Analysis of evolutionary algorithms for the phylogeny problem

Botond Draskoczy
University of Stuttgart
Germany
boti@jime.de

Nicole Weicker
University of Stuttgart
Germany
weicker@informatik.uni-stutttgart.de

Karsten Weicker
HTWK Leipzig
Germany
weicker@imn.htwk-leipzig.de

Abstract- In view of the strong increasing number of phylogeny problems and instances, the assessment of the respective optimization algorithms has become harder too. This paper contributes to this ongoing discussion by a first examination how the search dynamics are affected by the chosen algorithm and certain characteristics of the phylogeny problem. For this purpose a tunable problem generator is proposed and analyzed.

1 Introduction

The existing methods for analyzing the sequences of DNA, RNA, and amino acids produce lots of data to be used in classifying the degree of relationship of organisms. This classification problem is known as phylogeny problem, which is important in bioinformatics not only to reconstruct the descent-lines of living plants or animals but for example also to discover the relation between viruses in order to find effective vaccines quickly. In fact the phylogeny problem, i.e. the search for an phylogenetic tree that reflects the evolution of the given sequences in the leaves of the tree, is NP-complete in general (Day et al., 1986; Bodlaender et al., 1992; Foulds and Graham, 1982a,b).

Many heuristics for the phylogeny problem can be found in the scientific literature (e.g. Dress and Krüger, 1987; Platnick, 1987). Andreatta and Ribeiro (2002) list a bound of construction algorithms, neighborhoods (mutation operators), local search strategies, and meta-heuristics for this problem and compare them concerning time-to-target and quality. Additionally there exist several population based approaches to solve the phylogeny problem (e.g. Ribeiro and Vianna, 2003; Cotta and Moscato, 2002).

Most of the operators found in literature are problem specifically defined but usually used with standard parameter settings. A systematic analysis of phylogeny problem instances is missing which could give more insight into useful parameter calibration. The phylogeny problem properties that have to be considered are: size of the underlying alphabet, similarity of the sequences, number of sequences etc. Without knowledge concerning these factors and an adequate construction of artificial test problems the comparison of operators and heuristics can be misleading. In the extreme case (size of the alphabet $= 2$ and some additional properties) the phylogeny problems are solvable in polynomial time (Gusfield, 1991; Waterman, 1995).

As a first step towards a qualitative analysis of the phylogeny problem and the evolutionary operators on this problem the paper at hand makes the following three contributions.

First, we present our approach to analyze the difficulty of diverse problem instances using a tunable phylogeny problem generator.

Second, it is analyzed how two different operators that include a different amount of problem knowledge contribute to the optimization process in different phases of the search process.

Third, the influence of the number of parents and the number of offspring on the performance is examined.

2 Phylogeny problems

For the formal description of the phylogeny problem we abstract from the concrete types of sequences. Rather we view a sequence as a string $s \in \Sigma^k$ of length k over an arbitrary finite alphabet Σ. Now a phylogenetic tree is introduced which is crucial for the solution of phylogeny problems.

Definition 1 (Phylogenetic tree) *For a given set of sequences $M \subset \Sigma^k$ a phylogenetic tree is defined as graph $G = (V, E, \gamma)$ with $|V| = 2|M| - 2$ and the degree*

$$\forall\, v \in V: \; deg(v) = 3 \;\vee\; deg(v) = 1.$$

The vertices are labeled by $\gamma : V \rightarrow \Sigma^k$ where the $|M|$ leaves are mapped to the sequences in M, i.e.

$$\begin{aligned} \forall\, v \in V: \; deg(v) = 1 \;&\Rightarrow\; \gamma(v) \in M \\ \forall\, v, w \in V \;\wedge\; v \neq w: \; (deg(v) = 1 \;\wedge\; deg(w) = 1) & \\ \;&\Rightarrow\; \gamma(v) \neq \gamma(w) \end{aligned}$$

and the $|M| - 2$ internal vertices to hypothetical sequences.

It is noteworthy to emphasize that the phylogenetic tree has no given root. In fact, the root might be placed at any edge of the tree. In order to evaluate phylogenetic trees, a measure for the similarity of two sequences is necessary. This is based on the Hamming distance between two sequences.

Definition 2 (Hamming distance) *The Hamming distance of two sequences $s, t \in \Sigma^k$ is defined as*

$$d_H(s, t) \;=\; \big|\{i \mid 1 \leq i \leq k \;\wedge\; s_i \neq t_i\}\big|.$$

Definition 3 (Phylogeny problem) *A phylogeny problem is determined by a set of sequences $M \subset \Sigma^k$ for which a phylogenetic tree $G = (V, E, \gamma)$ is searched that has minimal cost, i.e. difference between neighbors in the tree*

$$cost(G) \;=\; \sum_{(v,w) \in E} d_H(\gamma(v), \gamma(w)).$$

Note that the definition above leaves the topology of the tree (including the assignment γ of the sequences) and the choice of the labels at the internal vertices open. However, Fitch (1971) has presented an algorithm for the computation of optimal hypothetical sequences at internal vertices for arbitrary phylogenetic trees. As a consequence the phylogeny problem is usually reduced to the search for the topology and the assignment of the given sequences to the leaves.

2.1 Different phylogeny problems

There are many different variants of the phylogeny problem. Where in the beginning morphological data was of primary interest, today the sequencing of DNA, RNA, and amino acid chains delivers data in a very large scale. However these data sets are only partially suited to gain deeper insight into the processing of particular optimization algorithms since the optimum and the characteristics of an optimal structure are not known and can only assessed by the best known solution.

2.2 Phylogeny problem generator

In order to produce sets of sequences with special characteristics that are similar to real DNA, RNA, or amino acid data we constructed a phylogeny problem generator. The parameters for this generator are the number of sequences n, the finite alphabet Σ, the length of the sequences k, and a mutation probability p. For a general discussion of the use of test problem generators see the repository of Spears and Potter (1999).

The number of sequences determine the size of the phylogenetic tree as defined in definition 1. The generator produces the structure of the phylogenetic tree by random; i.e. there are no sequences assigned and each possible topology has the same probability. Then, a random start vertex (leaf or inner vertex) is chosen and a random sequence $s \in \Sigma^k$ is assigned to this vertex.

In the next step the generator copies the assigned sequence, i.e. s in case of the start vertex, to the neighbored vertices and changes randomly each position in the sequence by the mutation probability p (point mutation). This step is iterated until all vertices of the tree have an assigned sequence.

Then the sequences of the leaves are saved and define an artificial instance of the phylogeny problem. The cost of the generated tree is saved as a reference value for optimizations of the instance.

The described generator has some advantages over the widely used practice to use uniformly created random 0-1 sequences as test problems.

- In the case of a uniform random generation no tree structure (or phylogeny) is contained in the data but all sequences have the same distance to each other on average. With our problem generator there is at least an artificial evolution that needs to be matched which is closer to real world problem instances.
- Furthermore, the problem generator is parameterized, i.e. the problem instances become more difficult with decreasing generating mutation rate. This is shown in Figure 1 where the algorithm described in Section 3 (with $n = 50, k = 50, \mu = 1, \lambda = 99$) is applied to five different problems and the results are averaged over 30 experiments. As it can be seen, the optimum cannot be found anymore for the problem generated with a small mutation rate.

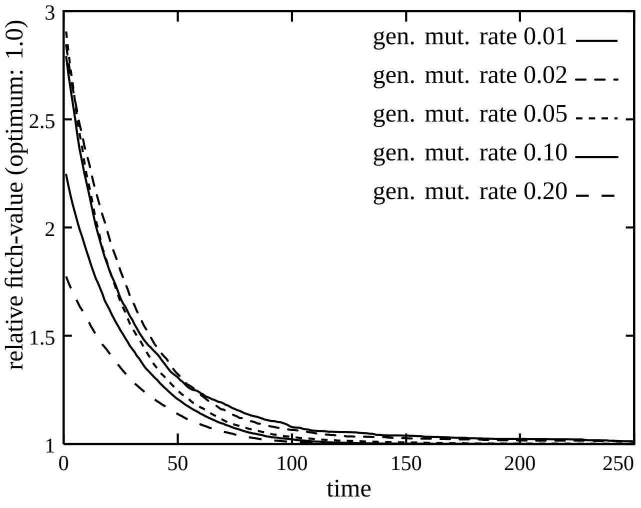


Figure 1: Problem difficulty depending on the generating mutation rate in the problem generator.

One advantage of the described generator is teh strong relation between the generated sequences because of the simulated evolution with a given mutation probability. Another advantage is the known cost value of the production, since this cost value is the aim of optimization.

In most cases this cost value of the production is the global optimum to be reached by the optimization. However, under certain circumstances it is possible that a evolutionary algorithm can find a phylogenetic tree with a better cost value. This is the case if the chosen mutation probability is high which leads to overlapping subtrees in the generated phylogenetic tree.

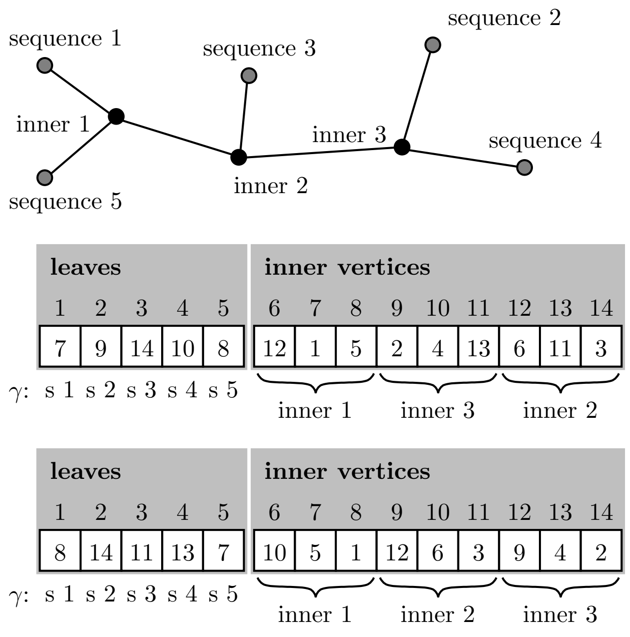
In order to get a benchmark producing problem generator we plan to include known statistics of the mutation probability between the different amino acids, i.e. BLOSUM-matrices (Henikoff and Henikoff, 1992) and PAM-matrices (Dayhoff, 1978). Also a probability to produces gaps should be added together with a rule that makes growing gaps more probable than newly create gaps – it is more probable to get a gap of size five than five gaps of length one. Nevertheless the described generator is practical for the following analysis.

3 Analysis of representation and evolutionary operators

There are at least two distinct possibilities to represent a phylogenetic tree. First, a rooted GP-like tree may be used (e.g. Cotta and Moscato, 2002) which appears to be plausible since the position of the root has no effect on the cost of a tree. Second, the tree is represented as general graph without root which was used in this paper. As we will argue later, the first representation restricts the evolutionary operators.

3.1 Representation

We decided to represent the tree by an array A of length $|M|+3(|M|-2)$ which is twice the number of edges in the tree. The first $|M|$ positions in the array contain the edge information of the leaves. The remaining fields the edge information of the inner vertices where for each vertex three fields are used. We require that for all indices i it holds that $A[A[i]] = i$. That means that an inner vertex is represented by three consecutive values, e.g. the first node by $|M|+1$, $|M|+2$, and $|M|+3$. The mapping γ from the leaves to the sequences is in a fixed order. Figure 2 shows two different individuals that both map to the same tree.



leaves					inner vertices								
1	2	3	4	5	6	7	8	9	10	11	12	13	14
7	9	14	10	8	12	1	5	2	4	13	6	11	3

leaves					inner vertices								
1	2	3	4	5	6	7	8	9	10	11	12	13	14
8	14	11	13	7	10	5	1	12	6	3	9	4	2

Figure 2: Two different encodings of the shown tree.

3.2 Size of the search space

As it can be shown easily by induction the number of phylogenetic trees is

$$T_n \;=\; \prod_{i=3}^{n}(2i-5) \;=\; \frac{(2n-4)!}{2^{n-2}\cdot(n-2)!}$$

for $n=|M|$ sequences and a fixed mapping γ of sequences to the leaves of the tree.

As it can be seen in Figure 2, the representation is highly redundant. Since the order of the inner vertices can be changed as well as the order of edges for each inner vertex, each phylogenetic tree is represented by

$$R_n \;=\; 6^{n-2}\cdot(n-2)!$$

individuals. As a consequence the size of the search space results as

$$|\Omega| \;=\; T_n \cdot R_n \;=\; (2n-4)!\cdot 3^{n-2}$$

in which R_n copies of the global optimum are contained.

In a GP-like representation, redundancy is introduced by the order of the subtrees at each node and the edge where the artificial root, which is not part of the phylogenetic tree, is placed. As a consequence the redundancy for this representation is

$$R'_n \;=\; 2^{n-1}\cdot(n-1).$$

3.3 Used operators

For the variation of phylogenetic trees two different mutation operators are used. First, the exchange of two leaves is described in Algorithm 1. Within our representation the mutation needs $\mathcal{O}(1)$ time. Note that this mutation is not able to change the topology of the tree. For a given tree one application of the mutation may result in $\frac{1}{2}\cdot|M|\cdot(|M|-1)$ different offspring. With a probability of $\frac{1}{|M|}$ the given parental individual is not changed.

Algorithm 1 Mutation: Exchange two leaves

```
1: INPUT: phylogenetic tree $G = (V, E, \gamma)$.
2: $u_1, u_2 \leftarrow$ choose random leaves from $V$
3: $v_1 \leftarrow$ vertex with $\{u_1, v_1\} \in E$
4: $v_2 \leftarrow$ vertex with $\{u_2, v_2\} \in E$
5: $E' \leftarrow (E\backslash\{\{u_1,v_1\},\{u_2,v_2\}\})\cup\{\{u_1,v_2\},\{u_2,v_1\}\}$
6: RETURN: $(V, E', \gamma)$
```

Second, an arbitrary leaf can be deleted in the tree which results in the transformation of the associated internal vertex into an edge. The leaf is reinserted at an arbitrary edge in the tree which is expanded to a new internal vertex. This mutation is described in Algorithm 2 and is also referred to as single step (Andreatta and Ribeiro, 2002). It needs again time $\mathcal{O}(1)$. Contrary to the first mutation this operator may change the topology of the tree. This operator can result in $|M|\cdot(2|M|-6)$ trees. Note that no neutral mutations can occur, however if two adjacent leaves are involved in a mutation, the same offspring can be created on two ways.

Algorithm 2 Mutation: Move one leaf

```
1: INPUT: phylogenetic tree $G = (V, E, \gamma)$.
2: $u \leftarrow$ choose a random leaf from $V$
3: $v \leftarrow$ vertex with $\{u, v\} \in E$
4: $e = \{w_1, w_2\} \leftarrow$ choose an edge from $E \setminus \{\{u, v\}\}$
   such that $\{v, w_1\} \notin E$ and $\{v, w_2\} \notin E$
5: $r, s \leftarrow$ vertices of $V$ with $\{v, r\} \in E$, $\{v, s\} \in E$,
   $r \neq u$, and $s \neq u$
6: $E' \leftarrow (E \setminus \{\{v, r\}, \{v, s\}\}) \cup \{\{r, s\}\}$
7: $E'' \leftarrow (E' \setminus \{\{w_1, w_2\}\}) \cup \{\{v, w_1\}, \{v, w_2\}\}$
8: RETURN: $(V, E'', \gamma)$
```

To recombine two different phylogenetic trees, the well-known prune-delete-graft recombination (pdg) is used (Cotta and Moscato, 2002). Algorithm 3 describes how a subtree is chosen in one phylogenetic tree, all vertices in the other tree that are assigned to the same sequences used at the leaves of the subtree are deleted, and the subtree is reintroduced at an arbitrary edge of the pruned tree. Within our representation the operator needs $\mathcal{O}(n)$ time where n is the size of the subtree and an additional space of $\mathcal{O}(n)$ is needed to memorize the positions of the deleted internal vertices. In our experiments we apply frequently the prune-delete-graft recombination to identical individuals. Then the operator just moves a subtree to a different edge in the tree. Then the mere modification of the tree is in $\mathcal{O}(1)$ time, but $\mathcal{O}(n)$ is needed to ensure that the subtree will be added at an edge not contained in the subtree itself. The different possible offsprings result from the consideration that $4 \cdot |M| - 6$ different subtrees may be selected. The distribution of the sizes of the subtrees depends on the topology of the parental tree. If the subtree contains m leaves it can be added to $2 \cdot |M| - 3 - 2 \cdot m$ edges in the other parent. If the recombination is applied to two identical individuals a neutral recombination is possible.

Remark that in our representation every subtree of the phylogenetic tree can be chosen while in a rooted tree this is not possible. The root restricts the number of heritable subtrees since no real subtree can include the root.

Algorithm 3 Prune-delete-graft (pdg)

```
1: INPUT: phylogenetic trees $G_1 = (V_1, E_1, \gamma_1)$ and $G_2 = (V_2, E_2, \gamma_2)$.
2: $e = \{u, v\} \leftarrow$ choose edge from $E_2$ where $u$ is the root of a subtree $G_2^{sub} = (V_2^{sub}, E_2^{sub}, \gamma_2^{sub})$ and $v$ does not belong to the subtree
3: for all leaves $w$ in the subtree of $u$ in $G_2$ do
4:    $w' \leftarrow$ vertex of $V_1$ with $\gamma_1(w') = \gamma_2(w)$
5:    $w'' \leftarrow$ vertex of $V_1$ with $\{w', w''\} \in E_2$
6:    $r, s \leftarrow$ vertices of $V_1$ with $\{w'', r\} \in E_2$, $\{w'', s\} \in E_2$, $r \neq w'$, and $s \neq w'$
7:    $E_1 \leftarrow (E_1 \setminus \{\{w', w''\}, \{w'', r\}, \{w'', s\}\}) \cup \{\{r, s\}\}$
8:    $V_1 \leftarrow V_1 \setminus \{w'\}$
9: end for
10: $f = \{x, y\} \leftarrow$ choose an edge from $E_1$
11: $E_1 \leftarrow E_1 \setminus \{f\}$
12: $V' \leftarrow V_2^{sub} \dot{\cup} V_1$
13: $E' \leftarrow E_2^{sub} \cup E_1 \cup \{\{x, u\}, \{y, u\}\}$
14: $\gamma' \leftarrow \gamma_1\Big|_{V_1} \cup \gamma_2\Big|_{V_2^{sub}}$
15: RETURN: $(V', E', \gamma')$
```

It is well known that the degree of redundancy has an impact on the evolutionary search dynamics. As we have shown for the knapsack problem the number of local optima decreases with increasing redundancy (Weicker and Weicker, 2001). However this is not true for this algorithm and representation since all operators only make phenotypical modifications.

4 Qualitative comparison of the evolutionary operators

The goal of our experiments is to gain insight into the impact of the operators during different phases of the search process. As it is known (Weicker and Weicker, 1999), the importance of different operators changes over the course of evolution.

4.1 Outline of the experiments

The evolutionary algorithms follow the usual scheme. After an initialization and evaluation of the population the cycle begins with a ranking selection of the parents. The offspring are produced by application of the prune-delete-graft and subsequent mutation (with a certain probability). After a new evaluation the best individuals among the parents and the offspring are chosen to be the new parents. The effective selection in both parental and environmental selection leads to a rather high selective pressure.

To setup an experiment the following parameters have to be determined:

- number of independent runs of the experiment (R)
- number of generations
- type of the sequences (amino acid, DNA): alphabet of size (l)
- number of sequences ($|M| = n$)
- length of each sequence (k)
- number of parents in the population (μ)
- number of offspring (λ)
- kind of mutation (by default move-mutation)
- probability to apply the mutation (p_m)
- type of problem (P – real data or produced by the phylogeny problem generator with a certain probability of point mutation)

For the comparison of all experiments hypothesis tests have been executed. In most cases we only refer to the test results in the text since they are plausible.

All performance graphs show the best values averaged over the executed independent experiments.

4.2 Comparison of the mutation operators

We compared the two mutation operators described in Algorithm 1 and 2. The setup of the two experiments using prune-delete-graft operator (pdg) as recombination on a generated test problem with amino acids as underlying alphabet is summarized in figure 3 which also shows the result. There is no significant difference between the move mutation and the exchange mutation.

This result is a little bit surprising, since the move mutation changes the topology contrary to the exchange mutation. Apparently the pdg operator compensates this effect.

For the following experiments we concentrate our analysis on the move mutation.

4.3 Effectivity of mutation and prune-delete-graft

For three different problems we analyzed the influence of the move mutation and the prune-delete-graft applied as a mutation operator. The latter uses more problem specific knowledge. All experiments in this section are set up with a (1+99) strategy and we are interested in the contribution

R	l	n	k	μ	λ	p_m	P
10	20	50	50	10	90	0.2	gen. 0.05

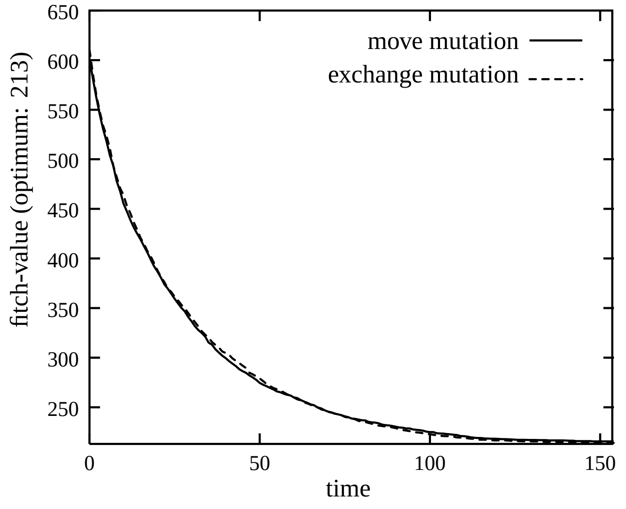


Figure 3: Different mutations together with prune-delete-graft recombination.

R	l	n	k	μ	λ	p_m	P
30	20	50	50	1	99	*	gen. 0.1

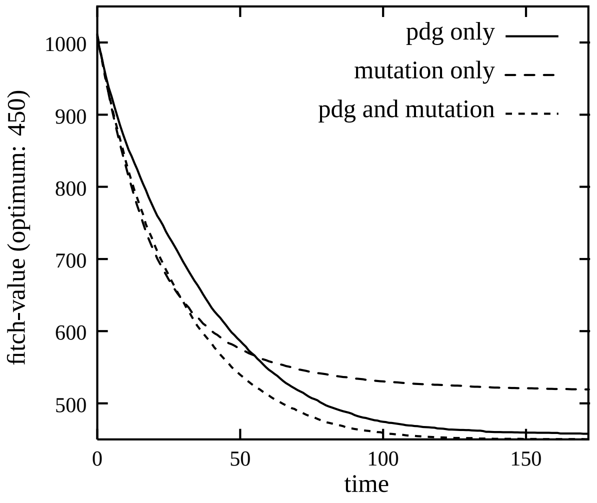


Figure 4: Algorithms with mutation only ($p_m = 1.0$), prune-delete-graft only, and the standard algorithm with both operators ($p_m = 0.2$) are compared on a small generated problem.

R	l	n	k	μ	λ	p_m	P
30	20	100	50	1	99	*	gen. 0.1

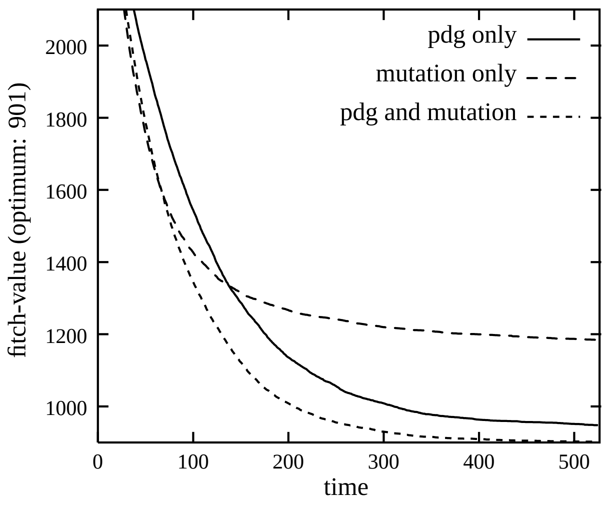


Figure 5: Algorithms with mutation only ($p_m = 1.0$), prune-delete-graft only, and the standard algorithm with both operators ($p_m = 0.2$) are compared on a large generated problem.

R	l	n	k	μ	λ	p_m	P
10	20	33	509	1	99	*	real

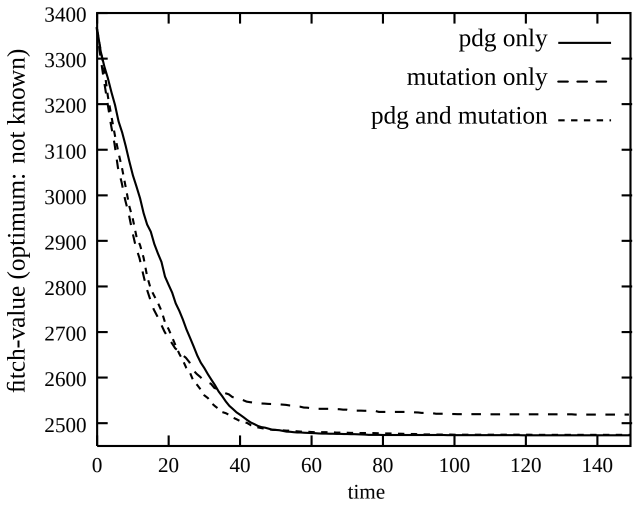


Figure 6: Algorithms with mutation only ($p_m = 1.0$), prune-delete-graft only, and the standard algorithm with both operators ($p_m = 0.2$) are compared on a large generated problem.

each operator yields depending on the stage of the search process. Therefore, the operators are applied both alone in an evolutionary algorithm as well as in combination. The results of the experiments are shown in Figures 4, 5, and 6.

The first two problems have been produced by the proposed phylogeny problem generator with different numbers of sequences. The third problem is a real data problem where the amino acid chains code a subset of the cytochrome-P-450-enzyme (van der Weide and Steijns, 1999).

The results of all these experiments are similar. In the first few generations (35 for the first problem, 75 for the second problem) the move mutation only is not significantly worse than the combination of move mutation and pdg. For the second the move mutation only is even significantly better than the combination. But although the experiments with pdg only are significant worse than the combination of both operators, the pdg only reaches the optimum in half of the experiments. The experiments with move mutation only get stuck in local optima on a rather worse fitch-value and the algorithm is not able to get out of it in even 1000 generations for the first problem.

The move mutation is needed to sort the leaves of the phylogenetic tree in order to produce good subtrees. The pdg mutation needs more time to optimize if it has to produce good subtrees itself. But if the phylogenetic tree is already quite good the pdg mutation is able to find the optimum in even half of the experiments. Although the move mutation is not able to bring a significant progress by its own, it helps the pdg mutation to find the optimum reliably.

From this results we can learn two things. First, the move mutation and the pdg complement each other. Second, it is not really helpful to follow the wide-spread memetic approach that tries in each step to find the optimum of the operator-defined neighborhood (using the move mutation).

4.4 Comparison of population based EAs

The impact of the parental population size is examined for the first problem of Section 4.3. The $(1+99)$-, $(10+99)$-, and $(20+99)$-strategies all find the optimum for the small problem. However, the algorithms prove to be sensitive concerning the population size which is displayed in Figure 7. $\mu = 10$ is only during the first 37 generations significantly better than $\mu = 1$. However, parental population size $\mu = 20$ is significantly worse almost until it is converged.

A second examination on the effects of the offspring population size is shown in Figure 8. Apparently the number of offspring individuals has a clear influence on the convergence speed. Again, the first problem of Section 4.3 is optimized with $(1+1)$-, $(1+3)$-, $(1+33)$-, and $(1+99)$-strategies. For a better comparison of the results the graphs are scaled such that the number of evaluations is shown in the x-axis. Obviously, a smaller number of offspring leads to faster convergence. As already discussed in Section 4.3 for the move mutation, the memetic approach appears to be disadvantageous for a combination of move mutation and pdg mutation too.

R	l	n	k	μ	λ	p_m	P
30	20	50	50	*	99	0.2	gen. 0.1

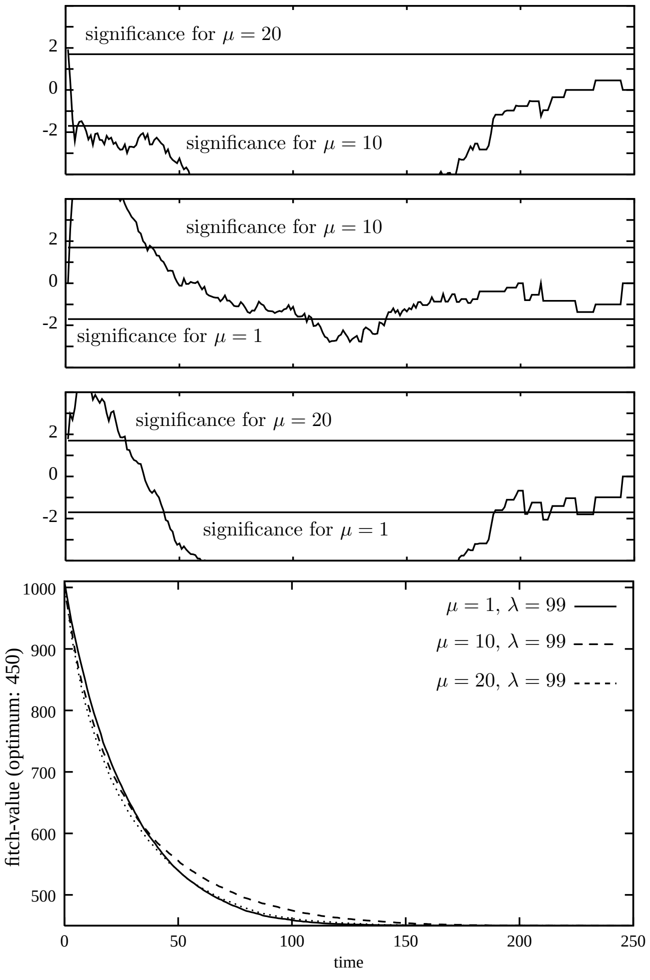


Figure 7: Comparison of different parental population sizes. The performance of the three different population sizes is shown as well as the results of pairwise hypothesis tests.

5 Discussion

5.1 Summary

This paper provides some punctual insight into how characteristics of phylogeny problems and parameter settings of the evolutionary algorithm influence the search process.

A phylogeny problem generator is proposed that simulates evolution to produce sets of sequences. This generator is tunable by the size of the underlying alphabet, the number of sequences, and the probability of the point mutation of the simulated evolution. As a consequence sets of problems with pre-defined difficulty can be produced.

The main results of our analysis of the evolutionary algorithm are the equivalence of move and exchange mutation in combination with pdg recombination, the insights regarding the role of move and pdg mutation at different phases of the search, and some findings concerning the role of the population size of two kinds the number of parents and the number of offspring.

R	l	n	k	μ	λ	p_m	P
10	20	50	50	1	*	0.2	gen. 0.1

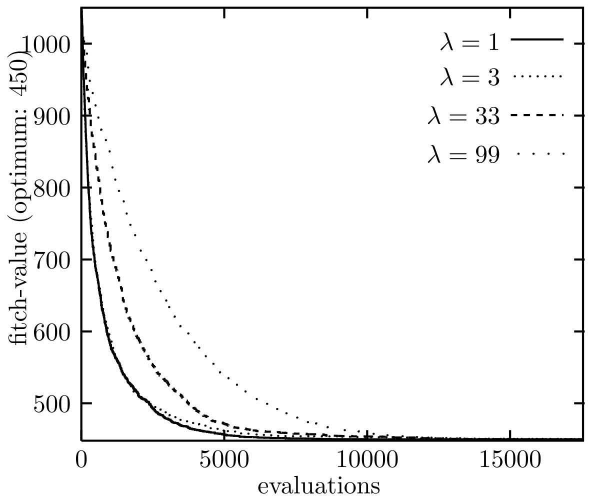


Figure 8: Comparison of different offspring population sizes.

5.2 Future work

As mentioned in section 2.2 we plan to extend the phylogeny problem generator. Especially, we will include more knowledge about the problem properties. The aim is to present a tunable benchmark problem generator that is able to produce realistic phylogeny problems with certain characteristics and known optimum.

The regarded real data problem in section 4.3 shows a very similar behavior to the generated problems. A next step is to analyze more real data problems whether the results of the proposed work will still hold for that problems.

Another planed examination concerns the scalability of the gained insights. Most of the experiments in the presented work are performed on rather small problems with 50 sequences of length 50. It is interesting whether a similar behavior can be observed for very big problems.

Bibliography

A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8(4):429–447, 2002.

H.L Bodlaender, M.R. Fellows, and T.J. Warnow. Two strikes against the perfect phylogeny problem. In *Proceedings of the International Conference on Algorithms, Languages and Programming*, pages 273–283, Wien, 1992. Springer Verlag.

Carlos Cotta and Pablo Moscato. Inferring phylogenetic trees using evolutionary algorithms. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José-Luis Fernández-Villacañas, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, pages 720–729, Berlin, 2002. Springer.

W.H.E. Day, D.S. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81:33–42, 1986.

M.O. Dayhoff. Atlas of protein sequence and structure. *National Biomedical Research Foundation*, 5(3):345–352, 1978.

A. Dress and M. Krüger. Parsimonious phylogenetic trees in metric spaces and simulated annealing. *Advances in Applied Mathematics*, 8:8–37, 1987.

W. M. Fitch. Towards defining the course of evolution: Minimum chances for a specific tree topology. *Systematic Zoology*, 20:406–419, 1971.

L.R. Foulds and R.L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982a.

L.R. Foulds and R.L. Graham. Unlikelihood that minimal phylogeny for a realistic biological study can be constructed in reasonable computational time. *Mathematical Biosciences*, 60:133–142, 1982b.

D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.

Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89:10915–10919, 1992.

N.I. Platnick. An empirical comparison of microcomputer parsimony programs. *Cladistics*, 3:121–144, 1987.

C.C. Ribeiro and D.S. Vianna. A genetic algorithm for the phylogeny problem using an optimized crossover strategy based on path-relinking. In *II Workshop Brasileiro de Bioinformatica*, pages 97–102, Macae, 2003. Springer.

William M. Spears and Mitchell A. Potter. Genetic algorithms (evolutionary algorithms): Repository of test problem generators. http://www.cs.uwyo.edu/˜wspears/generators.html, 1999.

Jan van der Weide and Linda S.W. Steijns. Cytochrome P450 enzyme system: genetic polymorphisms and impact on clinical pharmacology. *Ann Clin Biochem*, 36:722–729, 1999.

M.S. Waterman. *Introductino to Computation Biology – Maps, Seqeunces, and Genomes*. Chapman & Hall/CRC, London, 1995.

Karsten Weicker and Nicole Weicker. Locality vs. randomness – dependence of operator quality on the search state. In Wolfgang Banzhaf and Colin Reeves, editors, *Foundations of Genetic Algorithms 5*, pages 147–163. Morgan Kaufmann, San Francisco, CA, 1999.

Karsten Weicker and Nicole Weicker. Burden and benefits of redundancy. In Worthy N. Martin and William M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 313–333. Morgan Kaufmann, San Francisco, 2001.