GNU/Linux AI & Alife HOWTO

Table of Contents

GNU/Linux AI & Alife HOWTO....1
by John Eikenberry....1
1. Introduction....1
2. Symbolic Systems (GOFAI)....1
3. Connectionism....1
4. Evolutionary Computing....1
5. Alife & Complex Systems....1
6. Agents & Robotics....1
7. Statistical & Machine Learning....2
8. Missing & Dead....2
1. Introduction....2
1.1 Purpose....2
1.2 What's New....2
1.3 Where to find this software....4
1.4 Updates and comments....4
1.5 Copyright/License....4
2. Symbolic Systems (GOFAI)....4
2.1 AI class/code libraries....5
2.2 AI software kits, applications, etc....12
3. Connectionism....17
3.1 Connectionist class/code libraries....17
3.2 Connectionist software kits/applications....22
4. Evolutionary Computing....26
4.1 EC class/code libraries....26
4.2 EC software kits/applications....34
5. Alife & Complex Systems....35
5.1 Alife & CS class/code libraries....35
5.2 Alife & CS software kits, applications, etc....37
6. Agents & Robotics....43
6.1 Software Agents....43
6.2 Robotics and Simulators....54
7. Statistical & Machine Learning....59
7.1 Libraries....59
7.2 Applications....63
8. Missing & Dead....63
8.1 MIA - Projects missing linkage....64
8.2 Dead projects....70

GNU/Linux AI & Alife HOWTO

by John Eikenberry

v3.1, 31 Mar 2013

This howto mainly contains information about, and links to, various AI related software libraries, applications, etc. that work on the GNU/Linux platform. All of it is (at least) free for personal use. The new master page for this document is http://zhar.net/howto/

1. Introduction

- 1.1 Purpose
- 1.2 What's New
- 1.3 Where to find this software
- 1.4 Updates and comments
- 1.5 Copyright/License

2. Symbolic Systems (GOFAI)

- 2.1 AI class/code libraries
- 2.2 AI software kits, applications, etc.

3. Connectionism

- 3.1 Connectionist class/code libraries
- 3.2 Connectionist software kits/applications

4. Evolutionary Computing

- 4.1 EC class/code libraries
- 4.2 EC software kits/applications

5. Alife & Complex Systems

- 5.1 Alife & CS class/code libraries
- 5.2 Alife & CS software kits, applications, etc.

6. Agents & Robotics

- 6.1 Software Agents
- 6.2 Robotics and Simulators

7. Statistical & Machine Learning

- 7.1 Libraries
- 7.2 Applications

8. Missing & Dead

- 8.1 MIA - Projects missing linkage.
- 8.2 Dead projects.

---

1. Introduction

1.1 Purpose

The GNU/Linux OS has evolved from its origins in hackerdom to a full blown UNIX, capable of rivaling any commercial UNIX. It now provides an inexpensive base to build a great workstation. It has shed its hardware dependencies, having been ported to DEC Alphas, Sparcs, PowerPCs, and many others. This potential speed boost along with its networking support will make it great for workstation clusters. As a workstation it allows for all sorts of research and development, including artificial intelligence and artificial life.

The purpose of this Howto is to provide a source to find out about various software packages, code libraries, and anything else that will help someone get started working with (and find resources for) artificial intelligence, artificial life, etc. All done with GNU/Linux specifically in mind.

1.2 What's New

- v3.1 -

  New entries: SimpleAI .

  Resurfaced MIA entry: NuPIC .

  Deleted the section on "Programming languages". I moved the couple of agent-oriented domain specific languages, 2APL and APRIL , into the sections relevant to the work. I removed the rest as they were all just general purpose languages.

  Changed my Copyright/License to CC0. That is no copyright, it is Public Domain. I no longer wish to take part in copyright; http://questioncopyright.org/
- v3.0 -

  New entries: ORTS , FANN , OpenCV , CBR Microprograms , ConceptNet , FreeHAL , 2APL , Alchemy , plop , evolver , PyIE , Pyevolve , txevolver , python-dlp , CompLearn , dbacl , Maximum Entropy Toolkit , pebl , FLiP , Carmen , Orca , ROS , YARP , Neuroph , MLAP book samples , Elefant , Evocosm , Critterding , MRPT , PyBrain , peach , brain , FREVO , Vowpal Wabbit , ERESYE , Recast , EAP , GenePool , Milk , OpenCog , Pattern , CognitiveFoundry , clasp , timbl , MBT , scikits.learn , NeuroLab , Biogenesis , brain-simulator . Torch5 , Encog , Nengo , DEAP and Emergent .

Changed the name of the "Traditional" section to Symbolic Systems (GOFAI) . Added new section, Statistical & Machine Learning .

Seems someone has resuscitated EMA-XPS . Not very active, but enough to pull out of the dead projects area and place back amounst the living. I also fixed many links and moved quite a few dead projects to Missing & Dead .

- v2.4 -

  New entries: Eprover , Player , Logfun , Livingstone2 , Quackle , LingPipe , GATE , Infon Battle Arena , CLARAty , Reverend , Shogun , Nanopond , Polyworld , Fluidiom , NEAT , Framsticks , URBI , RobotFlow , Nero , ffnet , Alloy (removed), Pyke , NuPIC , Simbad , Robodeb , Loom , PowerLoom , tinygp , Curry (removed), JGAP , PyCLIPS , and STELLA (removed).

  I chopped the Agents section into two sub-sections, one for Software Agents and one for Robotics and Simulators . I play it a bit fast and loose in my deciding what goes into each category, but it is an improvement.

  MIA found! Cellular the cellular automata programming system. Fixed many bad links and cleaned out missing projects.
- v2.3 -

  New entries: Yampa , pygene , Push (removed), ANNEvolve , dgpf , Golly , IBAL , 3APL , OSCAR , and RobocodeNG .

  Updated information for some entries including Yale , Joone , Drone , Biome , ECLiPSe (removed), Xtoys , GECO , Creatures Docking Station and others.

  I also changed the MIA section to Missing & Dead which now groups into subsections entries with bad links that I can't find replacements for and long dead projects.
- v2.2 -

  Fixed a some bad links and was forced to move a few entries into the MIA (missing) section. I also removed one duplicate entry.

  New entries: MASON , spyse , AntWars , OpenSteer , Pyro , Robocode , Trend and Open BEAGLE .
- v2.1 -

  New entries: NLTK , NEURObjects , KANREN , Neural Networks at your Fingertips , SimWorld , SimAgent , Fuzzy sets for Ada , maxent , Evo , breve and AJA
- v2.0 - Ran linkchecker and for any bad links I either found a new link or removed the item. See the new section MIA for a list of the removed entries (please let me know if you know of a new home for them).

  New entries: Yale , DIET Agents , JASA , Jason , Noble Ape , Maude (removed), ECLiPSe (removed), lush (removed), and pygp
- v1.9 - One new entry (Bond) and fixed the link below to the dynamic list (now defunct).
- v1.8 - Cleaned up bad links, finding new ones where possible and eliminating those that seem to have disappeared. Quite a few new entries as well.

  New entries: Torch , Aleph , AI Kernel , OpenCyc , HTK , FFLL , JCK , Joone , scnANNlib , GAUL , Cougaar , and RoboTournament

- v1.7 - Another 9 new entries, a bunch of links fixed, and a few items removed that have vanished from the net.

  New entries: SPASS , CNNs , JCASim , Genetic , CAGE , AgentFarms , MATREM , OAA , and UTCS Neural Nets Research Group Software
- v1.6 - 9 new entries, a couple link fixes and one duplicate item removed.
- v1.5 - 26 new entries plus a couple link fixes.
- v1.4 - 10 new updates and fixed some lisp-related links.
- v1.3 - Putting a dent in the backlog, I added 30+ new entries today and submitted it to the LDP.
- Previous records were in a mixed format with site updates. See the old notes section of the master site for them.

1.3 Where to find this software

All this software should be available via the net (ftp || http). The links to where to find it will be provided in the description of each package. There will also be plenty of software not covered on these pages (which is usually platform independent) located on one of the resources listed on the links section of the Master Site (given above).

1.4 Updates and comments

If you find any mistakes, know of updates to one of the items below, or have problems compiling any of the applications, please mail me at: jae@zhar.net and I'll see what I can do.

If you know of any AI/Alife applications, class libraries, etc. **Please** email me about them. Include your name, ftp and/or http sites where they can be found, plus a brief overview/commentary on the software (this info would make things a lot easier on me... but don't feel obligated ;).

I know that keeping this list up to date and expanding it will take quite a bit of work. So please be patient (I do have other projects). I hope you will find this document helpful.

1.5 Copyright/License

CC0

To the extent possible under law, John Eikenberry has waived all copyright and related or neighboring rights to this work. This work is published from: United States.

---

2. Symbolic Systems (GOFAI)

Traditionally AI was based around the ideas of logic, rule systems, linguistics, and the concept of rationality. At its roots are programming languages such as Lisp and Prolog though newer systems tend to use more popular procedural languages. Expert systems are the largest successful example of this paradigm. An expert system consists of a detailed knowledge base and a complex rule system to utilize it. Such systems have been used for such things as medical diagnosis support and credit checking systems.

2.1 AI class/code libraries

These are libraries of code or classes for use in programming within the artificial intelligence field. They are not meant as stand alone applications, but rather as tools for building your own applications.

ACL2

◊ Web site: www.cliki.net/ACL2

ACL2 (A Computational Logic for Applicative Common Lisp) is a theorem prover for industrial applications. It is both a mathematical logic and a system of tools for constructing proofs in the logic. ACL2 works with GCL (GNU Common Lisp).

AI Kernel

◊ Web site: aikernel.sourceforge.net
◊ Sourceforge site: sourceforge.net/projects/aikernel/

The AI Kernel is a re-usable artificial intelligence engine that uses natural language processing and an Activator / Context model to allow multi tasking between installed cells.

AI Search II

◊ WEB site: http://www.neiu.edu/~kwtracy/ooai-book/

Basically, the library offers the programmer a set of search algorithms that may be used to solve all kind of different problems. The idea is that when developing problem solving software the programmer should be able to concentrate on the representation of the problem to be solved and should not need to bother with the implementation of the search algorithm that will be used to actually conduct the search. This idea has been realized by the implementation of a set of search classes that may be incorporated in other software through **C++**'s features of derivation and inheritance. The following search algorithms have been implemented:

◊ depth-first tree and graph search.
◊ breadth-first tree and graph search.
◊ uniform-cost tree and graph search.
◊ best-first search.
◊ bidirectional depth-first tree and graph search.
◊ bidirectional breadth-first tree and graph search.
◊ AND/OR depth tree search.
◊ AND/OR breadth tree search.

This library has a corresponding book, " Object-Oriented Artificial Intelligence, Using C++".

Alchemy

◊ Web site: http://alchemy.cs.washington.edu/

Alchemy is a software package providing a series of algorithms for statistical relational learning and probabilistic logic inference, based on the Markov logic representation. Alchemy allows you to easily develop a wide range of AI applications, including:

◊ Collective classification
◊ Link prediction

◊ Entity resolution
◊ Social network modeling
◊ Information extraction

Aleph

◊ Web site: http://www.cs.ox.ac.uk/activities/machlearn/Aleph/

This document provides reference information on A Learning Engine for Proposing Hypotheses (Aleph). Aleph is an Inductive Logic Programming (ILP) system. Aleph is intended to be a prototype for exploring ideas. Aleph is an ILP algorithm implemented in Prolog by Dr Ashwin Srinivasan at the Oxford University Computing Laboratory, and is written specifically for compilation with the YAP Prolog compiler

Microprograms

◊ Web site: http://www.cs.indiana.edu/~leake/cbr/code/

A collection of case-based reasoning "micro" versions of dissertation programs that were developed for pedagogical purposes. These programs are meant to distill key aspects of the original programs into a form that can be easily understood, modified, and extended.

Chess In Lisp (CIL)

◊ Web site: *found as part of the CLOCC archive at: clocc.sourceforge.net

The CIL (Chess In Lisp) foundation is a Common Lisp implementaion of all the core functions needed for development of chess applications. The main purpose of the CIL project is to get AI researchers interested in using Lisp to work in the chess domain.

clasp

◊ Web site: http://www.cs.uni-potsdam.de/clasp/

clasp is an answer set solver for (extended) normal logic programs. It combines the high-level modeling capacities of answer set programming (ASP) with state-of-the-art techniques from the area of Boolean constraint solving. The primary clasp algorithm relies on conflict-driven nogood learning, a technique that proved very successful for satisfiability checking (SAT). Unlike other learning ASP solvers, clasp does not rely on legacy software, such as a SAT solver or any other existing ASP solver. Rather, clasp has been genuinely developed for answer set solving based on conflict-driven nogood learning. clasp can be applied as an ASP solver (on LPARSE output format), as a SAT solver (on simplified DIMACS/CNF format), or as a PB solver (on OPB format).

ConceptNet

◊ Web site: http://conceptnet.media.mit.edu/
◊ Old Web site: http://web.media.mit.edu/~hugo/conceptnet/

ConceptNet aims to give computers access to common-sense knowledge, the kind of information that ordinary people know but usually leave unstated. The data in ConceptNet was collected from ordinary people who contributed it over the Web. ConceptNet represents this data in the form of a semantic network, and makes it available to be used in natural language processing and intelligent user interfaces.

This API provides Python code with access to both ConceptNet 3 and the development database that will become ConceptNet 4, and the natural language tools necessary to work with it. It uses Django for interacting with the database.

ERESYE

◊ Web site: http://sourceforge.net/projects/eresye/
◊ Tutorial:
http://www.trapexit.org/Artificial_Intelligence_with_Erlang:_the_Domain_of_Relatives
ERESYE means ERlang Expert SYstem Engine. It is a library to write expert systems and rule processing engines using the Erlang programming language. It allows to create multiple engines, each one with its own facts and rules to be processed.

FFLL

◊ Web site: ffll.sourceforge.net
The Free Fuzzy Logic Library (FFLL) is an open source fuzzy logic class library and API that is optimized for speed critical applications, such as video games. FFLL is able to load files that adhere to the IEC 61131-7 standard.

FLiP

◊ Web site: http://staff.washington.edu/jon/flip/www/
Flip is a logical framework written in Python. A logical framework is a library for defining logics and writing applications such as theorem provers. The checker can use different logics; Flip comes with several. You can add another logic, or add axioms and derived rules, by writing a module in Python. Python is both the object language and the metalanguage. Formulas, inference rules, and entire proofs are Python expressions. Prover commands are Python functions.

Fuzzy sets for Ada

◊ Web site: http://www.dmitry-kazakov.de/ada/fuzzy.htm
◊ Freshmeat: http://freshmeat.net/projects/fuzzy/
Fuzzy sets for Ada is a library providing implementations of confidence factors with the operations not, and, or, xor, +, and *, classical fuzzy sets with the set-theoretic operations and the operations of the possibility theory, intuitionistic fuzzy sets with the operations on them, fuzzy logic based on the intuitionistic fuzzy sets and the possibility theory; fuzzy numbers, both integer and floating-point with conventional arithmetical operations, and linguistic variables and sets of linguistic variables with operations on them. String-oriented I/O is supported.

HTK

◊ Web site: htk.eng.cam.ac.uk
The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems. The HTK release contains extensive documentation and examples.

JCK

◊ Web site: www.pms.informatik.uni-muenchen.de/software/jack/
JCK is a new library providing constraint programming and search for Java.

◊ JCK consists of three components:
◊ - JCHR: Java Constraint Handling Rules. A high-level language to write constraint solvers.
◊ - JASE: Java Abstract Search Engine. A generic search engine for JCHR to solve constraint problems.
◊ - VisualCHR: An interactive tool to visualize JCHR computations.

Source and documentation available from link above.

KANREN

◊ Web site: kanren.sourceforge.net

KANREN is a declarative logic programming system with first-class relations, embedded in a pure functional subset of Scheme. The system has a set-theoretical semantics, true unions, fair scheduling, first-class relations, lexically-scoped logical variables, depth-first and iterative deepening strategies. The system achieves high performance and expressivity without cuts.

LK

◊ Web site: www.cs.utoronto.ca/~neto/research/lk/

LK is an implementation of the Lin-Kernighan heuristic for the Traveling Salesman Problem and for the minimum weight perfect matching problem. It is tuned for 2-d geometric instances, and has been applied to certain instances with up to a million cities. Also included are instance generators and Perl scripts for munging TSPLIB instances.

This implementation introduces ``efficient cluster compensation'', an experimental algorithmic technique intended to make the Lin-Kernighan heuristic more robust in the face of clustered data.

LingPipe

◊ Web site: http://alias-i.com/lingpipe/

LingPipe is a state-of-the-art suite of natural language processing tools written in Java that performs tokenization, sentence detection, named entity detection, coreference resolution, classification, clustering, part-of-speech tagging, general chunking, fuzzy dictionary matching.

Logfun

◊ Web site: http://www.irisa.fr/lande/ferre/logfun/

Logfun is a library of logic functors. A logic functor is a function that can be applied to zero, one or several logics so as to produce a new logic as a combination of argument logics. Each argument logic can itself be built by combination of logic functors. The signature of a logic is made of a parser and a printer of formulas, logical operations such as a theorem prover for entailment between formulas, and more specific operations required by Logical Information Systems (LIS). Logic functors can be concrete domains like integers, strings, or algebraic combinators like product or sum of logics.

Logic functors are coded as Objective Caml modules. A logic semantics is associated to each of these logic functors. This enables to define properties of logics like the consistency and completeness of the entailment prover, and to prove under which conditions a generated entailement prover satisfies these properties given the properties of argument logics.

Loom

◊ Web site: http://www.isi.edu/isd/LOOM/

* Note: Loom has been succeeded by PowerLoom .

Loom is a language and environment for constructing intelligent applications. The heart of Loom is a knowledge representation system that is used to provide deductive support for the declarative portion of the Loom language. Declarative knowledge in Loom consists of definitions, rules, facts, and default rules. A deductive engine called a classifier utilizes forward-chaining, semantic unification and object-oriented truth maintainance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing.

The Loom system implements a logic-based pattern matcher that drives a production rule facility and a pattern-directed method dispatching facility that supports the definition of object-oriented methods. The high degree of integration between Loom's declarative and procedural components permits programmers to utilize logic programming, production rule, and object-oriented programming paradigms in a single application. Loom can also be used as a deductive layer that overlays an ordinary CLOS network. In this mode, users can obtain many of the benefits of using Loom without impacting the function or performance of their CLOS-based applications.

maxent

- ◊ Python/C++ version: http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html
- ◊ Java version: maxent.sourceforge.net

The Maximum Entropy Toolkit provides a set of tools and library for constructing maximum entropy (maxent) models in either Python or C++. Maxent Entropy Model is a general purpose machine learning framework that has proved to be highly expressive and powerful in statistical natural language processing, statistical physics, computer vision and many other fields.

It features conditional maximum entropy models, L-BFGS and GIS parameter estimation, Gaussian Prior smoothing, a C++ API, a Python extension module, a command line utility, and good documentation. A Java version is also available.

Nyquist

- ◊ Web site: www-2.cs.cmu.edu/~music/nyquist/

The Computer Music Project at CMU is developing computer music and interactive performance technology to enhance human musical experience and creativity. This interdisciplinary effort draws on Music Theory, Cognitive Science, Artificial Intelligence and Machine Learning, Human Computer Interaction, Real-Time Systems, Computer Graphics and Animation, Multimedia, Programming Languages, and Signal Processing. A paradigmatic example of these interdisciplinary efforts is the creation of interactive performances that couple human musical improvisation with intelligent computer agents in real-time.

OpenCyc

- ◊ Web site: http://www.opencyc.org/
- ◊ Alt Web site: http://sourceforge.net/projects/opencyc/

OpenCyc is the open source version of Cyc, the largest and most complete general knowledge base and commonsense reasoning engine. An ontology based on 6000 concepts and 60000 assertions about them.

Pattern

◊ Web site: http://www.clips.ua.ac.be/pages/pattern

Pattern is a web mining module for the Python programming language. It bundles tools for data retrieval (Google + Twitter + Wikipedia API, web spider, HTML DOM parser), text analysis (rule-based shallow parser, WordNet interface, syntactical + semantical n-gram search algorithm, tf-idf + cosine similarity + LSA metrics) and data visualization (graph networks).

PowerLoom

◊ Web site: http://www.isi.edu/isd/LOOM/PowerLoom/

PowerLoom is the successor to the Loom knowledge representation system. It provides a language and environment for constructing intelligent, knowledge-based applications. PowerLoom uses a fully expressive, logic-based representation language (a variant of KIF). It uses a natural deduction inference engine that combines forward and backward chaining to derive what logically follows from the facts and rules asserted in the knowledge base. While PowerLoom is not a description logic, it does have a description classifier which uses technology derived from the Loom classifier to classify descriptions expressed in full first order predicate calculus (see paper). PowerLoom uses modules as a structuring device for knowledge bases, and ultra-lightweight worlds to support hypothetical reasoning.

To implement PowerLoom we developed a new programming language called STELLA, which is a Strongly Typed, Lisp-like LAnguage that can be translated into Lisp, C++ and Java. PowerLoom is written in STELLA and therefore available in Common-Lisp, C++ and Java versions.

PyCLIPS

◊ Web site: http://pyclips.sourceforge.net/web/

PyCLIPS is an extension module for the Python language that embeds full CLIPS functionality in Python applications. This means that you can provide Python with a strong, reliable, widely used and well documented inference engine.

Pyke

◊ Web site: http://pyke.sourceforge.net/

Pyke is a knowledge-based inference engine (expert system) written in 100% python that can:

◊ Do both forward-chaining (data driven) and backward-chaining (goal directed) inferencing.
  · Pyke may be embedded into any python program.
◊ Automatically generate python programs by assembling individual python functions into complete call graphs.
  · This is done through a unique design where the individual python functions are attached to backward-chaining rules.
  · Unlike other approaches to code reuse (e.g. Zope adapters and generic functions), this allows the inference engine to ensure that all of the function's requirements are completely satisfied, by examining the entire call graph down to the leaves, before any of the functions are executed.
  · This is an optional feature. You don't need to use it if you just want the inferencing capability by itself.

python-dlp

◊ Web site: http://code.google.com/p/python-dlp/

python-dlp aims to be a contemporary expert system based on the Semantic Web technologies. Traditionally, expert systems are an application of computing and artificial intelligence with the aim of supporting software that attempts to reproduce the deterministic behavior of one or more human experts in a specific problem domain. It utilizes the efficient RETE_UL algorithm as the 'engine' for the expert system

Reverend

◊ Web site: http://sourceforge.net/projects/reverend/

Reverned is a general purpose Bayesian classifier written in Python. It is designed to be easily extended to any application domain.

Screamer

◊ Latest version is part of CLOCC: clocc.sourceforge.net

Screamer is an extension of Common Lisp that adds support for nondeterministic programming. Screamer consists of two levels. The basic nondeterministic level adds support for backtracking and undoable side effects. On top of this nondeterministic substrate, Screamer provides a comprehensive constraint programming language in which one can formulate and solve mixed systems of numeric and symbolic constraints. Together, these two levels augment Common Lisp with practically all of the functionality of both Prolog and constraint logic programming languages such as CHiP and CLP(R). Furthermore, Screamer is fully integrated with Common Lisp. Screamer programs can coexist and interoperate with other extensions to Common Lisp such as CLOS, CLIM and Iterate.

SimpleAI

◊ Web site: https://github.com/simpleai-team/simpleai

Python library that implements many of the artificial intelligence algorithms described on the book "Artificial Intelligence, a Modern Approach", from Stuart Russel and Peter Norvig. Emphasis on creating a stable, modern, and maintainable version. We are testing the majority of the lib, it's available via pip install, has a standard repo and lib architecture, well documented, respects the python pep8 guidelines, provides only working code (no placeholders for future things), etc. Even the internal code is written with readability in mind, not only the external API.

There is also https://code.google.com/p/aima-python/ which implements these algorithms as well. Though it hasn't seen activity in a while.

SPASS

◊ Web site: http://www.spass-prover.org/

SPASS: An Automated Theorem Prover for First-Order Logic with Equality

If you are interested in first-order logic theorem proving, the formal analysis of software, systems, protocols, formal approaches to AI planning, decision procedures, modal logic theorem proving, SPASS may offer you the right functionality.

Torch

◊ Web site: www.torch.ch
◊ Successor: Torch5

Torch is a machine-learning library, written in C++. Its aim is to provide the state-of-the-art of the best algorithms. It is, and it will be, in development forever.

- ◊ Many gradient-based methods, including multi-layered perceptrons, radial basis functions, and mixtures of experts. Many small "modules" (Linear module, Tanh module, SoftMax module, ...) can be plugged together.
- ◊ Support Vector Machine, for classification and regression.
- ◊ Distribution package, includes Kmeans, Gaussian Mixture Models, Hidden Markov Models, and Bayes Classifier, and classes for speech recognition with embedded training.
- ◊ Ensemble models such as Bagging and Adaboost.
- ◊ Non-parametric models such as K-nearest-neighbors, Parzen Regression and Parzen Density Estimator.

Torch◊s an open library whose authors encourage everybody to develop new packages to be included in future versions on the official website.

2.2 AI software kits, applications, etc.

These are various applications, software kits, etc. meant for research in the field of artificial intelligence. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

ASA - Adaptive Simulated Annealing

- ◊ Web site: http://www.ingber.com/#ASA-CODE

ASA (Adaptive Simulated Annealing) is a powerful global optimization C-code algorithm especially useful for nonlinear and/or stochastic systems.

ASA is developed to statistically find the best global fit of a nonlinear non-convex cost-function over a D-dimensional space. This algorithm permits an annealing schedule for 'temperature' T decreasing exponentially in annealing-time k, T = T_0 exp(-c k^1/D). The introduction of re-annealing also permits adaptation to changing sensitivities in the multi-dimensional parameter-space. This annealing schedule is faster than fast Cauchy annealing, where T = T_0/k, and much faster than Boltzmann annealing, where T = T_0/ln k.

Babylon

- ◊ Archive: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/babylon/

BABYLON is a modular, configurable, hybrid environment for developing expert systems. Its features include objects, rules with forward and backward chaining, logic (Prolog) and constraints. BABYLON is implemented and embedded in Common Lisp.

cfengine

- ◊ Web site: www.iu.hio.no/cfengine/

Cfengine, or the configuration engine is a very high level language for building expert systems which administrate and configure large computer networks. Cfengine uses the idea of classes and a primitive form of intelligence to define and automate the configuration of large systems in the most economical way possible. Cfengine is design to be a part of computer immune systems.

CLIPS

◊ Web site: http://clipsrules.sourceforge.net/

CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems.

CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented and procedural. Rule-based programming allows knowledge to be represented as heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. Object-oriented programming allows complex systems to be modeled as modular components (which can be easily reused to model other systems or to create new components). The procedural programming capabilities provided by CLIPS are similar to capabilities found in languages such as C, Pascal, Ada, and LISP.

EMA-XPS - A Hybrid Graphic Expert System Shell

◊ Web site: http://ema-xps.org/

EMA-XPS is a hybrid graphic expert system shell based on the ASCII-oriented shell Babylon 2.3 of the German National Research Center for Computer Sciences (GMD). In addition to Babylon's AI-power (object oriented data representation, forward and backward chained rules - collectible into sets, horn clauses, and constraint networks) a graphic interface based on the X11 Window System and the OSF/Motif Widget Library has been provided.

Eprover

◊ Web site: http://www.eprover.org/

◊ Web site: http://www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.html

The E Equational Theorem Prover is a purely equational theorem prover. The core proof procedure operates on formulas in clause normal form, using a calculus that combines superposition (with selection of negative literals) and rewriting. No special rules for non-equational literals have been implemented, i.e., resolution is simulated via paramodulation and equality resolution. The basic calculus is extended with rules for AC redundancy elemination, some contextual simplification, and pseudo-splitting. The latest version of E also supports simultaneous paramodulation, either for all inferences or for selected inferences.

E is based on the DISCOUNT-loop variant of the given-clause algorithm, i.e. a strict separation of active and passive facts. Proof search in E is primarily controlled by a literal selection strategy, a clause evaluation heuristic, and a simplification ordering. The prover supports a large number of preprogrammed literal selection strategies, many of which are only experimental. Clause evaluation heuristics can be constructed on the fly by combining various parameterized primitive evaluation functions, or can be selected from a set of predefined heuristics. Supported term orderings are several parameterized instances of Knuth-Bendix-Ordering (KBO) and Lexicographic Path Ordering (LPO).

FOOL & FOX

◊ Web site: rhaug.de/fool/

◊ FTP site: ftp.informatik.uni-oldenburg.de/pub/fool/

FOOL stands for the Fuzzy Organizer OLdenburg. It is a result from a project at the University of Oldenburg. FOOL is a graphical user interface to develop fuzzy rulebases. FOOL will help you to invent and maintain a database that specifies the behavior of a fuzzy-controller or something like that.

FOX is a small but powerful fuzzy engine which reads this database, reads some input values and calculates the new control value.

FreeHAL

◊ Web site: https://freehal.net

FreeHAL is a self-learning conversation simulator which uses semantic nets to organize its knowledge.

FreeHAL uses a semantic network, pattern matching, stemmers, part of speech databases, part of speech taggers, and Hidden Markov Models. Both the online and the download version support TTS.

FUF and SURGE

◊ Web site: http://www.cs.bgu.ac.il/surge/index.html

FUF is an extended implementation of the formalism of functional unification grammars (FUGs) introduced by Martin Kay specialized to the task of natural language generation. It adds the following features to the base formalism:

◊ Types and inheritance.
◊ Extended control facilities (goal freezing, intelligent backtracking).
◊ Modular syntax.

These extensions allow the development of large grammars which can be processed efficiently and can be maintained and understood more easily. SURGE is a large syntactic realization grammar of English written in FUF. SURGE is developed to serve as a black box syntactic generation component in a larger generation system that encapsulates a rich knowledge of English syntax. SURGE can also be used as a platform for exploration of grammar writing with a generation perspective.

GATE

◊ Web site: http://gate.ac.uk/
◊ Alt site: http://sourceforge.net/projects/gate

GATE (General Architecture for Text Engineering) is an architecture, framework and development environment for developing, evaluating and embedding Human Language Technology.

GATE is made up of three elements:

◊ An architecture describing how language processing systems are made up of components.
◊ A framework (or class library, or SDK), written in Java and tested on Linux, Windoze and Solaris.
◊ A graphical development environment built on the framework.

The Grammar Workbench

◊ Web site: ??? www.cs.kun.nl/agfl/

Seems to be obsolete??? Its gone from the site, though its parent project is still ongoing.

The Grammar Workbench, or GWB for short, is an environment for the comfortable development of Affix Grammars in the AGFL-formalism. Its purposes are:

◊ to allow the user to input, inspect and modify a grammar;
◊ to perform consistency checks on the grammar;

◊ to compute grammar properties;
◊ to generate example sentences;
◊ to assist in performing grammar transformations.

GSM Suite

◊ Alt site: www.ibiblio.org/pub/Linux/apps/graphics/draw/

The GSM Suite is a set of programs for using Finite State Machines in a graphical fashion. The suite consists of programs that edit, compile, and print state machines. Included in the suite is an editor program, gsmedit, a compiler, gsm2cc, that produces a C++ implementation of a state machine, a PostScript generator, gsm2ps, and two other minor programs. GSM is licensed under the GNU Public License and so is free for your use under the terms of that license.

Isabelle

◊ Web site: isabelle.in.tum.de

Isabelle is a popular generic theorem prover developed at Cambridge University and TU Munich. Existing logics like Isabelle/HOL provide a theorem proving environment ready to use for sizable applications. Isabelle may also serve as framework for rapid prototyping of deductive systems. It comes with a large library including Isabelle/HOL (classical higher-order logic), Isabelle/HOLCF (Scott's Logic for Computable Functions with HOL), Isabelle/FOL (classical and intuitionistic first-order logic), and Isabelle/ZF (Zermelo-Fraenkel set theory on top of FOL).

Jess, the Java Expert System Shell

◊ Web site: herzberg.ca.sandia.gov/jess/

Jess is a clone of the popular CLIPS expert system shell written entirely in Java. With Jess, you can conveniently give your applets the ability to 'reason'. Jess is compatible with all versions of Java starting with version 1.0.2. Jess implements the following constructs from CLIPS: defrules, deffunctions, defglobals, deffacts, and deftemplates.

learn

◊ Web site: www.ibiblio.org/pub/Linux/apps/cai/

Learn is a vocable learning program with memory model.

LISA

◊ Web site: lisa.sourceforge.net

LISA (Lisp-based Intelligent Software Agents) is a production-rule system heavily influenced by JESS (Java Expert System Shell). It has at its core a reasoning engine based on the Rete pattern matching algorithm. LISA also provides the ability to reason over ordinary CLOS objects.

Livingstone2

◊ Web site: http://ti.arc.nasa.gov/opensource/projects/livingstone2/

Livingstone2 (L2) is a reusable artificial intelligence (AI) software system designed to assist spacecraft, life support systems, chemical plants or other complex systems in operating robustly with minimal human supervision, even in the face of hardware failures or unexpected events.

NICOLE

◊ Web site: http://nicole.sourceforge.net/

NICOLE (Nearly Intelligent Computer Operated Language Examiner) is a theory or experiment that if a computer is given enough combinations of how words, phrases and sentences are related to one another, it could talk back to you. It is an attempt to simulate a conversation by learning how words are related to other words. A human communicates with NICOLE via the keyboard and NICOLE responds back with its own sentences which are automatically generated, based on what NICOLE has stored in it's database. Each new sentence that has been typed in, and NICOLE doesn't know about, is included into NICOLE's database, thus extending the knowledge base of NICOLE.

Otter: An Automated Deduction System

◊ Web site: http://www-unix.mcs.anl.gov/AR/otter/

Our current automated deduction system Otter is designed to prove theorems stated in first-order logic with equality. Otter's inference rules are based on resolution and paramodulation, and it includes facilities for term rewriting, term orderings, Knuth-Bendix completion, weighting, and strategies for directing and restricting searches for proofs. Otter can also be used as a symbolic calculator and has an embedded equational programming system.

PVS

◊ Web site: pvs.csl.sri.com/

PVS is a verification system: that is, a specification language integrated with support tools and a theorem prover. It is intended to capture the state-of-the-art in mechanized formal methods and to be sufficiently rugged that it can be used for significant applications. PVS is a research prototype: it evolves and improves as we develop or apply new capabilities, and as the stress of real use exposes new requirements.

SNePS

◊ Web site: www.cse.buffalo.edu/sneps/

The long-term goal of The SNePS Research Group is the design and construction of a natural-language-using computerized cognitive agent, and carrying out the research in artificial intelligence, computational linguistics, and cognitive science necessary for that endeavor. The three-part focus of the group is on knowledge representation, reasoning, and natural-language understanding and generation. The group is widely known for its development of the SNePS knowledge representation/reasoning system, and Cassie, its computerized cognitive agent.

Soar

◊ Web site: sitemaker.umich.edu/soar

Soar has been developed to be a general cognitive architecture. We intend ultimately to enable the Soar architecture to:

◊ work on the full range of tasks expected of an intelligent agent, from highly routine to extremely difficult, open-ended problems
◊ represent and use appropriate forms of knowledge, such as procedural, declarative, episodic, and possibly iconic
◊ employ the full range of problem solving methods
◊ interact with the outside world and
◊ learn about all aspects of the tasks and its performance on them.

In other words, our intention is for Soar to support all the capabilities required of a general intelligent agent.

TCM

◊ Web site: http://wwwhome.cs.utwente.nl/~tcm/

TCM (Toolkit for Conceptual Modeling) is our suite of graphical editors. TCM contains graphical editors for Entity-Relationship diagrams, Class-Relationship diagrams, Data and Event Flow diagrams, State Transition diagrams, Jackson Process Structure diagrams and System Network diagrams, Function Refinement trees and various table editors, such as a Function-Entity table editor and a Function Decomposition table editor. TCM is easy to use and performs numerous consistency checks, some of them immediately, some of them upon request.

Yale

◊ Web site: yale.sf.net/
◊ Alt Web site: rapid-i.com/

YALE (Yet Another Learning Environment) is an environment for machine learning experiments. Experiments can be made up of a large number of arbitrarily nestable operators and their setup is described by XML files which can easily created with a graphical user interface. Applications of YALE cover both research and real-world learning tasks.

WEKA

◊ Web site: lucy.cs.waikato.ac.nz/~ml/

WEKA (Waikato Environment for Knowledge Analysis) is an state-of-the-art facility for applying machine learning techniques to practical problems. It is a comprehensive software "workbench" that allows people to analyse real-world data. It integrates different machine learning tools within a common framework and a uniform user interface. It is designed to support a "simplicity-first" methodology, which allows users to experiment interactively with simple machine learning tools before looking for more complex solutions.

3. Connectionism

Connectionism is a technical term for a group of related techniques. These techniques include areas such as Artificial Neural Networks, Semantic Networks and a few other similar ideas. My present focus is on neural networks (though I am looking for resources on the other techniques). Neural networks are programs designed to simulate the workings of the brain. They consist of a network of small mathematical-based nodes, which work together to form patterns of information. They have tremendous potential and currently seem to be having a great deal of success with image processing and robot control.

3.1 Connectionist class/code libraries

These are libraries of code or classes for use in programming within the Connectionist field. They are not meant as stand alone applications, but rather as tools for building your own applications.

Software for Flexible Bayesian Modeling

◊ Web site: www.cs.utoronto.ca/~radford/fbm.software.html

This software implements flexible Bayesian models for regression and classification applications that are based on multilayer perceptron neural networks or on Gaussian processes. The implementation uses Markov chain Monte Carlo methods. Software modules that support Markov chain sampling are included in the distribution, and may be useful in other applications.

BELIEF

◊ Web site: www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/probabl/belief/

BELIEF is a Common Lisp implementation of the Dempster and Kong fusion and propagation algorithm for Graphical Belief Function Models and the Lauritzen and Spiegelhalter algorithm for Graphical Probabilistic Models. It includes code for manipulating graphical belief models such as Bayes Nets and Relevance Diagrams (a subset of Influence Diagrams) using both belief functions and probabilities as basic representations of uncertainty. It uses the Shenoy and Shafer version of the algorithm, so one of its unique features is that it supports both probability distributions and belief functions. It also has limited support for second order models (probability distributions on parameters).

bpnn.py

◊ Web site: http://arctrix.com/nas/python/bpnn.py

A simple back-propogation ANN in Python.

brain

◊ Web site: http://harthur.github.com/brain/

Brain is a lightweight JavaScript library for neural networks. It implements the standard feedforward multi-layer perceptron neural network trained with backpropagation.

brain-simulator

◊ Web site: http://www.briansimulator.org/

Brian is a clock-driven simulator for spiking neural networks. It is designed with an emphasis on flexibility and extensibility, for rapid development and refinement of neural models. Neuron models are specified by sets of user-specified differential equations, threshold conditions and reset conditions (given as strings). The focus is primarily on networks of single compartment neuron models (e.g. leaky integrate-and-fire or Hodgkin-Huxley type neurons). It is written in Python and is easy to learn and use, highly flexible and easily extensible. Features include:

◊ a system for specifying quantities with physical dimensions
◊ exact numerical integration for linear differential equations
◊ Euler, Runge-Kutta and exponential Euler integration for nonlinear differential equations
◊ synaptic connections with delays
◊ short-term and long-term plasticity (spike-timing dependent plasticity)
◊ a library of standard model components, including integrate-and-fire equations, synapses and ionic currents
◊ a toolbox for automatically fitting spiking neuron models to electrophysiological recordings

CNNs

◊ Web site: http://www.isiweb.ee.ethz.ch/haenggi/CNNsim.html
◊ Newer Version: http://www.isiweb.ee.ethz.ch/haenggi/CNNsim_adv_manual.html

◊ Old Page: http://www.ce.unipr.it/research/pardis/CNN/cnn.html

Cellular Neural Networks (CNN) is a massive parallel computing paradigm defined in discrete N-dimensional spaces. A visualizing CNN Simulator which allows to track the way in which the state trajectories evolve, thus gaining an insight into the behavior of CNN dynamics. This may be useful for forming an idea how a CNN 'works', especially for those people who are not experienced in CNN theory.

CONICAL

◊ Web site: strout.net/conical/

CONICAL is a C++ class library for building simulations common in computational neuroscience. Currently its focus is on compartmental modeling, with capabilities similar to GENESIS and NEURON. A model neuron is built out of compartments, usually with a cylindrical shape. When small enough, these open-ended cylinders can approximate nearly any geometry. Future classes may support reaction-diffusion kinetics and more. A key feature of CONICAL is its cross-platform compatibility; it has been fully co-developed and tested under Unix, DOS, and Mac OS.

Encog

◊ Web site: http://www.heatonresearch.com/

Encog is an advanced neural network and machine learning framework. Encog contains classes to create a wide variety of networks, as well as support classes to normalize and process data for these neural networks. Encog trains using multithreaded resilient propagation. Encog can also make use of a GPU to further speed processing time. A GUI based workbench is also provided to help model and train neural networks. Encog has been in active development since 2008. Encog is available for Java, .Net and Silverlight.

FANN

◊ Web site: http://leenissen.dk/fann/

Fast Artificial Neural Network Library is a free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks. Cross-platform execution in both fixed and floating point are supported. It includes a framework for easy handling of training data sets. It is easy to use, versatile, well documented, and fast. PHP, C++, .NET, Ada, Python, Delphi, Octave, Ruby, Prolog Pure Data and Mathematica bindings are available. A reference manual accompanies the library with examples and recommendations on how to use the library. A graphical user interface is also available for the library.

ffnet

◊ Web site: http://ffnet.sourceforge.net/

ffnet is a fast and easy-to-use feed-forward neural network training solution for python. Many nice features are implemented: arbitrary network connectivity, automatic data normalization, very efficient training tools, network export to fortran code.

Joone

◊ Web site: http://sourceforge.net/projects/joone/

Joone is a neural net framework to create, train and test neural nets. The aim is to create a distributed environment based on JavaSpaces both for enthusiastic and professional users, based on the newest Java technologies. Joone is composed of a central engine that is the fulcrum of all applications that

already exist or will be developed. The neural engine is modular, scalable, multitasking and tensile. Everyone can write new modules to implement new algorithms or new architectures starting from the simple components distributed with the core engine. The main idea is to create the basis to promote a zillion of AI applications that revolve around the core framework.

Matrix Class

◊ FTP site: ftp.cs.ucla.edu/pub/

A simple, fast, efficient C++ Matrix class designed for scientists and engineers. The Matrix class is well suited for applications with complex math algorithms. As an demonstration of the Matrix class, it was used to implement the backward error propagation algorithm for a multi-layer feed-forward artificial neural network.

NEAT

◊ Web site: http://nn.cs.utexas.edu/project-view.php?RECORD_KEY(Projects)=ProjID&ProjID(Projects)=14

◊ Web site: http://www.cs.ucf.edu/~kstanley/neat.html

Many neuroevolution methods evolve fixed-topology networks. Some methods evolve topologies in addition to weights, but these usually have a bound on the complexity of networks that can be evolved and begin evolution with random topologies. This project is based on a neuroevolution method called NeuroEvolution of Augmenting Topologies (NEAT) that can evolve networks of unbounded complexity from a minimal starting point.

The research as a broader goal of showing that evolving topologies is necessary to achieve 3 major goals of neuroevolution: (1) Continual coevolution: Successful competitive coevolution can use the evolution of topologies to continuously elaborate strategies. (2) Evolution of Adaptive Networks: The evolution of topologies allows neuroevolution to evolve adaptive networks with plastic synapses by designating which connections should be adaptive and in what ways. (3) Combining Expert Networks: Separate expert neural networks can be fused through the evolution of connecting neurons between them.

NeuroLab

◊ Web site: http://packages.python.org/neurolab/

NeuroLab - a library of basic neural networks algorithms with flexible network configurations and learning algorithms for Python. To simplify the using of the library, interface is similar to the package of Neural Network Toolbox (NNT) of MATLAB (c). The library is based on the package numpy (http://numpy.scipy.org), some learning algorithms are used scipy.optimize (http://scipy.org).

NuPIC

◊ Web site: http://www.numenta.org/

◊ Web site: https://github.com/numenta/nupic

The Numenta Platform for Intelligent Computing (NuPIC) is built around Cortical learning algorithms, a new variation of HTM networks (Hierarchical Temporal Memory). Based on Jeff Hawkins idea as laid out in his On Intelligence book. NuPIC consists of the Numenta Tools Framework and the Numenta Runtime Engine.

Pulcinella

◊ Web site: iridia.ulb.ac.be/pulcinella/

Pulcinella is written in CommonLisp, and appears as a library of Lisp functions for creating, modifying and evaluating valuation systems. Alternatively, the user can choose to interact with Pulcinella via a graphical interface (only available in Allegro CL). Pulcinella provides primitives to build and evaluate uncertainty models according to several uncertainty calculi, including probability theory, possibility theory, and Dempster-Shafer's theory of belief functions; and the possibility theory by Zadeh, Dubois and Prade's. A User's Manual is available on request.

scnANNlib

◊ Web site: www.sentinelchicken.org/projects/scnANNlib/

SCN Artificial Neural Network Library provides a programmer with a simple object-oriented API for constructing ANNs. Currently, the library supports non-recursive networks with an arbitrary number of layers, each with an arbitrary number of nodes. Facilities exist for training with momentum, and there are plans to gracefully extend the functionality of the library in later releases.

UTCS Neural Nets Research Group Software

◊ Web site: http://nn.cs.utexas.edu/soft-list.php

A bit different from the other entries, this is a reference to a collection of software rather than one application. It was all developed by the UTCS Neural Net Research Group. Here's a summary of some of the packages available:

- ◊ Natural Language Processing
  - · MIR - Tcl/Tk-based rapid prototyping for sentence processing
  - · SPEC - Parsing complex sentences
  - · DISCERN - Processing script-based stories, including
    - PROC - Parsing, generation, question answering
    - HFM - Episodic memory organization
    - DISLEX - Lexical processing
    - DISCERN - The full integrated model
  - · FGREPNET - Learning distributed representations
- ◊ Self-Organization
  - · LISSOM - Maps with self-organizing lateral connections.
  - · FM - Generic Self-Organizing Maps
- ◊ Neuroevolution
  - · Enforced Sub-Populations (ESP) for sequential decision tasks
    - Non-Markov Double Pole Balancing
  - · Symbiotic, Adaptive NeuroEvolution (SANE; predecessor of ESP)
    - JavaSANE - Java software package for applying SANE to new tasks
    - SANE-C - C version, predecessor of JavaSANE
    - Pole Balancing - Neuron-level SANE on the Pole Balancing task
  - · NeuroEvolution of Augmenting Topologies (NEAT) software for evolving neural networks using structure

Various (C++) Neural Networks

◊ Web site: www.dontveter.com/nnsoft/nnsoft.html

Example neural net codes from the book, The Pattern Recognition Basics of AI. These are simple example codes of these various neural nets. They work well as a good starting point for simple experimentation and for learning what the code is like behind the simulators. The types of networks available on this site are: (implemented in C++)

◊ The Backprop Package
◊ The Nearest Neighbor Algorithms
◊ The Interactive Activation Algorithm
◊ The Hopfield and Boltzman machine Algorithms
◊ The Linear Pattern Classifier
◊ ART I
◊ Bi-Directional Associative Memory
◊ The Feedforward Counter-Propagation Network

3.2 Connectionist software kits/applications

These are various applications, software kits, etc. meant for research in the field of Connectionism. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

Aspirin - MIGRAINES

(am6.tar.Z on ftp site)

◊ FTP site: sunsite.unc.edu/pub/academic/computer-science/neural-networks/programs/Aspirin/

The software that we are releasing now is for creating, and evaluating, feed-forward networks such as those used with the backpropagation learning algorithm. The software is aimed both at the expert programmer/neural network researcher who may wish to tailor significant portions of the system to his/her precise needs, as well as at casual users who will wish to use the system with an absolute minimum of effort.

DDLab

◊ Web site: http://www.ddlab.com/

DDLab is an interactive graphics program for research into the dynamics of finite binary networks, relevant to the study of complexity, emergent phenomena, neural networks, and aspects of theoretical biology such as gene regulatory networks. A network can be set up with any architecture between regular CA (1d or 2d) and "random Boolean networks" (networks with arbitrary connections and heterogeneous rules). The network may also have heterogeneous neighborhood sizes.

Emergent

◊ Web site: http://grey.colorado.edu/emergent/index.php/Main_Page

Note: this is a descendant of PDP++

emergent is a comprehensive, full-featured neural network simulator that allows for the creation and analysis of complex, sophisticated models of the brain in the world. With an emphasis on qualitative analysis and teaching, it also supports the workflow of professional neural network researchers. The GUI environment allows users to quickly construct basic networks, modify the input/output patterns, automatically generate the basic programs required to train and test the network, and easily utilize several data processing and network analysis tools. In addition to the basic preset network train and test programs, the high level drag-and-drop programming interface, built on top of a scripting language that has full introspective access to all aspects of networks and the software itself, allows one to write programs that seamlessly weave together the training of a network and evolution of its environment without ever typing out a line of code. Networks and all of their state variables are

visually inspected in 3D, allowing for a quick "visual regression" of network dynamics and robot behavior.

GENESIS

◊ Web site: http://genesis-sim.org/

GENESIS (short for GEneral NEural SImulation System) is a general purpose simulation platform which was developed to support the simulation of neural systems ranging from complex models of single neurons to simulations of large networks made up of more abstract neuronal components. GENESIS has provided the basis for laboratory courses in neural simulation at both Caltech and the Marine Biological Laboratory in Woods Hole, MA, as well as several other institutions. Most current GENESIS applications involve realistic simulations of biological neural systems. Although the software can also model more abstract networks, other simulators are more suitable for backpropagation and similar connectionist modeling.

JavaBayes

◊ Web site: http://www.cs.cmu.edu/~javabayes/

The JavaBayes system is a set of tools, containing a graphical editor, a core inference engine and a parser. JavaBayes can produce:

◊ the marginal distribution for any variable in a network.
◊ the expectations for univariate functions (for example, expected value for variables).
◊ configurations with maximum a posteriori probability.
◊ configurations with maximum a posteriori expectation for univariate functions.

Jbpe

◊ Web site: cs.felk.cvut.cz/~koutnij/studium/jbpe.html

Jbpe is a back-propagation neural network editor/simulator.

Features

◊ Standart back-propagation networks creation.
◊ Saving network as a text file, which can be edited and loaded back.
◊ Saving/loading binary file
◊ Learning from a text file (with structure specified below), number of learning periods / desired network energy can be specified as a criterion.
◊ Network recall

Nengo

◊ Web site: http://www.nengo.ca/

Nengo (Nengo Neural Simulator) is a graphical and scripting based software package for simulating large-scale neural systems.

To use it, you define groups of neurons in terms of what they represent, and then form connections between neural groups in terms of what computation should be performed on those representations. Nengo then uses the Neural Engineering Framework (NEF) to solve for the appropriate synaptic connection weights to achieve this desired computation. Nengo also supports various kinds of learning. Nengo helps make detailed spiking neuron models that implement complex high-level cognitive algorithms.

Among other things, Nengo has been used to implement motor control, visual attention, serial recall, action selection, working memory, attractor networks, inductive reasoning, path integration, and planning with problem solving.

The Spaun http://models.nengo.ca/spaun neural simulator is implemented in Nengo and its source is available as well.

Neural Network Generator

◊ FTP site: ftp.idsia.ch/pub/rafal

The Neural Network Generator is a genetic algorithm for the topological optimization of feedforward neural networks. It implements the Semantic Changing Genetic Algorithm and the Unit-Cluster Model. The Semantic Changing Genetic Algorithm is an extended genetic algorithm that allows fast dynamic adaptation of the genetic coding through population analysis. The Unit-Cluster Model is an approach to the construction of modular feedforward networks with a "backbone" structure.

NOTE: To compile this on Linux requires one change in the Makefiles. You will need to change '-ltermlib' to '-ltermcap'.

NEURON

◊ Web site: www.neuron.yale.edu/

NEURON is an extensible nerve modeling and simulation program. It allows you to create complex nerve models by connecting multiple one-dimensional sections together to form arbitrary cell morphologies, and allows you to insert multiple membrane properties into these sections (including channels, synapses, ionic concentrations, and counters). The interface was designed to present the neural modeler with a intuitive environment and hide the details of the numerical methods used in the simulation.

Neuroph

◊ Web site: http://neuroph.sourceforge.net/

Neuroph is lightweight Java neural network framework to develop common neural network architectures. It contains well designed, open source Java library with small number of basic classes which correspond to basic NN concepts. Also has nice GUI neural network editor to quickly create Java neural network components.

PDP++

◊ Web site: http://archive.cnbc.cmu.edu/Resources/PDP++/PDP++.html
◊ FTP mirror (US): ftp://grey.colorado.edu/pub/oreilly/pdp++/

NOTE: Renamed to Emergent

As the field of Connectionist modeling has grown, so has the need for a comprehensive simulation environment for the development and testing of Connectionist models. Our goal in developing PDP++ has been to integrate several powerful software development and user interface tools into a general purpose simulation environment that is both user friendly and user extensible. The simulator is built in the C++ programming language, and incorporates a state of the art script interpreter with the full expressive power of C++. The graphical user interface is built with the Interviews toolkit, and allows full access to the data structures and processing modules out of which the simulator is built. We have constructed several useful graphical modules for easy interaction with the structure and the contents

of neural networks, and we've made it possible to change and adapt many things. At the programming level, we have set things up in such a way as to make user extensions as painless as possible. The programmer creates new C++ objects, which might be new kinds of units or new kinds of processes; once compiled and linked into the simulator, these new objects can then be accessed and used like any other.

RNS

◊ Web site: www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/rns/

RNS (Recurrent Network Simulator) is a simulator for recurrent neural networks. Regular neural networks are also supported. The program uses a derivative of the back-propagation algorithm, but also includes other (not that well tested) algorithms.

Features include

◊ freely choosable connections, no restrictions besides memory or CPU constraints
◊ delayed links for recurrent networks
◊ fixed values or thresholds can be specified for weights
◊ (recurrent) back-propagation, Hebb, differential Hebb, simulated annealing and more
◊ patterns can be specified with bits, floats, characters, numbers, and random bit patterns with Hamming distances can be chosen for you
◊ user definable error functions
◊ output results can be used without modification as input

Semantic Networks in Python

◊ Web site: strout.net/info/coding/python/ai/index.html

The semnet.py module defines several simple classes for building and using semantic networks. A semantic network is a way of representing knowledge, and it enables the program to do simple reasoning with very little effort on the part of the programmer.

The following classes are defined:

◊ **Entity**: This class represents a noun; it is something which can be related to other things, and about which you can store facts.
◊ **Relation**: A Relation is a type of relationship which may exist between two entities. One special relation, "IS_A", is predefined because it has special meaning (a sort of logical inheritance).
◊ **Fact**: A Fact is an assertion that a relationship exists between two entities.

With these three object types, you can very quickly define knowledge about a set of objects, and query them for logical conclusions.

SNNS

◊ Web site: http://www-ra.informatik.uni-tuebingen.de/SNNS/

Stuttgart Neural Net Simulator (version 4.1). An awesome neural net simulator. Better than any commercial simulator I've seen. The simulator kernel is written in C (it's fast!). It supports over 20 different network architectures, has 2D and 3D X-based graphical representations, the 2D GUI has an integrated network editor, and can generate a separate NN program in C. SNNS is very powerful, though a bit difficult to learn at first. To help with this it comes with example networks and tutorials for many of the architectures. ENZO, a supplementary system allows you to evolve your networks with genetic algorithms.

TOOLDIAG

◊ Web site: www.inf.ufes.br/~thomas/home/soft.html
◊ Alt site:
http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/tooldiag/0.html

TOOLDIAG is a collection of methods for statistical pattern recognition. The main area of application is classification. The application area is limited to multidimensional continuous features, without any missing values. No symbolic features (attributes) are allowed. The program in implemented in the 'C' programming language and was tested in several computing environments.

XNBC

◊ Web site: www.b3e.jussieu.fr/xnbc/

XNBC v8 is a simulation tool for the neuroscientists interested in simulating biological neural networks using a user friendly tool.

XNBC is a software package for simulating biological neural networks.

Four neuron models are available, three phenomenologic models (xnbc, leaky integrator and conditional burster) and an ion-conductance based model. Inputs to the simulated neurons can be provided by experimental data stored in files, allowing the creation of `hybrid'' networks.

4. Evolutionary Computing

Evolutionary computing is actually a broad term for a vast array of programming techniques, including genetic algorithms, complex adaptive systems, evolutionary programming, etc. The main thrust of all these techniques is the idea of evolution. The idea that a program can be written that will *evolve* toward a certain goal. This goal can be anything from solving some engineering problem to winning a game.

4.1 EC class/code libraries

These are libraries of code or classes for use in programming within the evolutionary computation field. They are not meant as stand alone applications, but rather as tools for building your own applications.

ANNEvolve

◊ Web site: annevolve.sourceforge.net

A collection of programs using evolved artificial neural networks to solve a series of problems. The long term goal of the project is to advance our level of understanding about simulated evolution as a means to configure and optimize Artificial Neural Nets (ANNs). The medium term goal is to apply our methods to a series of interesting problems such as sail boat piloting and playing the game NIM.

A secondary goal is educational in nature. We attempt to write our software with ample explanation, not just for the user, but for the engineer/programmer/scientist who wants to understand the innermost detail. All of the source code is freely available to anyone to use without restriction.

All of the ANNEvolve software is implemented in C and Python.

EAP

◊ Web site: http://code.google.com/p/deap/

EAP has been built using the Python and UNIX programming philosophies in order to provide a transparent, simple and coherent environment for implementing your favourite evolutionary algorithms. EAP is very easy to use even for those who do not know much about the Python programming language. EAP uses the object oriented paradigm that is provided by Python in order to make development simple and beautiful. It also contains a 15 illustrative and diversified examples, to help newcomers to ramp up very quickly in using this environment.

It includes Genetic Algorithms using any imaginable representation, Genetic Programming with strongly and loosely typed trees in addition to automatically defined functions, Evolution Strategies (including Covariance Matrix Adaptation), multiobjective optimization techniques (NSGA-II and SPEA2), easy parallelization of algorithms and much more like milestones, genealogy, etc.

daga

◊ Web site: garage.cps.msu.edu/software/daga3.2/

daga is an experimental release of a 2-level genetic algorithm compatible with the GALOPPS GA software. It is a meta-GA which dynamically evolves a population of GAs to solve a problem presented to the lower-level GAs. When multiple GAs (with different operators, parameter settings, etc.) are simultaneously applied to the same problem, the ones showing better performance have a higher probability of surviving and "breeding" to the next macro-generation (i.e., spawning new "daughter"-GAs with characteristics inherited from the parental GA or GAs. In this way, we try to encourage good problem-solving strategies to spread to the whole population of GAs.

DEAP

◊ Web site: http://deap.googlecode.com

DEAP is intended to be an easy to use distributed evolutionary algorithm library in the Python language. Its two main components are modular and can be used separately. The first module is a Distributed Task Manager (DTM), which is intended to run on cluster of computers using TCP or a MPI connection. The second part is the Evolutionary Algorithms in Python (EAP) framework. EAP is the evolutionary core of DEAP, it provides data structures, methods and tools to design any kind of evolutionary algorithm. It works in perfect harmony with DTM, allowing easy parallelization of any demanding evolutionary task.

dgpf

◊ Web site: dgpf.sourceforge.net

The Distributed Genetic Programming Framework (DGPF) is a scalable Java environment for heuristic, simulation-based search algorithms of any kind and Genetic Algorithms in special. We use the broad foundation of a search algorithms layer to provide a Genetic Programming system which is able to create Turing-complete code.

It's under the LGPL license. It allows you to use heuristic searches like GA and randomized Hill Climbing for any problem space you like to with just minimal programming effort. Also, you may distribute all these searches over a network, using the client/server, the peer-to-peer, or even a client/server+ peer-to-peer hybrid distribution scheme. You also can construct heterogeneous search algorithms where GA cooperates with Hill Climbing without changing any code.

Ease

◊ Web site: www.sprave.com/Ease/Ease.html

Ease - Evolutionary Algorithms Scripting Evironment - is an extension to the Tcl scripting language, providing commands to create, modify, and evaluate populations of individuals represented by real number vectors and/or bit strings.

EO

◊ Web site: eodev.sourceforge.net

EO is a templates-based, ANSI-C++ compliant evolutionary computation library. It contains classes for any kind of evolutionary computation (specially genetic algorithms) you might come up to. It is component-based, so that if you don't find the class you need in it, it is very easy to subclass existing abstract or concrete class.

Evocosm

◊ Web site: http://www.coyotegulch.com/products/libevocosm/

Evocosm is a set of classes that abstract the fundamental components of an evolutionary algorithm. See the site for details, but here is a simple list of the provided classes; Random Numbers, OpenMP, Floating-Point Chromosomes, Roulettte Wheels, Organisms, Fitness Landscapes, Evocosms, Fitness Scaling, Migration, Selecting Survivors, Reproduction, Mutation Operations.

evolver

◊ Web site: https://launchpad.net/evolver

A Python library for creating and working with genetic algorithms and genetic (evolved) programs. Allows one to do Python-based evolutionary programming; string-based as well as source code tree node optimization/search solution discovery.

FORTRAN GA

◊ Web site: cuaerospace.com/carroll/ga.html

This program is a FORTRAN version of a genetic algorithm driver. This code initializes a random sample of individuals with different parameters to be optimized using the genetic algorithm approach, i.e. evolution via survival of the fittest. The selection scheme used is tournament selection with a shuffling technique for choosing random pairs for mating. The routine includes binary coding for the individuals, jump mutation, creep mutation, and the option for single-point or uniform crossover. Niching (sharing) and an option for the number of children per pair of parents has been added. More recently, an option for the use of a micro-GA has been added.

FREVO

◊ Web site: http://frevo.sourceforge.net/

FREVO is an open-source framework developed in Java to help engineers and scientists in evolutionary design or optimization tasks. The major feature of FREVO is the componentwise decomposition and separation of the key building blocks for each optimization tasks. We identify these as the problem definition, solution representation and the optimization method. This structure enables the components to be designed separately allowing the user to easily swap and evaluate different configurations and methods or to connect an external simulation tool.

GAlib: Matthew's Genetic Algorithms Library

- ◊ Web Site: http://lancet.mit.edu/ga/
- ◊ Download: http://lancet.mit.edu/ga/dist/
- ◊ Register GAlib at: http://lancet.mit.edu/ga/Register.html

GAlib contains a set of C++ genetic algorithm objects. The library includes tools for using genetic algorithms to do optimization in any C++ program using any representation and genetic operators. The documentation includes an extensive overview of how to implement a genetic algorithm as well as examples illustrating customizations to the GAlib classes.

GALOPPS

- ◊ Web site: http://garage.cse.msu.edu/software/galopps/
- ◊ FTP site: ftp://garage.cse.msu.edu/pub/GA/galopps/

GALOPPS is a flexible, generic GA, in 'C'. It was based upon Goldberg's Simple Genetic Algorithm (SGA) architecture, in order to make it easier for users to learn to use and extend.

GALOPPS extends the SGA capabilities several fold:

- ◊ (optional) A new Graphical User Interface, based on TCL/TK, for Unix users, allowing easy running of GALOPPS 3.2 (single or multiple subpopulations) on one or more processors. GUI writes/reads "standard" GALOPPS input and master files, and displays graphical output (during or after run) of user-selected variables.
- ◊ 5 selection methods: roulette wheel, stochastic remainder sampling, tournament selection, stochastic universal sampling, linear-ranking-then-SUS.
- ◊ Random or superuniform initialization of "ordinary" (non-permutation) binary or non-binary chromosomes; random initialization of permutation-based chromosomes; or user-supplied initialization of arbitrary types of chromosomes.
- ◊ Binary or non-binary alphabetic fields on value-based chromosomes, including different user-definable field sizes.
- ◊ 3 crossovers for value-based representations: 1-pt, 2-pt, and uniform, all of which operate at field boundaries if a non-binary alphabet is used.
- ◊ 4 crossovers for order-based reps: PMX, order-based, uniform order-based, and cycle.
- ◊ 4 mutations: fast bitwise, multiple-field, swap and random sublist scramble.
- ◊ Fitness scaling: linear scaling, Boltzmann scaling, sigma truncation, window scaling, ranking.
- ◊ **Plus** a whole lot more....

GAS

- ◊ Web site: http://starship.python.net/crew/gandalf/

GAS means "Genetic Algorithms Stuff". GAS is freeware.

Purpose of GAS is to explore and exploit artificial evolutions. Primary implementation language of GAS is Python. The GAS software package is meant to be a Python framework for applying genetic algorithms. It contains an example application where it is tried to breed Python program strings. This special problem falls into the category of Genetic Programming (GP), and/or Automatic Programming. Nevertheless, GAS tries to be useful for other applications of Genetic Algorithms as well.

GAUL

- ◊ Web site: http://gaul.sourceforge.net/

◊ SF project site: http://sourceforge.net/projects/gaul/

The Genetic Algorithm Utility Library (GAUL) is a flexible programming library designed to aid development of applications that require the use of genetic algorithms. Features include:

◊ Darwinian, Lamarckian or Baldwinian evolutionary schemes.
◊ Both steady-state and generation-based GAs included.
◊ The island model of evolution is available.
◊ Chromosome datatype agnostic. A selection of common chromosome types are built-in.
◊ Allows user-defined crossover, mutation, selection, adaptation and replacement operators.
◊ Support for multiple, simultaneously evolved,populations.
◊ Choice of high-level or low-level interface functions.
◊ Additional, non-GA, optimisation algorithms are built-in for local optimisation or comparative purposes.
◊ Trivial to extend using external code via the built-in code hooks.
◊ May be driven by, or extended by, powerful S-Lang scripts.
◊ Support for multiprocessor calculations.
◊ Written using highly portable C code.

GECO

◊ FTP site: http://common-lisp.net/project/geco/

GECO (Genetic Evolution through Combination of Objects), an extendible object-oriented tool-box for constructing genetic algorithms (in Lisp). It provides a set of extensible classes and methods designed for generality. Some simple examples are also provided to illustrate the intended use.

GenePool

◊ Web site: http://www.rubinsteyn.com/genepool/

GenePool is a framework for writing evolutionary optimization algorithms in OCaml. This library is not a complete solution but rather is a generic skeleton which takes care of the plumbing and nuisances of optimization. You provide GenePool with functions that give meaning to fitness and reproduction and after a specified number of generation, GenePool returns an array of the best "genomes" it evolved.

Genetic

◊ Web site: ???
◊ You can get it from the debian repository: http://packages.qa.debian.org/g/genetic.html

This is a package for genetic algorythms and AI in Python.

Genetic can typically solve ANY problem that consists to minimize a function.

It also includes several demos / examples, like the TSP (traveling saleman problem).

GPdata

◊ FTP site: ftp://ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/gp-code/

GPdata-3.0.tar.gz (C++) contains a version of Andy Singleton's GP-Quick version 2.1 which has been extensively altered to support:

◊ Indexed memory operation (cf. teller)
◊ multi tree programs

- ◊ Adfs
- ◊ parameter changes without recompilation
- ◊ populations partitioned into demes
- ◊ (A version of) pareto fitness

This ftp site also contains a small C++ program (ntrees.cc) to calculate the number of different there are of a given length and given function and terminal set.

gpjpp Genetic Programming in Java

- ◊ The code can be found in the tarball linked from "GP and Othello Java code and READMEs" on this page: http://www1.cs.columbia.edu/~evs/ml/hw4.html

gpjpp is a Java package I wrote for doing research in genetic programming. It is a port of the gpc++ kernel written by Adam Fraser and Thomas Weinbrenner. Included in the package are four of Koza's standard examples: the artificial ant, the hopping lawnmower, symbolic regression, and the boolean multiplexer. Here is a partial list of its features:

- ◊ graphic output of expression trees
- ◊ efficient diversity checking
- ◊ Koza's greedy over-selection option for large populations
- ◊ extensible GPRun class that encapsulates most details of a genetic programming test
- ◊ more robust and efficient streaming code, with automatic checkpoint and restart built into the GPRun class
- ◊ an explicit complexity limit that can be set on each GP
- ◊ additional configuration variables to allow more testing without recompilation
- ◊ support for automatically defined functions (ADFs)
- ◊ tournament and fitness proportionate selection
- ◊ demetic grouping
- ◊ optional steady state population
- ◊ subtree crossover
- ◊ swap and shrink mutation

jaga

- ◊ Web site: cs.felk.cvut.cz/~koutnij/studium/jaga/jaga.html

Simple genetic algorithm package written in Java.

JGAP

- ◊ Web site: http://jgap.sourceforge.net/

JGAP (pronounced "jay-gap") is a Genetic Algorithms and Genetic Programming component provided as a Java framework. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions.

JGAP was designed to be very easy to use "out of the box", while also designed to be highly modular so that more adventurous users can easily plug-in custom genetic operators and other sub-components.

lil-gp

- ◊ Web site: http://garage.cse.msu.edu/software/lil-gp/

patched lil-gp *

◊ Web site: http://cs.gmu.edu/~sean/research/lil-gp-patch/

lil-gp is a generic 'C' genetic programming tool. It was written with a number of goals in mind: speed, ease of use and support for a number of options including:

◊ Generic 'C' program that runs on UNIX workstations
◊ Support for multiple population experiments, using arbitrary and user settable topologies for exchange, for a single processor (i.e., you can do multiple population gp experiments on your PC).
◊ lil-gp manipulates trees of function pointers which are allocated in single, large memory blocks for speed and to avoid swapping.

* The patched lil-gp kernel is strongly-typed, with modifications on multithreading, coevolution, and other tweaks and features.

Lithos

◊ Web site: www.esatclear.ie/~rwallace/lithos.html

Lithos is a stack based evolutionary computation system. Unlike most EC systems, its representation language is computationally complete, while also being faster and more compact than the S-expressions used in genetic programming. The version presented here applies the system to the game of Go, but can be changed to other problems by simply plugging in a different evaluation function. ANSI C source code is provided.

Open BEAGLE

◊ Web site: beagle.gel.ulaval.ca

Open BEAGLE is a C++ evolutionary computation framework. It provides a high-level software environment to do any kind of evolutionary computation, with support for tree-based genetic programming, bit string and real-valued genetic algorithms, evolution strategy, co-evolution, and evolutionary multi-objective optimization.

PGAPack

Parallel Genetic Algorithm Library

◊ FTP site: ftp://ftp.mcs.anl.gov/pub/pgapack/

PGAPack is a general-purpose, data-structure-neutral, parallel genetic algorithm library. It is intended to provide most capabilities desired in a genetic algorithm library, in an integrated, seamless, and portable manner. Key features are in PGAPack V1.0 include:

◊ Callable from Fortran or C.
◊ Runs on uniprocessors, parallel computers, and workstation networks.
◊ Binary-, integer-, real-, and character-valued native data types.
◊ Full extensibility to support custom operators and new data types.
◊ Easy-to-use interface for novice and application users.
◊ Multiple levels of access for expert users.
◊ Parameterized population replacement.
◊ Multiple crossover, mutation, and selection operators.
◊ Easy integration of hill-climbing heuristics.
◊ Extensive debugging facilities.
◊ Large set of example problems.
◊ Detailed users guide.

PIPE

◊ FTP site: ftp.idsia.ch/pub/rafal

Probabilistic Incremental Program Evolution (PIPE) is a novel technique for automatic program synthesis. The software is written in C. It ...

◊ is easy to install (comes with an automatic installation tool).
◊ is easy to use: setting up PIPE_V1.0 for different problems requires a minimal amount of programming. User-written, application-independent program parts can easily be reused.
◊ is efficient: PIPE_V1.0 has been tuned to speed up performance.
◊ is portable: comes with source code (optimized for SunOS 5.5.1).
◊ is extensively documented(!) and contains three example applications.
◊ supports statistical evaluations: it facilitates running multiple experiments and collecting results in output files.
◊ includes testing tool for testing generalization of evolved programs.
◊ supports floating point and integer arithmetic.
◊ has extensive output features.
◊ For lil-gp users: Problems set up for lil-gp 1.0 can be easily ported to PIPE_v1.0. The testing tool can also be used to process programs evolved by lil-gp 1.0.

plop

◊ Web site: http://code.google.com/p/plop/

A Common Lisp framework for experimenting with meta-optimizing semantic evolutionary search (MOSES) and related approaches to learning with probability distributions over program spaces based on:

◊ reduction to normal form
◊ representation-building
◊ deme (sub-population) management
◊ probabilistic model-building (aka estimation-of-distribution algorithms)

Pyevolve

◊ Web site: http://pyevolve.sourceforge.net/

Pyevolve was developed to be a complete genetic algorithm framework, the main objectives of Pyevolve are:

◊ written in pure python, to maximize the cross-platform issue;
◊ easy to use API, the API must be easy for end-user;
◊ see the evolution, the user can and must see and interact with the evolution statistics, graphs and etc;
◊ extensible, the API must be extensible, the user can create new representations, genetic operators like crossover, mutation and etc;
◊ fast, the design must be optimized for performance;
◊ common features, the framework must implement the most common features: selectors like roulette wheel, tournament, ranking, uniform. Scaling schemes like linear scaling, etc;
◊ default parameters, we must have default operators, settings, etc in all options;
◊ open-source, the source is for everyone, not for only one.

pygp

◊ Web site: http://sourceforge.net/projects/pygp/files/

Your basic genetic algorithm package for python.

tinygp

◊ Web site: http://www.laserpirate.com/as3tinygp/

Small genetic programming library in C++ and ActionScript 3 (Javascript engine embedded in Flash) with flash demos.

This GP library uses the standard Koza expression tree program representation. It uses the 'grow' algorithm to generate random expressions. Mutation is performed by selecting a random subexpression in an expression tree, and replacing it with a new random expression (which satisfies the maximum tree depth constraint). Crossover (mating) between two expressions is performed by selecting a random subexpression in each parent, then exchanging them (although it only makes on child, not two).

In addition to the core code for creating, mutating, mating and evaluating expressions, the library includes a steady-state genetic algorithm with tournament selection, and a worst-out, elitist replacement policy (i.e. when a new child is created, it replaces the worse member of the population, only if it is better).

txevolver

◊ Web site: https://launchpad.net/txevolver

A Twisted-based set of libraries for performing calculations of genetic algorithms as well as genetic programs in an asynchronous, distributed manner.

4.2 EC software kits/applications

These are various applications, software kits, etc. meant for research in the field of evolutionary computing. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

ADATE

◊ Web site: www-ia.hiof.no/~rolando/adate_intro.html

ADATE (Automatic Design of Algorithms Through Evolution) is a system for automatic programming i.e., inductive inference of algorithms, which may be the best way to develop artificial and general intelligence.

The ADATE system can automatically generate non-trivial and novel algorithms. Algorithms are generated through large scale combinatorial search that employs sophisticated program transformations and heuristics. The ADATE system is particularly good at synthesizing symbolic, functional programs and has several unique qualities.

esep & xesep

◊ Web site(esep): www.iit.edu/~elrad/esep.html
◊ Web site(xesep): www.iit.edu/~elrad/xesep.html

This is a new scheduler, called Evolution Scheduler, based on Genetic Algorithms and Evolutionary Programming. It lives with original Linux priority scheduler.This means you don't have to reboot to change the scheduling policy. You may simply use the manager program esep to switch between them at any time, and esep itself is an all-in-one for scheduling status, commands, and administration. We didn't intend to remove the original priority scheduler; instead, at least, esep provides you with

another choice to use a more intelligent scheduler, which carries out natural competition in an easy and effective way.

Xesep is a graphical user interface to the esep (Evolution Scheduling and Evolving Processes). It's intended to show users how to start, play, and feel the Evolution Scheduling and Evolving Processes, including sub-programs to display system status, evolving process status, queue status, and evolution scheduling status periodically in as small as one mini-second.

Corewars

◊ SourceForge site: http://sourceforge.net/projects/corewars/

Corewars is a game which simulates a virtual machine with a number of programs. Each program tries to crash the others. The program that lasts the longest time wins. A number of sample programs are provided and new programs can be written by the player. Screenshots are available at the Corewars homepage.

JCASim

◊ Web site: http://www.jweimar.de/jcasim/

JCASim is a general-purpose system for simulating cellular automata in Java. It includes a stand-alone application and an applet for web presentations. The cellular automata can be specified in Java, in CDL, or using an interactive dialogue. The system supports many different lattice geometries (1-D, 2-D square, hexagonal, triangular, 3-D), neighborhoods, boundary conditions, and can display the cells using colors, text, or icons.

JGProg

◊ Web site: jgprog.sourceforge.net

Genetic Programming (JGProg) is an open-source Java implementation of a strongly-typed Genetic Programming experimentation platform. Two example "worlds" are provided, in which a population evolves and solves the problem.

5. Alife & Complex Systems

Alife takes yet another approach to exploring the mysteries of intelligence. It has many aspects similar to EC and Connectionism, but takes these ideas and gives them a meta-level twist. Alife emphasizes the development of intelligence through *emergent* behavior of *complex adaptive systems*. Alife stresses the social or group based aspects of intelligence. It seeks to understand life and survival. By studying the behaviors of groups of 'beings' Alife seeks to discover the way intelligence or higher order activity emerges from seemingly simple individuals. Cellular Automata and Conway's Game of Life are probably the most commonly known applications of this field. Complex Systems (abbreviated CS) are very similar to alife in the way the are approached, just more general in definition (ie. alife is a type of complex system). Usually complex system software takes the form of a simulator.

5.1 Alife & CS class/code libraries

These are libraries of code or classes for use in programming within the artificial life field. They are not meant as stand alone applications, but rather as tools for building your own applications.

AgentFarms

◊ Web site: http://www.agentfarms.net

Agent Farms is a system for modelling and simulation of complex, multi-agent based systems. The system can be used for:

◊ Creating models of multi-agent systems
◊ Interactive and distributed simulation
◊ Observation and visualisation of the simulation
◊ Population modification and migration

Biome

◊ Web site: http://sourceforge.net/projects/biome/

Biome is a C++ library aimed at individual-based/agent-based simulations. It is somewhat similar to Swarm, EcoSim or Simex but tries to be more efficient and less monolithic without compromising object- oriented design. Currently there is an event based scheduling system, a C++ified Mersenne-Twister RNG, several general analysis classes, some Qt-based GUI classes, a very basic persistence/database framework (used also for parameter storage) and many other small useful things.

CAGE

◊ Web site: http://www.alcyone.com/software/cage/

CAGE is a fairy generic and complete cellular automaton simulation engine in Python. It supports both 1D and 2D automata, a variety of prepackaged rules, and the concept of "agents" which can move about independently on the map for implementing agent behavior.

Cellular

◊ Web site: http://zhar.net/howto/homeless/cellular/

The Cellular automata programming system consists of a compiler for the Cellang cellular automata programming language, along with the corresponding documentation, viewer, and various tools. Postscript versions of the tutorial and language reference manual are available for those wanting more detailed information. The most important distinguishing features of Cellang, include support for:

◊ any number of dimensions;
◊ compile time specification of each dimensions size; cell neighborhoods of any size (though bounded at compile time) and shape;
◊ positional and time dependent neighborhoods;
◊ associating multiple values (fields), including arrays, with each cell;
◊ associating a potentially unbounded number of mobile agents [Agents are mobile entities based on a mechanism of the same name in the Creatures system, developed by Ian Stephenson (ian@ohm.york.ac.uk).] with each cell; and
◊ local interactions only, since it is impossible to construct automata that contain any global control or references to global variables.

Integrating Modelling Toolkit

◊ Web site: sourceforge.net/projects/imt/

The Integrating Modelling Toolkit (IMT) is a generic, comprehensive, and extensible set of abstractions allowing definition and use of interoperable model components. Modellers create an IMT "world" made of IMT "agents" that will perform each a particular phase of a modelling task. The core set of IMT agents can describe generic, modular, distributed model components, either native to the

IMT or integrating existing simulation toolkits, specialized for tasks that range from simple calculation of functions in an interpreted language to spatially explicit simulation, model optimization, GIS analysis, visualization and advanced statistical analysis. IMT agents are designed to easily "glue" together in higher-level simulations integrating different modelling paradigms and toolkits. The IMT can be easily extended by users and developers through a convenient plug-in mechanism

MAML

◊ Web site: http://www.maml.hu/

The current version of MAML is basically an extension to Objective-C (using the Swarm libraries). It consists of a couple of 'macro-keywords' that define the general structure of a simulation. The remaining must be filled with pure swarm-code. A MAML-to-Swarm (named xmc) compiler is also being developed which compiles the source code into a swarm application.

MASON

◊ Web site: http://cs.gmu.edu/~eclab/projects/mason/

MASON Stands for Multi-Agent Simulator Of Neighborhoods... or Networks... or something...

MASON is a fast discrete-event multi-agent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulations, and also to provide more than enough functionality for many lightweight simulation needs. MASON contains both a model library and an optional suite of visualization tools in 2D and 3D.

SimWorld

◊ Web site: http://www.nd.edu/ airolab/simworld/
◊ New Web site?: http://hrilab.tufts.edu/

SimWorld is a free artificial life simulation (based on the free SimAgent toolkit developed by Aaron Sloman), which provides functionality for running different interacting agents and objects in a simulated, continuous environment. The agents are controlled by rules written in the powerful rule interpreter. New behaviors of agents can be defined without any programming knowledge.

Swarm

◊ Web site: www.swarm.org/wiki/Swarm_main_page
◊ FTP site: ftp.swarm.org/pub/swarm/

The swarm Alife simulation kit. Swarm is a simulation environment which facilitates development and experimentation with simulations involving a large number of agents behaving and interacting within a dynamic environment. It consists of a collection of classes and libraries written in Objective-C and allows great flexibility in creating simulations and analyzing their results. It comes with three demos and good documentation.

5.2 Alife & CS software kits, applications, etc.

These are various applications, software kits, etc. meant for research in the field of artificial life. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

Achilles

◊ Web site: http://achilles.sourceforge.net/

Achilles is an evolution simulation based on Larry Yaeger's PolyWorld. It uses Hebbian neural networks, and an extremely simplified physical model that allows virtual organisms to interact freely in a simulated environment.

Avida

◊ Web site: http://avida.devosoft.org/
◊ Web site: http://sourceforge.net/projects/avida/

The computer program Avida is an auto-adaptive genetic system designed primarily for use as a platform in Artificial Life research. The Avida system is based on concepts similar to those employed by the Tierra program, that is to say it is a population of self-reproducing strings with a Turing-complete genetic basis subjected to Poisson-random mutations. The population adapts to the combination of an intrinsic fitness landscape (self-reproduction) and an externally imposed (extrinsic) fitness function provided by the researcher. By studying this system, one can examine evolutionary adaptation, general traits of living systems (such as self-organization), and other issues pertaining to theoretical or evolutionary biology and dynamic systems.

Biogenesis

◊ Web site: http://biogenesis.sourceforge.net/

Biogenesis is an artificial life program that simulates the processes involved in the evolution of organisms. It shows colored segment based organisms that mutate and evolve in a 2D environment. Biogenesis is based on Primordial Life.

breve

◊ Web site: www.spiderland.org/breve/

Breve is a free software package which makes it easy to build 3D simulations of decentralized systems and artificial life. Users define the behaviors of agents in a 3D world and observe how they interact. Breve includes physical simulation and collision detection so you can simulate realistic creatures, and an OpenGL display engine so you can visualize your simulated worlds.

BugsX

◊ FTP site: https://github.com/porridge/bugsx

Display and evolve biomorphs. It is a program which draws the biomorphs based on parametric plots of Fourier sine and cosine series and let's you play with them using the genetic algorithm.

Creatures Docking Station

◊ Linux info: http://www.creaturesdockingstation.com/

This is a free version of the Creatures3 ALife game. It has fewer species and a small 'space-station' world, but can connect to other worlds over the internet and (if you have the windows version of the game) can connect to your C3 world. The game itself revolves around breeding and training the alife creatures, 'Norns'. Its strikes a pretty nice balance between fun and science, or so I'm told.

(summary written by Steve Grand included below)

The eponymous creatures in this computer game are called Norns, and the world's population of them at one stage hovered around the five million mark, making them more common than many familiar natural species. Each norn is composed of thousands of tiny simulated biological components, such as neurons, biochemicals, chemoreceptors, chemoemitters and genes. The norns' genes dictate how these components are assembled to make complete organisms, and the creatures' behaviour then emerges from the interactions of those parts, rather than being explicitly 'programmed in'.

The norns are capable of learning about their environment, either by being shown things by their owners or through learning by their own mistakes. They must learn for themselves how to find food and how to interact with the many objects in their environment. They can interact with their owners, using simple language, and also with each other. They can form relationships and produce offspring, which inherit their neural and biochemical structure from their parents and are capable of open-ended evolution over time. They can fall prey to a variety of diseases (as well as genetic defects) and can be treated with appropriate medicines.

Critterding

◊ Web site: http://critterding.sourceforge.net/
◊ Web site (telepathic-critterdrug):
http://m-net.arbornet.org/~flamoot/telepathic-critterdrug.html

Critterding is a "Petri dish" universe in 3D that demonstrates evolving artificial life. Critters start out with completely random brains and bodies, but will automatically start evolving into something with much better survival skills. The lifeforms are placed in an artificial environment where there neural net 'brains' can evolve.

Telepathic-critterdrug is a fork of Critterding introducing a communications medium (called a retina) and psychoactive substances that effect this retina.

dblife & dblifelib

◊ FTP site: ibiblio.org/pub/Linux/science/ai/life/

dblife: Sources for a fancy Game of Life program for X11 (and curses). It is not meant to be incredibly fast (use xlife for that:-). But it IS meant to allow the easy editing and viewing of Life objects and has some powerful features. The related dblifelib package is a library of Life objects to use with the program.

dblifelib: This is a library of interesting Life objects, including oscillators, spaceships, puffers, and other weird things. The related dblife package contains a Life program which can read the objects in the Library.

Drone

◊ Web site: www.cscs.umich.edu/Software/Drone/

Drone is a tool for automatically running batch jobs of a simulation program. It allows sweeps over arbitrary sets of parameters, as well as multiple runs for each parameter set, with a separate random seed for each run. The runs may be executed either on a single computer or over the Internet on a set of remote hosts. Drone is written in Expect (an extension to the Tcl scripting language) and runs under Unix. It was originally designed for use with the Swarm agent-based simulation framework, but Drone can be used with any simulation program that reads parameters from the command line or from an input file.

EcoLab

◊ Web site: http://ecolab.sourceforge.net/

EcoLab is a system that implements an abstract ecology model. It is written as a set of Tcl/Tk commands so that the model parameters can easily be changed on the fly by means of editing a script. The model itself is written in C++.

Framsticks

◊ Web site: http://www.frams.alife.pl/

Framsticks is a three-dimensional life simulation project. Both mechanical structures ("bodies") and control systems ("brains") of creatures are modeled. It is possible to design various kinds of experiments, including simple optimization (by evolutionary algorithms), co-evolution, open-ended and spontaneous evolution, distinct gene pools and populations, diverse genotype/phenotype mappings, and species/ecosystems modeling.

Fluidiom

◊ Web site: http://ostatic.com/fluidiom

Evolutionary based alife platform. Has a game like feel which makes it fun while still allowing for some interesting experimentation.

It takes a minimalist approach to spatial structure to make a body, adds articulation in the form of muscles, and then lets evolution take over to see if these bodies can learn to walk, run, crawl, or slither from one place to the other.

Game Of Life (GOL)

◊ FTP site: ibiblio.org/pub/Linux/science/ai/life/

GOL is a simulator for conway's game of life (a simple cellular automata), and other simple rule sets. The emphasis here is on speed and scale, in other words you can setup large and fast simulations.

gant

◊ Web site: gant.sourceforge.net

This project is an ANSI C++ implementation of the Generalized Langton Ant, which lives on a torus.

gLife

◊ Web site: glife.sourceforge.net
◊ SourceForge site: sourceforge.net/projects/glife/

This program is similiar to "Conway's Game of Life" but yet it is very different. It takes "Conway's Game of Life" and applies it to a society (human society). This means there is a very different (and much larger) rule set than in the original game. Things need to be taken into account such as the terrain, age, sex, culture, movement, etc

Golly

◊ Web site: golly.sourceforge.net

An open source, cross-platform implementation of John Conway's Game of Life with an unbounded universe and capable of running patterns faster and further than ever before. It has many features such

as;

- ◊ Reads RLE, Life 1.05/1.06, and macrocell formats.
- ◊ Supports Wolfram's 1D rules.
- ◊ Can paste in patterns from the clipboard.
- ◊ Scriptable via Python.

Langton's Ant

- ◊ Web site: http://theory.org/software/ant/

Langton's Ant is an example of a finite-state cellular automata. The ant (or ants) start out on a grid. Each cell is either black or white. If the ant is on a black square, it turns right 90 and moves forward one unit. If the ant is on a white square, it turns left 90 and moves forward one unit. And when the ant leaves a square, it inverts the color. The neat thing about Langton's Ant is that no matter what pattern field you start it out on, it eventually builds a "road," which is a series of 117 steps that repeat indefinitely, each time leaving the ant displaced one pixel vertically and horizontally.

LEE

- ◊ Web site: www.informatics.indiana.edu/fil/LEE/

LEE (Latent Energy Environments) is both an Alife model and a software tool to be used for simulations within the framework of that model. We hope that LEE will help understand a broad range of issues in theoretical, behavioral, and evolutionary biology. The LEE tool described here consists of approximately 7,000 lines of C code and runs in both Unix and Macintosh platforms.

MATREM

- ◊ Web site: http://www.romansland.nl/matrem/

Matrem is a computer program that simulates life. It belongs to the emerging science of "artificial life", which studies evolution and complex systems in general by simulation. Matrem is also a game, where players compete to create the fittest life form. Their efforts are the driving force behind the program.

Nanopond

- ◊ Web site: http://adam.ierymenko.name/nanopond.shtml

Nanopond is a "corewar style" evolvable instruction set based virtual machine written in C. It is similar in design to Tierra and Avida but considerably smaller and simpler. Version 1.0 weights in at only 840 lines of C code, the majority of which are comments! It is very highly optimized and supports simple color visualization using the SDL (Simple Directmedia Layer) library.

More information can be learned by reading the Nanopond source code, which is very well commented.

Noble Ape

- ◊ Web site: www.nobleape.com/sim/

The Noble Ape Simulation has been developed (as the Nervana Simulation) since 1996. The aim of the simulation is to create a detailed biological environment and a cognitive simulation. The Simulation is intended as a palette for open source development. It provides a stable means of simulating large scale environments and cognitive processes.

It features a number of autonomous simulation components including a landscape simulation, biological simulation, weather simulation, sentient creature (Noble Ape) simulation and a simple intelligent-agent scripting language (ApeScript).

The code is currently (2007) used by Apple Inc and by INTEL for processor optimization and performance tuning. Apple includes it with their CHUD performance and debugging developer tool set.

Polyworld

◊ Web site: http://sourceforge.net/projects/polyworld

PolyWorld is a computational ecology that I developed to explore issues in Artificial Life. Simulated organisms reproduce sexually, fight and kill and eat each other, eat the food that grows throughout the world, and either develop successful strategies for survival or die. An organism's entire behavioral suite (move, turn, attack, eat, mate, light) is controlled by its neural network "brain". Each brain's architecture--it's neural wiring diagram--is determined from its genetic code, in terms of number, size, and composition of neural clusters (excitatory and inhibitory neurons) and the types of connections between those clusters (connection density and topological mapping). Synaptic efficacy is modulated via Hebbian learning, so, in principle, the organisms have the ability to learn during the course of their lifetimes. The organisms perceive their world through a sense of vision, provided by a computer graphic rendering of the world from each organism's point of view. The organisms' physiologies are also encoded genetically, so both brain and body, and thus all components of behavior, evolve over multiple generations. A variety of "species", with varying individual and group survival strategies have emerged in various simulations, displaying such complex ethological behaviors as swarming/flocking, foraging, and attack avoidance.

POSES++

◊ Web site: http://www.gpc.de/e_poses.html

The POSES++ software tool supports the development and simulation of models. Regarding the simulation technique models are suitable reproductions of real or planned systems for their simulative investigation.

In all industrial sectors or branches POSES++ can model and simulate any arbitrary system which is based on a discrete and discontinuous behaviour. Also continuous systems can mostly be handled like discrete systems e.g., by quantity discretion and batch processing.

Tierra

◊ Web site: http://life.ou.edu/tierra/

Tierra's written in the C programming language. This source code creates a virtual computer and its operating system, whose architecture has been designed in such a way that the executable machine codes are evolve-able. This means that the machine code can be mutated (by flipping bits at random) or recombined (by swapping segments of code between algorithms), and the resulting code remains functional enough of the time for natural (or presumably artificial) selection to be able to improve the code over time.

Trend

◊ Web site: www.complex.iastate.edu/download/Trend/

Trend is a general purpose cellular automata simulation environment with an integrated high level language compiler, a beautiful graphical user interface, and a fast, three stage cached simulation engine. This is the simulation system that was used to discover the first emergent self-replicating cellular automata rule set, and the first problem solving self-replication loop.

Since its simulator is very flexible with regard to cellular space sizes, cell structures, neighborhood structures and cellular automata rules, Trend can simulate almost all one or two-dimensional cellular automata models. It also has a smart backtracking feature which simplifies rule set development a lot by allowing users to go back to a previous stage of simulation! With other advanced features, Trend is probably the most easy to use 2-dimensional cellular automata simulator.

Also available is jTrend. A Java version of Trend.

XLIFE

◊ FTP site: http://linux.maruhn.com/sec/xlife.html

This program will evolve patterns for John Horton Conway's game of Life. It will also handle general cellular automata with the orthogonal neighborhood and up to 8 states (it's possible to recompile for more states, but very expensive in memory). Transition rules and sample patterns are provided for the 8-state automaton of E. F. Codd, the Wireworld automaton, and a whole class of `Prisoner's Dilemma' games.

Xtoys

◊ Web site: http://thy.phy.bnl.gov/www/xtoys/xtoys.html

xtoys contains a set of cellular automata simulators for X windows. Programs included are:

◊ xising --- a two dimensional Ising model simulator,
◊ xpotts --- the two dimensional Potts model,
◊ xautomalab --- a totalistic cellular automaton simulator,
◊ xsand --- for the Bak, Tang, Wiesenfeld sandpile model,
◊ xwaves --- demonstrates three different wave equations,
◊ schrodinger --- play with the Scrodinger equation in an adjustable potential.

6. Agents & Robotics

Software brains for computers that do stuff. Everythin from fun and games to data mining to physical robotics. This is a great hobbiest area of AI with many areas of interest to pursue. I've broken it down loosely into 2 sections. AI for purely software based agents and that for embodied agents (even if only simulated).

6.1 Software Agents

Also known as intelligent software agents or just agents, this area of AI research deals with simple applications of small programs that aid the user in his/her work. They can be mobile (able to stop their execution on one machine and resume it on another) or static (live in one machine). They are usually specific to the task (and therefore fairly simple) and meant to help the user much as an assistant would.

2APL

◊ Web site: http://apapl.sourceforge.net/

2APL (pronounced as double-a-p-l) is an agent-oriented programming language that facilitates the implementation of multi-agent systems. At the multi-agent level, it provides programming constructs to specify a multi-agent system in terms of a set of individual agents, a set of environments in which they can perform actions, and the access relation between the individual agents and the environments. At the individual agent level, it provides programming constructs to implement cognitive agents based on the BDI architecture.

3APL

◊ Web site: http://www.cs.uu.nl/3apl/
◊ Wikipedia entry: http://en.wikipedia.org/wiki/3APL
◊ Mobile version: http://www.cs.uu.nl/3apl-m/

3APL is a programming language for implementing cognitive agents. It provides programming constructs for implementing agents' beliefs, goals, basic capabilities (such as belief updates, external actions, or communication actions) and a set of practical reasoning rules through which agents' goals can be updated or revised. The 3APL programs are executed on the 3APL platform. Each 3APL program is executed by means of an interpreter that deliberates on the cognitive attitudes of that agent.

Agent

◊ FTP site: http://www.cpan.org/modules/by-category/23_Miscellaneous_Modules/Agent/

The Agent is a prototype for an Information Agent system. It is both platform and language independent, as it stores contained information in simple packed strings. It can be packed and shipped across any network with any format, as it freezes itself in its current state.

agentTool

◊ Web site: http://agenttool.cis.ksu.edu/

agentTool is a Eclipse-based graphical development environment to help users analyze, design, and implement multiagent systems. It is designed to support the highly tailorable Organization-based Multiagent Systems Engineering (O-MaSE) methodology. agentTool currently supports all O-MaSE models including the Goal Model, Agent Model, Role Model, Organization Model, Protocol Model, Plan Model, Capability-Action Model, Domain Model, and Policy Model.

Aglets Workbench

◊ Web site: http://www.trl.ibm.com/aglets/index_e.htm

An aglet is a Java object that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built-in security mechanism makes it safe for a computer to host untrusted aglets. The Java Aglet API (J-AAPI) is a proposed public standard for interfacing aglets and their environment. J-AAPI contains methods for initializing an aglet, message handling, and dispatching, retracting, deactivating/activating, cloning, and disposing of the aglet. J-AAPI is simple, flexible, and stable. Application developers can write platform-independent aglets and expect them to run on any host that supports J-AAPI.

AJA

◊ Web site: http://home.arcor.de/michal.badonsky/AJA/

AJA (Adaptable Java Agents) consists of two programming languages. HADL (Higher Agent Definition Language) is a higher-level language used for the description of the main agent parts. Java+ is the lower-level language used for the programming of the agent parts defined in HADL. It is actually Java enriched with the constructs for accessing higher-level agent parts defined in HADL.

A.L.I.C.E.

◊ Web site: http://www.alicebot.org/

◊ Other AIML implementations: http://pyaiml.sourceforge.net/

The ALICE software implements AIML (Artificial Intelligence Markup Language), a non-standard evolving markup language for creating chat robots. The primary design feature of AIML is minimalism. Compared with other chat robot languages, AIML is perhaps the simplest. The pattern matching language is very simple, for example permitting only one wild-card ('*') match character per pattern. AIML is an XML language, implying that it obeys certain grammatical meta-rules. The choice of XML syntax permits integration with other tools such as XML editors. Another motivation for XML is its familiar look and feel, especially to people with HTML experience.

APRIL

◊ Web site: http://sourceforge.net/projects/networkagent/

APRIL is a symbolic programming language that is designed for writing mobile, distributed and agent-based systems especially in an Internet environment. It has advanced features such as a macro sub-language, asynchronous message sending and receiving, code mobility, pattern matching, higher-order functions and strong typing. The language is compiled to byte-code which is then interpreted by the APRIL runtime-engine. APRIL now requires the InterAgent Communications Model (ICM) to be installed before it can be installed. [Ed. ICM can be found at the same web site]

Ara

◊ Web site: http://wwwagss.informatik.uni-kl.de/Projekte/Ara/index_e.html

Ara is a platform for the portable and secure execution of mobile agents in heterogeneous networks. Mobile agents in this sense are programs with the ability to change their host machine during execution while preserving their internal state. This enables them to handle interactions locally which otherwise had to be performed remotely. Ara's specific aim in comparison to similar platforms is to provide full mobile agent functionality while retaining as much as possible of established programming models and languages.

Bee-gent

◊ Web site: http://www.toshiba.co.jp/rdc/beegent/download/download.htm

◊ FAQ: http://www.toshiba.co.jp/rdc/beegent/faq/faq.htm

Bee-gent is a new type of development framework in that it is a 100% pure agent system. As opposed to other systems which make only some use of agents, Bee-gent completely "Agentifies" the communication that takes place between software applications. The applications become agents, and all messages are carried by agents. Thus, Bee-gent allows developers to build flexible open distributed systems that make optimal use of existing applications.

Bond

◊ Web site: http://bond.cs.ucf.edu/

Bond is a Java based distributed object system and agent framework. It implements a message based middleware and associated services like directory, persistence, monitoring and security. Bond allows to easily build multi agent, distributed applications. Another application of Bond will be a Virtual Laboratory supporting data annotation and metacomputing.

Cougaar

◊ Web site: http://www.cougaar.org/

Cougaar is java-based architecture for the construction of large-scale distributed agent-based applications. It is the product of a multi-year DARPA research project into large scale agent systems and includes not only the core architecture but also a variety of demonstration, visualization and management components to simplify the development of complex, distributed applications. [Yet another java based agent system -- ed.]

D'Agent (was AGENT TCL)

◊ Web site: http://agent.cs.dartmouth.edu/software/agent2.0/

A transportable agent is a program that can migrate from machine to machine in a heterogeneous network. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport to another machine and resume execution on the new machine. For example, an agent carrying a mail message migrates first to a router and then to the recipient's mailbox. The agent can perform arbitrarily complex processing at each machine in order to ensure that the message reaches the intended recipient.

DIET Agents

◊ Web site: http://diet-agents.sourceforge.net

DIET Agents is a lightweight, scalable and robust multi-agent platform in Java. It is especially suitable for rapidly developing P2P prototype applications and/or adaptive, distributed applications that use bottom-up, nature-inspired techniques.

FishMarket

◊ Web site: http://www.iiia.csic.es/Projects/fishmarket/newindex.html

FM - The FishMarket project conducted at the Artificial Intelligence Research Institute (IIIA-CSIC) attempts to contribute in that direction by developing FM, an agent-mediated electronic auction house which has been evolved into a test-bed for electronic auction markets. The framework, conceived and implemented as an extension of FM96.5 (a Java-based version of the Fishmarket auction house), allows to define trading scenarios based on fish market auctions (Dutch auctions). FM provides the framework wherein agent designers can perform controlled experimentation in such a way that a multitude of experimental market scenarios--that we regard as tournament scenarios due to the competitive nature of the domain-- of varying degrees of realism and complexity can be specified, activated, and recorded; and trading (buyer and seller) heterogeneous (human and software) agents compared, tuned and evaluated.

Grasshopper

◊ Web site: http://www.grasshopper.de/

Another Java agent system. Full featured and actively developed. Commercial, but free. Historically targeted at embedded systems.

Hive

◊ Web site: http://hive.sourceforge.net/

Hive is a Java software platform for creating distributed applications. Using Hive, programmers can easily create systems that connect and use data from all over the Internet. At its heart, Hive is an environment for distributed agents to live, communicating and moving to fulfill applications. We are trying to make the Internet alive.

ICM

◊ SourceForge site: http://sourceforge.net/projects/networkagent/

The Inter-Agent Communication Model (ICM) is a communication mechanism that can be used for sending messages between agents in an asynchronous fashion. Its intended application area is as a transportation mechanism for agent communication languages (ACLs), such as KQML and FIPA's ACL.

Jacomma

◊ Web site: http://jacomma.sourceforge.net/
◊ SourceForge site: http://sourceforge.net/projects/jacomma/

Jacomma is an agent development platform/framework for developing distributed, mobile, and reactive information agents with heterogeneous communication capabilities, in Java and JPython.

Jacomma provides a development framework and an execution environment, which sits on top of the Inter-Agent Communication Model infrastructure. The ICM defines a communication protocol, a store and forward messaging architecture, and low level communication infrastructure for message exchange. Communication is truly asynchronous, based on TCP sockets.

ICM has an entry in this howto, or you can find it via a link off the site.

Jade

◊ Web site: http://sharon.cselt.it/projects/jade/

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required.

JAM Agent

◊ Web site: http://www.marcush.net/IRS/irs_downloads.html

JAM supports both top-down, goal-based reasoning and bottom-up data-driven reasoning. JAM selects goals and plans based on maximal priority if metalevel reasoning is not used, or user-developed metalevel reasoning plans if they exist. JAM's conceptualization of goals and goal achievement is more classically defined (UMPRS is more behavioral performance-based than truly goal-based) and makes the distinction between plans to achieve goals and plans that simply encode behaviors. Goal-types implemented include achievement (attain a specified world state), maintenance (re-attain a specified world state), and performance. Execution of multiple simultaneous goals are

supported, with suspension and resumption capabilities for each goal (i.e., intention) thread. JAM plans have explicit precondition and runtime attributes that restrict their applicability, a postcondition attribute, and a plan attributes section for specifying plan/domain-specific plan features. Available plan constructs include: sequencing, iteration, subgoaling, atomic (i.e., non-interruptable) plan segments, n-branch deterministic and non-deterministic conditional execution, parallel execution of multiple plan segments, goal-based or world state-based synchronization, an explicit failure-handling section, and Java primitive function definition through building it into JAM as well as the invocation of predefined (i.e., legacy) class members via Java's reflection capabilities without having to build it into JAM.

JASA

◊ Web site: http://www.csc.liv.ac.uk/~sphelps/jasa
◊ Alt Web site: http://sourceforge.net/projects/jasa/

JASA is a high performance auction simulator suitable for conducting experiments in agent-based computational economics. It implements various auction mechanisms, trading strategies and experiments described in the computational economics literature, and as the software matures we hope that it will become a repository for reference implementations of commonly used mechanisms, strategies and learning algorithms.

Jason

◊ Web site: http://jason.sourceforge.net/

A Java-based interpreter for an extended version of AgentSpeak. Unlike other BDI (Beliefs-Desires-Intentions) agent tools, Jason implements the operational semantics of AgentSpeak, a BDI logic programming language extensively discussed in the literature. It is available as Open Source under GNU LGPL.

JATLite

◊ Web site: http://java.stanford.edu/

JATLite is providing a set of java packages which makes easy to build multi-agent systems using Java. JATLite provides only light-weight, small set of packages so that the developers can handle all the packages with little efforts. For flexibility JATLite provides four different layers from abstract to Router implementation. A user can access any layer we are providing. Each layer has a different set of assumptions. The user can choose an appropriate layer according to the assumptions on the layer and user's application. The introduction page contains JATLite features and the set of assumptions for each layer.

JATLiteBeans

◊ Web site: http://waitaki.otago.ac.nz/JATLiteBean/

◊ Improved, easier-to-use interface to JATLite features including KQML message parsing, receiving, and sending.
◊ Extensible architecture for message handling and agent "thread of control" management
◊ Useful functions for parsing of simple KQML message content
◊ JATLiteBean supports automatic advertising of agent capabilities to facilitator agents
◊ Automatic, optional, handling of the "forward" performative
◊ Generic configuration file parser
◊ KQML syntax checker

Java(tm) Agent Template

◊ Web site: www-cdr.stanford.edu/ABE/JavaAgent.html

The JAT provides a fully functional template, written entirely in the Java language, for constructing software agents which communicate peer-to-peer with a community of other agents distributed over the Internet. Although portions of the code which define each agent are portable, JAT agents are not migratory but rather have a static existence on a single host. This behavior is in contrast to many other "agent" technologies. (However, using the Java RMI, JAT agents could dynamically migrate to a foreign host via an agent resident on that host). Currently, all agent messages use KQML as a top-level protocol or message wrapper. The JAT includes functionality for dynamically exchanging "Resources", which can include Java classes (e.g. new languages and interpreters, remote services, etc.), data files and information inlined into the KQML messages.

lyntin

◊ Web site: lyntin.sourceforge.net/

Lyntin is an extensible Mud client and framework for the creation of autonomous agents, or bots, as well as mudding in general. Lyntin is centered around Python, a dynamic, object-oriented, and fun programming language and based on TinTin++ a lovely mud client.

Mole

◊ Web site: mole.informatik.uni-stuttgart.de/

Mole is an agent system supporting mobile agents programmed in Java. Mole's agents consist of a cluster of objects, which have no references to the outside, and as a whole work on tasks given by the user or another agent. They have the ability to roam a network of "locations" autonomously. These "locations" are an abstraction of real, existing nodes in the underlying network. They can use location-specific resources by communicating with dedicated agents representing these services. Agents are able to use services provided by other agents and to provide services as well.

Narval

◊ Web site: www.logilab.org

Narval is the acronym of "Network Assistant Reasoning with a Validating Agent Language". It is a personal network assistant based on artificial intelligence and agent technologies. It executes recipes (sequences of actions) to perform tasks. It is easy to specify a new action using XML and to implement it using Python. Recipes can be built and debugged using a graphical interface.

NeL

◊ Web site: www.nevrax.org

NeL is actually a game development library (for massive multi-player games), but I'm including it here as it (will) include a fairly sizable AI library. Here's a blurb from the whitepaper:

The purpose of the AI library is to provide a pragmatic approach to creating a distributed agents platform. Its focus is agents; individual entities that communicate regardless of location, using an action-reaction model.

OAA

◊ Web site: www.ai.sri.com/~oaa/

The Open Agent Architecture is a framework in which a community of software agents running on distributed machines can work together on tasks assigned by human or non-human participants in the community. Distributed cooperation and high-level communication are two ideas central to the foundation of the OAA.

It defines an interagent communication language and supports multiple platforms and programming languages.

OpenCV

◊ Web site: http://opencvlibrary.sourceforge.net/

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision.

Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.

OpenCog

◊ Web site: http://www.opencog.org/

The Open Cognition Framework (OpenCog) is software for the collaborative development of safe and beneficial Artificial General Intelligence.

OpenCog provides research scientists and software developers with a common platform to build and share artificial intelligence programs. The framework includes:

◊ a flexible and highly optimized in-memory database for knowledge representation,
◊ a plug-in architecture for cognitive algorithms and a cognitive process scheduler,
◊ a built-in LISP-like programming language, and
◊ other components to support artificial intelligence research and development.

OpenSteer

◊ Web site: opensteer.sourceforge.net

OpenSteer is a C++ library to help build steering behaviors for autonomous characters in games and animation. OpenSteer provides an app which displays predefined demos of steering behaviors. You can prototype, visualize and debug your own as a plug-in.

ORTS

◊ Web site: https://skatgame.net/mburo/orts/

ORTS is a programming environment for studying real-time AI problems such as pathfinding, dealing with imperfect information, scheduling, and planning in the domain of RTS games. These games are fast-paced and very popular. Furthermore, the current state of RTS game AI is bleak which is mainly caused by the lack of planning and learning - areas in which humans are currently much better than machines. Therefore, RTS games make an ideal test-bed for real-time AI research. Unfortunately, commercial RTS games are closed software which prevents researchers from connecting remote AI modules to them. Furthermore, commercial RTS games are based on peer-to-peer technology - which in a nutshell runs the entire simulation on all player machines and just hides part of the game state from the players. By tampering with the client software it is possible to reveal the entire game state and thereby gain an unfair advantage. We feel that this is unacceptable for playing games on the

internet. We therefore started the ORTS project to create a free software system that lets people and machines play fair RTS games. The communication protocol is public and all source code and artwork is freely available. Users can connect whatever client software they like. This is made possible by a server/client architecture in which only the currently visible parts of the game state are sent to the players. This openness leads to new and interesting possibilities ranging from on-line tournaments of autonomous AI players to gauge their playing strength to hybrid systems in which human players use sophisticated GUIs which allow them to delegate tasks to AI helper modules of increasing performance.

Penguin!

- ◊ FTP site: http://www.cpan.org/modules/by-category/23_Miscellaneous_Modules/Penguin/FSG/

Penguin is a Perl 5 module. It provides you with a set of functions which allow you to:

- ◊ send encrypted, digitally signed Perl code to a remote machine to be executed.
- ◊ receive code and, depending on who signed it, execute it in an arbitrarily secure, limited compartment.

The combination of these functions enable direct Perl coding of algorithms to handle safe internet commerce, mobile information-gathering agents, "live content" web browser helper apps, distributed load-balanced computation, remote software update, distance machine administration, content-based information propagation, Internet-wide shared-data applications, network application builders, and so on.

Ps-i

- ◊ Web site: ps-i.sourceforge.net

Ps-i is an environment for running agent-based simulations. It is cross-platform, with binaries available for Win32. Features include:

- ◊ declarative language for model specification
- ◊ industry standard Tcl/Tk scripting with built-in routine optimization, speculative evaluation and xf86 JIT compiler users can create complex models without sacrificing perfomance
- ◊ user friendly interface
- ◊ save and restore program runs
- ◊ change model parameters on the fly
- ◊ data visualization: field display with multiple agent shapes and color, statistics window, agent viewer, routine browser and highlight agents tool

Pyro

- ◊ Web site: http://pyrorobotics.org/

Pyro is a library, environment, graphical user interface, and low-level drivers to explore AI and robotics using the Python language. It works with many real robotics platforms and simulators. Extensive algorithms including behavior-based, vision (motion tracking, blobs, etc.), learning (back-propagation, self-organizing maps, etc.), evolutionary, and more.

Quackle

- ◊ Web site: http://www.quackle.org/
- ◊ Alt Web site: http://web.mit.edu/jasonkb/www/quackle/
- ◊ Alt Web site: http://sourceforge.net/projects/quackle

Quackle is a world-class crossword game artificial intelligence and analysis tool. It includes a move generator, simulator, and Qt-based user interface and can be used with any board layout, alphabet, lexicon, and tile distribution.

Recast

◊ Web site: http://code.google.com/p/recastnavigation/

Recast is state of the art navigation mesh construction toolset for games.

◊ It is automatic, which means that you can throw any level geometry at it and you will get robust mesh out
◊ It is fast which means swift turnaround times for level designers
◊ It is open source so it comes with full source and you can customize it to your hearts content.

The Library is free for commercial use and open source under the ZLib License.

Recast is accompanied with Detour, path-finding and spatial reasoning toolkit. You can use any navigation mesh with Detour, but of course the data generated with Recast fits perfectly.

Detour offers simple static navigation mesh which is suitable for many simple cases, as well as tiled navigation mesh which allows you to plug in and out pieces of the mesh. The tiled mesh allows to create systems where you stream new navigation data in and out as the player progresses the level, or you may regenerate tiles as the world changes.

Remembrance Agents

◊ Web site: www.remem.org

Remembrance Agents are a set of applications that watch over a user's shoulder and suggest information relevant to the current situation. While query-based memory aids help with direct recall, remembrance agents are an augmented associative memory. For example, the word-processor version of the RA continuously updates a list of documents relevant to what's being typed or read in an emacs buffer. These suggested documents can be any text files that might be relevant to what you are currently writing or reading. They might be old emails related to the mail you are currently reading, or abstracts from papers and newspaper articles that discuss the topic of your writing.

SimAgent

◊ Web site: www.cs.bham.ac.uk/research/projects/poplog/packages/simagent.html

The SimAgent toolkit provides a range of resources for research and teaching related to the development of interacting agents in environments of various degrees and kinds of complexity. It can be run as a pure simulation tool, or installed in a robot with a sufficiently powerful on-board computer, e.g. running linux. It was originally developed to support exploratory research on human-like intelligent agents, but has also been used for student projects developing a variety of interactive games and simulations.

spyse

◊ Web site: spyse.sf.net
◊ Alt Web site: zope.org/Members/drapmeyer/spyse

spyse is a development framework and platform for building multi-agent systems using the Python programming language. A multi-agent system (MAS) combines concepts from distributed computing and artificial intelligence. Agents are autonomously reasoning software entities that can collaborate

(or compete) in order to achieve a (common) goal. By cooperating they create emergent behaviour in the system (distributed artificial intelligence). The architecture of a MAS is specified in the FIPA standard.

Spyse provides multiple means for reasoning (BDI logics, CLIPS expert shell, etc.) and communicating locally and remotely.

Each agent has its own thread of control. Agents within and among instances of the platform communicate by exchanging messages based on ontologies. Spyse makes use of the Web Ontology Language (OWL) defined for the Semantic Web.

TKQML

◊ Web site: www.csee.umbc.edu/tkqml/

TKQML is a KQML application/addition to Tcl/Tk, which allows Tcl based systems to communicate easily with a powerful agent communication language.

The Tocoma Project

◊ Web site: www.tacoma.cs.uit.no/

An agent is a process that may migrate through a computer network in order to satisfy requests made by clients. Agents are an attractive way to describe network-wide computations.

The TACOMA project focuses on operating system support for agents and how agents can be used to solve problems traditionally addressed by operating systems. We have implemented a series of prototype systems to support agents.

TACOMA Version 1.2 is based on UNIX and TCP. The system supports agents written in C, Tcl/Tk, Perl, Python, and Scheme (Elk). It is implemented in C. This TACOMA version has been in public domain since April 1996.

We are currently focusing on heterogeneity, fault-tolerance, security and management issues. Also, several TACOMA applications are under construction. We implemented StormCast 4.0, a wide-area network weather monitoring system accessible over the internet, using TACOMA and Java. We are now in the process of evaluating this application, and plan to build a new StormCast version to be completed by June 1997.

UMPRS Agent

◊ Web site: http://www.marcush.net/IRS/

UMPRS supports top-down, goal-based reasoning and selects goals and plans based on maximal priority. Execution of multiple simultaneous goals are supported, with suspension and resumption capabilities for each goal (i.e., intention) thread. UMPRS plans have an integrated precondition/runtime attribute that constrain their applicability. Available plan constructs include: sequencing, iteration, subgoaling, atomic (i.e., non-interruptable) blocks, n-branch deterministic conditional execution, explicit failure-handling section, and C++ primitive function definition.

WebMate

◊ Web site: http://www.cs.cmu.edu/~softagents/webmate/

WebMate is a personal agent for World-Wide Web browsing and searching. It accompanies you when you travel on the internet and provides you what you want.

Features include:

- ◊ Searching enhancement, including parallel search, searching keywords refinement using our relevant keywords extraction technology, relevant feedback, etc.
- ◊ Browsing assistant, including learning your current interesting, recommending you new URLs according to your profile and selected resources, monitoring bookmarks of Netscape or IE, sending the current browsing page to your friends, etc.
- ◊ Offline browsing, including downloading the following pages from the current page for offline browsing.
- ◊ Filtering HTTP header, including recording http header and all the transactions between your browser and WWW servers, etc.
- ◊ Checking the HTML page to find the errors or dead links, etc.
- ◊ Programming in Java, independent of operating system, runing in multi-thread.

6.2 Robotics and Simulators

From fun battling robot games to full robot control systems. The idea is physical agents in the real world, or at least their control programming.

BattleBots

◊ Web site: www.bluefire.nu/battlebots/

AI programming game where you design the bot by selecting hardware and programming its CPU, then competing with other bots. Competitions can have teams and special rules for a game.

The hardware for use in your bot includes weapons, engine, scanners, CPU, etc. The programming lauguage is dependent on the CPU type and is similar to an assembly language.

Cadaver

◊ Web site: www.erikyyy.de/cadaver/

Cadaver is a simulated world of cyborgs and nature in realtime. The battlefield consists of forests, grain, water, grass, carcass (of course) and lots of other things. The game server manages the game and the rules. You start a server and connect some clients. The clients communicate with the server using a very primitive protocol. They can order cyborgs to harvest grain, attack enemies or cut forest. The game is not intended to be played by humans! There is too much to control. Only for die-hards: Just telnet to the server and you can enter commands by hand. Instead the idea is that you write artificial intelligence clients to beat the other artificial intelligences. You can choose a language (and operating system) of your choice to do that task. It is enough to write a program that communicates on standard input and standard output channels. Then you can use programs like "socket" to connect your clients to the server. It is NOT needed to write TCP/IP code, although i did so :) The battle shall not be boring, and so there is the so called spyboss client that displays the action graphically on screen.

Carmen

◊ Web site: http://carmen.sourceforge.net/

CARMEN, the Carnegie Mellon Robot Navigation Toolkit. CARMEN is an open-source collection of software for mobile robot control. CARMEN is modular software designed to provide basic navigation primatives including: base and sensor control, logging, obstacle avoidance, localization, path planning, and mapping.

CLARAty

◊ Web site: http://claraty.jpl.nasa.gov/man/overview/

CLARAty is an integrated framework for reusable robotic software. It defines interfaces for common robotic functionality and integrates multiple implementations of any given functionality. Examples of such capabilities include pose estimation, navigation, locomotion and planning. In addition to supporting multiple algorithms, it provides adaptations to multiple robotic platforms.

This is a public release of the some of the code used in the Mars rover projects at NASA. It is under a free for non-commercial use licence and consists of large number of modules and algorithms along with extensive documentation.

GNU Robots

◊ Web site: http://www.gnu.org/software/robots/

GNU Robots is a game/diversion where you construct a program for a little robot, then watch him explore a world. The world is filled with baddies that can hurt you, objects that you can bump into, and food that you can eat. The goal of the game is to collect as many prizes as possible before are killed by a baddie or you run out of energy. Robots can be written in Guile scheme or using a GUI.

Infon Battle Arena

◊ Web site: http://infon.dividuum.de/
◊ Web site: https://github.com/dividuum/infon

Infon Battle Arena is a networked multiplayer real-time programming game featuring little creatures fighting for food. You upload your Creature Code (written in Lua) to a game server using a telnet Interface. The game server then runs your code. The graphical client can be used to watch running games or replay recorded games.

Khepera Simulator

◊ Web site: http://diwww.epfl.ch/w3lami/team/michel/khep-sim/

Khepera Simulator is a public domain software package written by Olivier MICHEL during the preparation of his Ph.D. thesis, at the Laboratoire I3S, URA 1376 of CNRS and University of Nice-Sophia Antipolis, France. It allows to write your own controller for the mobile robot Khepera using C or C++ languages, to test them in a simulated environment and features a nice colorful X11 graphical interface. Moreover, if you own a Khepera robot, it can drive the real robot using the same control algorithm. It is mainly oriented toward to researchers studying autonomous agents.

MRPT

◊ Web site: http://babel.isa.uma.es/mrpt/index.php/Main_Page

The Mobile Robot Programming Toolkit (MRPT) is an extensive, cross-platform, and open source C++ library aimed to help robotics researchers to design and implement algorithms in the fields of Simultaneous Localization and Mapping (SLAM), computer vision, and motion planning (obstacle avoidance).

Nero

◊ Web site: http://www.nerogame.org/

Neuro-Evolving Robotic Operatives, or NERO for short, is a unique computer game that lets you play with adapting intelligent agents hands-on. Evolve your own robot army by tuning their artificial brains for challenging tasks, then pit them against your friends' teams in online competitions!

The goals of the project are (1) to demonstrate the power of state-of-the-art machine learning technology, (2) to create an engaging game based on it, and (3) to provide a robust and challenging development and benchmarking domain for AI researchers.

Closed source but free to download. They are working on OpenNERO which will be open source and more intended as a research platform.

Orca

◊ Web site: http://orca-robotics.sourceforge.net/
◊ Web site: https://github.com/naderman/orca-robotics

Orca is an open-source framework for developing component-based robotic systems. It provides the means for defining and developing the building-blocks which can be pieced together to form arbitrarily complex robotic systems, from single vehicles to distributed sensor networks.

Player

◊ Web site: http://playerstage.sourceforge.net/
◊ Player wiki: http://playerstage.sourceforge.net/wiki/Player

Player is a device server that provides a powerful, flexible interface to a variety of sensors and actuators (e.g., robots). Because Player uses a TCP socket-based client/server model, robot control programs can be written in any programming language and can execute on any computer with network connectivity to the robot. In addition, Player supports multiple concurrent client connections to devices, creating new possibilities for distributed and collaborative sensing and control.

RealTimeBattle

◊ Web site: http://realtimebattle.sourceforge.net/

RealTimeBattle is a programming game, in which robots controlled by programs are fighting each other. The goal is to destroy the enemies, using the radar to examine the environment and the cannon to shoot.

◊ Game progresses in real time, with the robot programs running as child processes to RealTimeBattle.
◊ The robots communicate with the main program using the standard input and output.
◊ Robots can be constructed in almost any programming language.
◊ Maximum number of robots can compete simultaneously.
◊ A simple messaging language is used for communication, which makes it easy to start constructing robots.
◊ Robots behave like real physical object.
◊ You can create your own arenas.
◊ Highly configurable.

Robocode

◊ Web site: http://robocode.sourceforge.net/

A java based robot combat programming game. It provides a simple API and class framework. It is designed as a means of learning Java and is easy to start using while not constraining the programmer from more advanced techniques. It has a built in security manager for running other peoples robots in a safe way.

Robodeb

◊ Web site: http://www.transterpreter.org/robodeb/

Robodeb is a complete robotics simulation environment for teaching concurrency and parallelism. It provides a unique environment for exploring concurrency and robotics. It provides a complete IDE for the occam-pi programming language, and leverages the Transterpreter, our portable and flexible runtime for the language. This combination is critical, as it provides a principled interface to the Player/Stage API, a set of widely used libraries for controlling the Pioneer3 robotics platform.

RobotFlow

◊ Web site: http://robotflow.sourceforge.net/

RobotFlow is a mobile robotics tookit based on the FlowDesigner project. FlowDesigner is a data-flow oriented architecture, similar to Simulink (Matlab) or Labview that is free (LGPL) and versatile. The visual programming interface provided in the FlowDesigner project will help people to better visualize & understand what is really happening in the robot's control loops, sensors, actuators, by using graphical probes and debugging in real-time.

RoboTournament

◊ Web site: http://robotournament.sourceforge.net/

RoboTournament is a RoboRally inspired game where players program their robots to vanquish their opponents. RoboTournament features: Multiple Game Types: Death Match, Rally, and Capture The Flag. Multi-Player through TCP/IP, Six weapons including BFG, Map Editor, and a wide variety of board elements.

ROS

◊ Web site: http://www.ros.org/wiki/

ROS is an operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

Simbad

◊ Web site: http://simbad.sourceforge.net/

Simbad is a Java 3d robot simulator for scientific and educationnal purposes. It is mainly dedicated to researchers/programmers who want a simple basis for studying Situated Artificial Intelligence, Machine Learning, and more generally AI algorithms, in the context of Autonomous Robotics and Autonomous Agents. It is not intented to provide a real world simulation and is kept voluntarily readable and simple.

Simbad enables programmers to write their own robot controller, modify the environment and use the available sensors. Don't think of it as a finite product but merely as an opened framework to test your own ideas.

SimRobot

◊ Web site: http://www.informatik.uni-bremen.de/simrobot/
◊ FTP site: ftp://ftp.uni-bremen.de/pub/ZKW/INFORM/simrobot/

SimRobot is a program for simulation of sensor based robots in a 3D environment. It is written in C++, runs under UNIX and X11 and needs the graphics toolkit XView.

◊ Simulation of robot kinematics
◊ Hierarchically built scene definition via a simple definition language
◊ Various sensors built in: camera, facette eye, distance measurement, light sensor, etc.
◊ Objects defined as polyeders
◊ Emitter abstractly defined; can be interpreted e.g. as light or sound
◊ Camera images computed according to the raytracing or Z-buffer algorithms known from computer graphics
◊ Specific sensor/motor software interface for communicating with the simulation
◊ Texture mapping onto the object surfaces: bitmaps in various formats
◊ Comprehensive visualization of the scene: wire frame w/o hidden lines, sensor and actor values
◊ Interactive as well as batch driven control of the agents and operation in the environment
◊ Collision detection
◊ Extendability with user defined object types
◊ Possible socket communication to e.g. the Khoros image processing software

TclRobots

◊ Web site: http://www.nyx.net/~tpoindex/

TclRobots is a programming game, similar to 'Core War'. To play TclRobots, you must write a Tcl program that controls a robot. The robot's mission is to survive a battle with other robots. Two, three, or four robots compete during a battle, each running different programs (or possibly the same program in different robots.) Each robot is equipped with a scanner, cannon, drive mechanism. A single match continues until one robot is left running. Robots may compete individually, or combine in a team oriented battle. A tournament can be run with any number of robot programs, each robot playing every other in a round-robin fashion, one-on-one. A battle simulator is available to help debug robot programs.

The TclRobots program provides a physical environment, imposing certain game parameters to which all robots must adhere. TclRobots also provides a view on a battle, and a controlling user interface. TclRobots requirements: a wish interpreter built from Tcl 7.4 and Tk 4.0.

URBI

◊ Web site: http://www.urbiforge.org/

URBI is a Universal Real-time Behavior Interface and gives you a simple but powerful way to control any robot or complex system like a video game, using a convenient and easy to use scripting language that can be interfaced with several popular programming languages (C++, Java, Matlab,...) and OS (Windows, Mac OSX, Linux). URBI is based on a client/server architecture, which give a great deal of flexibility. URBI includes powerful features compared to existing scripting solutions: parallel execution of commands, event programming, command tagging, dynamic variables,... Currently,

URBI is used as well by academic research labs, the industry and by hobbyists.

VWORLD

◊ Web site: http://zhar.net/projects/vworld/

Vworld is a simulated environment for research with autonomous agents written in prolog. It is currently in something of an beta stage. It works well with SWI-prolog, but should work with Quitnus-prolog with only a few changes. It is being designed to serve as an educational tool for class projects dealing with prolog and autonomous agents. It comes with three demo worlds or environments, along with sample agents for them. There are two versions now. One written for SWI-prolog and one written for LPA-prolog. Documentation is roughly done (with a student/professor framework in mind).

Yampa

◊ Web site: http://www.haskell.org/yampa/

FRP system with robotics library and graphical interactive robotics simulator.

Functional reactive programming, or FRP, is a paradigm for programming hybrid systems – i.e., systems containing a combination of both continuous and discrete components – in a high-level, declarative way. The key ideas in FRP are its notions of continuous, time-varying values, and time-ordered sequences of discrete events. Yampa is an instantiation of FRP as a domain-specific language embedded in Haskell.

YARP

◊ Web site: http://eris.liralab.it/yarp/

YARP is plumbing for robot software. It is a set of libraries, protocols, and tools to keep modules and devices cleanly decoupled. It is reluctant middleware, with no desire or expectation to be in control of your system.

More specifically, YARP supports building a robot control system as a collection of programs communicating in a peer-to-peer way, with a family of connection types that meet the diverse, sometimes contradictory, and always changing needs of advanced robotics. We also encourage compilation and use of hardware devices in a future-proof way. Our strategic goal is to increase the longevity of robot software projects.

7. Statistical & Machine Learning

All about getting machines to learn to do something rather than explicitly programming to do it. Tends to deal with pattern matching a lot and are heavily math and statistically based. Technically Connectionism falls under this category, but it is such a large sub-field I'm keeping it in a separate section.

7.1 Libraries

Libraries or frameworks used for writing machine learning systems.

CognitiveFoundry

◊ Web site: http://foundry.sandia.gov/

The Cognitive Foundry is a modular Java software library for the research and development of cognitive systems. It contains many reusable components for machine learning, statistics, and cognitive modeling. It is primarily designed to be easy to plug into applications to provide adaptive behaviors.

CompLearn

◊ Web site: http://complearn.org/

CompLearn is a software system built to support compression-based learning in a wide variety of applications. It provides this support in the form of a library written in highly portable ANSI C that runs in most modern computer environments with minimal confusion. It also supplies a small suite of simple, composable command-line utilities as simple applications that use this library. Together with other commonly used machine-learning tools such as LibSVM and GraphViz, CompLearn forms an attractive offering in machine-learning frameworks and toolkits.

Elefant

◊ Web site: http://elefant.developer.nicta.com.au/

Elefant (Efficient Learning, Large-scale Inference, and Optimisation Toolkit) is an open source library for machine learning licensed under the Mozilla Public License (MPL). We develop an open source machine learning toolkit which provides

◊ algorithms for machine learning utilising the power of multi-core/multi-threaded processors/operating systems (Linux, WIndows, Mac OS X),
◊ a graphical user interface for users who want to quickly prototype machine learning experiments,
◊ tutorials to support learning about Statistical Machine Learning (Statistical Machine Learning at The Australian National University), and
◊ detailed and precise documentation for each of the above.

Maximum Entropy Toolkit

◊ Web site: http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

The Maximum Entropy Toolkit provides a set of tools and library for constructing maximum entropy (maxent) model in either Python or C++.

Maxent Entropy Model is a general purpose machine learning framework that has proved to be highly expressive and powerful in statistical natural language processing, statistical physics, computer vision and many other fields.

Milk

◊ Web site: http://packages.python.org/milk/
◊ Web site: https://github.com/luispedro/milk

Milk is a machine learning toolkit in Python. It's focus is on supervised classification with several classifiers available: SVMs (based on libsvm), k-NN, random forests, decision trees. It also performs feature selection. These classifiers can be combined in many ways to form different classification systems. For unsupervised learning, milk supports k-means clustering and affinity propagation.

NLTK

◊ Web site: http://nltk.org/

NLTK, the Natural Language Toolkit, is a suite of Python libraries and programs for symbolic and statistical natural language processing. NLTK includes graphical demonstrations and sample data. It is accompanied by extensive documentation, including tutorials that explain the underlying concepts behind the language processing tasks supported by the toolkit.

NLTK is ideally suited to students who are learning NLP (natural language processing) or conducting research in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems.

peach

◊ Web site: http://code.google.com/p/peach/

Peach is a pure-python module, based on SciPy and NumPy to implement algorithms for computational intelligence and machine learning. Methods implemented include, but are not limited to, artificial neural networks, fuzzy logic, genetic algorithms, swarm intelligence and much more.

The aim of this library is primarily educational. Nonetheless, care was taken to make the methods implemented also very efficient.

pebl

◊ Web site: http://code.google.com/p/pebl-project/

Pebl is a python library and command line application for learning the structure of a Bayesian network given prior knowledge and observations. Pebl includes the following features:

◊ Can learn with observational and interventional data
◊ Handles missing values and hidden variables using exact and heuristic methods
◊ Provides several learning algorithms; makes creating new ones simple
◊ Has facilities for transparent parallel execution using several cluster/grid resources
◊ Calculates edge marginals and consensus networks
◊ Presents results in a variety of formats

PyBrain

◊ Web site: http://pybrain.org/

PyBrain is a modular Machine Learning Library for Python. It's goal is to offer flexible, easy-to-use yet still powerful algorithms for Machine Learning Tasks and a variety of predefined environments to test and compare your algorithms.

PyBrain contains algorithms for neural networks, for reinforcement learning (and the combination of the two), for unsupervised learning, and evolution. Since most of the current problems deal with continuous state and action spaces, function approximators (like neural networks) must be used to cope with the large dimensionality. Our library is built around neural networks in the kernel and all of the training methods accept a neural network as the to-be-trained instance. This makes PyBrain a powerful tool for real-life tasks.

MBT

◊ Web site: http://ilk.uvt.nl/mbt/

MBT is a memory-based tagger-generator and tagger in one. The tagger-generator part can generate a sequence tagger on the basis of a training set of tagged sequences; the tagger part can tag new sequences. MBT can, for instance, be used to generate part-of-speech taggers or chunkers for natural language processing. It has also been used for named-entity recognition, information extraction in domain-specific texts, and disfluency chunking in transcribed speech.

MLAP book samples

◊ Web site: http://seat.massey.ac.nz/personal/s.r.marsland/MLBook.html

Not a library per-say, but a whole slew of example machine learning algorithms from the book "Machine Learning: An Algorithmic Perspective" by Stephen Marsland. All code is written in python.

scikits.learn

◊ Web site: http://scikit-learn.org/stable/

scikits-learn is a Python module integrating classic machine learning algorithms in the tightly-knit world of scientific Python packages (numpy, scipy, matplotlib). It aims to provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.

Shogun

◊ Web site: http://www.shogun-toolbox.org/

The machine learning toolbox's focus is on large scale kernel methods and especially on Support Vector Machines (SVM). It provides a generic SVM object interfacing to several different SVM implementations, among them the state of the art LibSVM and SVMLight. Each of the SVMs can be combined with a variety of kernels. The toolbox not only provides efficient implementations of the most common kernels, like the Linear, Polynomial, Gaussian and Sigmoid Kernel but also comes with a number of recent string kernels as e.g. the Locality Improved, Fischer, TOP, Spectrum, Weighted Degree Kernel (with shifts). For the latter the efficient LINADD optimizations are implemented. Also SHOGUN offers the freedom of working with custom pre-computed kernels. One of its key features is the combined kernel which can be constructed by a weighted linear combination of a number of sub-kernels, each of which not necessarily working on the same domain. An optimal sub-kernel weighting can be learned using Multiple Kernel Learning. Currently SVM 2-class classification and regression problems can be dealt with. However SHOGUN also implements a number of linear methods like Linear Discriminant Analysis (LDA), Linear Programming Machine (LPM), (Kernel) Perceptrons and features algorithms to train hidden markov models. The input feature-objects can be dense, sparse or strings and of type int/short/double/char and can be converted into different feature types. Chains of preprocessors (e.g. substracting the mean) can be attached to each feature object allowing for on-the-fly pre-processing.

SHOGUN is implemented in C++ and interfaces to Matlab(tm), R, Octave and Python.

timbl

◊ Web site: http://ilk.uvt.nl/timbl/

The Tilburg Memory Based Learner, TiMBL, is a tool for NLP research, and for many other domains where classification tasks are learned from examples. It is an efficient implementation of k-nearest neighbor classifier.

TiMBL's features are:

◊ Fast, decision-tree-based implementation of k-nearest neighbor lassification;
◊ Implementations of IB1 and IB2, IGTree, TRIBL, and TRIBL2 algorithms;
◊ Similarity metrics: Overlap, MVDM, Jeffrey Divergence, Dot product, Cosine;
◊ Feature weighting metrics: information gain, gain ratio, chi squared, shared variance;
◊ Distance weighting metrics: inverse, inverse linear, exponential decay;
◊ Extensive verbosity options to inspect nearest neighbor sets;
◊ Server functionality and extensive API;
◊ Fast leave-one-out testing and internal cross-validation;
◊ and Handles user-defined example weighting.

7.2 Applications

Full applications that implement various machine learning or statistical systems oriented toward general learning (i.e., no spam filters and the like).

dbacl

◊ Web site: http://dbacl.sourceforge.net/

The dbacl project consist of a set of lightweight UNIX/POSIX utilities which can be used, either directly or in shell scripts, to classify text documents automatically, according to Bayesian statistical principles.

Torch5

◊ Web site: http://torch5.sourceforge.net/

Torch5 provides a matlab-like environment for state-of-the-art machine learning algorithms. It is easy to use and provides a very efficient implementation, thanks to a easy and fast scripting language (Lua) and a underlying C++ implementation. It is distributed under a BSD license.

This is the successor to the Torch3 project.

Vowpal Wabbit

◊ Web site: http://hunch.net/~vw/

Vowpal Wabbit is a fast online learning algorithm. It features:

◊ flexible input data specification
◊ speedy learning
◊ scalability (bounded memory footprint, suitable for distributed computation)
◊ feature pairing

The core algorithm is specialist gradient descent (GD) on a loss function (several are available), The code should be easily usable.

8. Missing & Dead

This is my area for old or bad entries. The MIA section is for entires for which I no longer have a valid home page. If you have any information regarding where I can find these now please let me know. The Dead section is for projects that seem dead. Moving them here allows me to keep my the main sections clean while

allowing for interested parties to correct me in which case I can just move it back.

8.1 MIA - Projects missing linkage.

AntWars

◊ Web site: http://ant-wars.net/

Ant Wars is a competition which pits clever programs against each other to do battle and compete for food in virtual worlds. Each contestant is a species of ant, which can visualize only the world immediately around him and pheromones left by fellow and enemy ants. Using this information, the ant brain (a simple state machine) must guide the ant towards collecting food at his home ant hill, while fending off or attacking enemies.

Clever use of pheromones and subtle behaviors can create large scale tactics such as raiding, defense, harvesting, and scouting when many ants cooperate.

CASE

◊ Web site: http://www.iu.hio.no/~cell/
◊ FTP site: ftp://ftp.iu.hio.no/pub/

CASE (Cellular Automaton Simulation Environment) is a C++ toolkit for visualizing discrete models in two dimensions: so-called cellular automata. The aim of this project is to create an integrated framework for creating generalized cellular automata using the best, standardized technology of the day.

CLEARS

◊ Web site: ????

The CLEARS system is an interactive graphical environment for computational semantics. The tool allows exploration and comparison of different semantic formalisms, and their interaction with syntax. This enables the user to get an idea of the range of possibilities of semantic construction, and also where there is real convergence between theories.

CLIG

◊ Web site: http://www.ags.uni-sb.de/~konrad/clig.html

CLIG is an interactive, extendible grapher for visualizing linguistic data structures like trees, feature structures, Discourse Representation Structures (DRS), logical formulas etc. All of these can be freely mixed and embedded into each other. The grapher has been designed both to be stand-alone and to be used as an add-on for linguistic applications which display their output in a graphical manner.

Corewar VM

◊ Web site: http://www.jedi.claranet.fr/

This is a virtual machine written in Java (so it is a virtual machine for another virtual machine !) for a Corewar game.

DAI

◊ Web site: http://starship.python.net/crew/gandalf/DNET/AI/

A library for the Python programming language that provides an object oriented interface to the CLIPS expert system tool. It includes an interface to COOL (CLIPS Object Oriented Language) that allows:

◊ Investigate COOL classes
◊ Create and manipulate with COOL instances
◊ Manipulate with COOL message-handler's
◊ Manipulate with Modules

DHARMI

◊ Web site: http://megazone.bigpanda.com/~wolf/DHARMI/

DHARMI is a high level spatial, tinker-toy like language who's components are transparently administered by a background process called the Habitat. As the name suggests, the language was designed to make modelling prototypes and handle living data. Programs can be modified while running. This is accomplished by blurring the distinction between source code, program, and data.

Dunce

◊ Web site: http://www.boswa.com/boswabits/

Dunce is a simple chatterbot (conversational AI) and a language for programming such chatterbots. It uses a basic regex pattern matching and a semi-neural rule/response firing mechanism (with excitement/decay cycles).

Dunce is listed about halfway down the page.

EcoSim

◊ Web site: http://www.offis.de/projekte/projekt.php?id=140

NOTE: the above web site has info on EcoSim but no code to download.

In EcoSim an ecosystem is described by all static and dynamic properties of the individuals involved in the system as well as time varying properties of the environment. Individuals change their state over time or due to internal and external events. The environment is also defined via dynamic objects which can change. Supports on the fly analysis and animation of generated data. It is a C++ class library designed to support individual-oriented modelling and simulation of ecological systems.

Evo

◊ Web site: http://omicrongroup.org/evo/

Evo is a software development framework that allows developers to build complex alife simulations. Using Evo, researchers can easily build systems of independent agents interacting with one another and with their environment. Evo implements biological operators such as genetic recombination and mutation to evolve the behavior of agents so that they are more adapted to their environment.

Grany-3

◊ Web site: http://zarb.org/ gc/html/grany.html

Grany-3 is a full-featured cellular automaton simulator, made in C++ with Gtk--, flex++/bison++, doxygen and gettext, useful to granular media physicists.

IBAL

◊ Web site: http://www.eecs.harvard.edu/~avi/IBAL/

IBAL (pronounced "eyeball") is a general-purpose language for probabilistic modeling, parameter estimation and decision making. It generalizes Bayesian networks, hidden Markov models, stochastic context free grammars, Markov decision processes, and allows many new possibilities. It also provides a convenient programming-language framework with libraries, automatic type checking and so on.

IDEAL

◊ Web site: http://yoda.cis.temple.edu:8080/ideal/

IDEAL is a test bed for work in influence diagrams and Bayesian networks. It contains various inference algorithms for belief networks and evaluation algorithms for influence diagrams. It contains facilities for creating and editing influence diagrams and belief networks.

IDEAL is written in pure Common Lisp and so it will run in Common Lisp on any platform. The emphasis in writing IDEAL has been on code clarity and providing high level programming abstractions. It thus is very suitable for experimental implementations which need or extend belief network technology.

At the highest level, IDEAL can be used as a subroutine library which provides belief network inference and influence diagram evaluation as a package. The code is documented in a detailed manual and so it is also possible to work at a lower level on extensions of belief network methods.

IDEAL comes with an optional graphic interface written in CLIM. If your Common Lisp also has CLIM, you can run the graphic interface.

Illuminator

◊ Web site: http://documents.cfar.umd.edu/resources/source/illuminator.html

Illuminator is a toolset for developing OCR and Image Understanding applications. Illuminator has two major parts: a library for representing, storing and retrieving OCR information, heretofore called dafslib, and an X-Windows "DAFS" file viewer, called illum. Illuminator and DAFS lib were designed to supplant existing OCR formats and become a standard in the industry. They particularly are extensible to handle more than just English.

The features of this release:

◊ 5 magnification levels for images
◊ flagged characters and words
◊ unicode support -- American, British, French, German, Greek, Italian, MICR, Norwegian, Russian, Spanish, Swedish, keyboards
◊ reads DAFS, TIFF's, PDA's (image only)
◊ save to DAFS, ASCII/UTF or Unicode
◊ Entity Viewer - shows properties, character choices, bounding boxes image fragment for a selected entity, change type, change content, hierarchy mode

Jet's Neural Architecture

◊ Web site: http://www.voltar-confed.org/jneural/

Jet's Neural Architecture is a C++ framework for doing neural net projects. The goals of this project were to make a fast, flexible neural architecture that isn't stuck to one kind of net and to make sure that end users could easily write useful applications. All the documentation is also easily readable.

Neural Networks at your Fingertips

◊ Web site: http://www.neural-networks-at-your-fingertips.com/

A set of ANSI C packages that illustrate Adaline networks, back-propagation, the Hopfield model, BAM, Boltzman, CPN, SOM, and ART1. Coded in portable, self-contained ANSI C. With complete example applications from a variety of well-known application domains.

Neureka ANS (nn/xnn)

◊ FTP site: ftp://ftp.ii.uib.no/pub/neureka/

nn is a high-level neural network specification language. The current version is best suited for feed-forward nets, but recurrent models can and have been implemented, e.g. Hopfield nets, Jordan/Elman nets, etc. In nn, it is easy to change network dynamics. The nn compiler can generate C code or executable programs (so there must be a C compiler available), with a powerful command line interface (but everything may also be controlled via the graphical interface, xnn). It is possible for the user to write C routines that can be called from inside the nn specification, and to use the nn specification as a function that is called from a C program. Please note that no programming is necessary in order to use the network models that come with the system (`netpack').

xnn is a graphical front end to networks generated by the nn compiler, and to the compiler itself. The xnn graphical interface is intuitive and easy to use for beginners, yet powerful, with many possibilities for visualizing network data.

NOTE: You have to run the install program that comes with this to get the license key installed. It gets put (by default) in /usr/lib. If you (like myself) want to install the package somewhere other than in the /usr directory structure (the install program gives you this option) you will have to set up some environmental variables (NNLIBDIR & NNINCLUDEDIR are required). You can read about these (and a few other optional variables) in appendix A of the documentation (pg 113).

NEURObjects

◊ Web site: http://www.disi.unige.it/person/ValentiniG/NEURObjects/

NEURObjects is a set of C++ library classes for neural networks development. The main goal of the library consists in supporting researchers and practitioners in developing new neural network methods and applications, exploiting the potentialities of object-oriented design and programming. NEURObjects provides also general purpose applications for classification problems and can be used for fast prototyping of inductive machine learning applications.

OSCAR

◊ Web site: http://oscarhome.soc-sci.arizona.edu/ftp/OSCAR-web-page/oscar.html

The goal of the OSCAR project is the formulation of a general theory of rationality and its implementation in an artificial rational agent. The function of artificial agents is to draw conclusions and make decisions on the basis of information supplied to them. OSCAR is a fully implemented architecture for rational agents, based upon a general purpose defeasible reasoner. OSCAR is written in Common Lisp and is free for educational and research purposes.

PAI

◊ Web site: http://utenti.quipo.it/claudioscordino/pai.html

AI (Programmable Artificial Intelligence) is a program capable of having a conversation in its mother tongue, English. Written in C++.

pygene

◊ Web site: http://www.freenet.org.nz/python/pygene/

pygene is a simple and easily understandable library for genetic algorithms and genetic programming in python. Includes examples such as the travelling salesman problem.

PyIE

◊ repo: https://www.dfwpython.org/repo/Projects/PyIE/

PyIE is a hypothesis based, agenda driven, object oriented inference engine written in Python. Inferencing modes include back chaining, opportunistic forward chaining and explicit forward chaining. The first-class object base supports metaclasses, classes, objects and multiple, dynamic inheritance. All objects are first class objects and all attributes are slot values, i.e. data members attached to some object.

PyIE uses a TMS (truth maintenance system) for first level (condition change) non-monotonic reasoning. User defined metaclasses are being explored to support second level (belief change) non-monotonic reasoning.

Simple Neural Net (in Python)

◊ Web site: http://www.amk.ca/python/unmaintained/

Simple neural network code, which implements a class for 3-level networks (input, hidden, and output layers). The only learning rule implemented is simple backpropagation. No documentation (or even comments) at all, because this is simply code that I use to experiment with. Includes modules containing sample datasets from Carl G. Looney's NN book. Requires the Numeric extensions.

Python Fuzzy Logic Module

◊ FTP site: ftp://ftp.csh.rit.edu/pub/members/retrev/

A simple python module for fuzzy logic. The file is 'fuz.tar.gz' in this directory. The author plans to also write a simple genetic algorithm and a neural net library as well. Check the 00_index file in this directory for release info.

QUANT1

◊ Web site: http://linux.irk.ru/projects/QUANT/

This project seems to have gone proprietary. The only trace I can find via google is at http://www.zurich.co.uk/strategicrisk/softwaresupport/Quant1.htm.

QUANT/1 stands for type QUANTifier. It aims to be an alternative to Prolog-like (Resulutional-like) systems. Main features include a lack of necessity for eliminating Quantifiers, scolemisation, ease of comprehension, large scale formulae operation, acceptance of nonHorn formulaes, and Iterative deeping. The actual library implemented in this project is called ATPPCF (Automatic Theorem Prover in calculus of Positively Constructed Formulae).

ATPPCF will be a library (inference engine) and an extension of the Predicate Calculus Language as a new logical language. The library will be incorporable in another software such as TCL, Python, Perl. The engine's primary inference method will be the "search of inference in language of Positively Constructed Formulas (PCFs)" (a subset of Predicate Calculus well translated in both directions). The language will be used as scripting language to the engine. But there will be possibility to replace it with extensions languages of main software.

SCNN

◊ Web site: http://www.uni-frankfurt.de/fb13/iap/e_ag_rt/SCNN/

SCNN is an universal simulating system for Cellular Neural Networks (CNN). CNN are analog processing neural networks with regular and local interconnections, governed by a set of nonlinear ordinary differential equations. Due to their local connectivity, CNN are realized as VLSI chips, which operates at very high speed.

Symbolic Probabilistic Inference (SPI)

◊ FTP site: ftp://ftp.engr.orst.edu/pub/dambrosi/spi/
◊ Paper (ijar-94.ps): ftp://ftp.engr.orst.edu/pub/dambrosi/

Contains Common Lisp function libraries to implement SPI type baysean nets. Documentation is very limited. Features:

◊ Probabilities, Local Expression Language Utilities, Explanation, Dynamic Models, and a TCL/TK based GUI.

SPRLIB/ANNLIB

◊ Web site: http://www.ph.tn.tudelft.nl/~sprlib/

SPRLIB (Statistical Pattern Recognition Library) was developed to support the easy construction and simulation of pattern classifiers. It consist of a library of functions (written in C) that can be called from your own program. Most of the well-known classifiers are present (k-nn, Fisher, Parzen, ...), as well as error estimation and dataset generation routines.

ANNLIB (Artificial Neural Networks Library) is a neural network simulation library based on the data architecture laid down by SPRLIB. The library contains numerous functions for creating, training and testing feed-forward networks. Training algorithms include back-propagation, pseudo-Newton, Levenberg-Marquardt, conjugate gradient descent, BFGS.... Furthermore, it is possible - due to the datastructures' general applicability - to build Kohonen maps and other more exotic network architectures using the same data types.

Sugal

◊ Web site: http://www.trajan-software.demon.co.uk/sugal.htm

Sugal [soo-gall] is the SUnderland Genetic ALgorithm system. The aim of Sugal is to support research and implementation in Genetic Algorithms on a common software platform. As such, Sugal supports a large number of variants of Genetic Algorithms, and has extensive features to support customization and extension.

ThoughtTreasure

◊ Web site: http://www.signiform.com/tt/htm/tt.htm

ThoughtTreasure is a project to create a database of commonsense rules for use in any application. It consists of a database of a little over 100K rules and a C API to integrate it with your applications. Python, Perl, Java and TCL wrappers are already available.

TIN

◊ Web site: http://www.jetlag.demon.nl/

This program simulates primitive life-forms, equipped with some basic instincts and abilities, in a 2D environment consisting of cells. By mutation new generations can prove their success, and thus passing on "good family values".

The brain of a TIN can be seen as a collection of processes, each representing drives or impulses to behave a certain way, depending on the state/perception of the environment (e.g. presence of food, walls, neighbors, scent traces) These behavior process currently are : eating, moving, mating, relaxing, tracing others, gathering food and killing. The process with the highest impulse value takes control, or in other words: the tin will act according to its most urgent need.

Ummon

◊ Web site: http://www.spacetide.com/projects/ummon/

Ummon is an advanced Open Source chatterbot. The main principle of the bot is that it has no initial knowledge of either words or grammar; it learns everything "on the fly." Numerous AI techniques will be explored in the development of Ummon to achieve realistic "human" communication with support for different, customizable personalities.

Virtual Secretary Project (ViSe)

(Tcl/Tk)

◊ Web site: http://www.vise.cs.uit.no/vise/

The motivation of the Virtual Secretary project is to construct user-model-based intelligent software agents, which could in most cases replace human for secretarial tasks, based on modern mobile computing and computer network. The project includes two different phases: the first phase (ViSe1) focuses on information filtering and process migration, its goal is to create a secure environment for software agents using the concept of user models; the second phase (ViSe2) concentrates on agents' intelligent and efficient cooperation in a distributed environment, its goal is to construct cooperative agents for achieving high intelligence. (Implemented in Tcl/TclX/Tix/Tk)

Zeus

◊ Web site: http://more.btexact.com/projects/agents/zeus/

The construction of multi-agent systems involves long development times and requires solutions to some considerable technical difficulties. This has motivated the development of the ZEUS toolkit, which provides a library of software components and tools that facilitate the rapid design, development and deployment of agent system

8.2 Dead projects.

FIPA-OS

◊ Web site: http://fipa-os.sourceforge.net/index.htm

FIPA-OS is an open source implementation of the mandatory elements contained within the FIPA specification for agent interoperability. In addition to supporting the FIPA interoperability concepts, FIPA-OS also provides a component based architecture to enable the development of domain specific agents which can utilise the services of the FIPA Platform agents. It is implemented in Java.

PDKB

◊ Web site: http://lynx.eaze.net/~pdkb/web/
◊ SourceForge site: http://sourceforge.net/projects/pdkb/

Public Domain Knowledge Bank (PDKB) is an Artificial Intelligence Knowledge Bank of common sense rules and facts. It is based on the Cyc Upper Ontology and the MELD language.

RobocodeNG

◊ Web site: http://robocodeng.sourceforge.net/

Merged together with original Robocode as of version 1.1.

Extension of Robocode, the battling bot AI programming game. Like its parent, it is written in Java and meant as a learning environment.

Sulawesi

◊ Web site: http://wearables.essex.ac.uk/sulawesi/

A framework called Sulawesi has been designed and implemented to tackle what has been considered to be important challenges in a wearable user interface. The ability to accept input from any number of modalities, and perform if necessary a translation to any number of modal outputs. It does this primarily through a set of proactive agents to act on the input.

TresBel

◊ Abstract: http://iridia.ulb.ac.be/Projects/imple.html
◊ Direct Download: ftp://iridia.ulb.ac.be/pub/hongxu/software/TresBel.tar.Z

This project seems to have been superseded by Pulcinella .

Libraries containing (Allegro) Common Lisp code for Belief Functions (aka. Dempster-Shafer evidential reasoning) as a representation of uncertainty. Very little documentation. Has a limited GUI.