PES UNIVERSITY




Project Progress Review #3

(High Level / Low-Level Design)

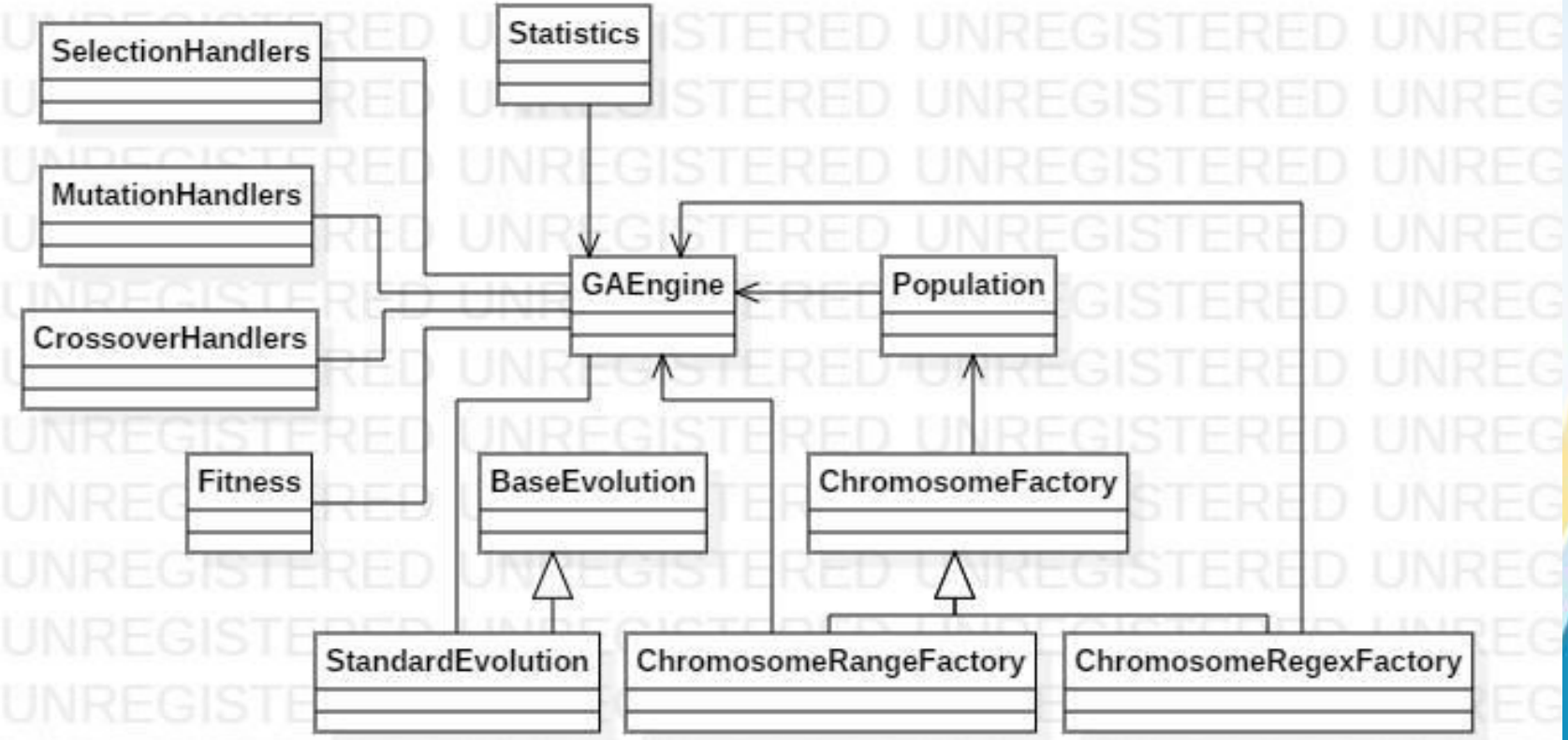
Project Title : Efficient Python Genetic Algorithm Framework
Project ID : PW19CGM01
Project Guide : Ms. Chitra G.M.
Project Team : Bharatraj S Telkar (01FB15ECS066)
Daniel I (01FB15ECS086)
Shreyas Vivek Patil (01FB15ECS286)

Project Abstract and Scope

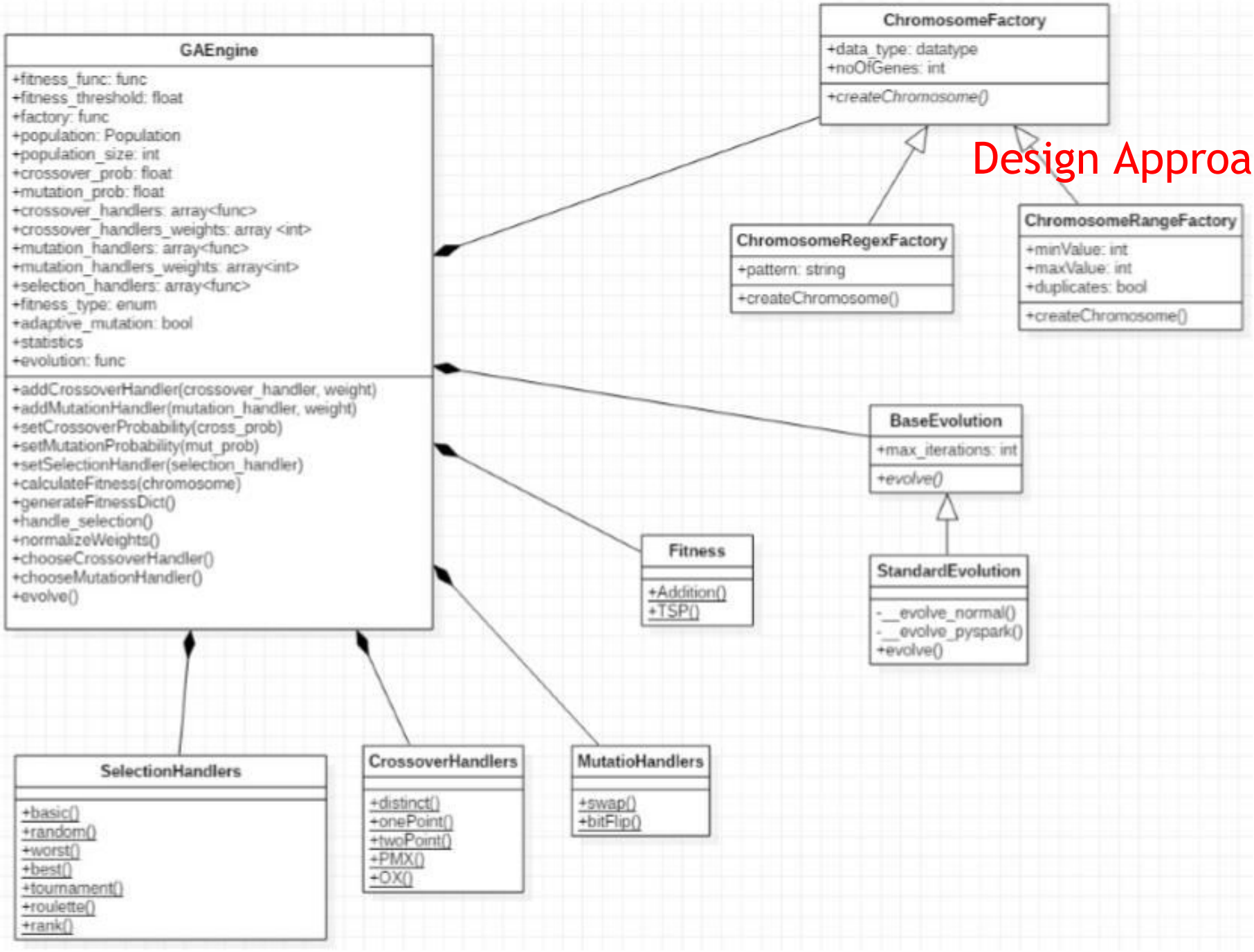
Our Genetic Algorithm Framework is proposed to be a very efficient, generic Framework where users can easily simulate all variations of Genetic Algorithms very easily. It is a usable way to **explore the problem solving ability of Genetic Algorithms.**

Usage of Spark to explore improvements in GA by using large populations

Design Approach



Design Approach



Design Approach

Benefits of this approach

- Follows good object oriented API principles of SOLID
- Is very generic in nature (users can define custom evolution style, crossovers, mutations, selection, chromosome factories easily)

Are there any drawbacks?

- Takes more lines of code than a less generic API.
- Very high dependence on good documentation

Alternate design approaches, if any.

High Level, less generic API with all the basic types of operations so as to reduce the number of lines of code needed

Design Constraints, Assumptions & Dependencies

Design Constraints:

- Need for spark to be correctly installed and configured
- Since it is generic in nature, need for more lines of code

Assumptions:

- Fitness type can be max, min, equal
- Crossover, mutation operations are expensive compared to generation of indexes RDD and spark operations

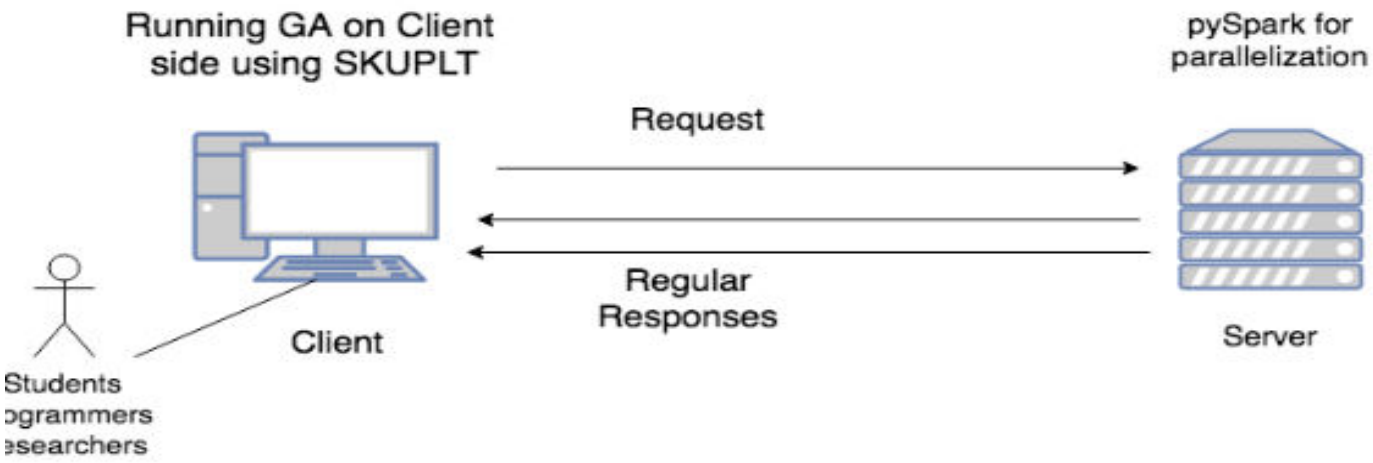
Dependencies:

Pyspark: Need for Apache Spark, Scala, JVM, pyspark installation for parallel execution using spark to work

Matplotlib: Needed for graph generations

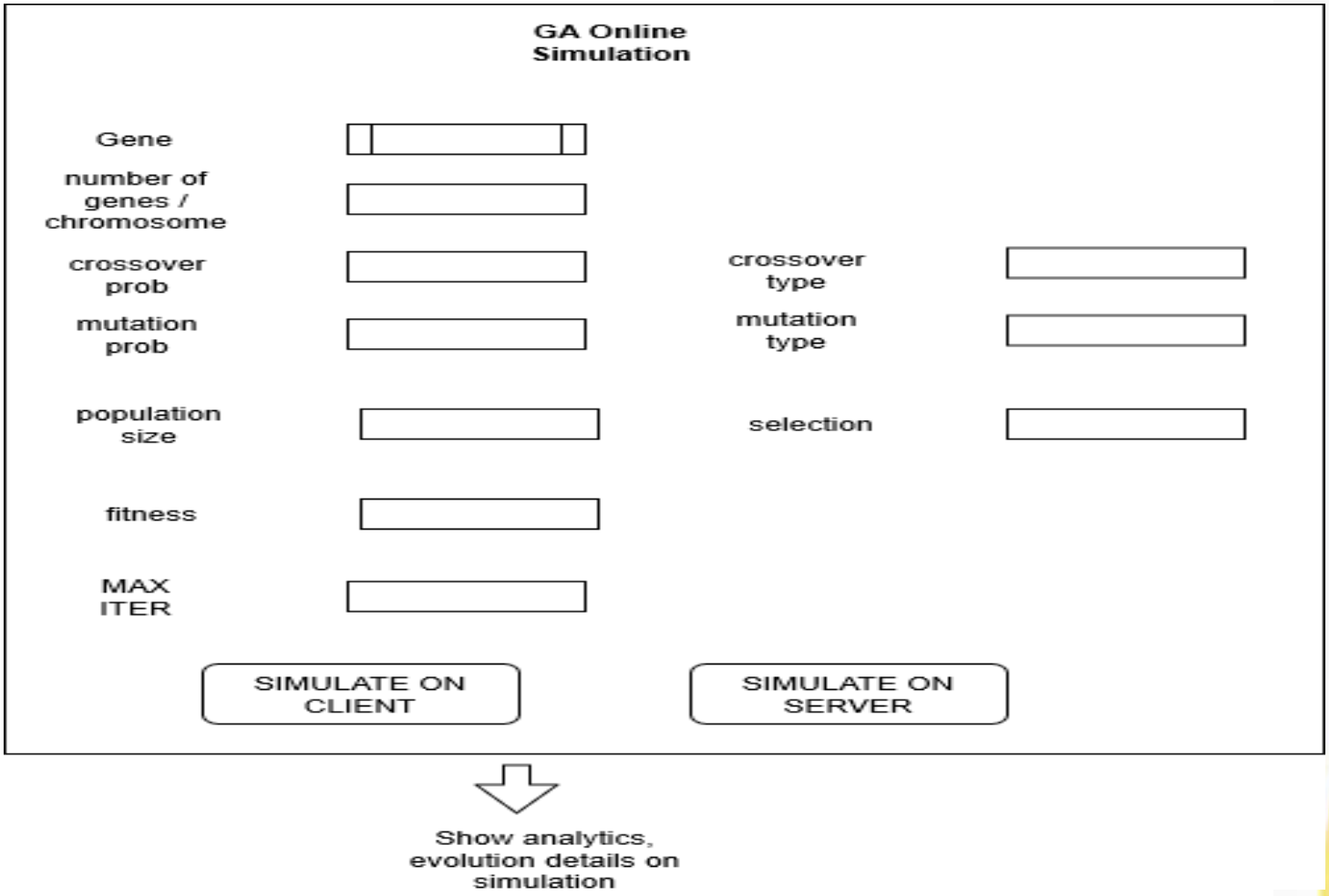
Design Description

Step 1: Take users inputs about the GA from the UI
Step 2: Convert the user inputs into python code which uses our GA API
Step 3: Run the code and observe GA execution and results

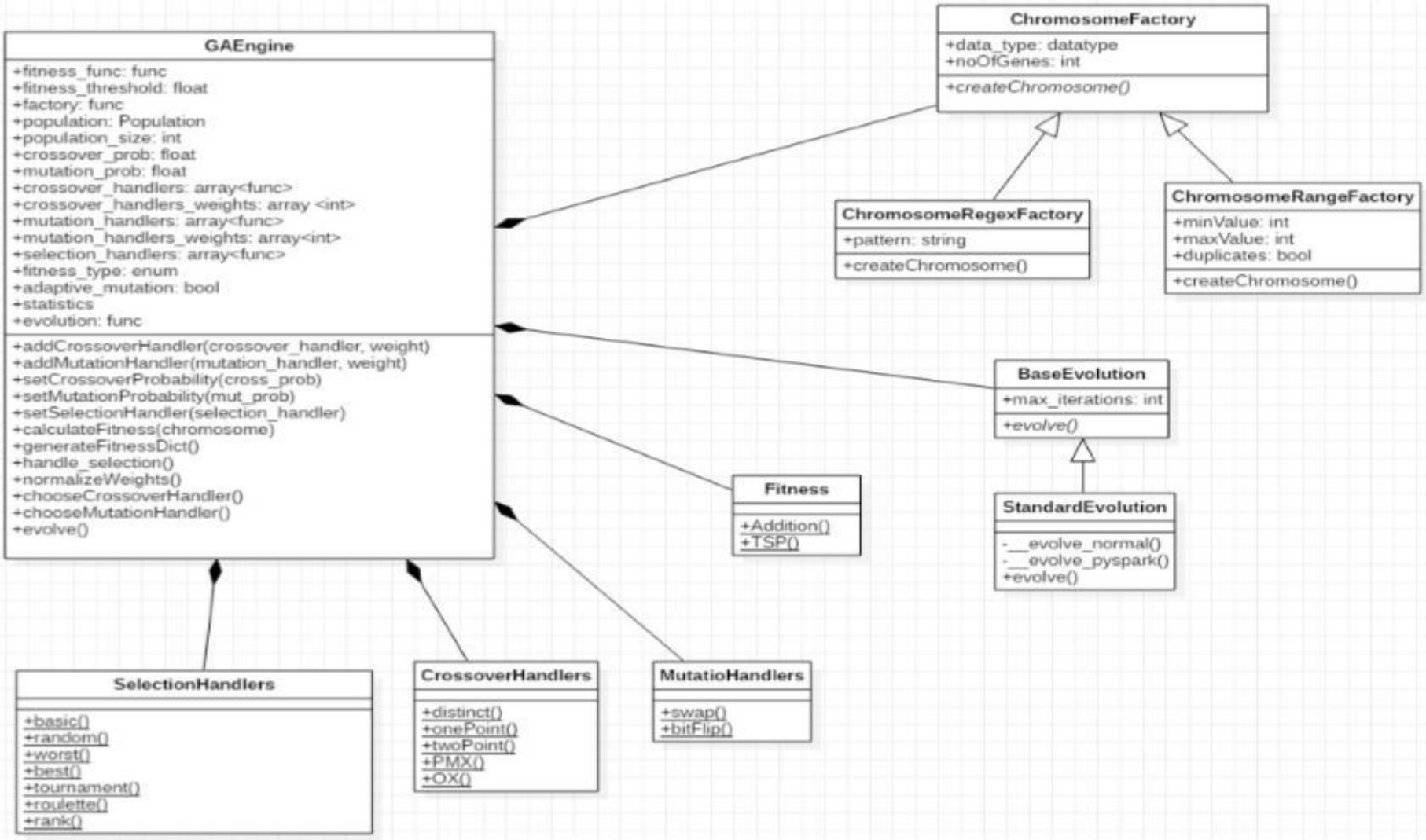


Choice: User can choose the run the code on the client side or the server side.
Client:no need to contact server for every GA generation update, no db support, no parallelization
Server:need for regular updates, db support, parallelization pySpark, "Genetic Algorithm as a Service"

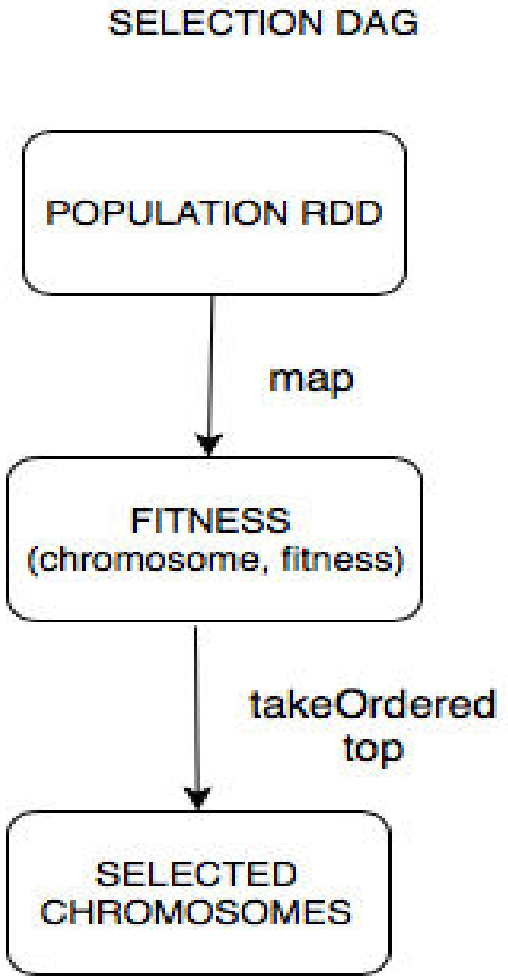
Design Description



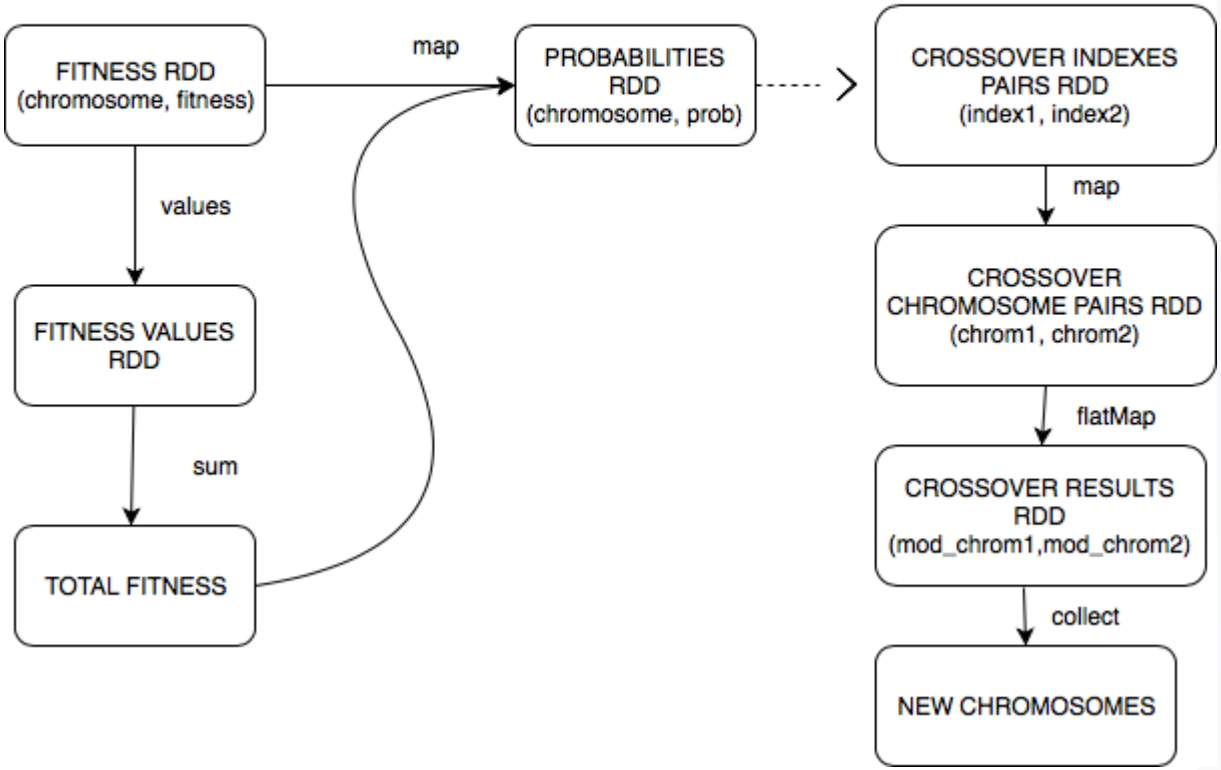
Design Description



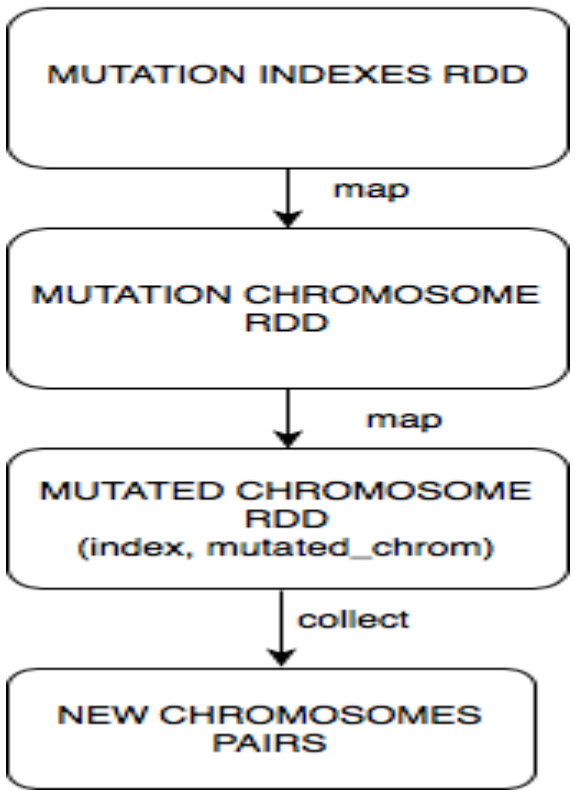
Spark DAG Design Description: Selection



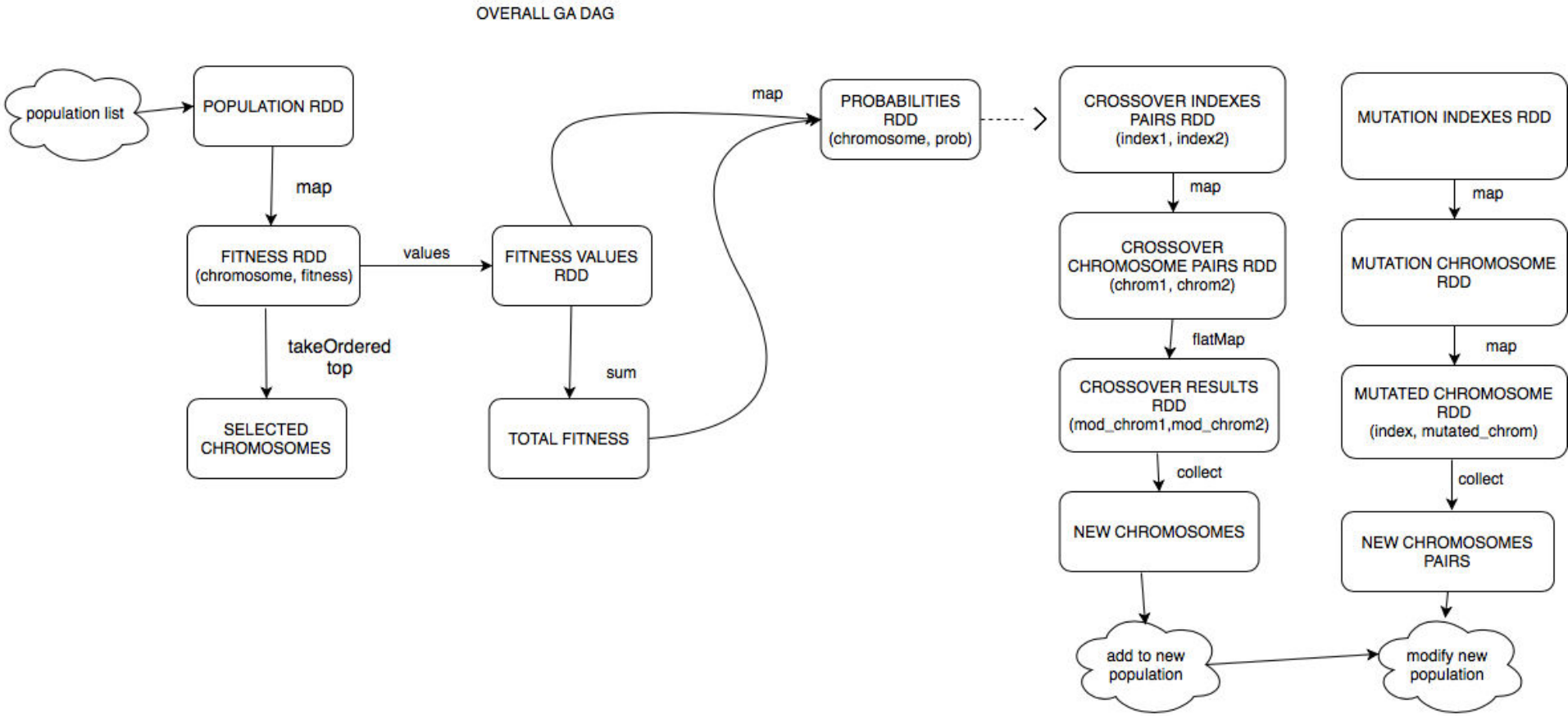
Spark DAG Design Description: Crossover



Spark DAG Design Description: Mutation



Spark DAG Design Description: Overall



Technologies Used

- Python - 3.0 and above
- Flask – for running algo as a service on server which receives parameters from UI.
- Pyspark - for parallelisation of operations in algo
- Pytest - for testing purposes.
- Matplotlib – For statistic graph plotting

PES UNIVERSITY




Project Progress So far

Divided work into three branches of activities

1. Implementation of classes
2. Website UI and functionality implementation
3. Documentation

Project Progress So far

1. Implementation of API

- Completed major parts of implementation of Regex/Range ChromosomeFactory, Utils, GAEngine, Statistics, Evolution classes
- Statistics of max,min,avg fitness
- Implementation of adaptive mutation GA optimization to change mutation rate based on diversity
- Implementation of a basic version of evolution using pySpark run on a single machine.

Project Progress So far

2. Website UI and functionality implementation

- Designed the basic frontend webpages using HTML, CSS, JS, jQuery.
- Designed basic Flask backend to serve webpages
- Done with conversion of user inputs into equivalent python code implementation
- Started with basic polling from website to get each generation data one by one

Project Progress So far

3. Documentation

- Worked on commenting every python file made to aid users in understanding our API
- Started on working on official pyGenetic documentation which is hosted at readthedocs.org

Project Progress So far

Work Left:

- Exploring GA on pySpark: Solving famous problems faster
- Implementation of more optimizations
- Implementation of other ML using GAs
- Website functionalities – polling, code download, enter custom code, security
- Documentation – better, more examples
- Testing
- Release to pypi

Work completed: 60 – 65 percent

Project Demo

DEMO

Problems encountered

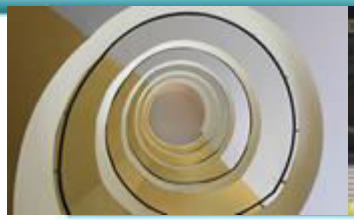
1. pySpark program takes longer time than a normal Python program to run GA

- Test on very large datasets
- Improve the dag ?
- Need to work on cluster ?

Problems encountered

2. Population Size: experiment with same size and increasing size on Spark ?

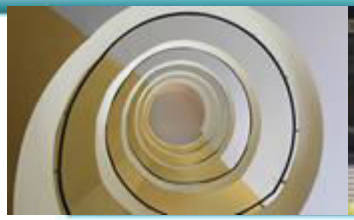
PES UNIVERSITY






Any Questions?

PES UNIVERSITY






Thank You