Programming Guidelines and Algorithms

- Programming Precepts and Program Development
- GNU programming tools
- Scientific Libraries
- Parallelisation and Benchmarking
- Algorithms
- Optimisation: Genetic Algorithm
- Optimisaiton: Ant Colony Optimisation

Programming Precepts

(adopted from Kruse, Tondo & Leung)

- Plan your program first, then write it
- Specify preconditions and postconditions of functions
- Document your code
- Give meaningful names to variables and functions
- One instruction per line, one task per function
- Test your program on a variety of input data
- Focus optimisation on the slowest functions
- Keep your algorithms as simple as possible
- Program in haste and debug forever

Programming Style

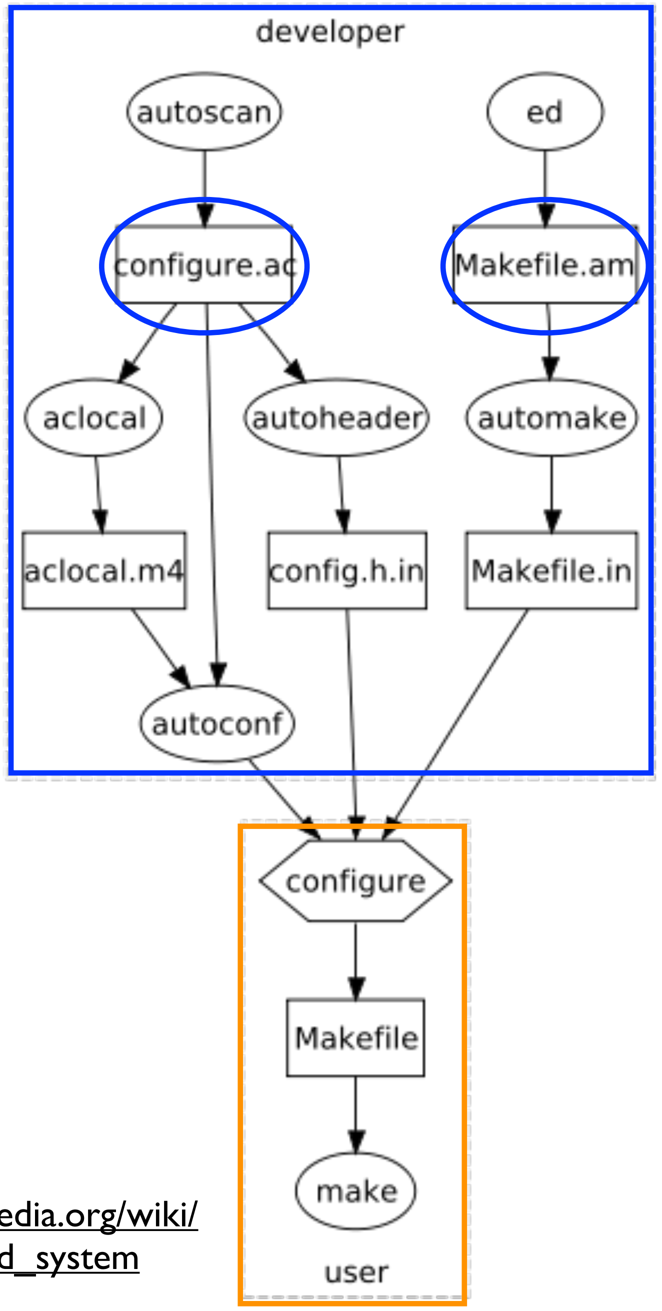
```
int main(args) {
 get_args(args);

 function_A(argA);

 return 0;
}


int function_A(argA) {
 compute_something(argA);

 return result;
}
```

make / Makefile

```
#______________________________________________________________________________
# MACROS
# compilers / flags
CC        = gcc
ARCHFLAG = -m64 # for 64bit version
CFLAGS   = -ggdb -Wall $(ARCHFLAG) # development binary
# GSL and math library
GSL_HOME = /opt/local
INCLIBS = -I$(GSL_HOME)/include
LDLIBS  = -lm -L$(GSL_HOME)/lib -lgsl -lgslcblas
# program / object names
TARGET = pdbencode
OBJS   = getpdb.o kabsch.o matrix.o parse_args.o pdbencode.o putpdb.o putseq.o \
         safe.o structure.o vector.o getdssp.o fragments.o getfragments.o
#____________________________________________________________________________
# TARGETS
# compile source
%.d %.o:    %.c
   $(CC) -c $(CFLAGS) $(INCLIBS) $<
# link objects
$(TARGET): $(OBJS)
   $(CC) -o $(TARGET) $(OBJS) $(LDLIBS) $(ARCHFLAG)
# make targets
all: $(TARGET)

clean:
    rm -f $(OBJS) $(DEPS) $(TARGET)
```

4

GNU Build System (autotools)



```
aclocal
autoheader
automake --add-missing --copy
autoconf
```

```
configure
make
make check
make install
make dist
...
```

http://en.wikipedia.org/wiki/GNU_build_system

5

configure.ac src/Makefile.am

```
AC_PREREQ([2.63])
AC_INIT(myprog, 1.0, me@gmail.com)
AC_CONFIG_SRCDIR([src/arg.c])
AC_CONFIG_HEADERS([src/config.h])
AM_INIT_AUTOMAKE($PACKAGE, $VERSION)
AC_SUBST(INTI_CFLAGS)
AC_SUBST(INTI_LIBS)
# Checks for programs.
AC_PROG_CC
AC_PROG_LN_S
# Checks for libraries.
AC_CHECK_LIB([m], [cos])
AC_CHECK_LIB([gslcblas],[cblas_dgemm])
AC_CHECK_LIB([gsl],[gsl_blas_dgemm])
# Checks for header files.
AC_FUNC_ALLOCA
AC_CHECK_HEADERS([limits.h stdlib.h string.h])
# Checks for typedefs and structures
AC_TYPE_SIZE_T
# Checks for library functions.
AC_FUNC_MALLOC
AC_CHECK_FUNCS([pow sqrt])
AC_CONFIG_FILES([Makefile]
                [src/Makefile]
                [tests/Makefile])
```

```
bin_PROGRAMS = myprog

AM_CPPFLAGS = $(INTI_CFLAGS)

myprog_SOURCES  = \
  align_3D.c align_3D.h getpdb.c getpdb.h \
  kabsch.c kabsch.h matrix.c matrix.h

myprog_LDADD = $(INTI_LIBS)

EXTRA_DIST = doxygen.cfg

CLEANFILES = $(TARGET) *.o
DISTCLEANFILES = \
  libtool config.cache config.log
```

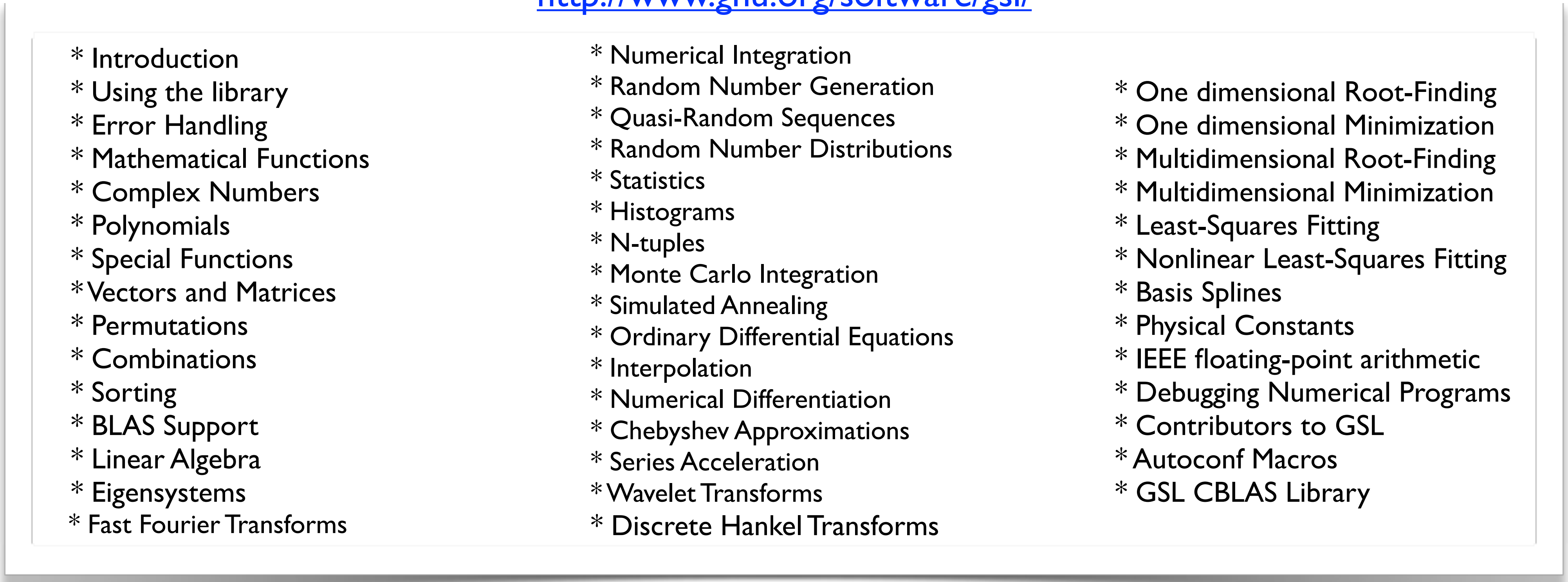
Profiling (gprof)

Compile with flag '-pg'. Run program, creates 'gmon'. Run 'gprof' to get information about performance.

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 33.34      0.02     0.02     7208     0.00     0.00  open
 16.67      0.03     0.01      244     0.04     0.12  offtime
 16.67      0.04     0.01        8     1.25     1.25  memccpy
 16.67      0.05     0.01        7     1.43     1.43  write
 16.67      0.06     0.01                             mcount
  0.00      0.06     0.00      236     0.00     0.00  tzset
  0.00      0.06     0.00      192     0.00     0.00  tolower
  0.00      0.06     0.00       47     0.00     0.00  strlen
  0.00      0.06     0.00       45     0.00     0.00  strchr
  0.00      0.06     0.00        1     0.00    50.00  main
  0.00      0.06     0.00        1     0.00     0.00  memcpy
  0.00      0.06     0.00        1     0.00    10.11  print
  0.00      0.06     0.00        1     0.00     0.00  profil
  0.00      0.06     0.00        1     0.00    50.00  report
...
```

GNU Scientific Library (C)

http://www.gnu.org/software/gsl/



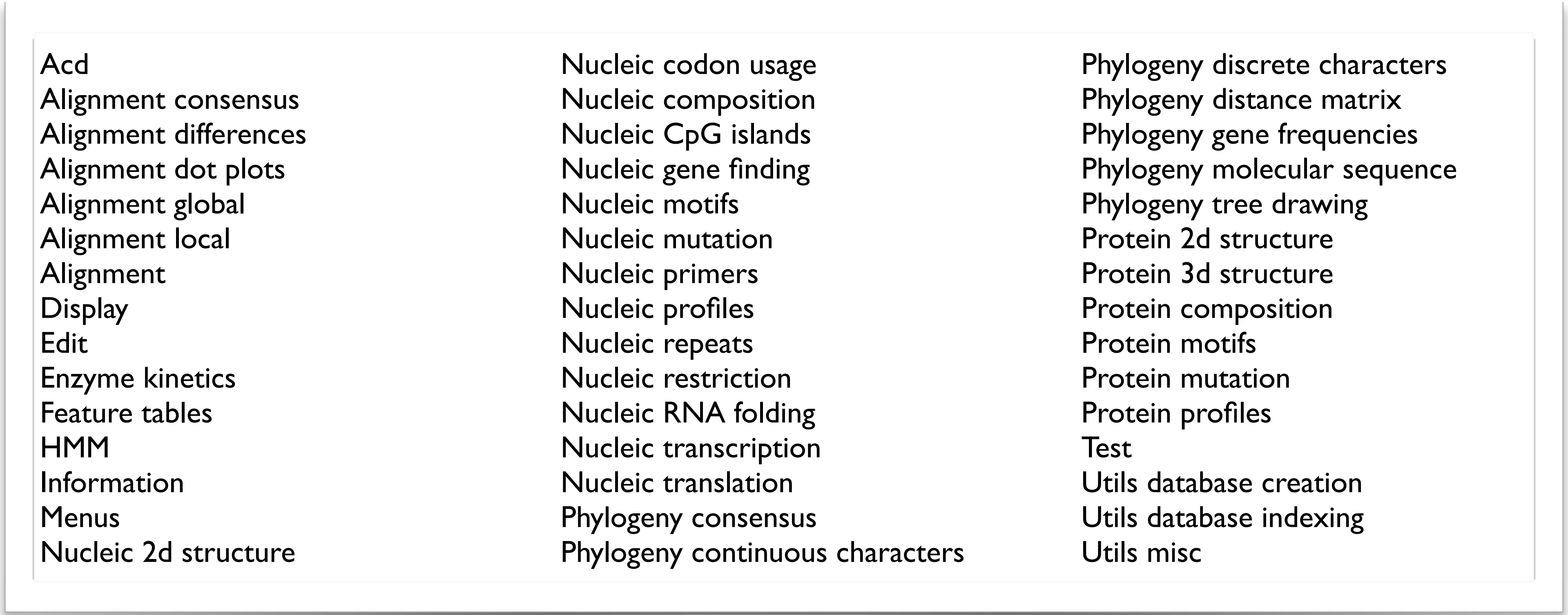
* Introduction
* Using the library
* Error Handling
* Mathematical Functions
* Complex Numbers
* Polynomials
* Special Functions
* Vectors and Matrices
* Permutations
* Combinations
* Sorting
* BLAS Support
* Linear Algebra
* Eigensystems
* Fast Fourier Transforms
* Numerical Integration
* Random Number Generation
* Quasi-Random Sequences
* Random Number Distributions
* Statistics
* Histograms
* N-tuples
* Monte Carlo Integration
* Simulated Annealing
* Ordinary Differential Equations
* Interpolation
* Numerical Differentiation
* Chebyshev Approximations
* Series Acceleration
* Wavelet Transforms
* Discrete Hankel Transforms
* One dimensional Root-Finding
* One dimensional Minimization
* Multidimensional Root-Finding
* Multidimensional Minimization
* Least-Squares Fitting
* Nonlinear Least-Squares Fitting
* Basis Splines
* Physical Constants
* IEEE floating-point arithmetic
* Debugging Numerical Programs
* Contributors to GSL
* Autoconf Macros
* GSL CBLAS Library

```c
#include <gsl/gsl_vector_double.h>
#include <gsl/gsl_matrix_double.h>

gsl_vector *eval = gsl_vector_alloc(3);
gsl_matrix *R = gsl_matrix_alloc(3,3);

gsl_vector_set_zero(eval);
gsl_matrix_set_zero(R);

gsl_vector_free(eval);
gsl_matrix_free(R);
```

EMBOSS (C)

http://emboss.sourceforge.net/

EMBOSS is "The European Molecular Biology Open Software Suite". EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community.



Acd	Nucleic codon usage	Phylogeny discrete characters
Alignment consensus	Nucleic composition	Phylogeny distance matrix
Alignment differences	Nucleic CpG islands	Phylogeny gene frequencies
Alignment dot plots	Nucleic gene finding	Phylogeny molecular sequence
Alignment global	Nucleic motifs	Phylogeny tree drawing
Alignment local	Nucleic mutation	Protein 2d structure
Alignment	Nucleic primers	Protein 3d structure
Display	Nucleic profiles	Protein composition
Edit	Nucleic repeats	Protein motifs
Enzyme kinetics	Nucleic restriction	Protein mutation
Feature tables	Nucleic RNA folding	Protein profiles
HMM	Nucleic transcription	Test
Information	Nucleic translation	Utils database creation
Menus	Phylogeny consensus	Utils database indexing
Nucleic 2d structure	Phylogeny continuous characters	Utils misc

9

BIOPERL

http://www.bioperl.org/wiki/Main_Page

Pages in category "Modules"

The following 200 pages are in this category, out of 1,452 total.

* Module:Bio::Tools::Run::StandAloneBlast
* Module:Bio::Tools::Run::RemoteBlast
* Module:Bio::SearchIO
* Module:Bio::Search::HSP::HSPI
* Module:Bio::Search::Result::ResultI
* Module:Bio::AlignIO
* Module:Bio::SimpleAlign
* Module:Bio::TreeIO
* Module:Bio::Search::Hit::HitI
* Module:Bio::SearchIO::blast
* Module:Bio::SearchIO::blasttable
* Module:Bio::SearchIO::blastxml
* Module:Bio::SearchIO::fasta
* Module:Bio::SearchIO::hmmer
* Module:Bio::SearchIO::wise
* Module:Bio::SearchIO::sim4
* Module:Bio::SearchIO::psl
* Module:Bio::SearchIO::axt
* Module:Bio::SearchIO::waba
* Module:Bio::TreeIO::newick
* Module:Bio::TreeIO::nexus
* Module:Bio::TreeIO::nhx
* Module:Bio::TreeIO::lintree
* Module:Bio::TreeIO::treecluster
* Module:Bio::TreeIO::pag
* Module:Bio::TreeIO::tab
* Module:Bio::TreeIO::svggraph
* Module:Bio::Search::HSP::GenericHSP
* Module:Bio::Search::HSP::FastaHSP
* Module:Bio::Search::HSP::BlastHSP
* Module:Bio::Search::HSP::HMMERHSP
* Module:Bio::Search::HSP::PSLHSP
* Module:Bio::Search::HSP::WABAHSP
* Module:Bio::Search::HSP::PsiBlastHSP
* Module:Bio::Tools::Tmhmm
* Module:Bio::Tools::Run::Alignment::StandAloneFasta
* Module:Bio::Tools::Run::TribeMCL
* Module:Bio::Tools::Run::Alignment::Blat
* Module:Bio::Tools::Run::Phylo::PAML::Baseml
* Module:Bio::Tools::Run::Phylo::PAML::Codeml
* Module:Bio::Tools::Run::Phylo::PAML::Yn00
* Module:Bio::Tools::Run::Phylo::PAML::Evolver
* Module:Bio::Tools::Phylo::PAML
* Module:Bio::PrimarySeqI

...

BIOPYTHON

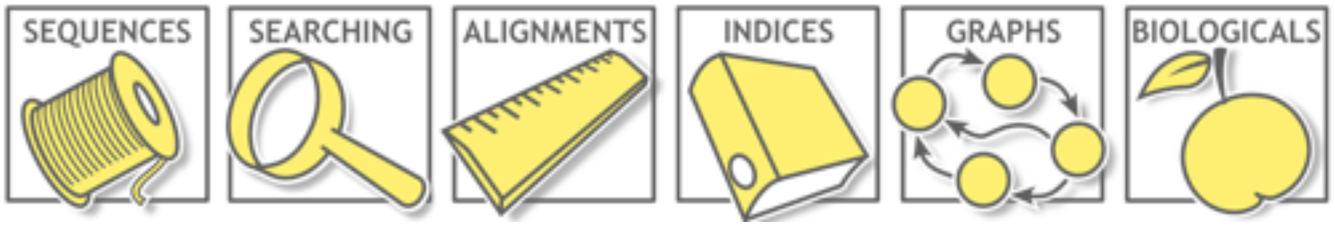
http://biopython.org/wiki/Main_Page

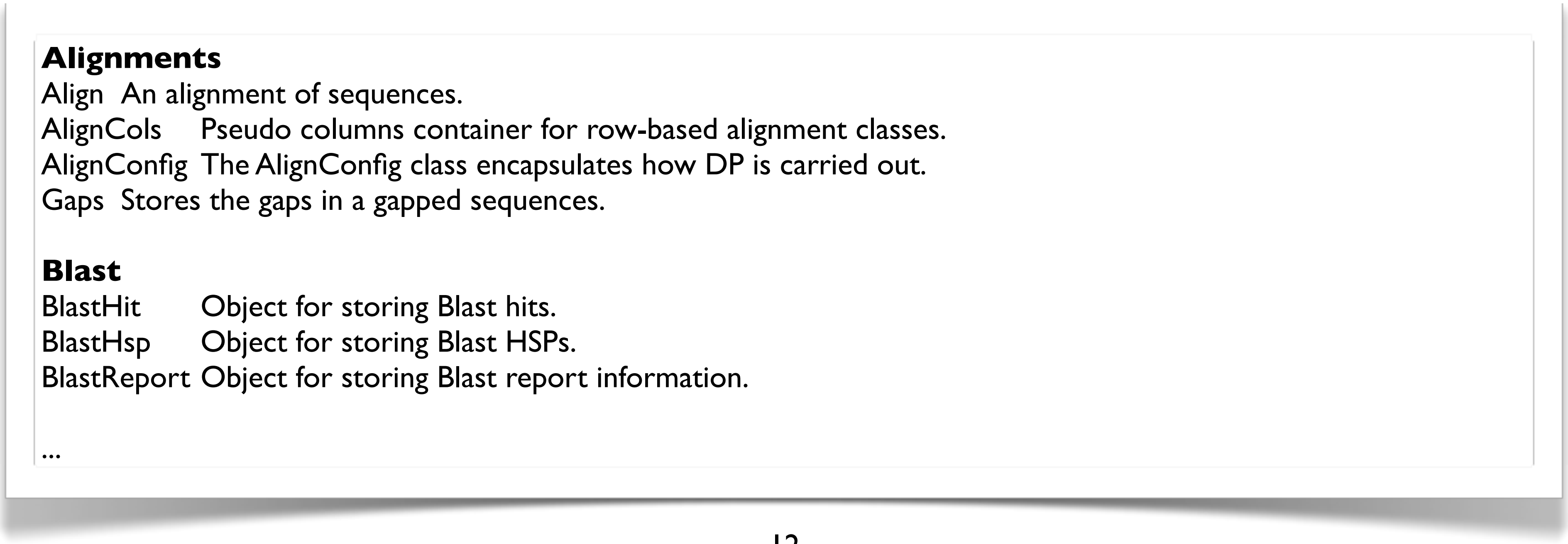
* Seq and SeqRecord objects
* Bio.SeqIO - sequence input/output
* Bio.AlignIO - alignment input/output
* Bio.PopGen - population genetics
* Biopython's BioSQL interface

SEQAN

http://www.seqan.de/

SeqAn is an open source C++ library of efficient algorithms and data structures for the analysis of sequences with the focus on biological data.





Alignments

Align	An alignment of sequences.
AlignCols	Pseudo columns container for row-based alignment classes.
AlignConfig	The AlignConfig class encapsulates how DP is carried out.
Gaps	Stores the gaps in a gapped sequences.

Blast

BlastHit	Object for storing Blast hits.
BlastHsp	Object for storing Blast HSPs.
BlastReport	Object for storing Blast report information.

...

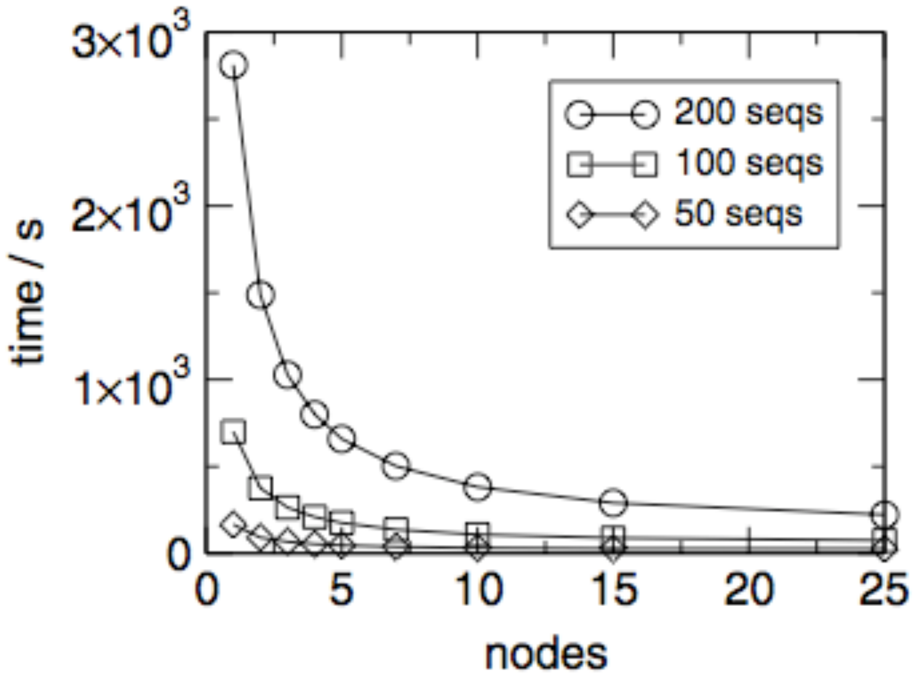
Parallelisation

Coarse grained

- use many computers (cluster)
- use multi-core for different jobs

Fine grained

- Thread (fork)
- Message Passing Interface
  - initialise
  - distribute jobs
  - communicate results
  - assign results
  - terminate

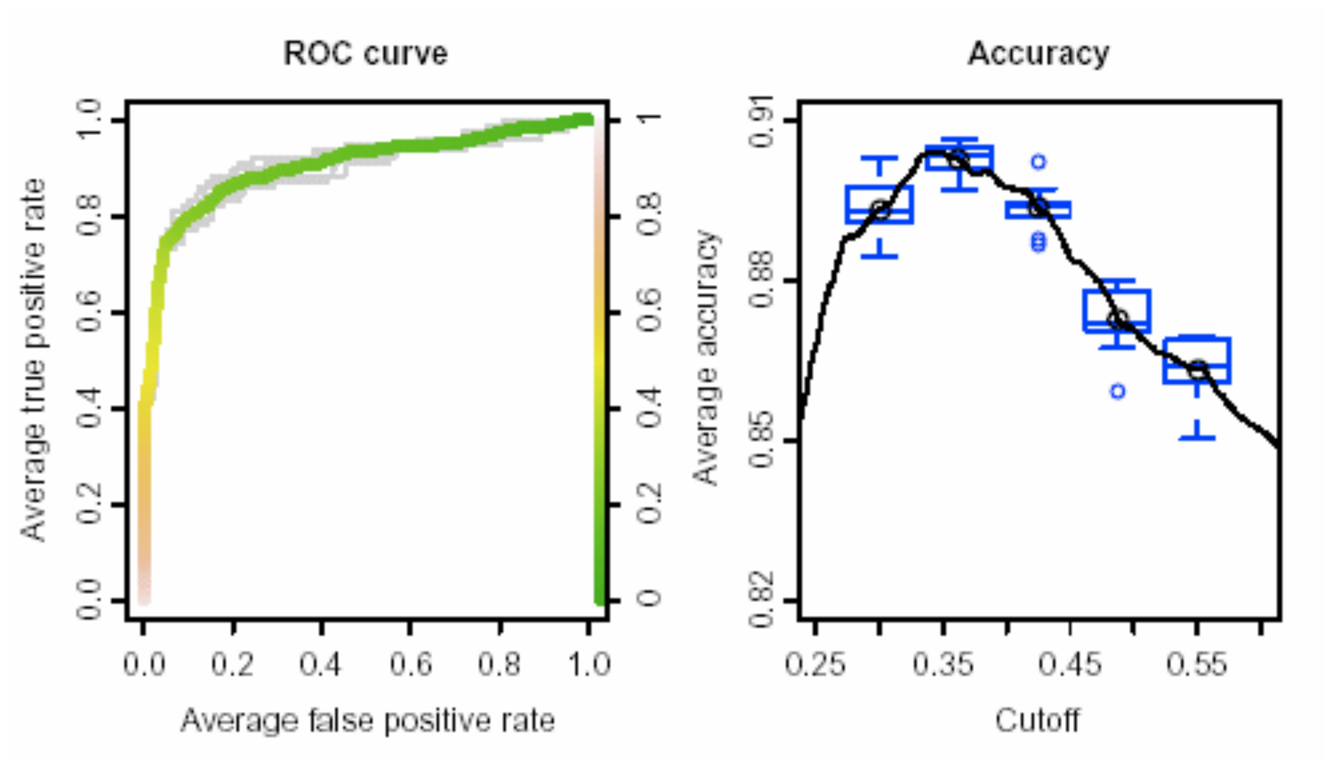


MPI

```
#include <stdio.h>
#include <mpi.h>

int main (int argc, char* argv[])
{
    int rank, size;

    MPI_Init (&argc, &argv);
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
    MPI_Comm_size (MPI_COMM_WORLD, &size);

    printf( "Hello world from process %d of %d\n", rank, size );

    MPI_Finalize();

    return 0;
}
```

```
Hello world from process 0 of 3
Hello world from process 1 of 3
Hello world from process 3 of 3
```

Benchmarking with ROCR



visualizing classifier performance in R, with only 3 commands

http://rocr.bioinf.mpi-sb.mpg.de/

Algorithms

Definition of 'algorithm'

A set of instructions to accomplish a task.

Time Complexity

Upper bound of run time dependent on problem size N:
O(log N) [logarithmic], O(N) [linear], O(N^2) [square], ...

Memory versus CPU

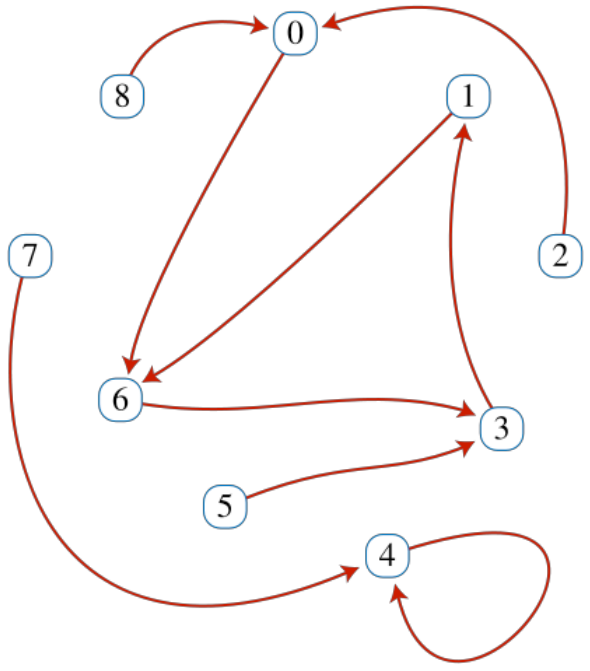
Memory can be saved at the cost of more CPU time and vice versa.
Example: Storing or computing the traceback in Dynamic Programming.

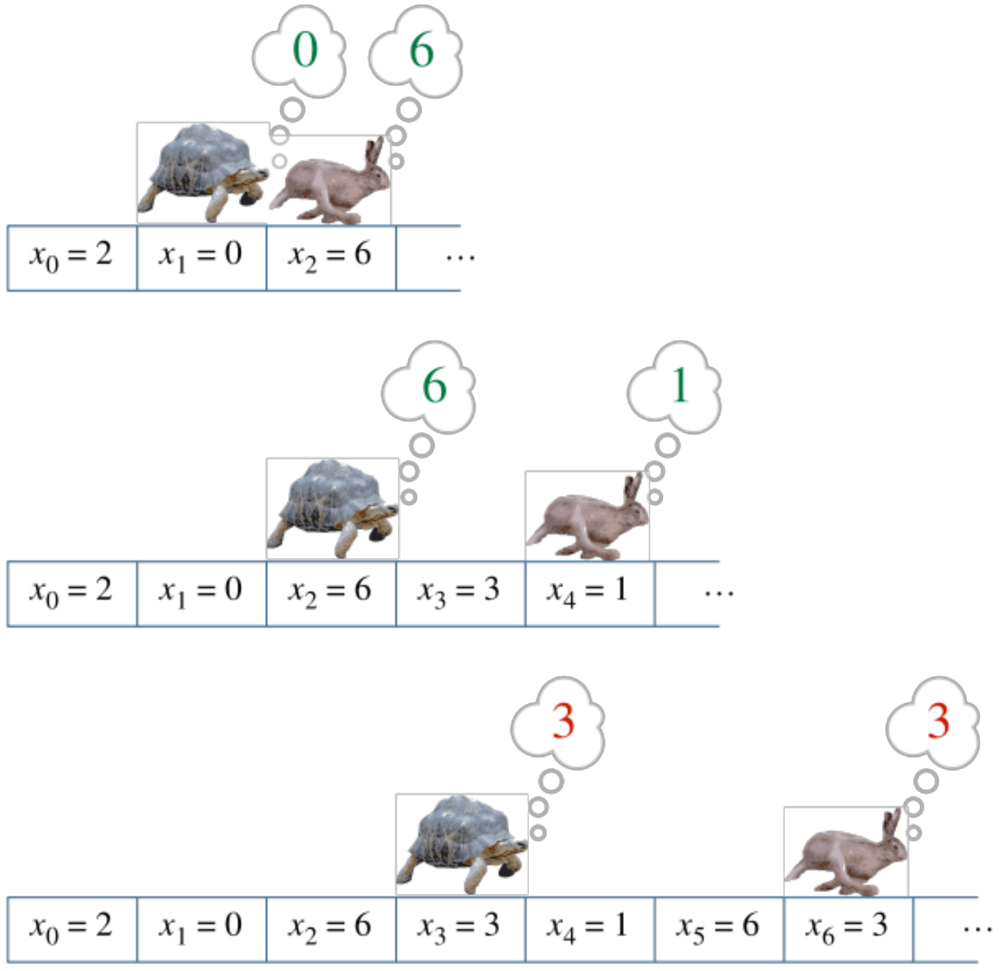
Pointers in Algorithms

Pointers are very efficient in algorithms (see next slide).

A Pointer Algorithm

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0

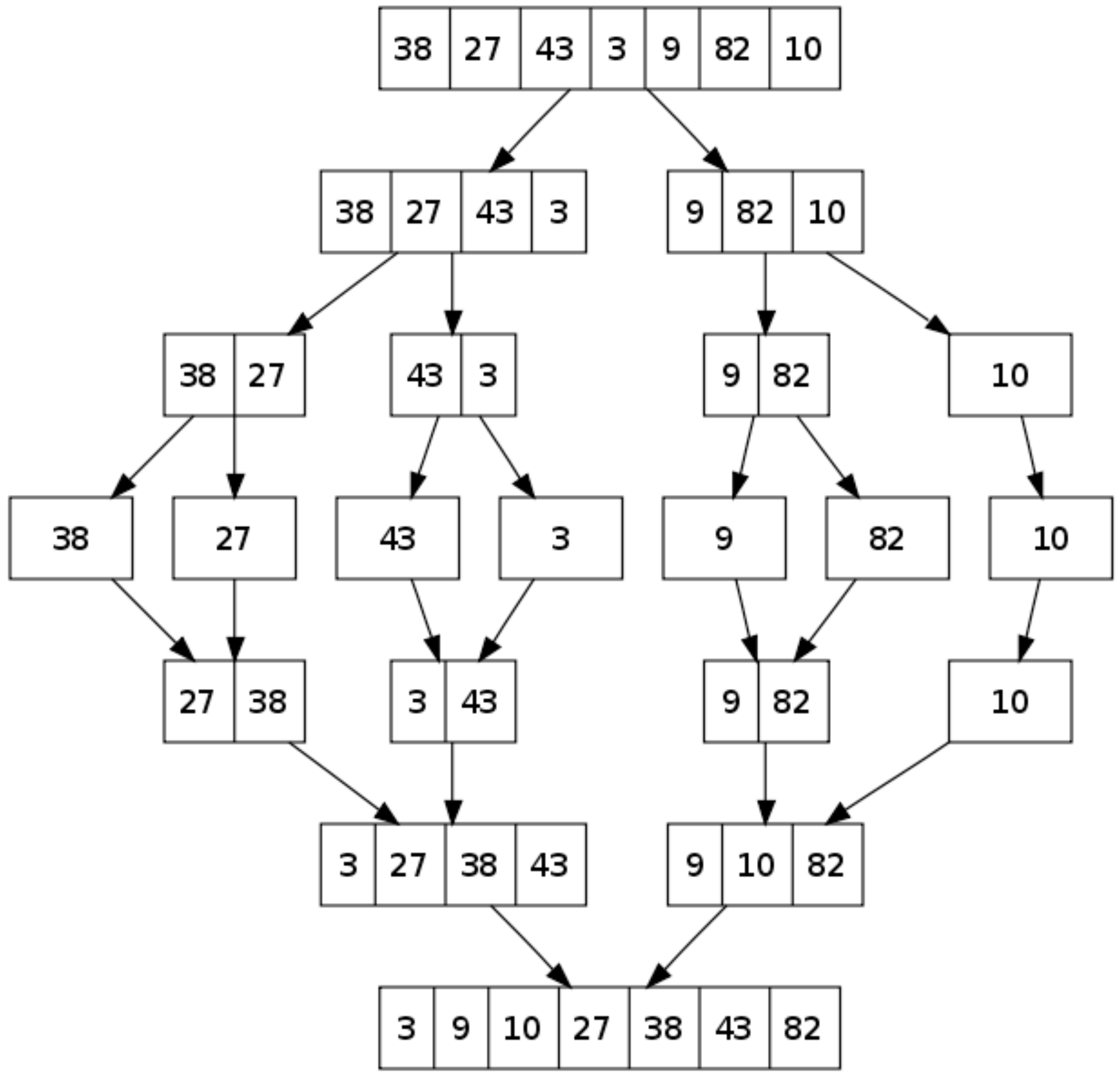




17

Merge Sort

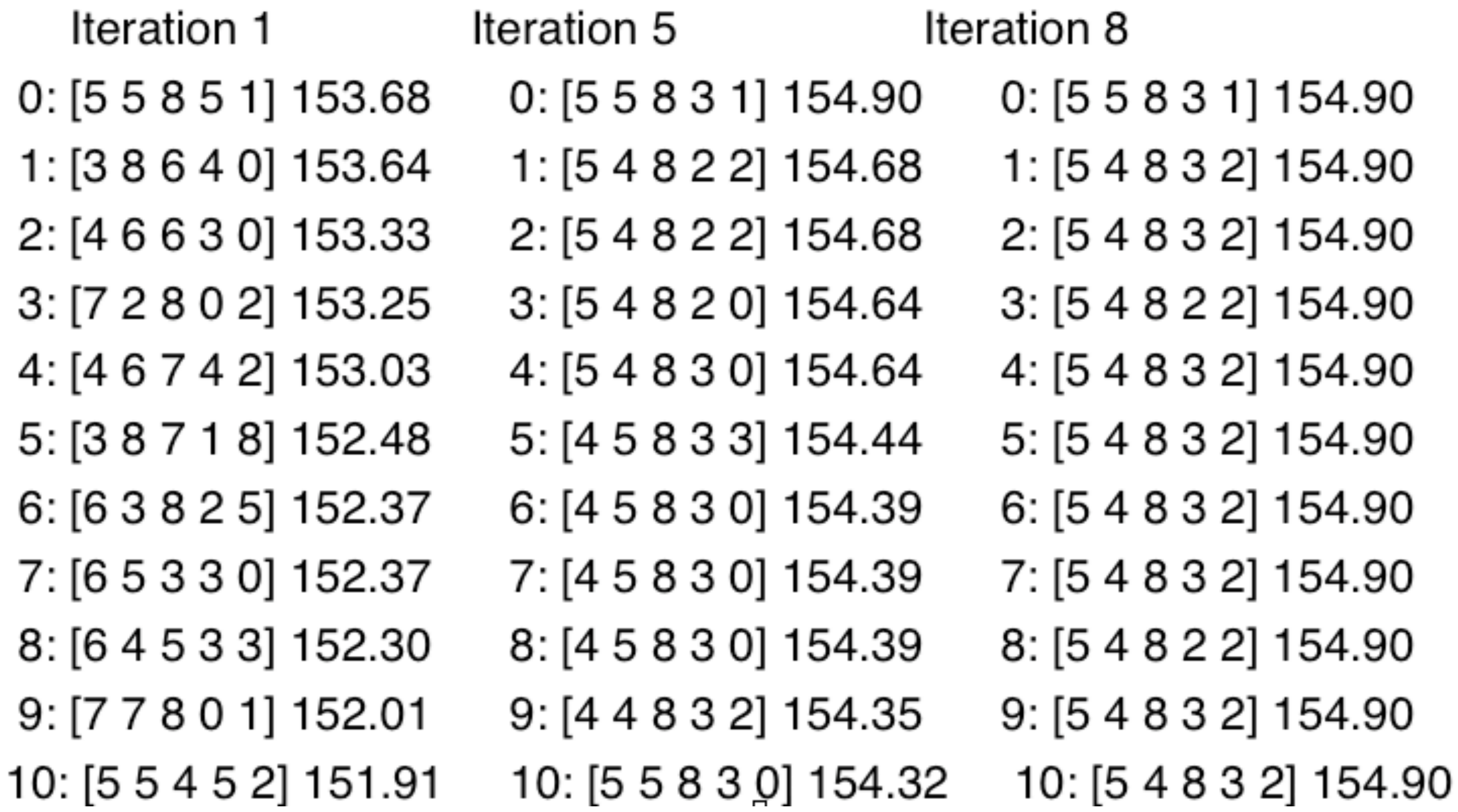
Complexity O(n log n), stable



18

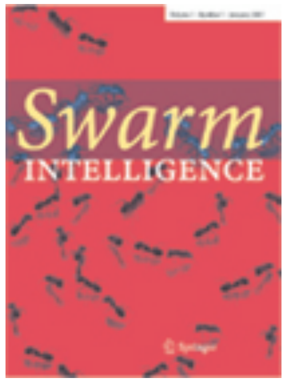
Genetic Algorithm

Idea: Encode parameters as ‘genes’, let many genomes compete, select the best ones and breed new generation. The result is a near-optimal parameter combination.



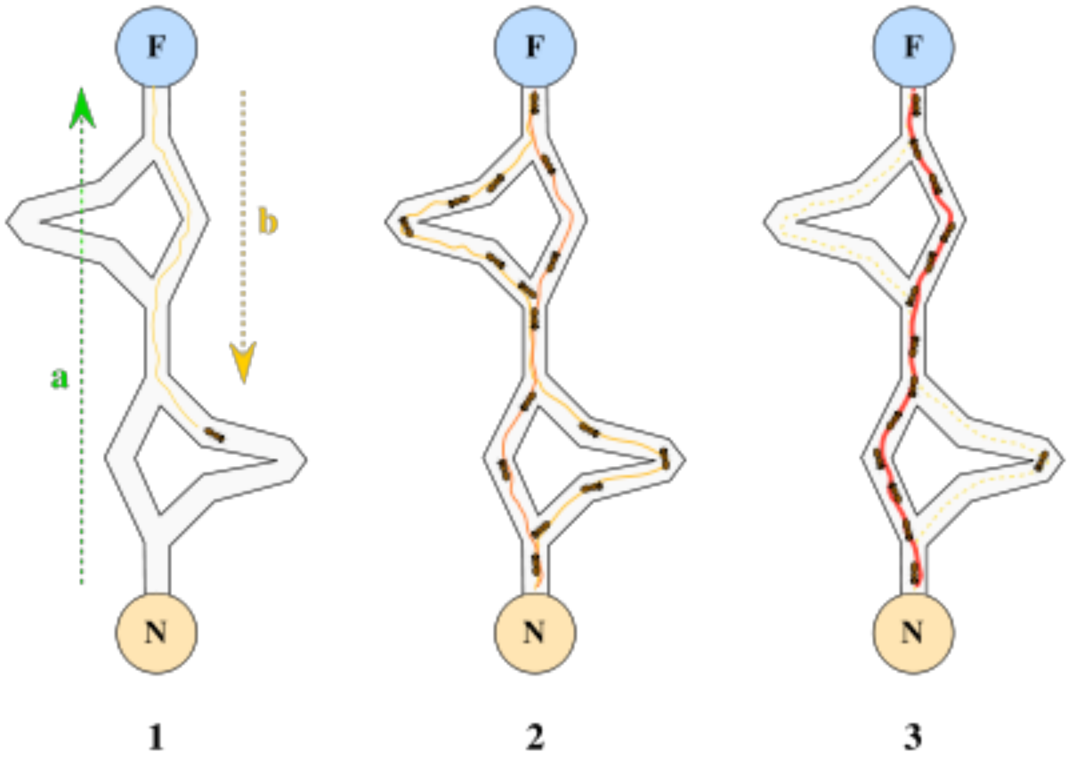
Iteration 1	Iteration 5	Iteration 8
0: [5 5 8 5 1] 153.68	0: [5 5 8 3 1] 154.90	0: [5 5 8 3 1] 154.90
1: [3 8 6 4 0] 153.64	1: [5 4 8 2 2] 154.68	1: [5 4 8 3 2] 154.90
2: [4 6 6 3 0] 153.33	2: [5 4 8 2 2] 154.68	2: [5 4 8 3 2] 154.90
3: [7 2 8 0 2] 153.25	3: [5 4 8 2 0] 154.64	3: [5 4 8 2 2] 154.90
4: [4 6 7 4 2] 153.03	4: [5 4 8 3 0] 154.64	4: [5 4 8 3 2] 154.90
5: [3 8 7 1 8] 152.48	5: [4 5 8 3 3] 154.44	5: [5 4 8 3 2] 154.90
6: [6 3 8 2 5] 152.37	6: [4 5 8 3 0] 154.39	6: [5 4 8 3 2] 154.90
7: [6 5 3 3 0] 152.37	7: [4 5 8 3 0] 154.39	7: [5 4 8 3 2] 154.90
8: [6 4 5 3 3] 152.30	8: [4 5 8 3 0] 154.39	8: [5 4 8 2 2] 154.90
9: [7 7 8 0 1] 152.01	9: [4 4 8 3 2] 154.35	9: [5 4 8 3 2] 154.90
10: [5 5 4 5 2] 151.91	10: [5 5 8 3 0] 154.32	10: [5 4 8 3 2] 154.90

Ant Colony Optimisation



```
Set parameters, initialize pheromone trails
SCHEDULE_ACTIVITIES
    ConstructAntSolutions
    DaemonActions    {optional}
    UpdatePheromones
END_SCHEDULE_ACTIVITIES
```

Ant Colony Optimisation



Learning Outcomes

- Program Design
- Profiling
- Usage of Scientific Libraries
- Parallelisation with MPI
- Benchmarking with ROCR
- Sorting
- Concept of Genetic Algorithm
- Concept of Ant Colony Optimisation