Bayesian optimization for demographic history inference

Summer project report

Ilia Sheshukov

Project goals

The project goal was to replace a BFGS optimization algorithm used in the tool moments with the Gaussian process based global Bayesian optimization and to study the effects. We aimed to come up with a way to get more accurate results of optimization in less time.

What was done over the course of project

- Tested 5 datasets with different parameters
- Automated report generation
- Added multidimensional plotting capabilities
- Added time constraints on optimization

Methods

We used `GPyOpt` library that, given function and bounding conditions optimizes said function, then we took the optimization routine in `moments` and replaced their algorithm with the calls to GPyOpt and some other of our changes to make them work together.

Methods

`GPyOpt` at its core uses *Bayesian optimization* which is the method of global optimization that does not use derivatives and is generally used when it is expensive to compute function at a point.

Datasets

We have used the following data sets:

- Butterfly *Euphydryas gillettii*
- Gaboon forest frog *Scotobleps gabonicus*
- Out of Africa model for 2 and 3 populations
- Synthetic model of 5 populations

Initial design

First phase of bayesian optimization is the so called *initial design*. It samples random points in a search space to better aid the actual optimization. We were testing different initial design sizes to find better one to use.

Local optimization

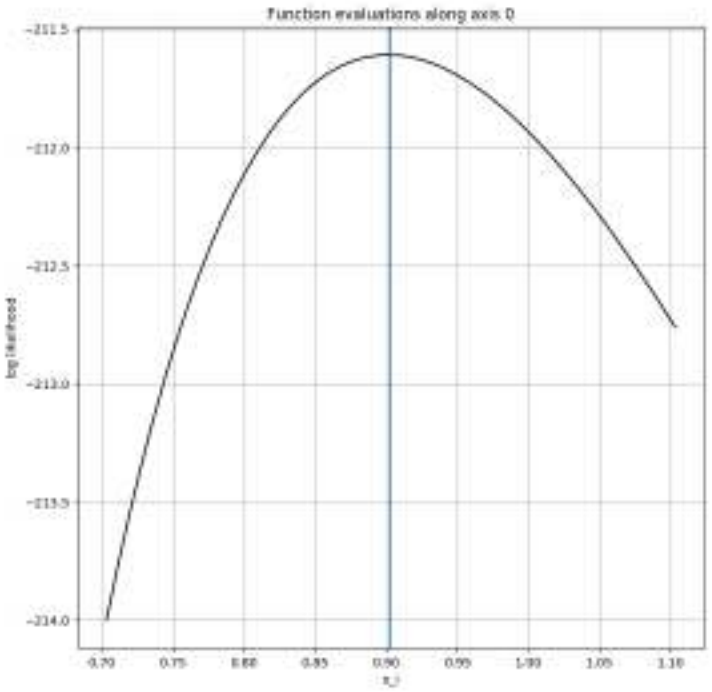
Bayesian optimization may not find the actual optimum of the function but point close to it. To pinpoint the actual optimum after the end of Bayesian optimization we employ the local one (BFGS).

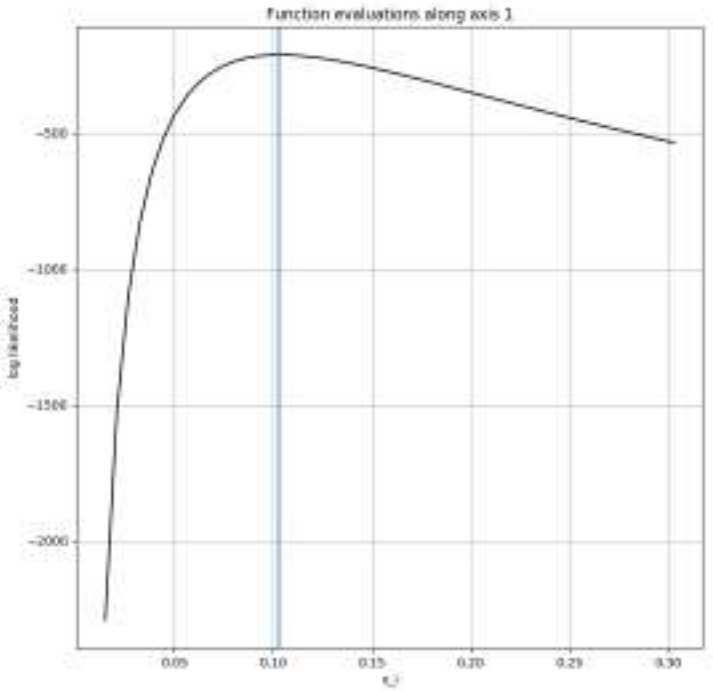
Kernel design

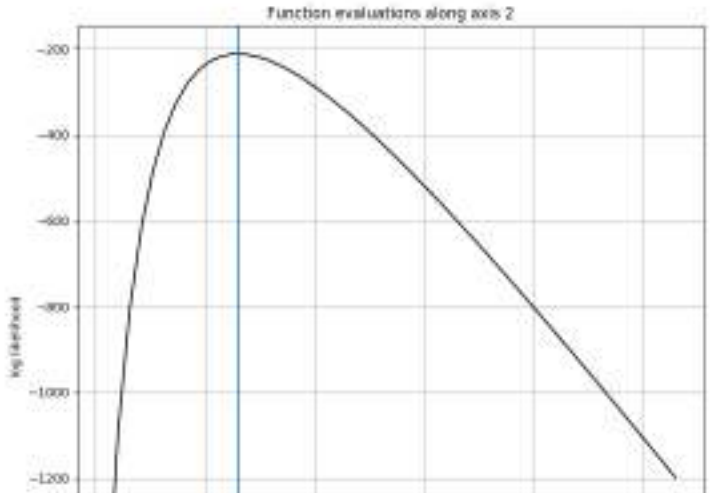
One of the most important hyperparameters of bayesian optimization is kernel. Informally, kernel is a measure of similarity between two points. Choice of kernel direcly influences quality of approximation and therefore optimum of a function.

We tried examining optimized function and trying to come up with a better kernel for our functions. To do it we tried projecting function in different ways which can be seen at the following plots:

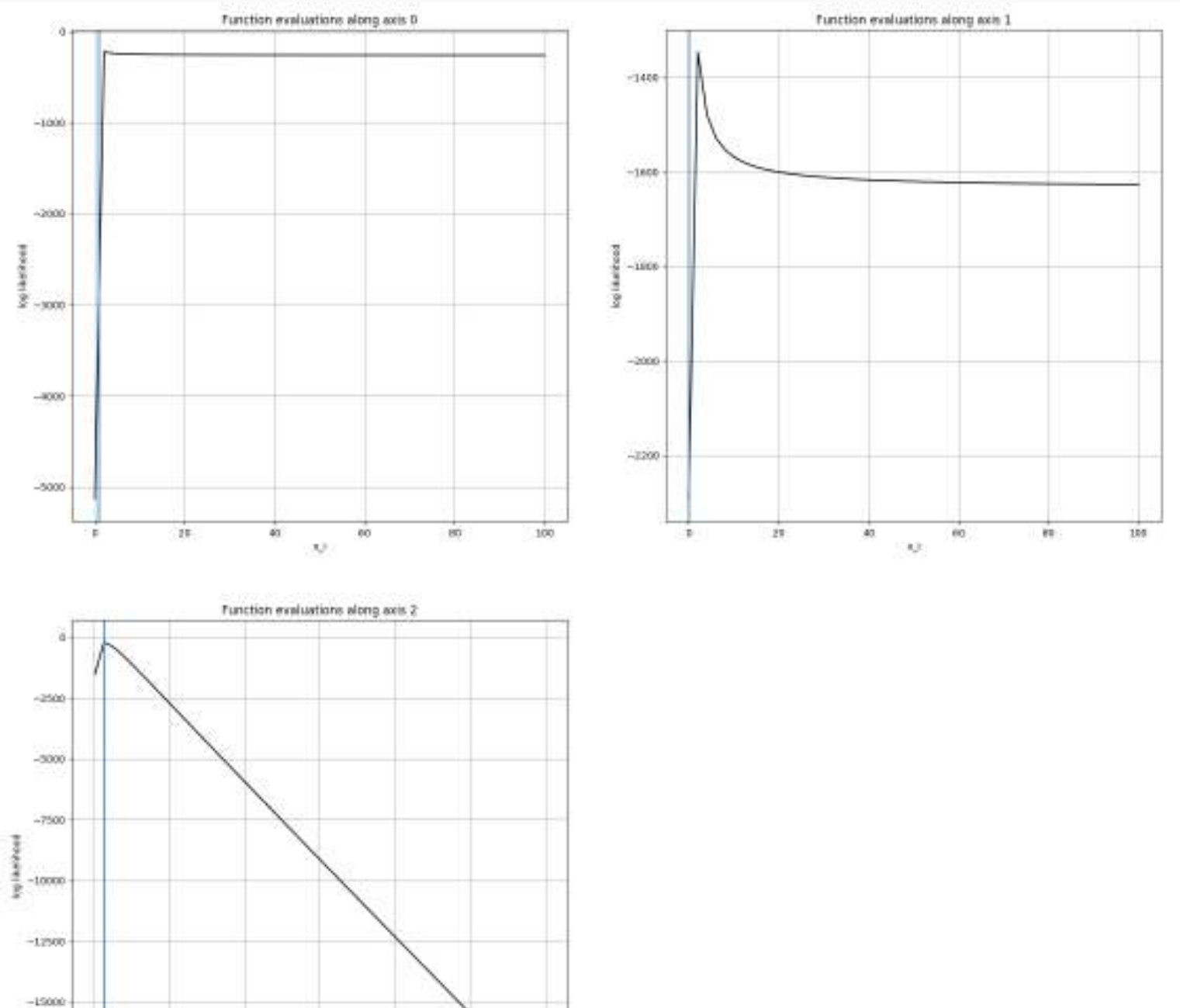
Plots


Function evaluations along axis 0
log-likelihood
-211.5
-212.0
-212.5
-213.0
-213.5
-214.0
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1.05
1.10

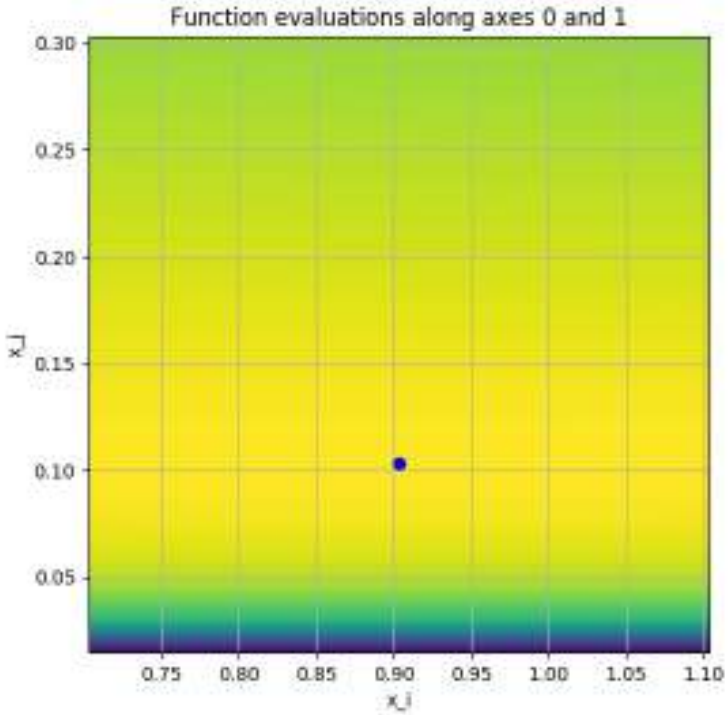

Function evaluations along axis 1
log-likelihood
-500
-1000
-1500
-2000
0.05
0.10
0.15
0.20
0.25
0.30

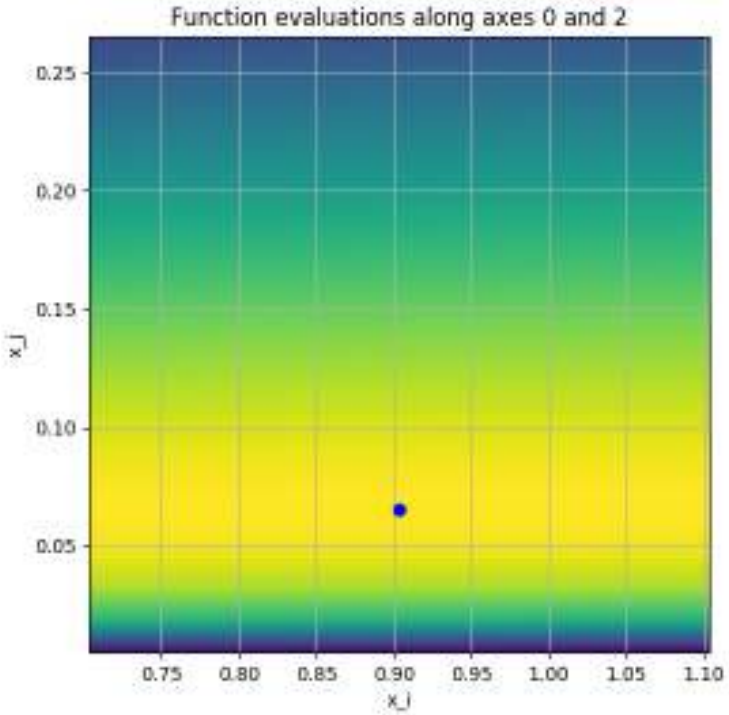

Function evaluations along axis 2
log-likelihood
-200
-400
-600
-800
-1000
-1200

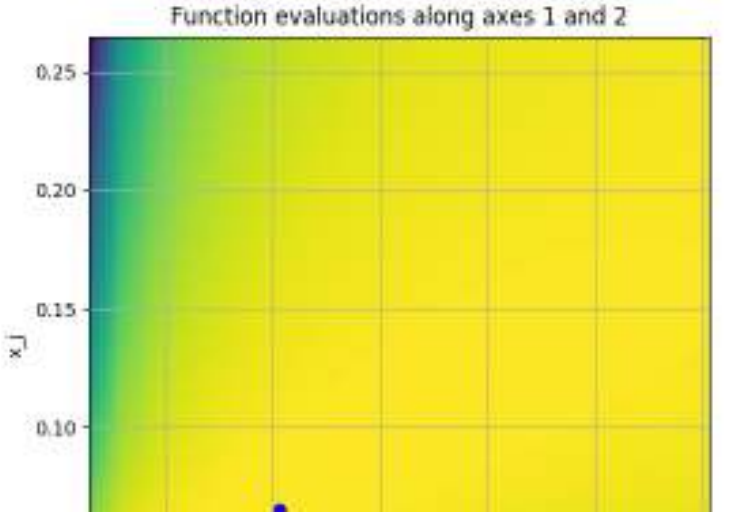
Plots


Function evaluations along axis 0
Function evaluations along axis 1
Function evaluations along axis 2

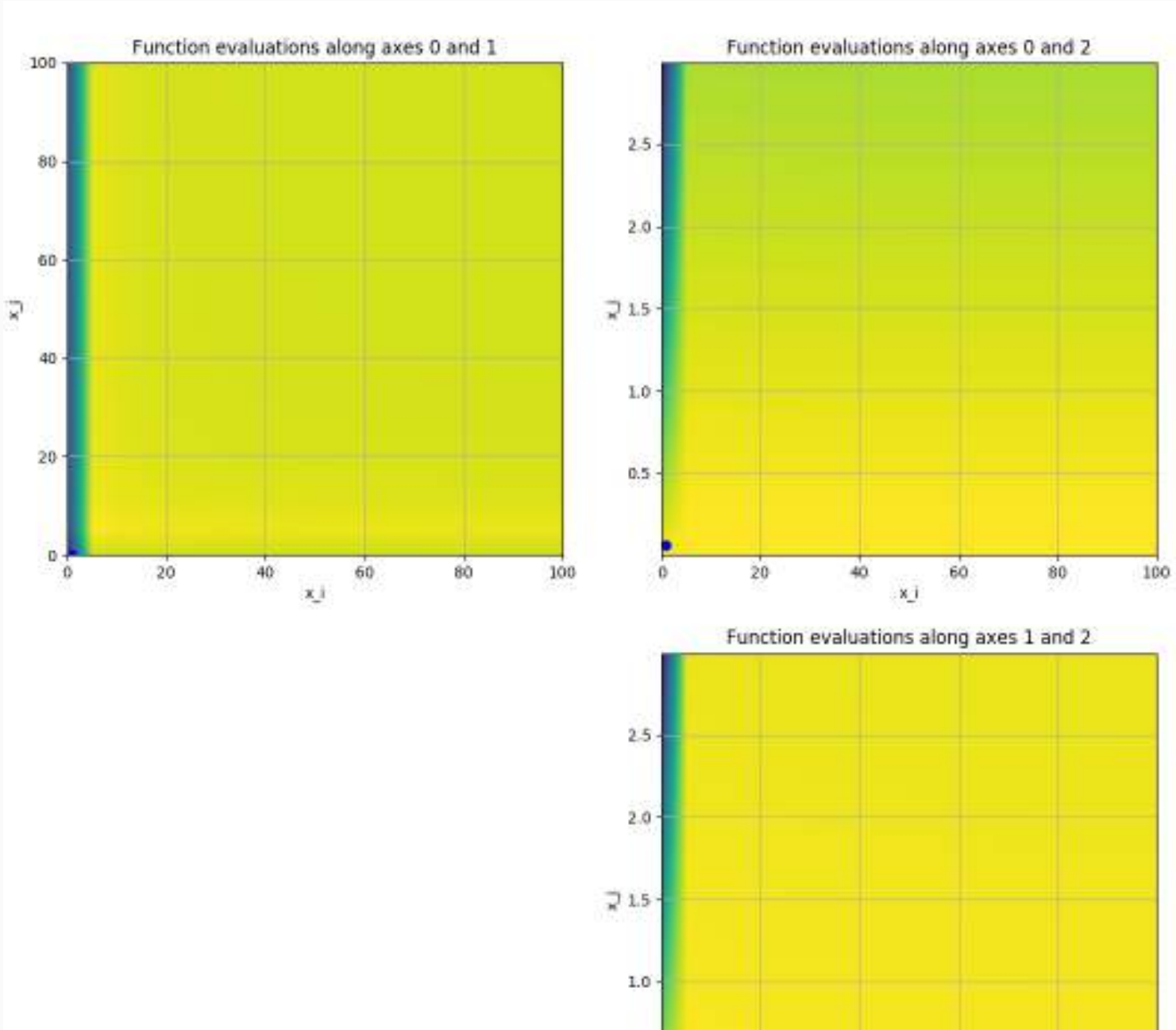
Plots


Function evaluations along axes 0 and 1
0.30
0.25
0.20
0.15
0.10
0.05
x_j
0.75
0.80
0.85
0.90
0.95
1.00
1.05
1.10
x_i


Function evaluations along axes 0 and 2
0.25
0.20
0.15
0.10
0.05
x_j
0.75
0.80
0.85
0.90
0.95
1.00
1.05
1.10
x_i


Function evaluations along axes 1 and 2
0.25
0.20
0.15
0.10
x_j

Plots


Function evaluations along axes 0 and 1
x_j
x_i
0
20
40
60
80
100
Function evaluations along axes 0 and 2
x_j
x_i
0.5
1.0
1.5
2.0
2.5
Function evaluations along axes 1 and 2
x_j
x_i

Kernel design

Unfortunately, different kernel specific to our task was not chosen due to lack of time.

Results

We performed a number of test runs to compare tool based on Bayesian optimization with already known optimum values. These results can be seen in following tables:

GADMA datasets

Each model ran 10 times.

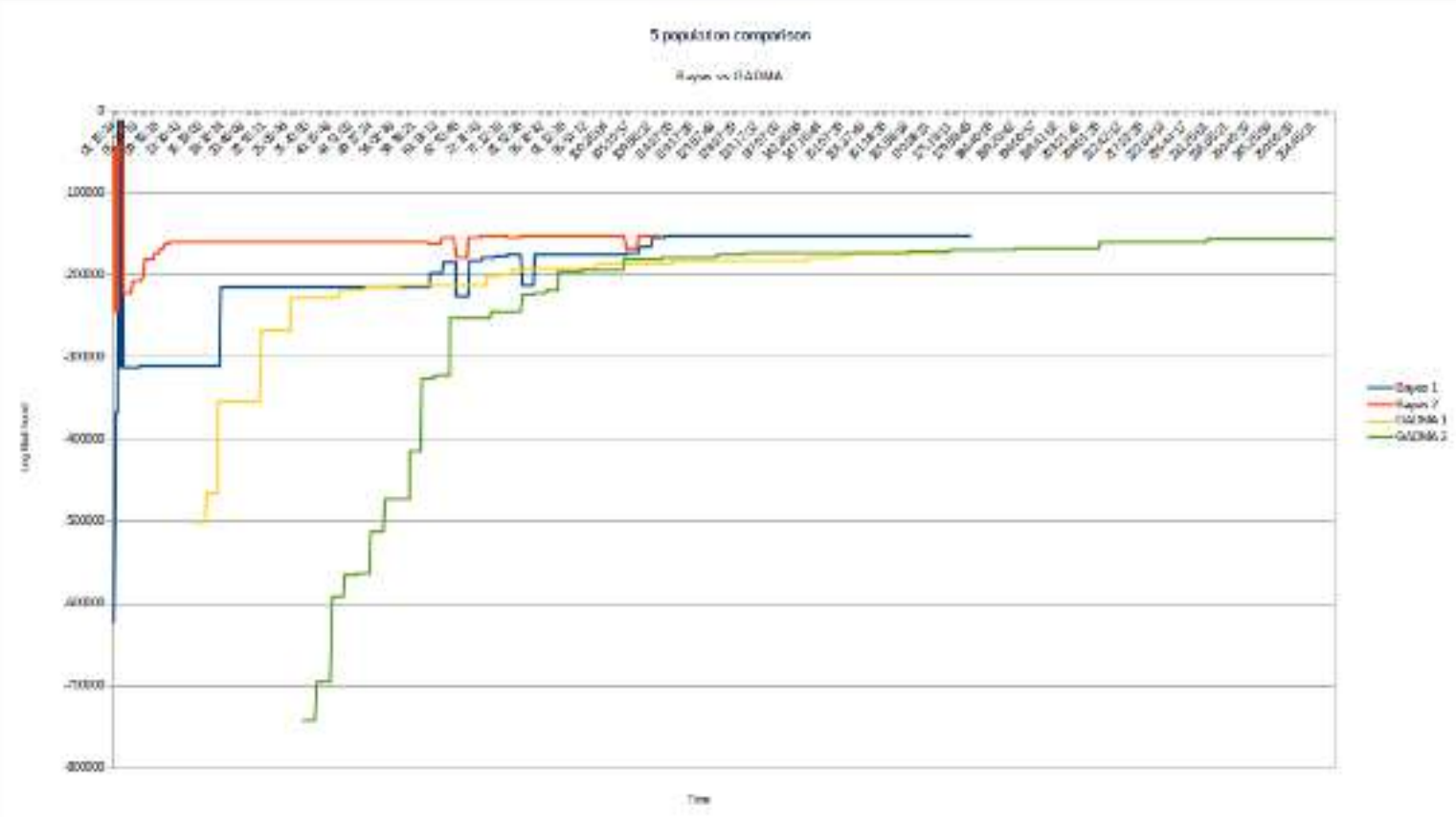
Average -LL	Best -LL	Average time	Known optimum
-283.541	-283.535	00:08:46	-283.530
-211.612	-211.608	00:08:48	-211.092
-211.584	-210.769	00:12:24	-210.830
-214.743	-210.781	00:17:10	-210.760
-609.246	-476.352	00:36:04	-455.167
-510.379	-453.800	00:35:11	-453.670
-407.200	-368.336	02:11:40	-368.232
-377.898	-374.555	00:23:38	-374.557
-388.332	-365.154	00:35:16	-365.181
-601.802	-413.190	00:24:46	-411.619
-589.171	-402.121	00:34:40	-402.165

Out of Africa (2 populations)

Model ran 50 times. Known optimum is -1066.822.

	Server time	-LL
Average	00:20:39	-1069.0656
Best	00:20:03	-1066.473

5 populations comparison



Conclusion

Results show that new method *may be* faster than `GADMA` for datasets with more than 3 populations and works more or less the same for small number of populations.

Thank you!