Hello and Welcome to MABE! This guide will walk you though everything you need to know about running MABE and Markov Brains.

0) Introduction to MABE

1) What is a Markov Brain?

2) Getting started

2) using MABE

4) Viewing results

3) Implementation Details

(things like how is the genome mutated and used to create Agents, how do worlds interface with agents and how do agents update, etc.)

4) reference for developing new code

(Including creating worlds, and gates, etc.)

**Section 0.**

**What is MABE?**

MABE (Modular Agent Based Evolver) is a digital evolution tool. MABE provides extensive support for Markov Brains, but is not limited to Markov Brains. As the name implies, MABE is modular. Almost all elements in the software have been designed so that the may be easily swapped out (i.e. worlds, gate types and behavior, types of genomes, types of brains, genome to brain conversions).

MABE creates populations of digital agents, evaluates these agents and generates new populations by means of natural and/or artificial selection. MABE has support to allow for various forms of genetic and direct mutation and supports multi-polidy genomes and sexual reproduction.

**What is MABE made of?**

MABE can be broken down into a number of interrelated modules.

- **Genomes – The Genome in MABE is a string of values that can be translated into a Brain.** It is possible to define a Brain (though some means of direct encoding) with out a genome, but we will generally assume that there is a genome. The Genome is what is used during reproduction to generate offspring and create brains. Genomes can be mutated (e.g. point mutation, insertions, deletions) and in the case of sexual recombination and/or multi-poidy genomes, crossover. The common genome in MABE is a multi-ploidy multi-chromosome genome. It can be defined as a haploid single chromosome genome made up of 256 characters (the standard for Markov Brains), a diploid genome with 23 pairs of chromosomes made up of 4 characters (a human genome – although achieving the size of an actual human genome may tax the memory and computational power of you computer!) or any other formatte.

- Genomes Handlers - Genomes Handlers can be thought of as helpers. They provide an interface to genomes. In addition to understanding the structure of and being able to navigate a genome, they are able to translate particular values found in a genome into standardized ranges usable by the rest of the software.

- Coding Regions - Genomes include something called coding regions. These will come up later, but in short, they provide a way track what parts of a genome are coding and what parts are ignored. Moreover, they provide information about the way in which a site in a genome is being used. This can be useful during analysis, and can also allow for targeted mutations.

**- Brains – Brains can be though of as controllers for the digital organisms that make up populations in MABE.** Brains interface with worlds (more on worlds later) though inputs provided from the world and outputs delivered to the world. Brains must have an update function which can be called by the world (a brain update). During a brain update a brain may read from the world inputs, process these inputs and deliver some outputs. Brains can have internal memory and/or states. Markov Brains will be explained in detail in the next section, which will include particular features of MABEs implementation.

Brains include a method to request information from Genomes and use this information to self assemble.

The interface between Brains and worlds is left intentionally vague - that is brains generally do not have functions which have knowledge about a particular world. This is done to maintain modularity and insures that brains and worlds can be used interchangeably. (If some problem requires that the brain have direct information about the world or type of world that it is interacting with, this can be supported though the input/output interface – but bewared, that this could result in incomparability with other worlds and may limit the future usefulness of these developments.)

**- Gates – Gates are subunits of brains.** Like brains, gates have an update function, only here, the gate is reading from brains values (either inputs that that brain has revived from the world, or internal memory/states of the brain), processing these outputs and providing output to the brain.

Gates also include a method to request information from Genomes and use this information to self assemble.

Gates are generally collected in gate lists.

**- Organisms – Organisms are generally defined as a Genome, a Brain, and any data associated with either this Genome or this Brain.** Organisms can be redefined though if needed. For example, a use may wish at add an evolve-able body to a organism.

**- Archivists – Archivists collect, maintain, record and retrieve data.** Different types of Archivist allow for different types and granularity of data to be collected. If a user requires some special form of data handling, that code should probably be added to an archivist.

**- Optimizers – An optimizer takes a population of organisms which has been evaluated, selects which organisms from the current population will be used to generate the next population and manages the process of creating the next generation.**

If a world includes an implicit method of offspring generation then an optimizer is not necessary.

**- Groups – Groups are defined as a population (a collection of living organisms), an optimizer(some generate new populations) and an archivist (which tracks and stores information about the group).** The optimizer maybe removed though if there is some implicit method of offspring generation. MABE can have multiple groups, and these groups can interact (i.e. a group of wolves and a group of sheep). It would be possible to create a group that contained both wolves and sheep, but this would require significant support (to control mating if these are sexual populations, and to segregate data). Groups exist to simplify this process. (If you wanted to see what would happen if your wolves could mate with your sheep... who a I to judge?... then you would most likely place them in the same group).

**- Worlds – Worlds define the evaluation function used to test organisms.** There is no limit to the types of evaluations worlds make. A world could for example pose a simple mathematical question, such as add x and y. On the other hand a world could be a complex simulation of a 3 dimensional space including resources and obstacles in which many organisms coexist.

Worlds must define a world to brain interface. That is the world provides inputs values to the brain and reads output values from the brain. Typically, a world will perform updates, during which the following events may occur:

- update world features (i.e. resource inflow/decay)

- for each organisms

- program organisms inputs

- call organisms update function

- update world in response to organisms output

- repeat...

**Section 1.** **What is a Markov Brain?**

Blah Blah....

**Section 2 Getting Started**

This section is designed to get a used up and running. It only provides limited information about MABE, but should provide a reasonable introduction.

**Section 2.0 Requirements**

MABE does not have a graphical interface (yet) so if you wish to use it, you must be able to navigate your file structure and execute commends. I suggest that some familiarity with working in a terminal (i.e. command line) will be helpful.

Below are links to introductions for various systems.

If you are on a Mac check out:

<http://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line>

or

<http://www.davidbaumgold.com/tutorials/command-line/>

If you are on a PC check out:

<http://www.computerhope.com/issues/chusedos.htm>

If you are on Linix (you pobably already know this stuff) check out:

<http://linuxcommand.org/lc3_lts0020.php>

**Section 2.1 Installing and compiling MABE**

In the future we hope to be able to provide executable versions of MABE but for now, rather then attempt to keep up to date versions of the software for every possible computer, we provide the code and leave it up to you to compile the code for your computer. Below are are instructions for various system.

*Mac:*

*get c++ compiler if you don't have it*

*get code from git*

*compile*

*test run - note, this is the command you will use to run MBs*

*you can press cntr & c at the same time to stop execution*

*PC:*

*get c++ compiler if you don't have it*

*get code from git*

*compile*

*test run - note, this is the command you will use to run MBs*

*you can press cntr & c at the same time to stop execution*

*LINUX:*

*get c++ compiler if you don't have it*

*get code from git*

*compile*

*test run - note, this is the command you will use to run MBs*

*you can press cntr & c at the same time to stop execution*

**Section 2.2 Generating a Settings files**

You should now have a working version of the MABE. If you just run the code now it will use all default values. The first thing you'll want to do is generate a settings file. This can be accomplished by running MABE with the 'makeConfigFile' flag set to 1.

./MABE makeConfigFile 1

you should see the following output:

-----

You can use your own configFile by setting the comand line flag "configFileName" to the name of your config file (default: configFile settings.cfg).

to write out a new config file set the comnad line flag "makeConfigFile" to 1.

-----

- loading Command Line.

- loading Config File "settings.cfg"

- Config File: WARNING unable to open file "settings.cfg". Proceeding without.

- "settings.cfg" has been created. Exiting.

If you now list the contents of the directory you should find 'settings.cfg'. Open 'settings.cfg' in an editor of your choices and you will see a number a settings divided into categories. Since this file is brand new, it will contain all the default settings. NOTE: values for bool variables must be 0 or 1 (not 'true' or 'false'). MABE is smart, when it comes to parameters at least. Along with default values, when a parameter is defined in MABE it also includes that parameters type, the category it belongs to and simple usage. Most questions about what a parameter does can be anwsered simply by looking at the settings file. If you need more information, you can find out more details about all settings in the appendix (***link to appendix)***

At this point it's important to note that the settings file is configured to run MABE with a single population of Markov Brains using genomes with an alphabet size of 256 in BerryWorld (see worlds). All of these features can be changed,.

In 'settings.cfg' find the following line:

detGate = 0

It will be under the heading 'GATE TYPES:'.

Change this line to read:

detGate = 0 (i.e. change the 0 to a 1). You have just enabled Deterministic Logic Gates.

Save 'settings.cfg' and rerun the program (make sure that you don't have 'makeConfigFile 1' after the program name)

As MABE is running you should see a max fitness increasing. If you wish to stop MABE, remember that you can use cntl-c to terminate execution.

In the next few sections I'll be providing information about working with settings, I would suggest not worrying about settings that you don't understand yet as some of them are rather esoteric (if it's really bothering you, you can look them up in the appendix).

**Section 2.3 Command line settings**

Any setting in the settings file can be set on the command line. The settings names and values are the same, just leave out the '='. This allows you to try things without needed to edit the settings file. More importantly, it simplifies the creation of scripts to create and manage multiple replicates and/or conditions.

Executing the following on the command line:

./MABE updates 30 popSize 100

will run MABE for 30 updates with a population size of 100.

**Section 2.4 Output**

During a run, various types of data can be saved. Different methods can be used by the program to write this data to disk. For the most part, MABE saves files in csv (comma separated value) format.

By default MABE saves ave.csv (a user defined collection of average values for the population) and dominant.csv (values from the highest scoring agent in a population).

Data.csv and Genome.csv can also be recored. Data.csv contains all data (aside from the genomes themselves) related to organisms (i.e. score, ancestors, number of gates, size of genome, world generated data). Genome files contain the sites that make up the genome and any other data that is needed to read the genome.

After running MABE with default settings you will see ave.csv and dominant.csv. Try changing the outputMethod from default to either LODwAP or SSwD and rerun MABE. You should now see data and genome files.

**Section 2.4.1 Managing Output**

settings.cfg contains a number of entries to determine what goes into the data.csv and genome.csv files and when they are written.

**updates** - how long do you want the program to run (generations).

**[file] interval** - how often do you want to write to a given file. Set this to 100 to output on updates 0,100,200,300...

**particular to LODwAP**

Line of Decent with Active Purning (LODwAP) Archivist – create one data file and one genome file. These files contain the data and genomes (respectively) for organisms on line of decent – (note: LODwAP can not be used with sexually reproducing populations since LOD can not be established in these populations).

**terminate after** – If running with LODwAP you may want to set MABE continue running after you reach updates in order to allow for coalescence (i.e. a single most recent conman ancestor for the current population). Since coalescence can take a long time, terminate after is how long we are willing to wait for coalescence. once updates + terminate\_after is reached the program will end. Any LOD based files that have not yet been written will be written assuming that a random agent in the current population is most recent conman ancestor.

**pruneInterval\_LODwAP** – determines how often the Archivist checks for coalescence and attempts to write data to file.

**Particular to SSwD**

SnapShot with Delay (SSwD) Archivists – create multiple data and genome files. Each file contains all of the related data for the time interval being recored.

**cleanupInterval\_SSwD** - How often to cleanup old checkpoints.

[**file] IntervalDelay** – How long to wait before writing data for an update. In SSwD, if an organism has no surviving decedents after this delay, that organisms data and genomes will not be saved. This can save space on disk... but more importantly can be used to filterer data to make more manageable data sets.

**Section 2.5 Brain Settings**

settings.cfg allows you to change the number of memory nodes that a Markov Brain has.

**BrainSize** – number of nodes in the brains.

**BitsPerBrainAddress** – when brains are being build one genome site at minimum will be used to determine gate addressing. This value is used to determine if more then one site is needed (i.e. this may be needed in the future to allow for brains that can gow in size).

**GATE TYPES** – allows the user to set which types of gates can exist in the brain.

**Section 2.6 Defining Mutation Rates**

Mutation rates can be set by changing values under the GENOME: header.