HPC User Experience in Genetic Algorithms and Bio-IT

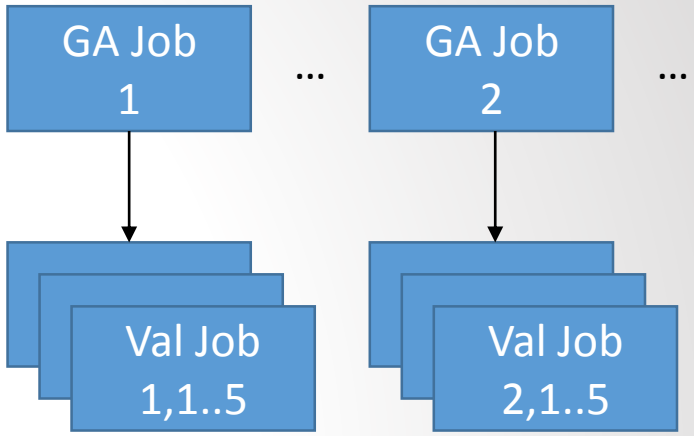
UL HPC School, Day 2 (June 26th, 2015)

By Sune S. Nielsen

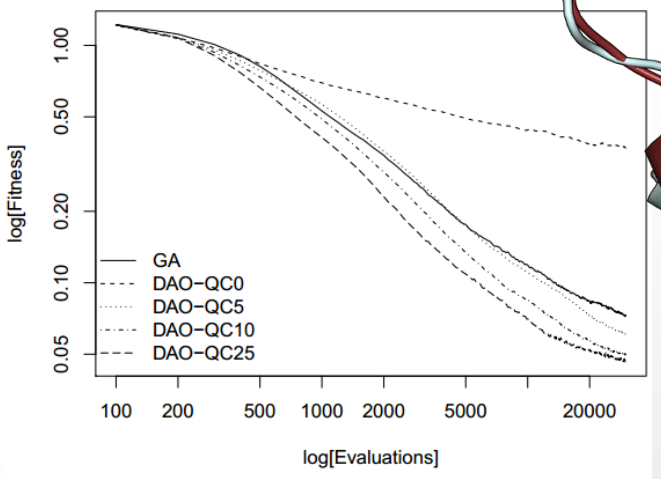
Outline

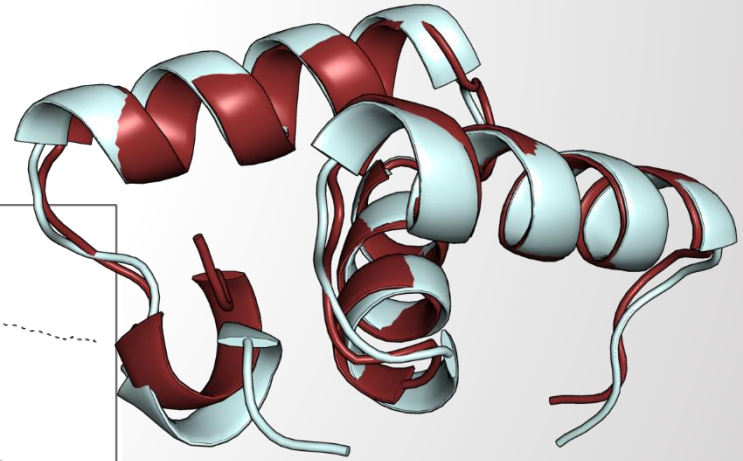
- Overview of workflow
- Scheduling commands
- Gather and present results

Workflow



- Genetic algorithm (GA)
  - Optimisation of bioinformatics objective functions
  - 4-8 hours execution time
  - 30 runs per setup (~5) and problem instance (~2)
- Post validation (Val)
  - Fold prediction with I-TASSER
  - 18-26 hours execution time
  - On 5 best solutions of each 30 runs





Scheduling commands

```
<xml>
  <configs>
    <config>
      <name repeat='1'>TS_1B3A_B…</name>
      <oarsub>-l nodes=1/cpu=1/core=4,walltime=48:00:00</oarsub>
      <commands>
        <command>…/run_tasser_lga.sh 1B3A:B …/data/1B3A_B.pdb </c
      </commands>
    </config>
  </configs>
</xml>
```

- Command files
  - Contain all information and parameters needed to launch job with OAR
- Command launching script
  - Script scans structure for matching directories and command files
  - Can run periodically in a screen session with watch
    - watch -n 300 -d=cumulative "python runXmlCommands.py -d .*QC25.* -p .*_30000_[0\|1\|2\|3\|4]\.cmd\.xml$ -s | tail“
    - If a command is either launched or finished it will be ignored
    - If 30 jobs are registered with OAR the script exits
    - If all requested commands have finished the script exits
    - A test switch (-t) will output all the OAR commands for sanity checks

Gather GA Results

- Output of GA run in single xml file:
- Reading with R

```
<xml>
<xopt.DefaultStatsLogger>
  <experiment>
    <settings class="prot.ProteinExperimentSettings">
      <evaluationLimit>30000.0</evaluationLimit>
      <defaultPopulationSize>100</defaultPopulationSize>
      <problemFriendlyName>1b3a_67</problemFriendlyName>
      <coevolution>false</coevolution>
…
<pop_status id='p50_0' gen='10' evals='500.0' div='74.233'>
<obj i='0' avg='-0.325906' min='-0.364654' max='-0.265129' />
<obj i='1' avg='15.848687' min='8.030303' max='23.383838' />
</pop_status>
…
```

```
xmlSettingsAttributes <- c('//settings/problemFriendlyName', '//settings/defaultPopulationSize',…)
xmlIteratorSubPaths <- c('@gen@evals@div', 'obj[1]@avg@min@max', 'obj[2]@avg@min@max')

xmlIteratorPath = '//pop_status'             # to get all intermediate values
xmlIteratorPath = '//pop_status[last()]'     # to get only the last

# magic function call

aggregatedConvergenceData <- convergenceDataTable[,
list(div = mean(div), avg_avg_f1=mean(avg), avg_best_f1=mean(min), min_avg_f1=min(avg), min_best_f1=min(min)),
by=list(problemFriendlyName, defaultPopulationSize, coevolution, islands, diversityAsObjective, quantileConstraint,
removeDoubles, averageNotMin, evals)]
```

Plot GA Results

- Configure what values to plot

```
experimentSettings = NULL
experimentSettings = rbind(experimentSettings, data.frame(list(
problemFriendlyName="1B3A_67", defaultPopulationSize=50, coevolution=FALSE, islands=0, diversityAsObjective=FALSE,
quantileConstraint=50, removeDoubles=TRUE, averageNotMin=FALSE, experimentNum = 1, seriesLabel = 'GA',
plotVals='evals,avg_avg_f1', plotValLabels='log[Evaluations],log[Fitness]', seriesLog='x', experimentTitle='1B3A
average fitness'
)))

experimentSettings = rbind(experimentSettings, data.frame(list(
problemFriendlyName="1B3A_67", defaultPopulationSize=100, coevolution=TRUE, islands=2, diversityAsObjective=TRUE,
quantileConstraint=10, removeDoubles=TRUE, averageNotMin=TRUE, experimentNum = 1, seriesLabel = 'GA-QC10avg',
plotVals=',', plotValLabels=',', seriesLog='', experimentTitle=''
)))

experimentSettings = rbind(experimentSettings, data.frame(list(
problemFriendlyName="1B3A_67", defaultPopulationSize=100, coevolution=TRUE, islands=5, diversityAsObjective=TRUE,
quantileConstraint=50, removeDoubles=TRUE, averageNotMin=TRUE, experimentNum = 1, seriesLabel = 'GA-QC50avg',
plotVals=',', plotValLabels=',', seriesLog='', experimentTitle=''
)))

# magic plotting code goes here
```

Gather Validation Results

- One .txt file for each of 150 runs
  - egrep -n -r 'PDB.*' tasser_output/*QC25*.txt > ~/tasser_full_qc25_c.txt
  - Excel?! :X

```
Sequence,Protein,Setting,idv#,model#,LGA_N,LGA_RMSD_N5N,LGA_RMSD_N5,LGA_GDT_TS,LGA_S3,LGA_Q
,TM_SCORE,TM_GDT_TS,TM_RMSD_N5N,TM_RMSD_N5,TM_RMSD,TM_ALIGN,H_MATCH,E_MATCH,L_MATCH,MODEL_S
EC,rmsd,Energy,ProfMatch,Identity,FeaturePenalty,b30,MatchScore

KESDPAELQKIMQILQQVHGTPEAIQEIAERNGMPIKEARQLCTEIMANGNTDPEELCK,1OAI_A,MDAO_QC25,0,1,59,35,2.62
,45.339,44.186,1.288,0.39,0.4534,32,2.29,9.784,0.203,82.93,0,83.33,LLHHHHHHHHHHHHHHHLLLLLL
HHHHHHHLLLLHHHHHHHHHHHHHHLLLLLLLLLL,0.148,-904.514,0.038136,10,98,46,0.292308

LQDSPEELAKRTTQWLQQGITESELQEWMEMMGVPEEQARRLLEEIAQNGTPTPEELCK,1OAI_A,MDAO_QC25,4,1,59,30,2.71
,41.525,39.153,1.069,0.3443,0.4068,27,2.29,9.014,0.136,95.12,0,83.33,LLLLLLHHHHHHHHHHHHLLLH
HHHHHHHHHHLLLHHHHHHHHHHHHHHLLLLHHHHHL,0.294,-845.634,0.037288,8,100,39,0.338462
```

  - Quickly back to R:

```
tasserData = read.csv(paste(searchPath, "tasser_final_formatted.csv", sep = '/'), header = TRUE)

tasserDataTable <- data.table(tasserData)

tasserAggregated <- tasserDataTable[, list(tm_score_best = max(TM_SCORE)), by=list(Sequence, Protein,  Setting)]

tasserMaxDataTable <- tasserDataTable[ , .SD[which.max(TM_SCORE)], by=Sequence]
tasserMaxDataTable <- tasserMaxDataTable[LGA_N > 0]
```

Questions?

- sune.nielsen@uni.lu