Genetic Programming in C# [closed]

Asked 16 years, 4 months ago Modified 7 years, 4 months ago

Viewed 34k times

60

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 7 years ago.

Improve this question

I've been looking for some good genetic programming examples for C#. Anyone knows of good online/book resources? Wonder if there is a C# library out there for Evolutionary/Genetic programming?

c# genetic-algorithm genetic-programming evolutionary-algorithm

Share

edited May 17, 2010 at 3:14

Improve this question

Follow

Jon Seigel

12.4k ● 8 ● 60 ● 93

asked Aug 17, 2008 at 23:25

Mac

2,705 ● 7 ● 36 ● 44

12 Answers

Sorted by: Highest score (default)

29

After developing [my own Genetic Programming didactic application], I found a complete Genetic Programming Framework called [AForge.NET Genetics]. It's a part of the [Aforge.NET library]. It's licensed under LGPL.

Share Improve this answer

Follow

edited Jul 16, 2009 at 19:11

answered Jun 23, 2009 at 0:22

Jader Dias

90.4k ● 159 ● 432 ● 632

MSDN had an article last year about genetic programming: Genetic Algorithms: Survival of the Fittest with Windows Forms

24

Share Improve this answer

Follow

answered Aug 17, 2008 at 23:28

Judah Gabriel Himango

60k • 39 • 161 • 215

14

I would recommend against actually generating assemblies unless you absolutely need to, particularly if you are just getting started with implementing the genetic algorithm.

The genetic algorithm is easiest to implement when the target language is functional and dynamically typed. That is generally why most genetic algorithm research is written in LISP. As a result, if you are going to implement it in C#, you are probably better off defining your own mini "tree language", having the algorithm generate trees, and just interpreting the trees when it comes time to run each iteration of the algorithm.

I did a project like this when I was in college (an implementation of the genetic algorithm in C#), and that was the approach I took.

Doing it that way will give you the advantage of only having 1 representation to work with (the AST

representation) that is optimally suited for both execution and the genetic algorithm "reproduction" steps.

Alternatively, if you try to generate assemblies you are probably going to end up adding a large amount of unneeded complexity to the app. Currently, the CLR does not allow an assembly to be unloaded from an App domain unless the entire app domain is destroyed. This would mean that you would need to spin up a separate app domain for each generated program in each iteration of the algorithm to avoid introducing a giant memory leak into your app. In general, the whole thing would just add a bunch of extra irritation.

Interpreted AST's, on the other hand, are garbage collectible just like any other object, and so you wouldn't need to monkey around with multiple app domains. If, for performance reasons you want to code-gen the final result you can add support for that later. However, I you would recommend that you do that using the DynamicMethod class. It will allow you to convert an AST into a compiled delegate dynamically at runtime. That will enable you to deploy a single DLL while keeping the code generation stuff as simple as possible. Also, DynamicMethod instances are garbage collectible so you could end up employing them as part of the genetic algorithm to speed things up there as well.

answered Aug 26, 2008 at 6:14

Scott Wisniewski

25k ● 8 ● 62 ● 90

Actually I have found that generated assemblies work quite well and the framework implementer can hide most of the complexity from the GA implementer. We can dynamically deploy versioned assemblies across a cluster
– Steve Severance Jan 23, 2009 at 6:59

10

You might be able to implement genetic programming using LINQ expression trees -- it's more likely to generate something usable than random IL generation.

Share Improve this answer

Follow

answered Aug 18, 2008 at 1:00

Curt Hagenlocher
20.9k ● 8 ● 62 ● 50

6

I saw a good high-level discussion of it on channel9 by Mike Swanson at http://channel9.msdn.com/posts/Charles/Algorithms-and-Data-Structures-Mike-Swanson-Genetic-Session-Scheduler/

Share Improve this answer

Follow

answered Aug 17, 2008 at 23:28

denis phillips
12.7k ● 5 ● 34 ● 47

5

If you're interested in genetic algorithms or heuristic optimization in general you might want to take a look at HeuristicLab. It is developed for several years, 1.5 years

since we released the new version. It is programmed in C# 4 and has a nice GUI. There are many algorithms already available like Genetic Algorithm, Genetic Programming, Evolution Strategy, Local Search, Tabu Search, Particle Swarm Optimization, Simulated Annealing and more. There are also several problems implemented like a vehicle routing problem, traveling salesman, real function optimization, knapsack, quadratic assignment problem, classification, regression, and many more. There are tutorials also and we have protocol buffers integrated so you can communicate with external programs for solution evaluation. It is licensed under GPL. In 2009 the software has received the Microsoft innovation award of Microsoft Austria.

We've also written a book on the subject: Genetic Algorithms and Genetic Programming.

Share Improve this answer Follow

edited Oct 17, 2011 at 20:24

answered Oct 17, 2011 at 17:56

Andreas

6,467 ● 2 ● 39 ● 47

4

Do you mean actual genetic programming, as opposed to genetic algorithms in general?

If so, C#/.net isn't the best language for it. LISP, for example, has always been a mainstay of GP.

However, if you must, you're probably going to want to dynamically generate CIL / MSIL. You could do this using System.Reflection.Emit, however I'd recommend Mono.Cecil. It lacks good docs (as if reflection emit has them).. But it offers much better assembly emission and reflection.

Another issue is that it is less than trivial to load code, and later dispose of it, in the .net framework. At least, you cannot unload assemblies. You can unload appdomains, but the whole business of loading code into a seperate appdomain, and calling it externally can get pretty messy. .NET 3.5's System.Addin stuff should make this easier.

Share Improve this answer

Follow

answered Aug 18, 2008 at 0:54

mgsloan

3,295 ● 22 ● 20

He does specify GP rather than GAs, but then the tags say both. – Chris S Oct 27, 2010 at 10:10

I am reading A Field Guide to Genetic Programming right now (free PDF download). It is also available as a paperback. It discuses the use of a library written in Java

3

called TinyGP. You might get some mileage out of that. I have not started doing any actual programming but am hoping to applies some of the concepts in C#.

Share Improve this answer

Follow

answered Sep 25, 2008 at 3:04

Jason Jackson

17.2k ● 8 ● 51 ● 75

3

I've forked ECJ to C# .NET 4.0 if you are interested in a full-featured Evolutionary Computation framework. The package includes everything from the original ECJ Java project, including all of the working samples.

I also wrote 500 unit tests to verify many aspects of the conversion. But many more tests are needed. In particular, the distributed computation aspects are not fully tested. That's because I plan on converting from ECJ's simple use of sockets to a more robust strategy using WCF and WF. I'll also be reworking the framework to utilize TPL (Task Parallel Library).

Anyway, you can download the initial conversion here:

http://branecloud.codeplex.com

I am also in the process of converting several other frameworks from Java to .NET that relate to "synthetic intelligence" research (when I can find the time).

Ben

Share Improve this answer

Follow

answered Jun 1, 2011 at 15:11

bstabile

209 ● 2 ● 2

3

You can try GeneticSharp.

It has all classic GA operations, like selection, crossover, mutation, reinsertion and termination.

It's very extensible, you can define your own chromosomes, fitness function, population generation strategy and all cited operations above too.

It can be used in many kind of apps, like C# libraries and Unity 3D games, there is samples running it in a GTK# app and Unity 3D checkers game.

It also works in Win and OSX.

Here is a basic sample how to use the library:

```
var selection = new EliteSelection();
var crossover = new OrderedCrossover();
var mutation = new ReverseSequenceMutation();
var fitness = new YourFitnessFunction();
var chromosome = new YourChromosome();
var population = new Population (50, 70, chromosome);

var ga = new GeneticAlgorithm(population, fitness, sel
mutation);

ga.Start();
```

Share Improve this answer

answered Jul 10, 2014 at 10:23

Follow

giacomelli
7,407 ● 2 ● 29 ● 32

3

The Manning book: "Metaprogramming in .NET" dedicates a large section on GP via expression trees.

Share Improve this answer

Follow

edited Dec 17, 2015 at 17:16

answered May 21, 2014 at 18:50

cs0815
17.4k ● 47 ● 152 ● 320

1

I maintain a port of ECJ in C#. It's great.

Share Improve this answer

Follow

edited Jan 29, 2009 at 13:25

mmcdole
92.7k ● 61 ● 188 ● 224

answered Oct 22, 2008 at 0:48

MrMan