem. See Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel (2000), pp. 345–354.

Gottlieb, J., Julstrom, B. A., Raidl, G. R., & Rothlauf, F. (2001). *Prüfer numbers: A poor representation of spanning trees for evolutionary search* (IlliGAL Report No. 2001001). Urbana: University of Illinois at Urbana-Champaign. to appear in Genetic and Evolutionary Computation Conference (GECCO2001).

Gottlieb, J., & Raidl, G. R. (2000). The effects of locality on the dynamics of decoder-based evolutionary search. In Whitley, D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, L., & Beyer, H.-G. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 2000* (pp. 283–290). San Francisco, CA: Morgan Kaufmann Publishers.

Institute of Electrical and Electronics Engineers (Ed.) (1998). *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*. Piscataway, NJ: 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.

Julstrom, B. A. (2000). Comparing lists of edges with two other genetic codings of rectilinear steiner trees. In *Late Breaking Papers at the 2000 Genetic And Evolutionary Computation Conference* (pp. 155–161). Madison, WI: Omni Press.

Kim, J. R., & Gen, M. (1999). Genetic algorithm for solving bicriteria network topology design problem. See Angeline, Michalewicz, Schoenauer, Yao, Zalzala, and Porto (1999), pp. 2272–2279.

Krishnamoorthy, M., Ernst, A. T., & Sharaiha, Y. M. (1999). *Comparison of algorithms for the degree constrained minimum spanning tree* (Tech. Rep.). Clayton, Australia: CSIRO Mathematical and Information Sciences.

Li, Y., Gen, M., & Ida, K. (1998). Fixed charge transportation problem by spanning tree-based genetic algorithm. *Beijing Mathematics*, *4*(2), 239–249.

Mahfoud, S. W., & Goldberg, D. E. (1995). Parallel recombinative simulated annealing: A genetic algorithm. In *Parallel Computing*, Volume 21 (pp. 1–28). Amsterdam, The Netherlands: Elsevier Science.

Palmer, C. C. (1994). *An approach to a problem in network design using genetic algorithms.* unpublished PhD thesis, Polytechnic University, Troy, NY.

Palmer, C. C., & Kershenbaum, A. (1994a). Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, Volume 1 (pp. 379–384). Piscataway, NJ: IEEE Service Center.

Palmer, C. C., & Kershenbaum, A. (1994b). *Two algorithms for finding optimal communication spanning trees.* IBM research report RC-19394.

Prüfer, H. (1918). Neuer Beweis eines Satzes ueber Permutationen. *Archiv für Mathematik und Physik*, *27*, 742–744.

Rothlauf, F., & Goldberg, D. E. (1999). Tree network design with genetic algorithms - an investigation in the locality of the pruefernumber encoding. In Wu, A. S. (Ed.), *Late Breaking Papers at the Genetic and Evolutionary Computation Conference 1999* (pp. 238–244). Madison, WI: Omni Press.

Rothlauf, F., & Goldberg, D. E. (2000). Pruefernumbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. See Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel (2000), pp. 395–404.

Rothlauf, F., Goldberg, D. E., & Heinzl, A. (2000). *Network random keys – a tree network representation scheme for genetic and evolutionary algorithms* (Technical Report No. 8/2000). University of Bayreuth, Germany.

Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., & Schwefel, H.-P. (Eds.) (2000). *Parallel Problem Solving from Nature, PPSN VI.* Berlin: Springer-Verlag.

van Laarhoven, P. J. M., & Aarts, E. H. L. (1988). *Simulated Annealing: Theory and Applications.* Dordrecht, The Netherlands: Kluwer.

Zhou, G., & Gen, M. (1997). Approach to degree-constrained minimum spanning tree problem using genetic algorithm. *Engineering Design & Automation*, *3*(2), 157–165.