Evolution Strategies, Network Random Keys, and the One-Max Tree Problem

Barbara Schindler[1], Franz Rothlauf[1], and Hans-Josef Pesch[2]

[1] Department of Information Systems, University of Bayreuth/Germany
[2] Department of Applied Mathematics, University of Bayreuth/Germany
barbara.schindler@stud.uni-bayreuth.de,
rothlauf@uni-bayreuth.de,hans-josef.pesch@uni-bayreuth.de

Abstract. Evolution strategies (ES) are efficient optimization methods for continuous problems. However, many combinatorial optimization methods can not be represented by using continuous representations. The development of the network random key representation which represents trees by using real numbers allows one to use ES for combinatorial tree problems.
In this paper we apply ES to tree problems using the network random key representation. We examine whether existing recommendations regarding optimal parameter settings for ES, which were developed for the easy sphere and corridor model, are also valid for the easy one-max tree problem.
The results show that the $\frac{1}{5}$-success rule for the $(1+1)$-ES results in low performance because the standard deviation is continuously reduced and we get early convergence. However, for the $(\mu+\lambda)$-ES and the (μ,λ)-ES the recommendations from the literature are confirmed for the parameters of mutation τ_1 and τ_2 and the ratio μ/λ. This paper illustrates how existing theory about ES is helpful in finding good parameter settings for new problems like the one-max tree problem.

1 Introduction

Evolution strategies [1–3] are a class of direct, probabilistic search and optimization methods gleaned from the model of organic evolution. In contrast to genetic algorithms (GAs) [4], which work on binary strings and process schemata, ES have been dedicated to continuous optimization problems. The main operator of ES is mutation, whereas recombination is only important for the self-adaption of the strategy parameters. Random network keys (NetKeys) have been proposed by [5] as a way to represent trees with continuous variables. This work was based on [6] and allows to represent a permutation by a sequence of continuous variables.

In this work we investigate the performance of ES for tree problems when using the continuous NetKey representation. Because ES have been designed for solving continuous problems and have shown good performance therein, we expect ESs to perform well for network problems when using NetKeys. Furthermore, we want to examine whether the recommendations for the setting of ES parameters, that are derived for the sphere and the corridor model, are also valid

for the easy 8 node one-max tree problem. In analogy to the sphere and corridor models, the one-max tree problem [5, 7] is also easy and ES are expected to perform well. Finally, we compare the performance of ES and GAs for the one-max tree problem. We wanted to know which of the two search approaches, mutation versus crossover, performs better for this specific test problem.

The paper is structured as follows. In section 2 we present the NetKey encoding and present its major characteristics. This is followed by a short review of the one-max tree problem. In section 4, after taking a closer look at the different types of ES (subsection 4.1), we perform an analysis of the adjustment of ES parameters for the one-max problem (subsection 4.2), and finally compare the performance of ESs to GAs (subsection 4.3). The paper ends with concluding remarks.

2 Network Random Keys

This section gives a short overview about the NetKey encoding.

Network random keys are adapted random keys (RKs) for the representation of trees. RKs allow us to represent permutations and were first presented in [6]. Like the LNB encoding [8] NetKeys belong to the class of weighted representations. Other tree representations are Prüfer numbers [9], direct encodings [10], or the determinant encoding [11].

When using NetKeys, a key sequence of l random numbers $r_i \in [0, 1]$, where $i \in \{0, \ldots l-1\}$, represents a permutation r^s of length l. From the permutation r^s of length $l = n(n-1)/2$ a tree with n nodes and $n-1$ links is constructed using the following algorithm:

(1) Let $i = 0$, G be an empty graph with n nodes, and r^s the permutation of length $l = n(n-1)/2$ that can be constructed from the key sequence r. All possible links of G are numbered from 1 to l.
(2) Let j be the number at the ith position of the permutation r^s.
(3) If the insertion of the link with number j in G would not create a cycle, then insert the link with number j in G.
(4) Stop, if there are $n-1$ links in G.
(5) Increment i and continue with step 2.

With this calculation rule, a unique, valid tree can be constructed from every possible key sequence. We give some properties of the encoding:

- Standard crossover and mutation operators work properly and the encoding has high locality and heritability.
- NetKeys allow a distinction between important and unimportant links.
- There is no over- or underspecification of a tree possible.
- The decoding process goes with $O(l \log(l))$, where $l = n(n-1)/2$.

Examining NetKeys reveals that the mutation of one key results either in the same tree, or in a tree with no more than two different links. Therefore, NetKeys have high locality. Furthermore, standard recombination operators, like x-point

or uniform crossover, create offspring that inherit the properties of their parents that means they have the same links like their parents. If a link exists in a parent, the value of the corresponding key is high in comparison to the other keys. After recombination, the corresponding key in the offspring has the same, high value and is therefore also used with high probability for the construction of the tree.

A benefit of the NetKey encoding is that genetic and evolutionary algorithms (GEAs) are able to distinguish between important and unimportant links. The algorithm which constructs a tree from the key sequence uses high-quality links with high key values and ensures that they are not lost during the GEA run.

NetKeys always encode valid trees. No over- or underspecification of a tree is possible because the construction rule ensures that only valid solutions are decoded. Thus, no additional repair mechanism are needed.

The key sequence that is used for representing a tree has length $l = n(n-1)/2$. Constructing a a tree results in sorting the l keys that goes with $O(l \log(l))$. Therefore, in comparison to other representations like Prüfer numbers the decoding process is more complex and demanding. For a more detailed description of the NetKey encoding the reader is referred to [5].

3 The One-Max Tree Problem

This section gives a short overview of the one-max tree problem. For further information please refer to [5].

For the one-max tree problem an optimal solution (tree) is chosen either randomly or by hand. The structure of this tree can be determined: It can be a star, a list, or an arbitrary tree with n nodes. In this work we only consider the optimal solution to be an arbitrary tree.

For the calculation of the fitness of the individuals, the distance d_{ab} between two trees G_a and G_b is used. It is defined as

$$d_{ab} = \frac{1}{2} \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} |l_{ij}^a - l_{ij}^b|,$$

where l_{ij}^a is 1 if the link from node i to node j exists in tree G_a and 0 if it does not exist in G_a. n denotes the number of nodes. This definition of distance between two trees is based on the Hamming distance [12] and $d_{ab} \in \{0, 1, \ldots, n-2\}$.

When using this distance metric for a minimization problem the fitness of an individual G_i is defined as the distance $d_{i,opt}$ to the optimal solution G_{opt}. Therefore, $f_i = d_{i,opt}$, and $f_i \in \{0, 1, \ldots, n-2\}$. An individual has fitness (cost) of $n-2$ if it has only one link in common with the best solution. If the two individuals do not differ ($G_i = G_{opt}$), the fitness (cost) of G_i is $f_i = 0$. In this work we only want to use a minimization problem. Because this test problem is similar to the standard one-max-problem it is easy to solve for mutation-based GEAs, but somewhat harder for recombination-based GAs [13].

4 Performance of Evolution Strategies and Adjustment of Parameters

In this section, after a short introduction into the functionality of evolution strategies, we present an investigation into the adjustment of ES parameters for the one-max tree problem when using the NetKey encoding. The section ends with a short comparison of ES and GA for this specific problem.

4.1 A Short Introduction into Evolution Strategies

ESs were developed by Rechenberg and Schwefel in the 1960s at the Technical University of Berlin in Germany [14]. First applications were experimental and dealt with hydrodynamical problems like shape optimization of a bended pipe, drag minimization of a joint plate [2] and a structure optimization of a two-phase flashing nozzle [3].

The simple $(1 + 1)$-ES uses n-dimensional real valued vectors and creates one offspring $x' = \{x'_1, \dots x'_n\}$ from one parent $x = \{x_1, \dots x_n\}$ by applying mutation with identical standard deviations σ to each parental allele x_i.

$$x'_i = x_i + \sigma \cdot N_i(0,1) \qquad \forall\, i = 1, \dots, n.$$

$N(0,1)$ denotes a normal distributed one-dimensional random variable with expectation zero and standard deviation one. $N_i(0,1)$ indicates that the random variable is sampled anew for each possible value of the counter i. The resulting individual is evaluated and compared to its parent, and the better one survives to become the parent of the next generation. For the $(1 + 1)$-ES a theoretical convergence model for two specific problems, the sphere model and the corridor model, exists. The $\frac{1}{5}$-success rule reflects the theoretical result that, to achieve fast convergence, on average one out of five mutations should result in higher fitness values [14, p. 123].

To incorporate the principle of a population, [15] introduced the $(\mu + \lambda)$-ES and the (μ, λ)-ES. This notation considers the selection mechanism and the number of parents μ and offspring λ. For the $(\mu + \lambda)$-ES, the μ best individuals survive out of the union of the μ parents and the λ offspring. In the case of the (μ, λ)-ES, only the best μ offspring form the next parent generation. Both population-based ES start with a parent population of μ individuals. Each individual a consists of an n-dimensional vector $x \in \mathbb{R}^n$ and l standard deviations $\sigma \in \mathbb{R}_+$. One individual is described as $a = (x, \sigma)$ [16].

For both, $(\mu + \lambda)$-ES and (μ, λ)-ES, recombination is used for the creation of the offspring. Mostly, discrete recombination is used for the decision variables x'_i and intermediate recombination is used for the standard deviations σ'_i. Discrete recombination means that x'_i is randomly taken from one parent, whereas intermediate recombination creates σ'_i as the arithmetic mean of the parents standard deviations.

However, the main operator in ES is mutation. It is applied to every individual after recombination:

$$\begin{aligned} \sigma'_k &= \sigma_k \cdot \exp(\tau_1 \cdot N(0,1) + \tau_2 \cdot N_k(0,1)) \quad \forall\, k = 1, 2, \dots, l, \\ x'_i &= x_i + \sigma'_i \cdot N_i(0,1) \qquad\qquad\qquad\qquad\quad \forall\, i = 1, 2, \dots, n. \end{aligned}$$

The standard deviations σ_k are mutated using a multiplicative, logarithmic normally distributed process with the factors τ_1 and τ_2. Then, the decision variables x_i are mutated by using the modified σ'_k. This mutation mechanism enables the ES to evolve its own strategy parameters during the search, exploiting an implicit link between appropriate internal model and good fitness values. One of the major advantages of ES is seen in its ability to incorporate the most important parameters of the strategy, e.g standard deviations, into the search process. Therefore, optimization not only takes place on object variables, but also on strategy parameters according to the actual local topology of the object function. This capability is called self-adaption.

4.2 Adjustment of Parameters

Over time, many recommendations for choosing ES parameters have been developed mainly for the simple sphere and the corridor model. We want to investigate if these recommendations also hold true for simple one-max tree problems represented using the real-valued NetKey encoding.

When using the $(1+1)$-ES there are two possibilities for choosing the standard deviation σ. It can be either fixed to some value or adapted according to the $\frac{1}{5}$-success rule. For sphere and corridor models this rule results in fastest convergence. However, sometimes the probability of success cannot exceed $\frac{1}{5}$. For problems, where the objective function has discontinuous first partial derivatives, or at the edge of the allowed search space, the $\frac{1}{5}$-success rule does not work properly. Especially in the latter case, the success rule progressively forces the sequence of iteration points nearer to the boundary and the step lengths are continuously reduced without the optimum being approached with comparable accuracy [17].

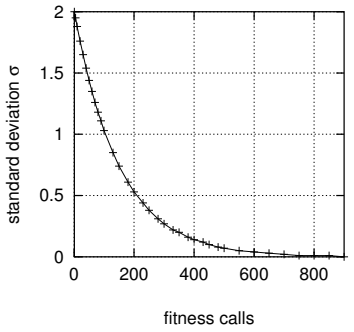


Fig. 1. Standard deviation σ over number of fitness calls. We use the $\frac{1}{5}$-rule and a $(1+1)$-ES for the 8 node one-max tree problem.

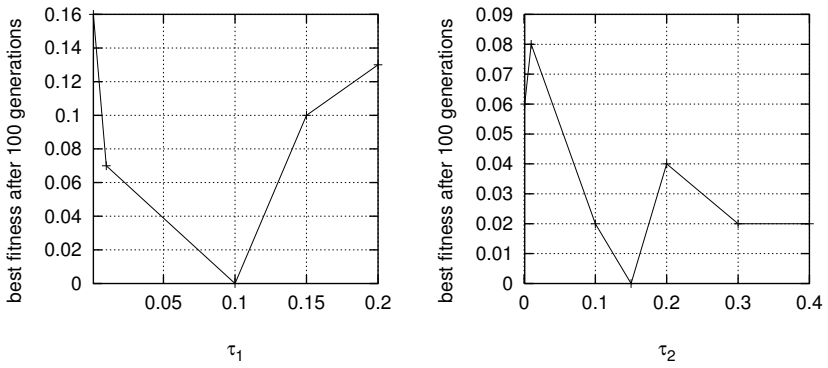
Figure 3(a) and Figure 1 illustrate the problems of the $\frac{1}{5}$-success rule when using ES for solving an 8 node one-max tree problem. The plots show the fitness and the standard deviation σ over the number of fitness calls. The initial standard deviation $\sigma_0 = 2$ and we performed 200 runs. Due to the success rule the standard deviation is continuously reduced and we get early convergence. The same results have been obtained for larger 16 and 32 node problem instances. This behavior of $(1+1)$-ES can be explained when examining the NetKey encoding. In section 2 we saw that only $n-1$ out of $n(n-1)/2$ links are used for constructing the tree. Therefore, a mutation of one allele often does not result in a change of the represented tree. Many mutations do not result in a different phenotype but only change the

genotype. However, for the $\frac{1}{5}$-rule we assume that every mutation results in a different phenotype and about every fifth new phenotype is superior to its parent. Therefore, the one-max tree problem is more difficult than the fully easy sphere and corridor models, and the $\frac{1}{5}$-rule can not be used.

Instead, we can use a fixed standard deviation σ. In Figure 3(b) we show the fitness after 10 000 iterations over the standard deviation σ for the 8 and 16 node one-max tree problem. The results indicate that the $(1+1)$-ES shows the best performance for a fixed standard deviation of $\sigma \approx 0.2-0.4$. Larger standard deviations do not result in a faster convergence but the search becomes random. With smaller standard deviations we also do not get better solutions, because with small standard deviations of mutation we only make slow progress.

To overcome the problem of the $(1+1)$-ES getting stuck in local optima, population-based ES approaches like $(\mu+\lambda)$-ES and (μ,λ)-ES have been proposed. We want to examine how one can adjust the strategy parameters τ_1 and τ_2. The standard deviations are mutated using a multiplicative, logarithmic normally distributed process. The logarithmic normal distribution is motivated as follows. A multiplicative modification process for the standard deviations guarantees positive values for σ and smaller modifications must occur more often than larger ones [18]. Because the factors τ_1 and τ_2 are robust parameters, [18] suggests setting them as follows: $\tau_1 \propto (\sqrt{2\sqrt{n}})^{-1}$ and $\tau_2 \propto (\sqrt{2n})^{-1}$. Newer investigations indicate that optimal adjustments are in the interval $[0.1, 0.2]$ [19, 20]. τ_1 and τ_2 can be interpreted in the sense of "learning rates" as in artificial neural networks, and preliminary experiments with proportionality factors indicate that the search process can be tuned for particular objective functions by modifying these factors. We investigated in Figure 2 for the $(\mu+\lambda)$-ES whether the recommendations of Schwefel or Kursawe are also valid for the 8 node one-max tree problem. The plots show how the best fitness after 100 generations depends on τ_1 and τ_2. The results confirm the recommendations from Kursawe to initialize the parameters in the interval $[0.1, 0.2]$, where $\tau_2 > \tau_1$. The best solutions are $\tau_1 = 0.1$ and $\tau_2 = 0.15$ for the $(\mu+\lambda)$-ES and $\tau_1 = 0.15$ and $\tau_2 = 0.2$ for the (μ,λ)-ES.

The next part of our investigation focuses on the optimal proportion of μ parents to λ offspring to maximize the convergence velocity. [17] proposed a $(1,5)$-ES or a $(1,6)$-ES that is nearly optimal for sphere and corridor models. Figure 3(c) ($(\mu+\lambda)$-ES) and Figure 3(d) ((μ,λ)-ES) show the fitness over the number of fitness calls for the 8 node one-max tree problem. We used a population size of $N = \mu + \lambda = 200$, $\tau_1 = 0.13$, $\tau_2 = 0.16$, $\sigma_0 = 0.5$, and performed 1000 runs for every parameter setting. The results show that a ratio of $\frac{\mu}{\lambda} \in \{\frac{1}{4} \dots \frac{1}{7}\}$ results in good performance for the $(\mu+\lambda)$-ES and the (μ,λ)-ES. These results confirm the recommendations from [21] and [16]. The investigations for the sphere model indicated that the ratio of $\frac{\mu}{\lambda} \approx \frac{1}{7}$ is optimal concerning the accelerating effect of self-adaption. This ratio also provides the basic parameterization instrument for controlling the character of the search. Decreasing μ/λ emphasizes on path-oriented search and convergence velocity, while increasing μ/λ leads to a more volume-oriented search.



(a) τ_2 fixed at 0.15 (b) τ_1 fixed at 0.1

Fig. 2. Best fitness at end of the run over the strategy parameters τ_1 and τ_2 for the $(\mu + \lambda)$-ES. In Figure 2(a) we fixed τ_2 at 0.15 and varied τ_1, and in Figure 2(b) we fixed τ_1 at 0.1 and varied τ_2. We used $N = 200$, $\sigma_0 = 0.5$, $\mu/\lambda = 0.25$, 100 generations, and 1000 runs per plot.

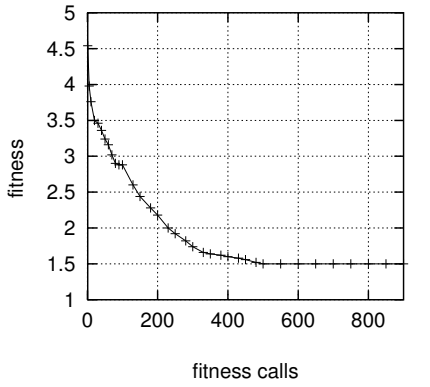
Finally, we want to examine the population size N which provides optimal convergence velocity. The population size mainly depends on the representation of the individuals and the optimization problem. Guidelines for choosing proper population sizes N when using NetKeys for the one-max tree problem and using selectorecombinative GAs were shown in [5]. In Figure 3(e) we compare the fitness over the number of fitness calls for the $(\mu+\lambda)$-ES for different population sizes $N = \mu + \lambda$. We used $\tau_1 = 0.13$, $\tau_2 = 0.16$, $\sigma_0 = 0.5$, $\mu/\lambda = 0.25$ and performed 1000 runs for every parameter setting. The results reveal that for a 8 node problem, a population size $N = 200$ is enough to allow ES to find the optimal solution reliably and fast.

Our investigations indicate that the simple $\frac{1}{5}$-rule for the $(1 + 1)$-ES from [14] does not work when using NetKeys. However, when using $(\mu + \lambda)$-ES or (μ, λ)-ES the recommendations for the simple sphere and corridor models from [18], [19], and [17] can also be used for the one-max tree problem using NetKeys. The existing guidelines help us to choose proper strategy parameters τ_1, τ_2, and the ratio $\frac{\mu}{\lambda}$. For further information about the use of ES for tree problems using the NetKey encoding the reader is referred to [22].

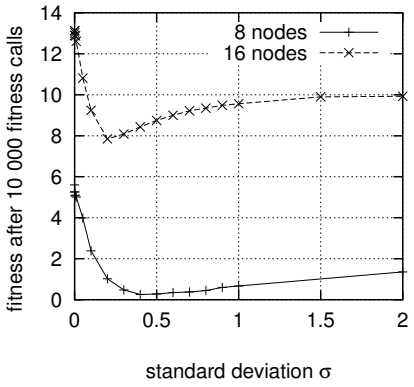
4.3 A Comparison to Genetic Algorithms for the One-Max Tree Problem

After identifying optimal strategy parameters for ES we want to compare the performance of ES with GAs for the one-max tree problem using NetKeys.

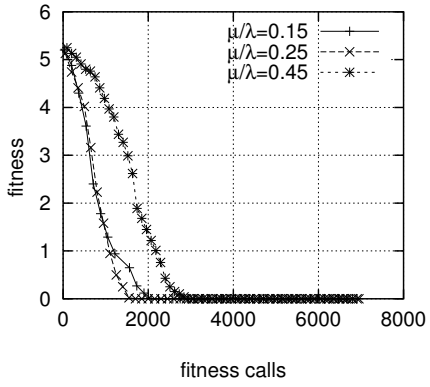
For both optimization methods, ES and GA, we use uniform crossover. For the GA we implemented a roulette-wheel selection scheme and used $N = 200$. Mutation in the context of GA means that the value of one key is randomly



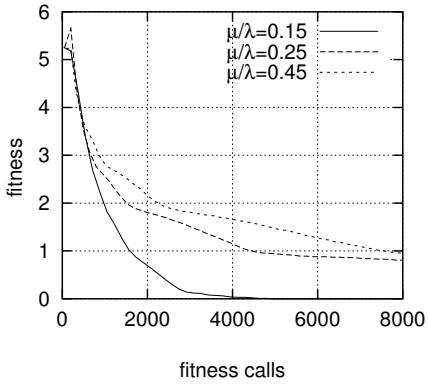
(a) Performance of $(1+1)$-ES using the $\frac{1}{5}$-success rule.



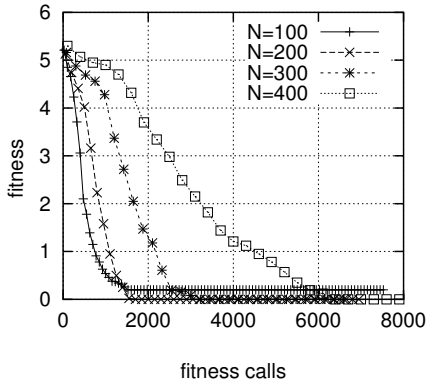
(b) $(1+1)$-ES using fixed σ (fitness after 10 000 fitness calls over σ).



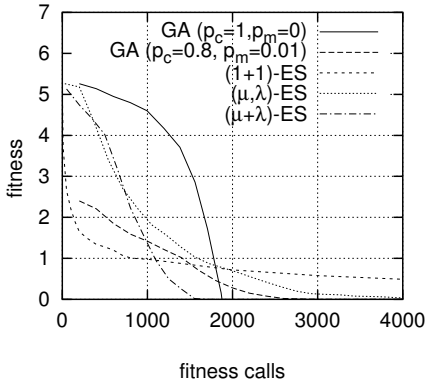
(c) $(\mu+\lambda)$-ES for different μ/λ ratios.



(d) (μ, λ)-ES for different μ/λ ratios.



(e) $(\mu+\lambda)$-ES for different $N = \mu+\sigma$.



(f) Comparison between ES and GA.

Fig. 3. Adjustment of ES parameters for the 8 node one-max tree problem. All plots show the fitness over the number of fitness calls (except Figure 3(a) which shows best fitness after 10 000 fitness calls over σ.)

changed. As before, we used for the ES $\tau_1 = 0.13$, $\tau_2 = 0.16$, $\sigma_0 = 0.5$, $N = \mu + \lambda = 200$, and performed 1000 runs for every parameter setting.

Figure 3(f) compares the performance of ES and GAs for the one-max tree problem. We plot the fitness over the number of fitness calls. The results show that a $(\mu + \lambda)$-ES has the highest performance. The $(1 + 1)$-ES gets stuck and is not able to find the optimal solution.

5 Conclusions

In this paper we extended the use of evolution strategies to combinatorial tree problems. Evolution strategies are designed for continuous optimization problems and can be applied to trees when using the continuous network random key (NetKey) representation. We examined for the small 8 node one-max tree problem how to adjust the parameters of the $(1 + 1)$-, $(\mu + \lambda)$-, and (μ, λ)-ES and compared their performance to a simple GA.

The results showed that the recommendations regarding the adjustment of the ES parameters (τ_1, τ_2, and μ/λ) for simple sphere and corridor models can also be used for the easy one-max tree problem when using the NetKey encoding. Only the $\frac{1}{5}$-success rule for the $(1 + 1)$-ES does not hold true for the one-max tree problem because most of the mutations do not change the represented tree. Therefore, the strategy parameter σ is continously reduced and the algorithm gets stuck.

The results indicate that existing theory about ES can often help in finding good parameter settings for new types of problems. We want to encourage researchers when developing new representations or techniques to first look at existing theory, to check if they can be used advantageously, and not to reinvent the wheel.

References

1. H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Master's thesis, Technische Universität Berlin, 1965.
2. I. Rechenberg. Cybernetic solution path of an experimental problem. Technical Report 1122, Royal Aircraft Establishment, Library Translation, Farnborough, Hants., UK, 1965.
3. H.-P. Schwefel. Experimentelle Optimierung einer Zweiphasendüse. Bericht 35, AEG Forschungsinstitut Berlin, Projekt MHD-Staustahlrohr, 1968.
4. J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, MI, 1975.
5. F. Rothlauf, D. E. Goldberg, and A. Heinzl. Network random keys – a tree network representation scheme for genetic and evolutionary algorithms. Technical Report No. 8/2000, University of Bayreuth, Germany, 2000. to be published in Evolutionary Computation.
6. J. C. Bean. Genetics and random keys for sequencing and optimization. Technical Report 92-43, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, June 1992.

7. Franz Rothlauf. *Towards a Theory of Representations for Genetic and Evolutionary Algorithms: Development of Basic Concepts and their Application to Binary and Tree Representations*. PhD thesis, University of Bayreuth/Germany, 2001.
8. C. C. Palmer. *An approach to a problem in network design using genetic algorithms*. unpublished PhD thesis, Polytechnic University, Troy, NY, 1994.
9. H. Prüfer. Neuer Beweis eines Satzes über Permutationen. *Archiv für Mathematik und Physik*, 27:742–744, 1918.
10. Günther R. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of 2000 IEEE International Conference on Evolutionary Computation*, pages 43–48, Piscataway, NJ, 2000. IEEE.
11. F. Abuali, R. Wainwright, and D. Schoenefeld. Determinant factorization and cycle basis: Encoding schemes for the representation of spanning trees on incomplete graphs. In *Proceedings of the 1995 ACM/SIGAPP Symposium on Applied Comuting*, pages 305–312, Nashville, TN, February 1995. ACM Press.
12. R. Hamming. *Coding and Information Theory*. Prentice-Hall, 1980.
13. D. E. Goldberg, K. Deb, and D. Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993.
14. I. Rechenberg. Bionik, Evolution und Optimierung. *Naturwissenschaftliche Rundschau*, 26(11):465–472, 1973.
15. H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technical University of Berlin, 1975.
16. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
17. H.-P. Schwefel. *Evolution and Optimum Seeking*. Wisley & Sons, New York, 1995.
18. H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhuser, Basel, 1977. from Interdisciplinary Systems Research, volume 26.
19. F. Kursawe. *Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien - Metastrategien*. PhD thesis, University of Dortmund, 1999.
20. V. Nissen. *Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution*. Vieweg, Wiesbaden, 1997.
21. H.-P. Schwefel. Collective phenomena in evolutionary systems. In P. Checkland and I. Kiss, editors, *Problems of Constancy and Change - The Complementarity of Systems Approaches to Complexity*, volume 2, pages 1025–1033, Budapest, 1987. Papers presented at the 31st Annual Meeting of the International Society for General System Research.
22. Barbara Schindler. Einsatz von Evolutionären Stratgien zur Optimierung baumförmiger Kommunikationsnetzwerke. Master's thesis, Universität Bayreuth, Lehrstuhl für Wirtschaftsinformatik, Mai 2001.