ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-783

CTRNN Parameter Learning using Differential Evolution

Ivanoe De Falco[1] and **Antonio Della Cioppa**[2] and **Francesco Donnarumma**[3] and **Domenico Maisto**[1] and **Roberto Prevete**[3] and **Ernesto Tarantino**[1]

Abstract. Target behaviours can be achieved by finding suitable parameters for Continuous Time Recurrent Neural Networks (CTRNNs) used as agent control systems. Differential Evolution (DE) has been deployed to search parameter space of CTRNNs and overcome granularity, boundedness and blocking limitations. In this paper we provide initial support for DE in the context of two sample learning problems.

Key words: CTRNN, Differential Evolution, Dynamical Systems, Genetic Algorithms

1 INTRODUCTION

Insofar as Continuous Time Recurrent Neural Networks (CTRNNs) are universal dynamics approximators [1], the problem of achieving target agent behaviours is redefined as the problem of identifying suitable network parameters.

Although a variety of different learning algorithms exists, evolutionary approaches like Genetic Algorithms (GA) are usually deployed to perform searches in the parameter space of CTRNNs [5]. However GAs require some kind of network encoding which may greatly influence parameter searches. In fact, the resolution of the parameters is limited by the bit resolution of the encoding (***granularity***) and the parameters cannot assume values falling outside an encoding a priori fixed interval (***boundedness***).

Yamauchi and Beer [5] proposed a real-valued encoding for CTRNNs, which improves the learning process allowing parameter values to be in $\mathbb{R}$. However, problems arise that, in not rare cases, prevent real-valued GAs (rvGA) from finding global optima (***blocking)*** [2].

Here we propose an approach based on a Differential Evolution (DE) algorithm [4] which combines fast learning with the possibility of overcoming the limitations mentioned above.

Section 2 introduces the DE algorithm. In Section 3 two sample CTRNN parameter search problems are solved with DE. Finally in Section 4, the obtained results are discussed and future developments of this approach are proposed.

2 DIFFERENTIAL EVOLUTION

DE is a stochastic, population-based evolutionary algorithm [4] which addresses a generic optimization problem with m real parameters by starting with a randomly initialized population consisting of n individuals, each made up of m real values, and, subsequently, by updating the population from a generation to the next one by means of many different transformation schemes commonly named as *strategies* [4]. In all of these strategies DE generates new individuals by adding to an individual a number of weighted difference vectors made up of couples of population individuals.

In the strategy chosen, starting from $\mathbf{x}_i$, the i-th individual, a new trial one $\mathbf{x}'_i$ is generated, by perturbing the best individual $\mathbf{x}_{best}$ by means of 2 difference vectors. The generic j-th component candidate is:

$$x'_{i,j} = x_{best,j} + F \cdot [(x_{r_1,j} - x_{r_2,j}) + (x_{r_3,j} - x_{r_4,j})]$$

with 4 randomly generated integer numbers r_1, r_2, r_3, r_4 in $\{1, \ldots, n\}$, differing from one another and F the parameter which controls the *magnitude* of the differential variation.

So in DE new candidate solutions are created by using vector differences, whereas traditional rvGAs rely on probabilistic selection, random perturbation (mutation) and on mixing (recombination) of individuals. The three phases of a standard rvGA, selection, recombination and mutation, are *combined* in DE in one operation which is carried out for each individual. According to this, in rvGA not all the elements are involved in each phase of the generation of the new population, while, by contrast, DE algorithm iterates through the entire population and generates a candidate for each individual.

3 EXPERIMENTS

We tested the efficacy of CTRNN training by DE on two sample experiments where the approach seems to solve problems outlined in Section 1. Parameters ruling the DE algorithm were assigned experimentally via a set of training trials.

3.1 Cusp point learning

Let us consider a CTRNN made up of a single self-connected neuron. The equation of the system is given by

$$\tau \cdot \dot{y} = -y + w\sigma(y + \theta) + I \quad (1)$$

where for simplicity we set the time constant $\tau = 1$ and the bias $\theta = 0$. Notice that no elementary expression for the solution of (1) exists. Such system has a *cusp point*, that is the only bifurcation point in which the system undergoes a

[1] ICAR-CNR, Naples, Italy -{ivanoe.defalco, domenico.maisto, ernesto.tarantino}@na.icar.cnr.it
[2] DIIIE, Università di Salerno - adellacioppa@unisa.it
[3] Università di Napoli Federico II - {donnarumma, prevete}@na.infn.it

picthfork bifurcation [3]. The goal of the experiment is to find such cusp point. To evaluate each network candidate (I', w') we let it evolve for a *sufficient* time T so that we can consider $y'(T) \simeq \bar{y}'$. Then we choose as fitness function $F_{CP}(y'(I', w')) = f_{fixed} + f_{tan} + f_{cusp}$ with terms rewarding respectively fixed point, non hyperbolic and cusp curve intersection condition.

Average and standard deviation values found for (I, w) in 10 runs using the DE algorithm are $\bar{I} = -2.00015$ with a standard deviation equal to $1.6 \cdot 10^{-4}$ and $\bar{w} = 4.0003$ with standard deviation $3.1 \cdot 10^{-4}$. These values are absolutely close to the coordinates $(\tilde{I}, \tilde{w}) = (-2, 4)$ of the cusp point which can be formally inferred. Figure (1) shows fitness trend as a function of the generation number for average, best and worst case. The constant and smooth decrease suggests a gradual and continuous learning improvement as the generation number grows. In addition, the evident increasing resolution of the parameter values observable during DE runs demonstrates the possibility of tackling the granularity problem, theoretically having the machine precision as only limit.

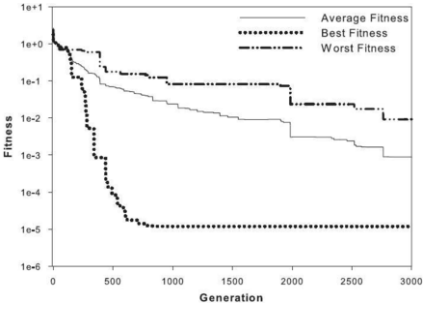

Figure 1. Cusp point learning: fitness plots of runs corresponding to the average, worst and best solutions as a function of the generation number

3.2 Sequence generator task

The goal of this task is to train a control network able to switch between two different behaviours (fixed points 0 and 1) anytime a signal trigger is detected [5]. Focusing on a network of three neurons, we generate a random sequence for each generation $\mathbf{I}' = [bit^1, \ldots, bit^M]$, where M is the length of the sequence and $bit^i \in \{0, 1\}\ \forall i \in M$. The length of every sequence of 0 (no signal) or 1 (trigger) has been extracted from a Gaussian distribution. For every sequence generation we generate the desired target $\mathbf{t}' = \{t^1, \ldots, t^M\}$. We measure the output candidate $\mathbf{y} = \{\bar{y}_3^1, \ldots, \bar{y}_3^M\}$ with a fitness function $F_{SG}(\mathbf{y}(\mathbf{w})) = F_{HM}(\mathbf{y}(\mathbf{w})) + k \cdot F_{HD}(\mathbf{y}(\mathbf{w}))$ with the first term (the Hamming distance) and the second term respectively measuring *how many times* and *how different* the fixed point values are from the desired targets. We set $k = 0.01$ so as to weight the first contribute more than the second.

In each of the 10 runs DE is able to find optimal solutions, even reaching the global minima. It is worth remarking that the weights found are very sparse (e.g. $w \approx 21.19$ and $w \approx 1.86 \cdot 10^{18}$) so that by fixing a priori intervals many good solutions would become inaccessible. This sparseness suggests that DE is almost able to investigate the entire parameter space allowing the surmounting of the boundedness problem. Moreover each run passes through a different sequence of local minima, from which the DE algorithm has to escape. So the descent of the function towards the global minimum occurs in "steps" (see **Left** of Figure 2). **Right** of Figure 2 illustrates how the search of the parameters continues even in the very proximity of optimal values, finding better and better solutions. Moving by vector differences in the parameter space is "as if" DE is capable of *calibrating* the magnitude and the direction towards the reaching of the minima in it. The result is that every run is able to overcome the blocking problem.

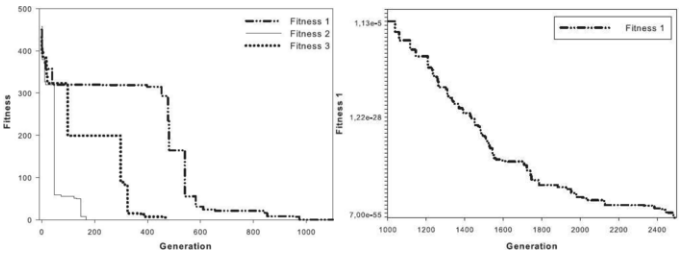

Figure 2. Sequence generator task. **Left:** fitness of three different runs plotted as a function of the generation number. **Right:** Fitness 1 from 1000-th to 2500-th generation. Figures show DE avoiding blocking by escaping from local minima.

4 CONCLUSIONS

We showed two experiments solved by means of DE which provides a simple and a "physical" way to perform CTRNN parameter space search. The first experiment provides an example of how the *granularity* problem can be overcome. DE showed a high precision in determining the parameter values which can be still improved by letting the execution run. The second experiment points to ways in which *boundedness* and *blocking* can be overcome, too, by a DE approach. Using only three neurons we solve the sequence generator task. The found parameter values are sparse, so fixing a priori intervals would have cut many possible solutions. Furthermore, although each run passes through a sequence of local minima, DE algorithm can escape from them jumping step by step towards a better approximation of a global minimum.

After this encouraging results next studies will concern a direct comparison with rvGAs particularly on local minima trapping issue and a deeper investigation on theoretical details of DE approach for CTRNN learning.

References

[1] Ken-ichi Funahashi and Yuichi Nakamura, 'Approximation of dynamical systems by continuous time recurrent neural networks', *Neural Networks*, **6**(6), 801–806, (1993).
[2] David E. Goldberg, 'Real-coded genetic algorithms, virtual alphabets, and blocking', *Complex Systems*, **5**, 139–167, (1991).
[3] J. K. Hale and H Kocac, *Dynamics and Bifurcations*, Springer-Verlag, 1991.
[4] K Price, R Storn, and Lampinen J, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer-Verlag, 2005.
[5] Brian M. Yamauchi and Randall D. Beer, 'Sequential behavior and learning in evolved dynamical neural networks', *Adaptive Behavior*, **2**(3), 219–246, (1994).