Genetic Engine Programming Guide

The Genetic Engine libraries (GeneticEngine.dll and GeneticEnginePlugin.dll) provides a flexible framework for developing and running genetic algorithms. GeneticEnginePlugin.dll contains the interface definitions and support classes required to implement custom plug-ins, defining the behaviour of the various stages of the algorithm. GeneticEngine.dll contains the classes necessary to load plug-ins and run the algorithm.

This document outlines the important interfaces and classes and how they are used. For more detailed information see the API documentation.

GeneticEnginePlugin.dll

GeneticAlgorithm.Plugin.IPopulator

A class which implements this interface defines a populator plug-in. A populator is used to generate the initial population of individuals for the algorithm.

Populate(ArrayList individuals) is called when the GeneticEngine is initialised and whenever its *Reset()* method is called. It should populate the ArrayList with a number of objects. This set of objects will be used as the initial population for the algorithm. All objects should be instances of the same class. This will be the class expected by the other plug-ins to define individuals.

GeneticAlgorithm.Plugin.IEvaluator

A class which implements this interface defines an evaluator plug-in. An evaluator is used to calculate a fitness value for each individual.

Initialise(int generationCount, ArrayList individuals) is called once for every generation before any individuals are evaluated. It gives the evaluatior an opportunity to prepare to process the generation. If no preparation is required it can be ignored.

Evaluate(object individual) is called once for every individual. It should return an unsigned int representing the fitness of the individual.

GeneticAlgorithm.Plugin.IGeneticOperator

A class which implements this interface defines a genetic operator plug-in. A genetic operator generates a new population from the current generation.

Operate(IGeneration source, ArrayList destination) is called once every generation. It should use the individuals in the IGeneration and their fitness values to generate a new set of individuals which it inserts into the ArrayList.

GeneticAlgorithm.Plugin.ITerminator

A class which implements this interface defines a terminator plug-in. A terminator determines when the algorithm is complete.

When the *Run()* method of the GeneticEngine has been started *Terminate(IGeneration generation)* is called once every generation to determine if the algorithm is complete. It should return true when the generation meets some termination condition.

GeneticAlgorithm.Plugin.IOutputter

A class which implements this interface defines an outputter plug-in. An outputter saves or displays some or all of the individuals in each generation along with statistics about the entire generation.

The *Status* property is checked before any call to the outputter. Open indicates that it is ready to receive generations for output, Closed indicates that it is not ready but can be made ready by calling *OpenOutput()* and Locked indicates that it is not ready and cannot be made ready.

OutputGeneration(IGeneration generation, int generationNumber) is called once every generation after the individuals have been evaluated. If Status is Closed then *OpenOutput()* is called first. If Status is Locked then OutputGenration is not called. This method should not modify the generation or the individuals within it.

CloseOutput() is called when the *FinishOutput()* method of the GeneticEngine is called. It should finish writing and close any files that are open.

GeneticAlgorithm.Plugin.IGenerationFactory

A class which implements this interface defines a generation factory plug-in. A generation factory produces empty instances of a class which implements the IGeneration interface.

CreateGeneration(ArrayList individuals) is called once per generation. It should return an empty instance of a class which implements the IGeneration interface. The ArrayList contains the unevaluated individuals which will later be added to the generation. This method should not add any of these individuals to the generation, it is only supplied to inform the plug-in about the size and type of data the generation will be expected to hold.

Other Interfaces and Classes:

- **GeneticAlgorithm.Plugin.IGeneration** - Holds a set of individuals and their fitness values.
- **GeneticAlgorithm.Plugin.IndividualWithFitness** - Simple container which allows lookups in an IGeneration to return an individual and its fitness value together.
- **GeneticAlgorithm.Plugin.Generic.AATreeGeneration** - Default implementation of IGeneration. Supports the standard IGeneration methods and lookup by fitness sum to support fitness proportionate sampling and similar random sampling methods. All operations work in O(lg(n)) time where n is the number of individuals in the generation.
- **GeneticAlgorithm.Plugin.Generic.AATreeGenerationFactory** - Default Generation Factory. Produces instances of AATreeGeneration.
- **GeneticAlgorithm.Plugin.Generic.FitnessThresholdTerminator** - Simple terminator plug-in which terminates the algorithm when the fitness of at least one individual meets or exceeds a chosen threshold.
- **GeneticAlgorithm.Plugin.Generic.XmlOutputter** - Abstract class implementing IOutputter. Writes all generations to a series of XML files. Classes which extend this class must implement *WriteIndividual(object individual, XmlWriter writer)* to allow individuals of a specific type to be written.
- **GeneticAlgorithm.Plugin.Util.FitnessConverter** - Contains static methods to convert positive floating point values to/from standard the unsinged integer fitness values used by the GeneticEngine.
- **GeneticAlgorithm.Plugin.Xml.GenerationIndex** - Manages an XML index file and loads generations saved by XmlOutputter.

- **GeneticAlgorithm.Plugin.Xml.IIndividualReader** - Classes which implement this interface define how individuals of a specific type are loaded through GenerationIndex.

GeneticEngine.dll

GeneticAlgorithm.GeneticEngine

This class manages the genetic algorithm. It's constructor takes one of each plug-in type:

```
GeneticEngine(
        IPopulator populator,
        IEvaluator evaluator,
        IGeneticOperator geneticOperator,
        ITerminator terminator,
        IOutputter outputter,
        IGenerationFactory generationFactory)
```

outputter and generationFactory are optional and default to null. When outputter is null or omitted no outputter is used. When generationFactory is null the default generation factory (AATreeGenerationFactory) is used.

The constructor generates an initial set if individuals using the populator then evaluates each and uses them to populate the current generation.

Properties and Methods:

- *GenerationNumber* indicates which generation the algorithm is up to. This counts from 0.
- Generation gets the current generation.
- *IsComplete* indicates whether the termination condition has been met.
- *Reset()* Finishes the current output then starts the algorithm again from a new initial population.
- *FinishOutput()* checks whether the outputter's Status is Open. If it is then it calls *outputter.CloseOutput().*
- *Step()* runs the algorithm for a single generation.
- *Repeat(int n)* runs the algorithm for n generations.
- *Run()* runs the algorithm until the termination condition is satisfied.
- *Stop()* sets a flag indicating that no more generations should be processed. This is used to terminate the Repeat(int n) and Run(n) methods running in another thread. Note: the other methods of GeneticEngine are not designed to be thread safe. Stop() is the only method which should ever be called on an running GeneticEngine from another thread.

GeneticAlgorithm.Util.PluginLoader

This class allows plug-in classes to be loaded from arbitrary .Net Class Libraries (.dll files) at run time. Only classes which have a constructor which accepts one object as its only parameter are recognised as plug-ins. This object is intended to hold configuration information which the plug-in will cast to whatever type it expects.

Methods:

- *LoadDll(string path)* loads the .dll file specified by path and finds any plug-in classes. Multiple .dll files may be loaded however plug-in classes with the same name (including namespace) as already loaded plug-ins will be ignored.
- *GetPluginNames(Type pluginType)* returns a list containing the names of classes which can be cast to pluginType. If pluginType is omitted or null then all plug-in classes are listed.
- *GetInstance(string pluginName, object config)* Returns an instance of the plug-in class specified by pluginName. The config object is passed to the constructor.

Other Interfaces and Classes:

GeneticAlgorithm.GeneticEngineException - A custom exception type which is thrown by the GeneticEngine and PluginLoader. It contains a *SourcePlugin* property which indicates which plug-in the error occurred in.