Numerical Optimization with Neuroevolution

Brian Greer
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78705 USA
bgreer@cs.utexas.edu
Senior Honors Thesis. Advisor: Risto Miikkulainen

December 8, 2001

Abstract

Neuroevolution techniques have been successful in many sequential decision tasks such as robot control and game playing. This paper aims at establishing whether they can be useful in numerical optimization more generally, by comparing neuroevolution to Linear Programming in a manufacturing optimization domain. It turns out that neuroevolution can learn to compensate for uncertainty in the data and outperform Linear Programming when the number of variables in the problem is small and the required accuracy is low, but the current techniques do not (yet) provide an advantage in problems where many variables must be optimized with high accuracy.

1 Introduction

Evolving neural networks with Genetic Algorithms has proven to be a powerful approach to many sequential decision making tasks such as robot control and game playing (Gomez and Miikkulainen 1999, 2001; Moriarty et al. 1999; Whitley et al. 1993; Yao 1999). In such domains, a policy needs to be learned where optimal actions are identified for each state. Usually the state and the actions are discrete, or require only a limited amount of numerical accuracy. An open question at this point is whether neuroevolution methods could also solve large-scale numerical optimization problems such as those occurring in resource management or manufacturing.

This paper presents an application of neuroevolution to such a task: optimizing a metal recycling process. The system has to represent a large number of mass, concentration, and cost variables accurately, and optimize these variables under uncertainty. Neural networks were evolved using a method called Enforced Sub-Populations (ESP), because this method has been shown more powerful than other reinforcement learning techniques in many control tasks (Gomez and Miikkulainen 1997, 1999, 2001).

The main result was that neuroevolution found more accurate solutions than Linear Programming when fewer variables were present. Neuroevolution was also able to improve its performance with a constant number of variables as the amount of accuracy required diminished. This is a promising result, which also presents a significant challenge for further neuroevolution research.

2 Operations Research

Operations research attempts to optimize the use of time, money, raw materials, production facilites, and shipping capabilities. Increasing overall profits is always the bottom line, which means that cost minimization is central to the equation. Optimization of the cost function is commonly formulated as a multi-objective constrained optimization problem, which attempts to minimize the cost function within the constraints of the process domain. A generic formulation of the problem is of the form:

$$\begin{array}{rc} \text{minimize} & F(x), \\ \text{subject to} & b_1 \leq Ax \leq b_2 \end{array}$$

where $F(x)$ is the cost function, x is a vector of variables that are involved in the manufacturing process, A is the matrix of known coeefficients, b_1 and b_2 are the lower and upper bounds, respectively, of the manufacturing process.

Inventory scheduling is the area of operations research that deals with the efficient allotment of raw materials to create finished goods. Costs, amounts, and specifications for each of the raw material must be considered in order to efficiently optimize the overall cost of producing a finished product. Optimization in the metal recycling plant is an instance of such an inventory scheduling problem.

2.1 Linear Programming

Numerical optimization has been effectively tackled using the well-known technique of Linear Programming (LP; see e.g. Dantzig 1967; Taha 1987), which can solve the problem exactly, given that the constraints and the objective function are linear and deterministic.

The simplex method is the most common LP method used in operations research because it gaurantees that an optimal solution will be found within a finite amount of time. It explores the boundary between the feasible and infeasible regions of the the state space by fixing some of the variables in the optimization equation to one of their boundary conditions in order to create a “basic” solution. The algorithm then moves along that boundary looking for a minimal feasible solution by assigning the remaining variables unique values (Dantzig 1967).

Uncertainty in manufacturing data often makes the optimization problem nonlinear, which causes LP to deliver only approximate solutions. One way to solve this problem, in principle, would be to analyze the distribution of the uncertainty in the data and use the observed regularities to improve the optimization process. For example, Genetic Algorithms can be used to detect these kinds of distributions, and they can be encoded in neural networks, which then utilize them during decision making. The ability of neural networks to utilize these observed regularities is the motivation for their use.

3 The Neuroevolution Method

3.1 Genetic Algorithms

Genetic Algorithms (GAs) are inspired by Darwin’s theory of biological evolution (Goldberg 1989; Holland 1975). GAs are distinguished from other global search techniques by the fact that they maintain a population of prospective solutions instead of only optimizing a single solution. The result of the process, however, is still a single individual that represents an optimal solution. Complex problems are approached from the standpoint that a population of parallel search agents is able to converge on a global optimum much more efficiently than a single agent.

Biological organisms encode their traits and characteristics (genomes) in a chromosome. The chromosome is a fundamental aspect in evolution since it contains the hereditary information of the organism that will be passed on to its offspring. Genetic computing must reproduce the chromosome in such a way as to facilitate the use of evolutionary operators (crossover and mutation) on members of the population to create offspring. Neuroevolution methodologies often diverge with respect to the definition of the chromosome in an attempt to find an optimal encoding structure. Often the entire network may be encoded in the chromosome so that the search space contains all networks of a certain structure, and all network weights are evolved simultaneously. Other encoding schemas include encoding individual neurons (Moriarty and Miikkulainen 1996a) or encoding the topology of the network (Stanley and Miikkulainen 2001).

As in biological evolution, the genetic operators of selection, crossover, and mutation are applied to the members of the simulated populations to create successive generations. Selection determines which members of the population are allowed to reproduce and with what frequency. Crossover represents biological reproduction, and mutation simulates random mutations that may occur during reproduction. During each generation, members of the population are evaluated on a given task and assigned a fitness value. Those members of the population with higher fitness values are more likely to be selected for reproduction thereby allowing whatever dominant traits they posses to help solve the problem to carryover to the next generation. This model is constructed in a similar fashion to a “survival of the fittest” approach in Darwinian evolution where members of the population with desireable traits are more likely to be selected by other members of the population for reproduction.

GAs have proven to be effective in many real world optimization tasks such as electromagnetic system design (Michielssen and Weile 1995) and aerodynamic design (Periaux et al. 1995).

3.2 Nueroevolution

GAs have been used in conjunction with neural netwoks to evolve weights in fixed architecture networks, evolve network topologies, and to select training data and interpret output behavior of neural networks. When neural networks are evolved using GAs the method is referred to as neuroevolution.

The training is done in neuroevolution using a reinforcement system whereby individual networks are tested on a given task and assigned a fitness, which then determines their relative position in a population of networks. Superior networks (networks that perform well in the given task) are allowed to reproduce more often (have a higher selection probability). A population of networks is evolved in parallel in this manner until some threshold fitness is met or the performance of the population stagnates.

The reinforcement aspect of neuroevolution stems from the fact that networks do not require immediate feedback during the training process, rather a general measure of fitness is calculated after the entire task has been completed. Previous knowledge about the domain is also not required with this method since pertinent knowledge is acquired through the evolution process. This allows the networks to ascertain what knowledge is required in order to solve the problem sufficiently.

Stagnation of the population is handled differently for different methodologies. One possible solution to stagnation is to replace some percentage of the unfit members of the population by random mutations of the best network. This is how Delta-Coding in ESP works as discussed below.

3.3 SANE

David Moriarty altered the classic approach to neuroevolution by evolving populations of neurons instead of populations of entire networks. This method is known as Symbiotic Adaptive Neuroevolution (SANE; Moriarty and Miikkulainen 1996a, 1997). Each neuron is encoded with its input and output weights in a string (chromosome). Three-layer feedforward networks are built using a randomly chosen neuron from each population to form the hidden layer. The network is then evaluated and given a fitness value. This fitness value is passed on to each neuron that participated in the network. Neurons are then selected for based on how well they performed in each of the networks in which they participated.

Since good networks require different kinds of neurons, diversity is automatically maintained in the population. The network architecture itself, however, is fixed during evolution so that only the weight space of the individual neurons is explored.

3.4 ESP

SANE was extended by Faustino Gomez by grouping hidden neurons into subpopulations in a method known as Enforced Sub-populations (ESP; Gomez and Miikkulainen 1997, 1999, 2001). Crossover only occurs between neurons of a particular subpopulation allowing each neuron to specialize in its role in the network (figure 1). In effect, evolution not only maintains diversity automatically, it also defines compatible subtasks, and finds the solution by optimizing each subtask. Such subtasking makes the search for solution more efficient.

ESP was also used with Delta-Coding turned on in order to handle stagnation. Whenever ESP detects stagnation in the current population, a percentage of the population with the lower fitness values is repleced by perturbations of the best neuron.