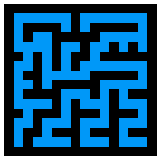
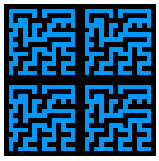
TREE-PUZZLE - Maximum likelihood analysis for nucleotide, amino acid, and two-state data





TREE-PUZZLE Manual
Version 5.2 (July 2004)

Heiko A. Schmidt
von Neumann Institute for Computing (NIC),
Research Center Jülich, D-52425 Jülich, Germany.
email: `hschmidt @ cs.uni-duesseldorf.de`

Korbinian Strimmer
Department of Statistic, University of Munich,
Ludwigstr. 33, D-80539 Munich, Germany.
email: `strimmer @ stat.uni-muenchen.de`

Arndt von Haeseler
von Neumann Institute for Computing (NIC),
Research Center Jülich, D-52425 Jülich, Germany
and
Bioinformatics, Düsseldorf University,
Universitätsstr. 1, D-40225 Düsseldorf, Germany.
email: `haeseler @ cs.uni-duesseldorf.de`

Remarks:

- A bug has been fixed, which tended to produce less resolved trees. Hence, versions prior to 5.2 should not be used anymore!
- Some menus have been extended. Hence, the order of some options might have changed. Please check existing scripts and parameter files!
- Until 2000 TREE-PUZZLE was distributed under the name PUZZLE.

General Information

TREE-PUZZLE is a computer program to reconstruct phylogenetic trees from molecular sequence data by maximum likelihood. It implements a fast tree search algorithm, *quartet puzzling*, that allows analysis of large data sets and automatically assigns estimations of support to each internal branch. TREE-PUZZLE also computes pairwise maximum likelihood distances as well as branch lengths for user specified trees. Branch lengths can be calculated with and without the molecular-clock assumption. In addition, TREE-PUZZLE offers *likelihood mapping*, a method to investigate the support of a hypothesized internal branch without computing an overall tree and to visualize the phylogenetic content of a sequence alignment. TREE-PUZZLE also conducts a number of statistical tests on the data set (chi-square test for homogeneity of base composition, likelihood ratio to test the clock hypothesis, one and two-sided Kishino-Hasegawa test, Shimodaira-Hasegawa test, Expected Likelihood Weights). The models of substitution provided by TREE-PUZZLE are TN, HKY, F84, SH for nucleotides, Dayhoff, JTT, mtREV24, BLOSUM 62, VT, WAG for amino acids, and F81 for two-state data. Rate heterogeneity is modeled by a discrete Gamma distribution and by allowing invariable sites. The corresponding parameters can be inferred from the data set.

TREE-PUZZLE is available free of charge from

- `http://www.tree-puzzle.de/` (TREE-PUZZLE home page)
- `http://www.dkfz-heidelberg.de/tbi/tree-puzzle/` (TREE-PUZZLE home page mirror at DKFZ)
- `http://iubio.bio.indiana.edu/soft/molbio/evolve` (IUBio-Archive, USA)
- `ftp://ftp.pasteur.fr/pub/GenSoft` (Institut Pasteur, France)

TREE-PUZZLE is written in ANSI/ISO C. It will run on most personal computers and workstations if compiled by an appropriate C compiler. The tree reconstruction part of TREE-PUZZLE has been parallelized using the Message Passing Interface (MPI) library standard (Snir *et al.*, 1998; Gropp *et al.*, 1998). If desired to run TREE-PUZZLE in parallel you need an implementation of the MPI library installed on your system. Please read the *Installation* section (2) for more details.

We suggest that this documentation should be read before using TREE-PUZZLE the first time. If you do not have the time to read this manual completely please do read at least the sections *Input/Output Conventions* (3.1) and *Quick Start* (4) below. Then you should be able to use the TREE-PUZZLE program, especially if you have some experience with the PHYLIP programs. The other sections should then be read at a later time.

To find out what's new in the current version please read the *Version History* (section 16).

Contents

Chapter 1

Legal Stuff

Chapter 2

Installation

The source code of the TREE-PUZZLE software is 100% identical across platforms. However, installation procedures differ. There is a source distribution (`tree-puzzle-5.2.tar.gz`, `tree-puzzle-5.2.tar.zip`, `tree-puzzle-5.2.tar.sit`) and binary distributions (`tree-puzzle-5.2-linux.tar.gz`, `tree-puzzle-5.2-macosx.sit`, `tree-puzzle-5.2-windows.zip`) with executables for Linux, Mac OS X, and Windows respectively.

2.1 UNIX/Source Distribution

Get the file `tree-puzzle-5.2.tar.gz` (or the `.sit` or `.zip` file for Mac or Windows). Decompress it first (using `gunzip` command or another program capable of handling `gz` format) and then untar the file with

```
gunzip tree-puzzle-5.2.tar.gz
tar xvf tree-puzzle-5.2.tar
```

The newly created directory `tree-puzzle-5.2` contains four subdirectories called `doc`, `data`, `bin`, and `src`. The `doc` directory contains this manual in HTML format. The `data` directory contains example input files. The `src` directory contains the ANSI/ISO C sources of TREE-PUZZLE. Switch to this directory by typing

```
cd tree-puzzle-5.2
```

To compile we recommend the GNU gcc (or GNU egcs) compiler. If gcc is installed just type

```
sh ./configure
make
make install
```

and the executable `puzzle` is compiled and put into the `/usr/local/bin` directory. If you want to have `puzzle` installed into another directory you can set this by setting the `--prefix=/name/of/the/wanted/directory` directive at the `sh ./configure` command line. The parallel version should have been built and installed as well, if `configure` found a known MPI compiler/installation (cf. 2.1.6 Parallel TREE-PUZZLE).

Then type

```
make clean
```

and everything will be nicely cleaned up.

If your compiler is not the GNU gcc compiler and not found by `configure` you will have to modify that, by setting the `CC` variable (e.g. `setenv CC cc` under `csh` or `CC=cc; export CC` under `sh`/`bash`) before running `sh ./configure`. If you still cannot compile properly then your compiler or its runtime library is most probably not ANSI compliant (e.g., very old SUN compilers). In most cases, however, you will succeed to compile by changing some parameters in the `makefile`. Ask your local Unix expert/system administrator for help.

2.1.1 Linux (binary distribution)

Get the file `tree-puzzle-5.2-linux.tar.gz`. After decoding this file (if you have problems, please ask your local Linux expert), you will find a folder called `tree-puzzle-5.2` on your hard disk. This folder contains the subfolders `doc`, `data`, and `src`. The `doc` folder contains this manual in PDF format. The `data` folder contains example input files. The `src` folder contains the ANSI/ISO C sources of TREE-PUZZLE as well as the precompiled executable.

Rename the Linux executable `puzzle-linux-gcc-static` to `puzzle` and copy it to a folder in your `PATH`, e.g., `/usr/local/bin` is generally a good choice.

The Linux executable have been compiled using the GNU C compiler (`http://gcc.gnu.org`). If you need a compiler to prepare the executable yourself, you might download one from the list below (section 2.2). Then proceed as described in the UNIX/Source Installation section (2.1).

2.1.2 Mac OS X (binary distribution)

Get the file `tree-puzzle-5.2-macosx.sit`. After decoding this file (if you have problems, please ask your local Mac OS X expert), you will find a folder called `tree-puzzle-5.2` on your hard disk. This folder contains the subfolders `doc`, `data`, and `src`. The `doc` folder contains this manual in PDF format. The `data` folder contains example input files. The `src` folder contains the ANSI/ISO C sources of TREE-PUZZLE as well as the precompiled executable.

Rename the Mac OS X executable `puzzle-macosx-Xcode` to `puzzle` and copy it to a folder in your `PATH`, e.g., `/usr/local/bin` is generally a good choice.

The Mac OS X executable have been compiled using Apple's Xcode Tools (`http://developer.apple.com/macosx/`). If you need a compiler to prepare the executable yourself, you might download one from the list below (section 2.2). Then proceed as described in the *UNIX/Source Installation* section (2.1).

2.1.3 Older Mac OSes

Due to missing access we do not support Mac OS 9/Classic anymore. We recommend changing to Mac OS X, which combines the advantages out UNIX systems with the Mac OS's graphical interface.

If you want to prepare an executable for your Mac OS yourself, you might get Metrowerks' CodeWarrior (refer to section 2.2) to compile the C sources included in all TREE-PUZZLE distribution packages.

2.1.4 Windows 95/98/NT/... (binary distribution)

Get the file `tree-puzzle-5.2-windows.zip`. After un-zipping this file (if you have problems, please ask your local Windows expert), you will find a folder called `tree-puzzle-5.2` on your hard disk. This folder contains the subfolders `doc`, `data`, and `src`. The `doc` folder contains this manual in PDF format. The `data` folder contains example input files. The `src` folder contains the ANSI/ISO C sources of TREE-PUZZLE as well as the precompiled executable.

Rename the Windows executable `puzzle-windows-mingw` to `puzzle` and copy it to a folder in Windows' search path.

The Windows executable has been compiled using MinGW (`http://www.mingw.org`). If you need a compiler to prepare the executable yourself, you might download one from the list below (section 2.2). Then proceed as described in the UNIX/Source Installation section (2.1).

If you have a Linux partition on your PC we recommend to install and use TREE-PUZZLE under Linux (see section 2.1.1) because it runs TREE-PUZZLE faster than Windows.

2.1.5 VMS

TREE-PUZZLE for VMS is not supported anymore. Nevertheless TREE-PUZZLE should still be compilable for VMS using the sources (see section 2.1 and ask your local VMS expert for help).

2.1.6 Parallel TREE-PUZZLE

To compile and run the parallelized TREE-PUZZLE you need an implementation of the Message Passing Interface (MPI) library, a widely used message passing library standard. Implementations of the MPI libraries are available for almost all parallel platforms and computer systems, and there are free implementations for most platforms as well.

To find an MPI implementation suitable for your platform visit the following web sites:

- `http://www.lam-mpi.org/mpi/implementations/`
- `http://www-unix.mcs.anl.gov/mpi/implementations.html`
- `http://WWW.ERC.MsState.Edu/labs/hpcl/projects/mpi/implementations.html`

Although MPI is also available on Macintosh and Windows systems, the developers never ran the parallel version on those platforms.

To install the parallel version of TREE-PUZZLE you need the source distribution for TREE-PUZZLE and install the package on your computer as described above (2.1). The `configure` should configure the Makefiles appropriately. If there is no known MPI compiler found on the system the parallel version is not configured. (If problems occur ask your local system administrator for help.)

Than you should be able to compile the parallel version of TREE-PUZZLE using the following commands:

```
sh ./configure
make
make install
```

and the executable `ppuzzle` is compiled and put into the `/usr/local/bin` directory. If you want to have the executable installed into another directory please proceed as described in section 2.1.

If your compiler is non out of `mpcc` (IBM), `hcc` (LAM), `mpicc_lam` (LAM under LINUX), `mpicc_mpich` (MPICH under LINUX), and `mpicc` (LAM, MPICH, HP-UX, etc.) and not found by `configure` you will have to modify that by setting the `MPICC` variable (e.g. `setenv MPICC /another/mpicc` under `csh` or `MPICC=/another/mpicc; export MPICC` under `sh`) before running `sh ./configure`.

Some compilers (e.g., IBM) have problems to compile the SPRNG random number generator source code. In this case you can use the old 'leapfrog generator' by setting the `CFLAGS` variable to `-DNOSPRNG` (e.g., `setenv CFLAGS '-DNOSPRNG'` under `csh` or `CFLAGS='-DNOSPRNG'; export CFLAGS` under `sh`/`bash`) before running `sh ./configure`.

The way you have to start `ppuzzle` depends on the MPI implementation installed. So please refer to your MPI manual or ask your local MPI expert for help.

Note: The parallelization of the tree reconstruction method follows a master-worker-concept, i.e., a master process handles the scheduling of the computation to the n worker processes, while the worker processes are doing almost all the computation work of evaluating the quartets and constructing the *puzzling step* trees.

Since the master process does not require a lot of CPU time, it can be scheduled sharing one processor with a worker process. Thus, you can run `ppuzzle` by assigning $n + 1$ processes.

If you want to evaluate a usertree or perform *likelihood mapping* analysis it is not recommended to do a parallel run, because all the computation will be done by the master process. Hence a run of the sequential version of TREE-PUZZLE is more appropriate for usertree or *likelihood mapping* analysis.

2.2 ANSI/ISO C Compilers

If there is no binary version of TREE-PUZZLE available for your computer or if you want to compile TREE-PUZZLE yourself, an ANSI/ISO C compiler is needed to produce an executable suitable for your machine. There are a number of free compilers available the different operating systems:

GCC - Gnu Compiler Collection (UNIX/Linux, Mac OS X, Windows), available for most OSes, widely spread in the UNIX/Linux community where it is included in virtually every distribution. GCC is also the basis to most compilers mentioned below. (`http://gcc.gnu.org`)

ICC - Intel C++ Compiler (Linux, Windows), ICC has shown to have good performance. It is free to a certain extend - please refer to license. (`http://www.intel.com/software/products/compilers/`)

MinGW - Minimalist GNU for Windows (Windows), a GCC based package that provides all basic tool necessary for software development with Gnu tools under Windows (`http://www.mingw.org`)

CygWin - GNU+Cygnus+Windows (Windows), a GCC based package that provides the (almost) full Gnu environment and tools known from UNIX/Linux systems (including X-Windows). Unfortunately, executables compiled with CygWin need CygWin installed to be run on other Windows computers. (`http://www.cygwin.com`)

Xcode Tools (Mac OS X), the GCC based software development tools from Apple for Mac OS X. It seems to be free, but you have to register. Please check the license. (`http://developer.apple.com/macosx/`)

CodeWarrior (Mac OS Classic/9/X, Windows), this is a commercial development environment from Metrowerks. Although not supported by the TREE-PUZZLE developers, CodeWarrior is able to produce executables for older Mac OSes. (`http://www.metrowerks.com/MW/Develop/CodeWarrior.htm`)

For MPI (message passing interface) libraries which are needed to build the parallel version of TREE-PUZZLE refer to the *Parallel TREE-PUZZLE* section (2.1.6).

2.3 Contributed TREE-PUZZLE Packages

There are a number of other distributions for different operating systems (ordered alphabetically)

BioLinux.org `http://www.biolinux.org/tree-puzzle.html`, (thanks to Luc Ducazu), RPMs for RedHat, Fedora, SuSE

Debian Linux `http://packages.debian.org/stable/science/tree-puzzle` (thanks to Andreas Tille and Stephane Bortzmeyer)

FreeBSD `http://www2.de.freebsd.org/ports/biology.html` (thanks to Jan Lentfer)

NetBSD `http://pkgsrc.netbsd.se/?cat=search&34844` (thanks to Marc Baudoin)

Chapter 3

Introduction

TREE-PUZZLE is an ANSI/ISO C application to reconstruct phylogenetic trees from molecular sequence data by maximum likelihood. It implements a fast tree search algorithm, *quartet puzzling*, that allows analysis of large data sets and automatically assigns estimations of support to each internal branch. Rate heterogeneity (invariable sites plus Gamma distributed rates) is incorporated in all models of substitution available (nucleotides: SH, TN, HKY, F84, and submodels; amino acids: Dayhoff, JTT, mtREV24, BLOSUM 62, VT, and WAG; two-state data: F81). All parameters including rate heterogeneity can be estimated from the data by maximum likelihood approaches. TREE-PUZZLE also computes pairwise maximum likelihood distances as well as branch lengths for user specified trees. In addition, TREE-PUZZLE offers a *likelihood mapping* to investigate the support of internal branches without computing an overall tree.

3.1 Input/Output Conventions

A few things of the name conventions have changed compared to earlier (< 5.0) PUZZLE releases. From version 5.0 onwards names of the sequence input file and the usertree file can be specified at the command line (e.g. '`puzzle infilename intreename`', where `infilename` is the name of the sequence file and `intreename` is the name of the usertree file). If only the input filename or no filename is given at the command line the TREE-PUZZLE software searches for input files named `infile` and/or `intree` respectively.

The naming conventions of the output files have changed as well. As prefix of the output filenames the name of the sequence input file (or the usertree file in the usertree analysis case) is used and an extension added to denote the content of the file. If no input filename is given at the command line the default filenames of the earlier versions are used.

The following extensions/default filenames are possible:

Extension	default filename	file content
`.puzzle`	`outfile`	TREE-PUZZLE report
`.dist`	`outdist`	ML distances
`.tree`	`outtree`	final tree(s)
`.qlist`	`outqlist`	list of unresolved quartets
`.ptorder`	`outptorder`	list of unique *puzzling step* tree topologies
`.pstep`	`outpstep`	list of *puzzling step* tree topologies in chronological order
`.eps`	`outlm.eps`	EPS graphics file generated in the *likelihood mapping* analysis

The file types are described in detail below. In the following "`INFILENAME`" denotes the prefix, which is the sequence input filename or the usertree filename respectively.

3.1.1 Sequence Input

TREE-PUZZLE requests sequence input in PHYLIP INTERLEAVED format (sometimes also called PHYLIP 3.4 format). Many sequence editors and alignment programs (e.g., CLUSTAL W) output data in this format. The `data` directory contains four example input files (`globin.a`, `marswolf.n`, `atp6.a`, `primates.b`) that can be used as templates for own data files. The default name of the sequence input file is `infile`, if no input filename is given at the command line. If an `infile` or a file with the given name is not present TREE-PUZZLE will request an alternative file name. Sequences names in the input file are allowed to contain blanks but all blanks will internally be converted to underscores '_'. Sequences can be in upper or lower case, any spaces or control characters are ignored. The dot '.' is recognized as character matching to the first sequence, it can be used in all sequences except in the first sequence. Valid symbols for nucleotides are A, C, G, T and U, and for amino acids A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y. All other visible characters (including gaps, question marks etc.) are treated as N (DNA/RNA) or X (amino acids). For two-state data the symbols 0 and 1 are allowed. The first sequence in the data set is considered the default outgroup.

3.1.2 General Output

All results are written to the TREE-PUZZLE report file (`INFILENAME.puzzle` or `outfile`). If the option "List all unresolved quartets" is invoked a file called "`INFILENAME.qlist`" or "`outqlist`" is created showing all these quartets. If the option "List puzzling step trees" is set accordingly the files "`INFILENAME.pstep`" or "`outpstep`" and/or "`INFILENAME.ptorder`" or "`outptorder`" are generated.

The "`INFILENAME.ptorder`" or "`outptorder`" file contains the unique tree topologies in PHYLIP format preceded by PHYLIP-format comment (in parenthesis). A typical line in the ptorder file looks like this:

```
[ 2. 60 6.00 2 5 1000 ](chicken,((cat,(horse,(mouse,rat))),
```

```
(opossum,platypus)));
```

The entries (separated by single blanks) in the parenthesis mean the following:

- **2.** - Topology occurs second-most among all intermediate tree topologies (= order number).
- **60** - Topology occurs 60 times.
- **6.00** - Topology occurs in 6.00 % of the intermediate tree topologies.
- **2** - unique topology ID (needed for the pstep file)
- **5** - Sum of uniquely occurring topologies.
- **1000** - Sum of intermediate trees estimated during the analysis.

The "`INFILENAME.pstep`" or "`outpstep`" file contains a log of the *puzzling steps* performed and the occurring tree topologies.

A typical line in the pstep file contains the following entries (separated by tabstops):

```
6. 55 698 3 5 828
```

The entries in the rows mean the following:

- **6.** - 6th block of intermediate trees performed.
- **55** - number of intermediate trees inferred in this block.
- **698** - occurrences of this topology so far.
- **3** - unique topology ID (for lookup in the ptorder file).
- **5** - number unique topologies occurred so far.
- **828** - number of *puzzling step* performed so far.

In the case of a sequential run (`puzzle`) the entries of this file are more resolved, because every block consists of one intermediate tree.

3.1.3 Distance Output

TREE-PUZZLE automatically computes pairwise maximum likelihood distances for all the sequences in the data file. They are written in the TREE-PUZZLE report file "`INFILENAME.puzzle`" or "`outfile`" and in the separate file "`INFILENAME.dist`" or "`outdist`". The format of distance file is PHYLIP compatible (i.e. it can directly be used as input for PHYLIP distance-based programs such as `neighbor`).

3.1.4 Tree Output

The *quartet puzzling* tree with its support values and with maximum likelihood branch lengths is displayed as ASCII drawing in the TREE-PUZZLE report in "`INFILENAME.puzzle`" or "`outfile`". The same tree is written into the "`INFILENAME.tree`" or "`outtree`" file in CLUSTAL W format. If clock-like maximum-likelihood branch lengths are computed there will be both an unrooted and a rooted tree in the "`INFILENAME.puzzle`" or "`outfile`". The tree convention follows the NEWICK format (as implemented in PHYLIP or CLUSTAL W): the tree topology is described by the usual round brackets `(a,b,(c,d));` where branch lengths are written after the colon a:0.22,b:0.33. Support values for each branch are displayed as internal node labels, i.e., they follow directly after each node before the branch length to each node. Here is an example:

```
(Gibbon:0.1393, ((Human:0.0414, Chimpanzee:0.0538)99:0.0175,
Gorilla:0.0577)98:0.0531, Orangutan:0.1003);
```

The likelihood value of each tree is added in parenthesis before the tree string (e.g. "`[ lh=-1621.201605 ]`"). Parenthesis mark comments in the Newick or PHYLIP tree format. In some cases the comment has to be removed before using them with other programs.

With the programs TreeView and TreeTool it is possible to view a tree both with its branch lengths and simultaneously with the support values for the internal branches (here 98% and 99%). Note, the PHYLIP programs DRAWTREE and DRAWGRAM may also be used with the CLUSTAL W treefile format. However, in the version 3.5 they ignore the internal labels and simply print the tree topology along with branch lengths.

3.1.5 Tree Input

TREE-PUZZLE optionally also reads input trees. The default name for the file containing the input tree is `intree`, if not given at the command line, but if you choose the input tree option and there is no file with the given name or `intree` present you will be prompted for an alternative name. The format of the input trees is identical to the trees in the "`INFILENAME.tree`" or "`outtree`" file. However, it is sufficient to provide the tree topology only, you don't need to specify branch lengths (that are ignored anyway) or internal labels (that are read, stored, and written back to the "`INFILENAME.tree`" or "`outtree`" file). The input trees needs not to be unrooted, they can also be rooted. It is important that sequence names in the input tree file do not contain blanks (use underscores!). The trees can be multifurcating. The format of the tree input file is easy: just put the trees into the file. TREE-PUZZLE counts the ';' at the end of each tree description to determine how many input trees there are. Any header (e.g., with the number of trees) is ignored (this is useful in conjunction with programs like MOLPHY that need this header). If there is more than one tree TREE-PUZZLE performs the Shimodaira-Hasegawa test (Shimodaira and

Hasegawa, 1999), the two-sided and two-sided Kishino-Hasegawa test (Kishino and Hasegawa, 1989; Goldman *et al.*, 2000) and computes the *expected likelihood weights* (ELW Strimmer and Rambaut, 2002) or, if chosen in the menu by the user, constructs a consensus tree from the input trees.

3.1.6 Likelihood Mapping Output

TREE-PUZZLE also offers *likelihood mapping* analysis, a method to investigate support for internal branches of a tree without computing an overall tree and to graphically visualize phylogenetic content of a sequence alignment. The results of *likelihood mapping* are written in ASCII to the "`INFILENAME.puzzle`" or "`outfile`" as well as to a file called "`INFILENAME.eps`" or "`outlm.eps`" respectively. This file contains in encapsulated Postscript format (EPSF) a picture of the triangle that forms the basis of the *likelihood mapping* analysis. You may print it out on a Postscript capable printer or view it with a suitable program. The "`INFILENAME.eps`" or "`outlm.eps`" file can be edited by hand (it is plain ASCII text!) or by drawing programs that understand the Postscript language (e.g., Adobe Illustrator). It can also be converted to other formats with tools like ghostscript, a free Postscript interpreter which is available for the different operating systems (`http://www.ghostscript.com`).

Chapter 4

Quick Start

Prepare your sequence input file and, optionally, your tree input file. Then start the TREE-PUZZLE program. TREE-PUZZLE will choose automatically the nucleotide or the amino acid mode. If more than 85% of the characters (not counting '-' and '?') in the sequences are A, C, G, T, U or N, it will be assumed that the sequences consists of nucleotides. If your data set contains amino acids TREE-PUZZLE suggests whether you have amino acids encoded on mtDNA or on nuclear DNA, and selects the appropriate model of amino acid evolution. If your data set contains nucleotides the default model of sequence evolution chosen is the HKY model. Parameters need not to be specified, they will be estimated by a maximum likelihood procedure from the data. If TREE-PUZZLE detects a usertree file stated at the command line or one called `intree` it automatically switches to the input tree mode.

Then, a menu (PHYLIP "look and feel") appears with default options set. It is possible to change all available options. For example, if you want to incorporate rate heterogeneity you have to select option 'w' as rate heterogeneity is switched off by default. Then type 'y' at the input prompt and start the analysis. You will see a number of status messages on the screen during computation. When the analysis is finished all output files (e.g., `outfile`, `outtree`, `outdist`, `outqlist`, `outlm.eps`, `outpstep`, `outptlist` or `INFILENAME.puzzle`, `INFILENAME.tree`, `INFILENAME.dist`, `INFILENAME.qlist`, `INFILENAME.eps`, `INFILENAME.pstep`, `INFILENAME.ptorder`) will be in the same directory as the input files.

To obtain a high quality picture of the output tree (including node labels) you might want to use use the TreeView program by Roderic Page. It is available free of charge and runs on Mac OS and MS-Windows. It can be retrieved from `http://taxonomy.zoology.gla.ac.uk/rod/treeview.html`. TreeView understands the CLUSTAL W treefile conventions, reads multifurcating trees and is able to simultaneously display branch lengths and support values for each branch. Open the "`INFILENAME.tree`" or "`outtree`" file with TreeView, choose "Phylogram" to draw branch lengths, and select "Show internal edge labels".

On a Unix you can use the TreeTool program (`ftp://rdp.life.uiuc.`

edu/pub/RDP/programs/TreeTool) to display and manipulate TREE-PUZZLE trees. Precompiled SUN executables are available from the TreeTool homepage. A statically linked version for Linux is available from http://www.tree-puzzle.de.

Chapter 5

Models of Sequence Evolution

Here we give a brief overview over the models implemented in TREE-PUZZLE.

5.1 Models of Substitution

The substitution process is modeled as reversible time homogeneous stationary Markov process. If the corresponding stationary nucleotide (amino acid) frequencies are denoted π_i the most general rate matrix for the transition from nucleotide (amino acid) i to j can be written as

$$R_{ij} = \begin{cases} Q_{ij}\pi_j & \text{for } i \neq j \\ -\sum_m Q_{im}\pi_m & \text{for } i = j \end{cases}$$

The matrix Q_{ij} is symmetric with $Q_{ii} = 0$ (diagonals are zero). For nucleotides the most general model implemented is the general time reversible or GTR model (Lanave *et al.*, 1984; Tavaré, 1986; Rodriguez *et al.*, 1990). The GTR model allows for six different substitution rates ($R_{AC} = R_{CA}$, $R_{AG} = R_{GA}$, $R_{AT} = R_{TA}$, $R_{CG} = R_{GC}$, $R_{CT} = R_{TC}$, $R_{GT} = R_{TG}$). However, there is no automatic parameter estimation implemented (yet) to estimate these parameters from the input data. Hence the parameters have to be entered by the user. The most general model built into TREE-PUZZLE with parameter estimation is the Tamura-Nei model (TN, Tamura and Nei (1993)).

The matrix Q_{ij} for this model equals

$$Q_{ij} = \begin{cases} \frac{4t\gamma}{\gamma+1} & \text{for } i \to j \text{ pyrimidine transition} \\ \frac{4t}{\gamma+1} & \text{for } i \to j \text{ purine transition} \\ 1 & \text{for } i \to j \text{ transversion} \end{cases}$$

The parameter γ is called the "Y/R transition parameter" whereas t is the "Transition/transversion parameter". If γ is equal to 1 we get the HKY model

(Hasegawa *et al.*, 1985). Note, the ratio of the transition and transversion rates (without frequencies) is $\kappa = 2t$. There is a subtle but important difference between the *transition-transversion parameter*, the *expected transition-transversion ratio*, and the *observed transition transversion ratio*. The *transition-transversion parameter* simply is a parameter in the rate matrix. The *expected transition-transversion ratio* is the ratio of actually occurring transitions to actually occurring transversions taking into account nucleotide frequencies in the alignment. Due to saturation and multiple hits not all substitutions are observable. Thus, the *observed transition-transversion ratio* counts observable transitions and transversions only. If the base frequencies in the HKY model are homogeneous ($\pi_i = 0.25$) HKY further reduces to the Kimura model. In this case t is identical to the expected transition/transversion ratio. If t is set to 0.5 the Jukes-Cantor model is obtained. The F84 model (as implemented in the various PHYLIP programs, (Felsenstein, 1984) is a special case of the Tamura-Nei model.

For amino acids the matrix Q_{ij} is fixed and does not contain any free parameters. Depending on the type of input data four different Q_{ij} matrices are available in TREE-PUZZLE. The Dayhoff (Dayhoff *et al.*, 1978) and JTT (Jones *et al.*, 1992) matrices are for general use with proteins encoded on nuclear DNA, the mtREV24 (Adachi and Hasegawa, 1996) matrix is for use with proteins encoded on mtDNA.

The WAG matrix has been inferred from a database of 3905 globular protein sequences, forming 182 distinct gene families spanning a broad range of evolutionary distances (Whelan and Goldman, 2001), and is, hence applicable for a wide range of protein families.

The VT model is based an new estimator for amino acid replacement rates, the resolvent method. The VT matrix has been computed from a large set alignments of varying degree of divergence. Hence VT is for use with proteins of distant relatedness as well (Müller and Vingron, 2000).

Although implemented in TREE-PUZZLE on request, the BLOSUM 62 matrix (Henikoff and Henikoff, 1992) has been been developed for protein searches. Hence, it does not represents an evolutionary process and should not really be used for phylogeny reconstruction.

For doublets (pairs of dependent nucleotides) the SH model (Schöniger and von Haeseler, 1994) is implemented in TREE-PUZZLE. The corresponding matrix Q_{ij} reads

$$Q_{ij} = \begin{cases} 2t & \text{for } i \rightarrow j \text{ transition substitution} \\ 1 & \text{for } i \rightarrow j \text{ transversion substitution} \\ 0 & \text{for } i \rightarrow j \text{ two substitutions} \end{cases}$$

The SH model basically is a F81 model (Felsenstein, 1981) for single substitutions in doublets.

5.2 Models of Rate Heterogeneity

Rate heterogeneity is taken into account by considering invariable sites and by introducing Gamma-distributed rates for the variable sites.

For invariable sites the parameter θ ("Fraction of invariable sites") determines the probability of a given site to be invariable. If a site is invariable the probability for the constant site patterns is π_i, the frequency of each nucleotide (amino acid).

The rates r for variable sites are determined by a discrete Gamma distribution that approximates the continuous Gamma distribution

$$g(r) = \frac{\alpha^\alpha r^\alpha}{e^{\alpha r}\Gamma(\alpha)}$$

where the parameter alpha ranges from $\alpha = \infty$ (no rate heterogeneity) to $\alpha < 1$ (strong heterogeneity). The mean expectation of r under this distribution is 1.

A mixed model of rate heterogeneity (Gamma plus invariable sites) is also available. In this case the total rate heterogeneity ρ (as defined by Gu *et al.* (1995)) computes as $\rho = \frac{(1+\theta\alpha)}{(1+\alpha)}$.

Chapter 6

Possible Analysis

A number of analyses are possible with the TREE-PUZZLE package. The main ones are described in the following.

6.1 Tree Reconstruction Using Quartet Puzzling

The main purpose of the TREE-PUZZLE package is to reconstruct trees. It uses the *quartet puzzling algorithm* as described by Strimmer and von Haeseler (1996). *Quartet Puzzling* is a three step algorithm. Preceeded by an initializing step that reads input data and infers missing parameters. The three steps are:

ML Step: In this first step, all quartet tree topologies are evaluated to get a set of quartets, which are best supported by the underlying n sequence alignment. To that end, all $\binom{n}{4}$ quartets, i.e., groups of four sequences, are evaluated using maximum likelihood. The three quartet topologies $ab|cd$, $ac|bd$, and $ac|bd$ (cf. Fig 6.1) are then weighted by their posterior probabilities. Then one, two, or even all three quartet topologies to the set of supported quartets according to their weights (Strimmer *et al.*, 1997).

Puzzling Step: Starting from one quartet tree the remaining $n-4$ sequences are added one-by-one to get a full n-tree. A sequence is added to the branch which is least contradicted by the neighborhood relationships from the quartet set constructed in the ML step as described by Strimmer and von Haeseler (1996), ties broken randomly. Differing from their description we use a recursive $O(n^4)$ algorithm that gives the same result in faster time.

This step is repeated very often producing a large number of so-called intermediate trees.

Consensus Step: From the set of intermediate trees from the ML step a majority consensus tree is built and its branch lengths and ML value are estimated. By default a 50% majority consensus (M_{50} consensus *sensu*

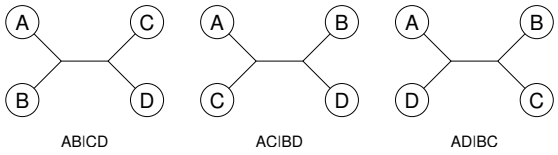


Figure 6.1: The 3 fully resolved tree topologies for the quartet (A,B,C,D).

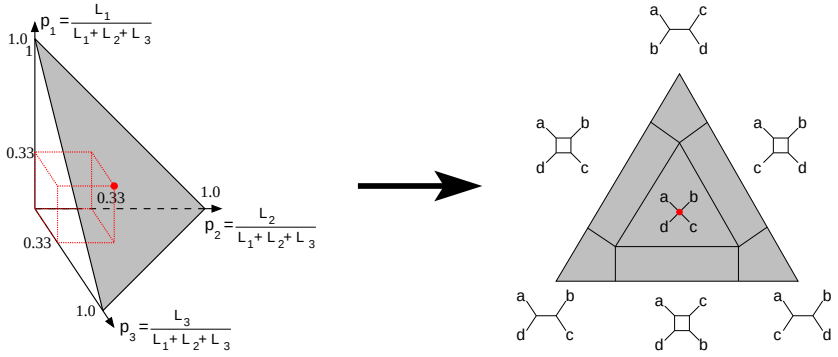


Figure 6.2: What does the likelihood diagrams show? The posterior weights of the three resolved tree topologies for each quartet plotted into 3-dimensional coordinate system (left side). All points fall into the triangular grey surface. Hence, shifting the view point produces the triangular diagram.

McMorris and Neumann, 1983) is constructed. However this behavior can be changed to a *relative consensus* (M_{rel}, cf. 6.4 for details) by adding the '`-consmrel`' flag at the commandline.

For more detailed descriptions of the *quartet puzzling* algorithm refer to Strimmer and von Haeseler (1996), Strimmer *et al.* (1997), as well as Schmidt and von Haeseler (2003). Further informations might also be found in chapters 7, 8, and 9.

6.2 Likelihood Mapping

The quartet topology weighting introduced by Strimmer *et al.* (1997) led to three values that add up to 1.0. Plotting these three posterior weights into a 3-dimensional coordinate system makes all points fall into a triangular surface, a so-called simplex (Fig. 6.2). This led enables a way to analyze phylogenetic information in datasets, *likelihood mapping* (Strimmer and von Haeseler, 1997).

Plotting all $\binom{n}{4}$ quartets (or a random subsample) in a *likelihood mapping* diagram provides an overview of how many quartets cannot be resolved (Fig. 6.3, left side). The higher this percentage, the less suited a dataset is for phylogenetic analysis. How can a tree be resolved reliably, if even its quartet subtrees cannot?

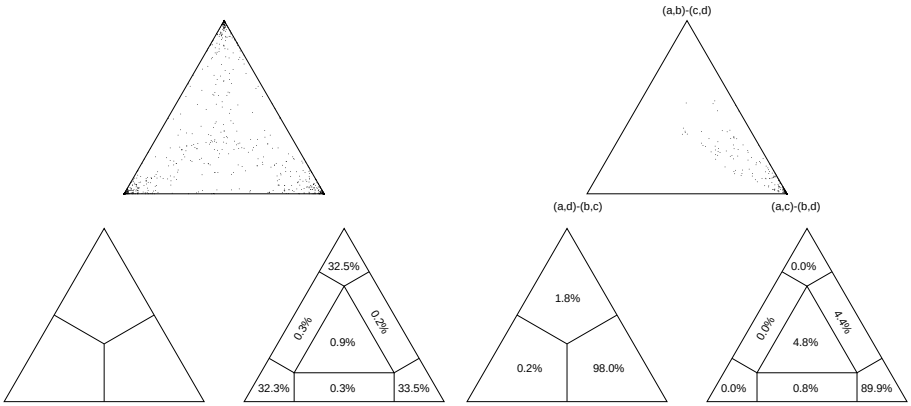


Figure 6.3: Likelihood mapping diagrams produced from the elongation factor dataset (`EF.phy`). Left side: measurement of phylogenetic signal in the dataset showing only 0.9% unresolved and 0.8 % partly resolved quartets. Right side: grouped likelihood mapping diagram of the branching pattern of EF-2/G sequences. The EF dataset was grouped into 4 clusters: Crenarchaeota (a), Bacteria (b), Eucaryota (c), EF-1α/Tu sequences as outgroup (d). The clustering of Crenarchaeota (a) and Eucaryota (c) is supported suggesting the Bacteria at the root of the EF-2/G subtree.

Alternatively, clusters of sequences can be analyzed by *likelihood mapping*. This analyzes the support for internal branches or groupings in a tree without having to compute an overall tree. Every internal branch in a completely resolved tree defines up to four clusters of sequences. Sometimes only the relationship of these groups are of interest and not details of the structure of the clusters themselves. Then a *likelihood mapping* analysis is sufficient. The corresponding *likelihood mapping* triangle diagrams (as contained in various output files generated by TREE-PUZZLE) will elucidate the possible relationships in detail (Fig. 6.3, right side).

For more detailed descriptions refer to Strimmer and von Haeseler (1997) as well as Schmidt and von Haeseler (2003). Further informations might also be found in chapters 7, 8, and 9.

6.3 Usertree Evaluation and Testing

It is also possible to estimate ML branch lengths and likelihood values for a set of tree topologies provided by the user. If more than one input tree are provided, several tests are performed to compare these and infer some kind of 'confidence sets'. The SH test (Shimodaira and Hasegawa, 1999) is a multi-comparison test taking into account that any tree in the test set could be the maximum likelihood tree. Two types of Kishino and Hasegawa test (KH test) are implemented to check which trees are significantly worse than the best tree. Included are, first,

the original two-sided KH test as described in Kishino and Hasegawa (1989), and second, the one-sided KH test (Goldman *et al.*, 2000) using pairwise SH tests to cope with the fact that we test against the best tree. Goldman *et al.* (2000) pointed out the two-sided KH test is not suited for testing against the ML tree since all other trees can hardly be better in terms of likelihood (cf. that paper for details). Furthermore expected likelihood weights (ELW, Strimmer and Rambaut 2002) are implemented which infer a narrow 'confidence set' on the tree set. For the interpretation and applicability of the different tests, the articles cited above have to be read carefully.

6.4 Consensus Tree Construction

Alternatively, a consensus tree can be constructed from all the trees in a set. ML branch lengths and ML value are estimated for the consensus topology.

Currently, TREE-PUZZLE by default constructs a 50% majority rule consensus (M_{50} consensus *sensu* McMorris and Neumann, 1983), that means all splits (bipartitions of the sequence set) occurring in more than 50% of the dataset are incorporated in the consensus tree.

A *relative majority consensus tree* (M_{rel} consensus) is constructed, if the user adds the '`-consmrel`' flag at the commandline. In this M_{rel} consensus all splits are incorporated into the consensus tree that do not contradict any other split occurring equally or more often in the set of user trees. This means, all splits are used, even below 50% occurrence, until the first incongruence is observed. (This is comparable to a *relative majority* vote.)

Further informations might also be found in chapters 7, 8, and 9.

6.5 Parameter Estimation and Pairwise Distances

It is also possible to only infer parameters and compute a maximum likelihood distance matrix. That means, no tree is reconstructed and TREE-PUZZLE ends after writing the estimated parameters and the pairwise distance matrix to the according output files (`*.puzzle`, `*.dist`).

This can be very useful if one wants TREE-PUZZLE to estimate parameters for the use in another program, e.g., to perform further analyses.

Further informations might also be found in chapters 7, 5, 8, and 9.

Chapter 7

Available Options

All options can be selected and changed after TREE-PUZZLE has read the input file. Depending on the input files options are preselected and displayed in a menu ("PHYLIP look and feel"):

```
GENERAL OPTIONS
 b                      Type of analysis?  Tree reconstruction
 k                 Tree search procedure?  Quartet puzzling
 v        Approximate quartet likelihood?  No
 u              List unresolved quartets?  No
 n              Number of puzzling steps?  1000
 j              List puzzling step trees?  No
 o                   Display as outgroup?  Gibbon
 z     Compute clocklike branch lengths?  No
 e                   Parameter estimates?  Approximate (faster)
 x             Parameter estimation uses?  Neighbor-joining tree
SUBSTITUTION PROCESS
 d           Type of sequence input data?  Nucleotides
 m                 Model of substitution?  HKY (Hasegawa et al. 1985)
 t    Transition/transversion parameter?  Estimate from data set
 f                Nucleotide frequencies?  Estimate from data set
RATE HETEROGENEITY
 w           Model of rate heterogeneity?  Uniform rate

Quit [q], confirm [y], or change [menu] settings:
```

By typing the letters shown in the menu you can either change settings or enter new parameters. Some options (for example 'm' and 'w') can be invoked several times to switch through a number of different settings. The parameters of the models of sequence evolution can be estimated from the data by a variety of procedures based on maximum likelihood. The analysis is started by typing 'y' at the input prompt. To quit the program type 'q'.

The following table lists in alphabetical order all TREE-PUZZLE options. Be aware, however, not all of them are accessible at the same time:

Option Description

a - Gamma rate heterogeneity parameter alpha. This is the so-called shape parameter of the Gamma distribution.

b - Type of analysis. Allows to switch between tree reconstruction/analysis by maximum likelihood and *likelihood mapping*.

c - Number of rate categories (4-16) for the discrete Gamma distribution (rate heterogeneity).

d - Data type. Specifies whether nucleotide, amino acid sequences, or two-state data serve as input. The default is automatically set by inspection of the input data. After TREE-PUZZLE has selected an appropriate data type (marked by 'Auto:') the 'd'-option changes the type in the following order: automatically selected type → Nucleotides → Amino acids → automatically selected type.

e - Approximation option. Determines whether an approximate or the exact likelihood function is used to estimate parameters of the models of sequence evolution. The approximate likelihood function is in most cases sufficient and is faster.

f - Base frequencies. The maximum likelihood calculation needs the frequency of each nucleotide (amino acid, doublet) as input. TREE-PUZZLE estimates these values from the sequence input data. This option allows specification of other values.

g - Group sequences in clusters. Allows to define clusters of sequences as needed for the *likelihood mapping* analysis. Only available when *likelihood mapping* is selected ('b' option).

h - Codon positions or definition of doublets. For nucleotide data only. If the TN or HKY model of substitution is used and the number of sites in the alignment is a multiple of three the analysis can be restricted to each of the three codon positions and to the 1st and 2nd positions. If the SH model is used this options allows to specify that the 1st and 2nd codon positions in the alignment define a doublet.

i - Fraction of invariable sites. Probability of a site to be invariable. This parameter can be estimated from the data by TREE-PUZZLE (only if the approximation option for the likelihood function is turned off).

j - List *puzzling step* trees. Writes all intermediate trees (*puzzling step* trees) used to compute the quartet puzzling tree into a file, either as a list of topologies ordered by number of occurrences (*.ptorder), or as list about the chronological occurrence of the topologies (*.pstep), or both.

k - Tree search. Determines how the overall tree is obtained. The topology is either computed with the *quartet puzzling* algorithm or a set of trees is provided by the user. If there are more than two trees in such a set, maximum likelihood branch lengths will be computed for this tree and a number of tests (KH-test, SH-test, and ELW) will be performed on the trees by default. Instead of the evaluation a consensus can be computed for all the trees for which ML branch lengths and ML value are estimated. Alternatively, a maximum likelihood distance matrix only can also be computed (no overall tree).

l - Location of root. Only for computation of clock-like maximum likelihood branch lengths. Allows to specify the branch where the root should be placed in an unrooted tree topology. For example, in the tree (a,b,(c,d)) l = 1 places the root at the branch leading to sequence a whereas l=5 places the root at the internal branch.

m - Model of substitution. The following models are implemented for nucleotides: the general time reversible model (Tavaré, 1986, GTR, e.g.,) model, the Tamura and Nei (TN) model, the Hasegawa *et al.* (HKY) model, and the Schöniger and von Haeseler (SH) model. The SH model describes the evolution of pairs of dependent nucleotides (pairs are the first and the second nucleotide, the third and the fourth nucleotide and so on). It allows for specification of the transition-transversion ratio. The original model (Schöniger and von Haeseler, 1994) is obtained by setting the transition-transversion parameter to 0.5. The Jukes and Cantor (1969), the Felsenstein (1981), and the Kimura (1980) model are all special cases of the HKY model.

For amino acid sequence data the Dayhoff *et al.* (Dayhoff) model, the Jones *et al.* (JTT) model, the Adachi and Hasegawa (mtREV24) model, the Henikoff and Henikoff (BLOSUM 62), the Müller and Vingron (VT), and the Whelan and Goldman (WAG) substitution model are implemented in TREE-PUZZLE. The mtREV24 model describes the evolution of amino acids encoded on mtDNA, and BLOSUM 62 is for distantly related amino acid sequences, as well as the VT model. After TREE-PUZZLE has selected an appropriate amino acid substitution model (marked by 'Auto:') the 'm'-option changes the model in the following order: automatically selected model → Dayhoff → JTT → mtREV24 → BLOSUM62 → VT → WAG → automatically selected model

For more information please read the section in this manual about models of sequence evolution (p. 19). See also option 'w' (model of rate heterogeneity).

n - If tree reconstruction is selected: number of *puzzling steps*. Parameter of the *quartet puzzling* tree search. Generally, the more sequences are used the more *puzzling steps* are advised. The default value varies depending on the number of sequences (at least 1000).

If *likelihood mapping* is selected: number of quartets in a *likelihood mapping* analysis. Equal to the number of dots in the *likelihood mapping* diagram. By default 10000 dots/quartets are assumed. To use all possible quartets in clustered *likelihood mapping* you have to specify a value of n=0.

o - Outgroup. For displaying purposes of the unrooted *quartet puzzling* tree only. The default outgroup is the first sequence of the data set.

p - Constrain the TN model to the F84 model. This option is only available for the Tamura-Nei model. With this option the expected (!) transition-transversion ratio for the F84 model have to be entered and TREE-PUZZLE computes the corresponding parameters of the TN model (this depends on base frequencies of the data). This allows to compare the results of TREE-PUZZLE and the PHYLIP maximum likelihood programs which use the F84 model.

q - Quits analysis.

r - Y/R transition parameter. This option is only available for the TN model. This parameter is the ratio of the rates for pyrimidine transitions and purine transitions. You do not need to specify this parameter as TREE-PUZZLE estimates it from the data. For precise definition please read the section in this manual about models of sequence evolution (p. 19).

s - Symmetrize doublet frequencies. This option is only available for the SH model. With this option the doublet frequencies are symmetrized. For example, the frequencies of "AT" and "TA" are then set to the average of both frequencies.

t - Transition/transversion parameter. For nucleotide data only. You do not need to specify this parameter as TREE-PUZZLE estimates it from the data. The precise definition of this parameter is given in the section on models of sequence evolution in this manual (p. 19).

u - Show unresolved quartets. During the *quartet puzzling* tree search TREE-PUZZLE counts the number of unresolved quartet trees. An unresolved quartet is a quartet where the maximum likelihood values for each of the three possible quartet topologies are so similar that it is not possible to prefer one of them (Strimmer *et al.*, 1997). If this option is selected you will get a detailed list of all star-like quartets. Note, for some data sets there may be a lot of unresolved quartets. In this case a list of all unresolved quartets is probably not very useful and also needs a lot of disk space.

v - Approximate quartet likelihood. For the *quartet puzzling* tree search only. Only for very small data sets it is necessary to compute an exact maximum likelihood. For larger data sets this option should always be turned on.

w - Model of rate heterogeneity. TREE-PUZZLE provides several different models of rate heterogeneity: uniform rate over all sites (rate homogeneity), Gamma distributed rates, two rates (1 invariable + 1 variable), and a mixed model (1 invariable rate + Gamma distributed rates). All necessary parameters can be estimated by TREE-PUZZLE. Note that whenever invariable sites are taken into account the parameter estimation will invoke the 'e' option to use an exact likelihood function. For more detailed information please read the section in this manual about models of sequence evolution (p. 19). See also option 'm' (model of substitution).

x - Selects the methods used in the estimation of the model parameters. Neighbor-joining tree means that a NJ tree is used to estimate the parameters. Quartet sampling means that a number of random sets of four sequences are selected to estimate parameters.

y - Starts analysis.

z - Computation of clock-like maximum likelihood branch lengths. This option also invokes the likelihood ratio clock test.

1 - The A-C rate $R_{AC} = R_{CA}$. For nucleotide data with GTR model only.

2 - The A-G rate $R_{AG} = R_{GA}$. For nucleotide data with GTR model only.

3 - The A-T rate $R_{AT} = R_{TA}$. For nucleotide data with GTR model only.

4 - The C-G rate $R_{CG} = R_{GC}$. For nucleotide data with GTR model only.

5 - The C-T rate $R_{CT} = R_{TC}$. For nucleotide data with GTR model only.

6 - The G-T rate $R_{GT} = R_{TG}$. For nucleotide data with GTR model only.

Chapter 8

Other Features

- For nucleotide data TREE-PUZZLE computes the expected transition/transversion ratio and the expected pyrimidine transition/purine transition ratio corresponding to the selected model. Base frequencies play an important role in the calculation of both numbers.
- TREE-PUZZLE also tests with a 5% level chi-square-test whether the base composition of each sequence is identical to the average base composition of the whole alignment. All sequences with deviating composition are listed in the TREE-PUZZLE report file. It is desired that no sequence (possibly except for the outgroup) has a deviating base composition. Otherwise a basic assumption implicit in the maximum likelihood calculation is violated.
- A hidden feature of TREE-PUZZLE (since version 2.5) is the employment of a weighting scheme of quartets (Strimmer *et al.*, 1997) in the *quartet puzzling* tree search.
- TREE-PUZZLE also computes the average distance between all pairs of sequences (maximum likelihood distances). The average distances can be viewed as a rough measure for the overall sequence divergence.
- If more than one input tree several tests are performed to compare these and infer some kind of 'confidence sets'. The SH test Shimodaira and Hasegawa (1999) is a multi-comparison test taking into account that any tree in the test set could be the maximum likelihood tree. Two types of Kishino and Hasegawa test (KH test) are used to check which trees are significantly worse than the best tree, first, the original two-sided KH test as described in Kishino and Hasegawa (1989) and second, the one-sided KH test (Goldman *et al.*, 2000) using pairwise SH tests to cope with the fact that we test against the best tree. Furthermore expected likelihood weights (ELW, Strimmer and Rambaut 2002) are implemented which infers a narrow 'confidence set' on the tree set. For the interpretation and

applicability of the different tests, the articles cited above have to be read carefully.

- If clock-like maximum-likelihood branch lengths are computed TREE-PUZZLE checks with the help of a likelihood-ratio test (Felsenstein, 1988) whether the data set is clock-like.

- To utilize phylogenetic in columns with gaps, gap characters are interpreted as wildcards (like 'N' in DNA or 'X' in protein sequences. If the percentage of gapped columns is very high it might be reasonable to strip them from the alignment prior to analysis.

- A statistic on the amount of gaps and wildcards is calculated and printed in the puzzle report file and should be examined carefully. If the percentage of those characters is very high certain sequences or even the whole dataset might lack enough overlapping sequence information for proper phylogenetic analysis.

- TREE-PUZZLE also detects sequences that occur more than once in the data and that therefore can be removed from the data set to speed up analysis. Equal sequences are detected via their evolutionary distance, i.e., if their distance equals zero. Since TREE-PUZZLE treats gaps as wildcards, two sequences seem identical if they just differ by a gap or wildcard.

- If rate heterogeneity is taken into account in the analysis TREE-PUZZLE also computes the most probable assignment of rate categories to sequence positions, according to Felsenstein (1996).

- If *quartet puzzling* or *likelihood mapping* analysis is performed the number of fully, partly, and completely unresolved quartets is printed into the puzzle report file for each sequence as well as the whole dataset. These values can help to identify un-informative sequences and whether they or even the whole dataset is suited for for phylogenetic analysis.

Chapter 9

Interpretation and Hints

9.1 Quartet Puzzling Support Values

The *quartet puzzling* (QP) tree search estimates support values for each internal branch. They can be interpreted in much the same way as bootstrap values (though they should not be confused with them). Branches showing a QP reliability from 90% to 100% can be considered very strongly supported. Branches with lower reliability ($> 70\%$) can in principle be also trusted but in this case it is advisable to check how well the respective internal branch does in comparison to other branches in the tree (i.e. check relative reliability). If you are interested in a branch with a low confidence it is also important to check the alternative groupings that are not included in the QP tree (they are listed in the TREE-PUZZLE report file in `*.**` format). There should be a substantial gap between the lowest reliability value of the QP tree and the most frequent grouping that is not included in the QP tree.

9.2 Percentage of Unresolved Quartets

TREE-PUZZLE computes the number and the percentage of resolved, partly resolved, and completely unresolved maximum likelihood quartets for the whole dataset as well as each sequence, i.e. observing all quartets the sequence is part of. A partly resolved or completely unresolved quartet is a quartet where the maximum likelihood values are so similar for two or all three, respectively, of the three possible quartet topologies that it is not possible to prefer only one of them Strimmer *et al.* (1997). The percentage of the unresolved quartets among all possible quartets is an indicator of the suitability of the data for phylogenetic analysis. A high percentage usually results in a highly multifurcating *quartet puzzling* tree. If you only have a few unresolved quartets we recommend to invoke option 'u' to get a list of all these quartets. In a *likelihood mapping* analysis the percentage of completely unresolved quartets is shown in the central region of the triangle diagram.

If the reconstructed tree is by and large unresolved, i.e. it contains many multifurcation, and a single sequences have a high percentage of unresolved quartets, this sequence should be discarded from the dataset, because it might be a source of ambiguity.

9.3 Percentage of Ambiguous Characters in the Alignment

Besides the percentages of the three quartet types (resolved, partly resolved, and unresolved) TREE-PUZZLE also counts the amount of gap and ambiguous characters per sequence and for the full alignment. Ambiguous characters are such that stand for several characters, e.g. 'X' in amino acid and 'N' in nucleotide sequences. Note that TREE-PUZZLE interprets any unknown character as 'X' or 'N' according to the data type.

If single sequences contain a large quantity of gaps and ambiguous characters they might be another source of inaccuracy due to a lack of information as well as a possible lack of common sites with other sequences.

9.4 Automatic Parameter Estimation

TREE-PUZZLE estimates both the parameters of the models of substitution (TN, HKY) and of the model of rate variation (Gamma distribution, fraction of invariable sites) without prior knowledge of an overall tree by a number of different strategies based on maximum likelihood. For all estimated parameters a corresponding standard error (S.E.) is computed. If you have good arguments to choose a different set of parameters than the values obtained by TREE-PUZZLE don't hesitate to use them. If sequences are extremely similar it is very hard for every algorithm to extract information about the model of substitution from the data set. Also, be careful if the estimated parameter values are very close to the internal upper and lower bounds:

Parameter (Symbol)	**Minimum**	**Maximum**
Transition/transversion parameter (t)	0.20	30.00
Y/R transition parameter (γ)	0.10	6.00
Fraction of invariable sites (θ)	0.00	0.99
Gamma rate heterogeneity parameter (α)	0.01	99

It is also possible to use the GTR model (Lanave *et al.*, 1984; Tavaré, 1986; Rodriguez *et al.*, 1990). Unfortunately, there is no automatic estimation procedure (yet) implemented to estimate the six substitution rates. Hence, to use GTR the user has to provide these parameter to the program.

9.5 Batch Mode

Running TREE-PUZZLE from a Unix batch file is straightforward despite the lack of command line switches. Hence you have to pipe the parameters to the standard input of the program. For example, to run TREE-PUZZLE with a the transition/transversion parameter equal to 10 the following lines in a shell script are sufficient:

```
puzzle << EOF
t
10
y
EOF
```

Another possibility is to pipe a parameter file into it, i.e. `puzzle < params` or `cat params | puzzle` where in this example the parameter file `params` contains:

```
t
10
y
```

All other parameters can also be accessed the same way. Note that the `y` parameter is always needed at the end.

Chapter 10

Limits and Error Messages

TREE-PUZZLE has a built-in limit to allow data sets only up to 257 sequences in order to avoid overflow of internal integer variables. At least 32767 sites should be possible depending on the compiler used. Computation time will be the largest constraint even if sufficient computer memory is available. If rate heterogeneity is taken into account every additional category slows down the overall computation by the amount of time needed for one complete run assuming rate homogeneity.

If problems are encountered TREE-PUZZLE terminates program execution and returns a plain text error message. Depending on the severity errors can be classified into three groups:

"HALT" errors: Very severe. You should never ever see one of these messages. If so, please contact the developers!

"Unable to proceed" errors: Harmless but annoying. Mostly problems with the format of the input files and sometimes memory errors (i.e., not enough RAM).

Other errors: Completely uncritical. Occur mostly when options of TREE-PUZZLE are being set.

A standard machine (1996 Unix workstation) with 32 to 64 MB RAM TREE-PUZZLE can easily do maximum likelihood tree searches including estimation of support values for data sets with 50-100 sequences. As *likelihood mapping* is not memory consuming and computationally quite fast it can be applied to large data sets as well.

Chapter 11

Are Quartets Reliable?

Quartets may be intrinsically one of the most difficult phylogenies to resolve accurately (cf. Hillis *et al.*, 1996). It has been asked whether this is a problem for *quartet puzzling* because it works with quartets.

However, this is not true. According to Hillis' findings (Hillis *et al.*, 1996), quartets can be hard, but extra information helps. That is, if all you have are data on species (A, B, C, D) then it might be relatively difficult to find the correct tree for them. But if you have additional data (species E, F, G, ...) and try to find a tree for all the species, then that part of the tree relating (A, B, C, D) will more likely be correct than if you had just the data for (A, B, C, D). In Hillis' big 'model' tree, there are many examples of subsets of 4 species which in themselves might be hard to resolve correctly, but which are correctly resolved thanks to the (...large amount of...) additional data. TREE-PUZZLE (*quartet puzzling*) also gains advantage from extra data in the same way. It's 'understanding' or resolution of the quartet (A, B, C, D) might be incorrect, but the information on the relationships of (A, B, C, D) implicit in its treatment of (A, B, C, E), (A, B, E, D), (A, E, C, D), (E, B, C, D), (A, B, C, F), (A, B, F, D), (A, F, C, D), (F, B, C, D), (A, B, C, G), etc. should overcome this problem.

The facts about how well TREE-PUZZLE actually works have been investigated in the Strimmer and von Haeseler (1996) and Strimmer *et al.* (1997) papers. Their results cannot be altered by Hillis' findings. Considered as a heuristic search for maximum likelihood trees, *quartet puzzling* works very well.

(This section follows N. Goldman, personal communication).

Chapter 12

Other Programs

12.1 Related Links and Programs

Some links related to TREE-PUZZLE:

Puzzle-Server: a web interface to TREE-PUZZLE.
`http://bioweb.pasteur.fr/seqanal/interfaces/Puzzle.html`

PUZZLEBOOT: a program to perform bootstrap analyses with TREE-PUZZLE.
`http://hades.biochem.dal.ca/Rogerlab/Software/software.html`

12.2 Supporting Programs

An (incomplete) list of programs supporting TREE-PUZZLE by providing graphical presentation of results produced by PUZZLE are:

TreeView: (UNIX/Linux, Mac OS X, Windows) a tree viewer
`http://taxonomy.zoology.gla.ac.uk/rod/treeview.html`

TreeTool: (UNIX/Linux) a tree viewer
`ftp://rdp.life.uiuc.edu/pub/RDP/programs/TreeTool/`, a statically linked Linux executable is available from `http://www.tree-pzzle.de`

GhostView: (UNIX/Linux, Mac OS X, Windows) a free PostScript viewer/converter
`http://www.ghostscript.com`

epstopdf: (UNIX/Linux, Mac OS X, Windows) a free encapsulated PostScript (EPS) to PDF converter. The program needs GhostScript (see above) and Perl (`http://www.cpan.org`) installed.
`http://www.ctan.org/tex-archive/support/epstopdf/`

12.3 Other Phylogenetic Programs

There are a number of other very useful and widespread programs to reconstruct phylogenetic relationships and to analyze molecular sequence data that are available free of charge. Here are the URLS of some web pages that provide links to most of them (including the PHYLIP package and the MOLPHY and PAML maximum likelihood programs):

- Joe Felsenstein's list of programs (well-organized and pretty exhaustive): `http://evolution.genetics.washington.edu/phylip/software.html`
- "Tree of Life" software page: `http://phylogeny.arizona.edu/tree/programs/programs.html`
- European Bioinformatics Institute: `http://www.ebi.ac.uk/biocat/biocat.html`

12.4 Compilers and Other Software

A list of available compilers is provided in section 2.2. Links to lists of implementations of the Message Passing Interface (MPI) library for parallel computing are given in section 2.1.6.

Chapter 13

TREE-PUZZLE References and Further Reading

To get more hands-on details on the usage of the program we recommend to refer to

> Schmidt, H.A. and A. von Haeseler (2003) Maximum-Likelihood Analysis Using TREE-PUZZLE. In A.D. Baxevanis, D.B. Davison, R.D.M. Page, G. Stormo, and L. Stein (eds.) *Current Protocols in Bioinformatics*, Unit 6.6, Wiley and Sons, New York. ISBN 0-471-25093-7 ISBN 0-471-25093-7

More information can be gained also from the following:

If you intend to use TREE-PUZZLE in a publication please cite the program as

> Schmidt, H.A., K. Strimmer, M. Vingron, and A. von Haeseler (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*. 18:502-504. (PubMed: 11934758)

for the current TREE-PUZZLE version and

> Strimmer, K., and A. von Haeseler (1996) *Quartet puzzling*: A quartet maximum likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.* 13: 964-969.

for the *quartet puzzling* algorithm.

For *likelihood mapping* please use:

> Strimmer, K., and A. von Haeseler (1997) Likelihood-mapping: A simple method to visualize phylogenetic content of a sequence alignment. *Proc. Natl. Acad. Sci. USA*. 94:6815-6819. (PubMed: 9192648)

For the quartet selection taking into account 2nd and 3rd best quartets:

> Strimmer, K., N. Goldman and A. von Haeseler (1997) Bayesian Probabilities and *Quartet Puzzling*. *Mol. Biol. Evol.* 14:210-213.

For the current parallel version of TREE-PUZZLE please use:

> Schmidt, H.A., K. Strimmer, M. Vingron, and A. von Haeseler (2002) TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*. 18:502-504. (PubMed: 11934758)

More information on the performance of the parallel implementation can be found in

> H.A. Schmidt, E. Petzold, M. Vingron, and A. von Haeseler (2003) Molecular Phylogenetics: Parallelized Parameter Estimation and *Quartet Puzzling*. *J. Parallel Distrib. Comput.*, 63, 719-727. DOI: 10.1016/S0743-7315(03)00129-1

Chapter 14

Acknowledgments and Credits

The maximum likelihood kernel of TREE-PUZZLE is an offspring of the program NucML/ProtML version 2.2 by Jun Adachi and Masami Hasegawa (`ftp://sunmh.ism.ac.jp/pub/molphy`). We thank them for generously allowing us to use the source code of their program. As a scalable random number generator we use source code of SPRNG (Scalable Pseudo Random Number Generator) library (Mascagni and Srinivasan, 2000).

We would also like to thank the European Bioinformatics Institute (EBI), the Institut Pasteur, and the University of Indiana (i.e., Don Gilbert) for kindly distributing the TREE-PUZZLE program.

We especially thank Stephane Bortzmeyer, Ross H. Crozier, Peter Foster for their help in debugging and their valuable remarks and suggestions, Ekkehard Petzold for the parallelizing the parameter estimation part.

We also thank John M. Archibald, Andreas Bernauer, Christian Blouin, Steve Cannon, Ignazio Carbone, Iñaki Comas, Ross H. Crozier, Anthony A. Echelle, Nick Goldman, Matthias Görlach, David Horner, Nicolas Joly, Patrick Joost, Jan Lentfer, Pawel Mackiewicz, Benoit Moury, Aris Parmakelis, Ekkehard Petzold, Herve Philippe, Jean-Pierre Szikora, Atro Tossavainen, Falk Tschierske, Lutz Voigt, Felipe Wettstein, Simin Whelan, Olga Zhaxybayeva, Christian Zmasek (in alphabetical order), and many others for contributions, tips, suggestions, bug reports, and other contributions.

Furthermore we thank for preparing packages and making TREE-PUZZLE available in various formats: Jan Lentfer (FreeBSD), Marc Baudoin (NetBSD), Andreas Tille and Stephane Bortzmeyer (Debian Linux), Luc Ducazu (BioLinux.org).

Finally we thank the Deutsche Forschungsgemeinschaft and the Max-Planck-Society and the von-Neumann-Institute for Computing (NIC), FZ Jülich, for financial support.

Bibliography

Adachi, J. and Hasegawa, M. (1996) Model of amino acid substitution in proteins encoded by mitochondrial DNA. *J. Mol. Evol.*, **42**, 459–468.

Dayhoff, M. O., Schwartz, R. M. and Orcutt, B. C. (1978) A model of evolutionary change in proteins. In Dayhoff, M. O. (ed.), *Atlas of Protein Sequence Structure*, volume 5, pp. 345–352, National Biomedical Research Foundation, Washington DC.

Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, **17**, 368–376.

Felsenstein, J. (1984) Distance methods for inferring phylogenies: A justification. *Evolution*, **38**, 16–24.

Felsenstein, J. (1988) Phylogenies from molecular sequences: Inference and reliability. *Annu. Rev. Genet.*, **22**, 521–565.

Felsenstein, J. (1996) Inferring phylogenies from protein sequences by parsimony, distance, and likelihood methods. *Methods Enzymol.*, **266**, 418–427.

Goldman, N., Anderson, J. P. and Rodrigo, A. G. (2000) Likelihood-based tests of topologies in phylogenetics. *Syst. Biol.*, **49**, 652–670.

Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W. and Snir, M. (1998) *MPI: The Complete Reference - The MPI Extensions*, volume 2. 2nd edition, The MIT Press, Cambridge, Massachusetts.

Gu, X., Fu, Y.-X. and Li, W.-H. (1995) Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Mol. Biol. Evol.*, **12**, 546–557.

Hasegawa, M., Kishino, H. and Yano, T.-A. (1985) Dating of the human–ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, **22**, 160–174.

Henikoff, S. and Henikoff, J. G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

Hillis, D. M., Moritz, C. and Mable, B. K. (eds.) (1996) *Molecular Systematics*. 2nd edition, Sinauer Associates, Sunderland, Massachusetts.

Jones, D. T., Taylor, W. R. and Thornton, J. M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.*, **8**, 275–282.

Jukes, T. H. and Cantor, C. R. (1969) Evolution of protein molecules. In Munro, H. N. (ed.), *Mammalian Protein Metabolism*, volume 3, pp. 21–123, Academic Press, New York.

Kimura, M. (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, **16**, 111–120.

Kishino, H. and Hasegawa, M. (1989) Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. *J. Mol. Evol.*, **29**, 170–179.

Lanave, C., Preparata, G., Saccone, C. and Serio, G. (1984) A new method for calculating evolutionary substitution rates. *J. Mol. Evol.*, **20**, 86–93.

Mascagni, M. and Srinivasan, A. (2000) SPRNG: A scalable library for pseudorandom number generation. *ACM Trans. Math. Software*, **26**, 436–461.

McMorris, F. R. and Neumann, D. A. (1983) Consensus functions defined on trees. *Math. Soc. Sci*, **4**, 131–136.

Müller, T. and Vingron, M. (2000) Modeling amino acid replacement. *J. Comput. Biol.*, **7**, 761–776.

Rodriguez, F., Oliver, J. L., Main, A. and Medina, J. R. (1990) The general stochastic model of nucleotide substitution. *J. theor. Biol.*, **142**, 485–501.

Schmidt, H. A. and von Haeseler, A. (2003) Maximum likelihood analysis using TREE-PUZZLE. In Baxevanis, A. D., Davison, D. B., Page, R. D. M., Stormo, G. and Stein, L. (eds.), *Current Protocols in Bioinformatics*, pp. 6.6.1–6.6.25, Wiley and Sons, New York, USA.

Schöniger, M. and von Haeseler, A. (1994) A stochastic model for the evolution of autocorrelated DNA sequences. *Mol. Phylogenet. Evol.*, **3**, 240–247.

Shimodaira, H. and Hasegawa, M. (1999) Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol. Biol. Evol.*, **16**, 1114–1116.

Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D. W. and Dongarra, J. (1998) *MPI: The Complete Reference - The MPI Core*, volume 1. 2nd edition, The MIT Press, Cambridge, Massachusetts.

Strimmer, K., Goldman, N. and von Haeseler, A. (1997) Bayesian probabilities and quartet puzzling. *Mol. Biol. Evol.*, **14**, 210–213.

Strimmer, K. and von Haeseler, A. (1996) Quartet puzzling: A quartet maximum–likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, **13**, 964–969.

Strimmer, K. and von Haeseler, A. (1997) Likelihood–mapping: A simple method to visualize phylogenetic content of a sequence alignment. *Proc. Natl. Acad. Sci. USA*, **94**, 6815–6819.

Strimmer, K. and Rambaut, A. (2002) Inferring confidence sets of possibly misspecified gene trees. *Proc. R. Soc. Lond. B*, **269**, 137–142.

Tamura, K. and Nei, M. (1993) Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.*, **10**, 512–526.

Tavaré, S. (1986) Some probabilistic and statistical problems on the analysis of DNA sequences. *Lec. Math. Life Sci.*, **17**, 57–86.

Whelan, S. and Goldman, N. (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum likelihood approach. *Mol. Biol. Evol.*, **18**, 691–699.

Chapter 15

Known Bugs

On Alpha based computers *floating point exception* errors have occurred. We hope that we have fixed them all.

For occurrences of *floating point exceptions* or other errors we need information about the operating system to reproduce and debug those errors. The datasets and options used when such errors occurred would be very helpful.

Optimizations and Estimations of model parameters and branch lengths are in general performed sequentially until the increase of quality (i.e., likelihood) is below certain thresholds. Additionally TREE-PUZZLE uses upper and lower bounds for parameters (cf. 9.4) and branch lenghts. These thresholds and bounds are a tradeoff between accuracy and and speed. Hence, the inferred likelihood values might differ from other programs. However, these values should only be 'shifted' compared to other programs, but should never show different results in general like, e.g., in the order of the likelihoods of a user defined set of trees.

If you observe such behavior or other bugs please contact the developers!

Chapter 16

Version History

The TREE-PUZZLE program has first been distributed in 1995 under the name PUZZLE. Since then it has been continually improved. Here is a list of the most important changes.

5.2 The *quartet puzzling step* is now implemented as a fast $O(n^4)$ algorithm. The GTR model is possible to be used for DNA sequences. However, the parameters have to be provided by the user. The threshold of $4 <= n <= 257$ has been removed for tree evaluation, pairwise distance computation, and *likelihood mapping*. The limits are still there for ML quartet based tree reconstruction using *quartet puzzling*. Beside the evaluation of a set of user trees, also a consensus tree can now be constructed from these trees. The random number generation has been changed SPRNG (Mascagni and Srinivasan, 2000, Scalable Parallel Pseudo Random Number Generators Library) which has better performance both for the sequential as well as the parallel code. Furthermore, the parameter estimation has been parallelized in close collaboration with Ekkehard Petzold.

Some bug fixes have been incorporated: The WAG matrix has been fixed with corrected data provided by the authors. A bug that tended to make trees reconstructed less resolved has been rectified. KH tests have been altered to not artificially reject a tree if the variance between it and the best tree is too small. Several minor bug fixes have been applied.

5.1 The TREE-PUZZLE manual is now distributed as a PDF file. Quartet and ambiguous character statistics per sequence/dataset added. One sided Kishino-Hasegawa test, Shimodaira-Hasegawa test, and ELW (expected likelihood weights, Strimmer-Rambaut) added for usertree comparison.“ Prints all congruent splits before the *consensus step*, even if below 50%. Gaps and ambiguous characters statistics per sequence/dataset to the puzzle report. The programm stops if there are sequences with only gaps/wildcards.
Several minor bugs fixed: File input fixed to accept also Mac and Windows file formats/linefeeds. (Fixes the bug that restricted user trees to

be in one line.) root search bug, rate categories output bug, output of infile name fixed, if the not named 'infile', minor clock bug corrected, VT matrix corrected by its authors, `radixsort` renamed to `tp_radixsort` to avoid name clashes on Mac OS X and FreeBSD. Compiler linker flag order fixed. 200% bug fixed. FPE errors fixed on Compaq Alpha machines.

5.0 Puzzle tree reconstruction part is now parallelized using the MPI standard (Message Passing Interface).
Possibility added to specify the names of the input file and the user tree file at the command line. Output files renamed to the form PREFIX.EXTENSION, where PREFIX is the input file name or, if used, the user tree file name. The EXTENSION could be one of the following: `puzzle` (PUZZLE report), `tree` (tree file), `dist` (ML distance file), `eps` (*likelihood mapping* output in eps format), `qlist` (unresolved quartets), `qstep` (*puzzling step* tree IDs as they occur in the analysis), or `qtorder` (sorted unique list of *puzzling step* trees).
The tree likelihood value is added to the treefile as a leading comment ('`[ lh=x.xxx ]`') to the tree string.
VT (variable time) matrix (Müller and Vingron, 2000) and WAG matrix (Whelan and Goldman, 2001) are added to the AA substitution models.
The Data type and AA-model options in the menu now show the automatically set type/model first. These can now be changed by using the 'd' or 'm' key independently from the type/model selected. This makes it possible to select a desired AA substitution model or data type by piping letters to the standard input without knowing PUZZLE's preselection.
Parameters are written to file when estimated before evaluation of the quartets.
The inconsistency with respect to other programs in handling invariable sites has been fixed.
Some minor bug fixes (e.g. the clockbug and another in the optimization routine have been fixed).
Source code organization adopted to the GNU standards (`configure`, `make`, `make install` under UNIX)

4.0.2 Update to provide precompiled Windows 95/98/NT executables. In addition: Internal rearrangement of rate matrices. Improved BLOSUM 62 matrix. Endless input loop for input files restricted to 10 trials. Source code clean up to remove compile time warnings. Explicit quit option in menu. Changes in NJ tree code. Updates of documentation (address changes, correction of errors).

4.0.1 Maintenance release. Correction of mtREV matrix. Fix of the "intree bug". Removal of stringent runtime-compatibility check to allow out-of-the-box compile on Alpha. More accurate gamma distribution allowing 16 instead of 8 categories and ensuring a better alpha > 1.0. Update of documentation (mainly address changes). More Unix-like file layout, and change of license to GPL.

4.0 Executables for Windows 95/NT and OS/2 instead of MS-DOS. Computation of clock-like branch lengths (also for amino acids and for non-binary trees). Automatic likelihood ratio clock test. Model for two-state sequences data (0,1) included. Display of most probable assignment of rates to sites. Identification of groups of identical sequences. Possibility to read multiple input trees. Kishino-Hasegawa test to check whether trees are significantly different. BLOSUM 62 model of amino acid substitution (Henikoff and Henikoff, 1992). Use of parameter alpha instead of $\eta = 1/(1 + \alpha)$ (for rate heterogeneity).

Improvements to user interface. SH model can be applied to 1st and 2nd codon positions. Automatic check for compatible compiler settings. Workaround for severe runtime problem when the gcc compiler was used.

3.1 Much improved user interface to rate heterogeneity (less confusing menu, rearranged outfile, additional out-of-range check). Possibility to read rooted input trees (automatic removal of basal bifurcation). Computation of average distance between all pairs of sequences. Fix of a bug that caused PUZZLE 3.0 to crash on some systems (DEC Alpha). Cosmetic changes in program and documentation.

3.0 Rate heterogeneity included in all models of substitution (Gamma distribution plus invariable sites). *Likelihood mapping* analysis with Postscript output added. Much more sophisticated maximum likelihood parameter estimation for all model parameters including those of rate heterogeneity. Codon positions selectable. Update to mtREV24. New icon. Less verbose runtime messages. HTML documentation. Better internal error classification. More information in outfile (number of constant positions etc.).

2.5.1 Fix of a bug (present only in version 2.5) related to computation of the variance of the maximum likelihood branch lengths that caused occasional crashes of PUZZLE on some systems when applied to data sets containing many very similar sequences. Drop of support for non-FPU Macintosh version. Corrections in manual.

2.5 Improved QP algorithm (Strimmer *et al.*, 1997). Bug fixes in ML engine, computation of ML distances and ML branch lengths, optional input of a user tree, F84 model added, estimation of all TN model parameters and corresponding standard errors, CLUSTAL W treefile convention adopted to allow to show branch lengths and QP support values simultaneously, display of unresolved quartets, update of mtREV matrix, source code more compatible with some almost-ANSI compilers, more safety checks in the code.

2.4 Automatic data type recognition, chi-square-test on base composition, automatic selection of best amino acid model, estimation of transition-transversion parameter, ASCII plot of *quartet puzzling* tree into the outfile.

2.3 More models, many usability improvements, built-in consensus tree routines, more supported systems, bug fixes, no more dependencies of input order. First EBI distributed version.

2.2 Optimized internal data structure requiring much less computer memory. Bug fixes.

2.1 Bug fixes concerning algorithm and transition/transversion parameter.

2.0 Complete revision merging the maximum likelihood and the *quartet puzzling* routines into one user friendly program. First electronic distribution.

1.0 First public release, presented at the 1995 phylogenetic workshop (June 15-17, 1995) at the University of Bielefeld, Germany.