

Robust Evolution Strategies Use Adaptive Search Strategies to Design Continuous-Time Recurrent Neural Networks

Y. Matsumura¹, X. Yao¹, J. L. Wyatt¹, K. Ohkura² and K. Ueda³

¹ School of Computer Science, The University of Birmingham,
Edgbaston, Birmingham B15 2TT, U.K.

{Y.Matsumura, X.Yao, J.L.Wyatt}@cs.bham.ac.uk

² Faculty of Engineering, Kobe University,
Rokko, Nada, Kobe 657-8501, JAPAN

ohkura@mech.kobe-u.ac.jp

³ Research into Artifacts, Center for Engineering, The University of Tokyo,
Komaba, Meguro, Tokyo 153-8904, JAPAN

ueda@race.u-tokyo.ac.jp

Abstract

This paper empirically investigates the use and behaviour of Evolution Strategies (ES) algorithms on problems such as function optimisation and the use of evolutionary artificial neural networks in evolutionary robotics. Computer simulations are conducted which compare the performance of Classical-ES (CES) and Robust-ES (RES). We show that the performance of the RES algorithm improves on that of the CES algorithm. Most importantly statistical analyses of the evolutionary behaviour show that the CES algorithm keeps the same search strategy regardless of domain while the RES algorithm changes the search strategy to fit the problem.

1 Introduction

Evolutionary Artificial Neural Networks (EANN) [16][18] are referred to as the combination of Evolutionary Algorithms (EAs) and Artificial Neural Networks (ANN). EANNs are great interested in some fields, such as Evolutionary Robotics (ER) [5], Traffic Flow Prediction in Telecommunication, Breast Cancer Diagnosis and so on. Especially, ER researchers propose EANN as a methodology for the control of autonomous mobile robot involving artificial evolution for designing ANN.

ER researchers use different types of ANN, such as Continuous-Time Recurrent Neural Networks (CTRNNs) [2][3], GasNet[8], and Spike Response Models [4]. CTRNNs have some good features such as the possibility of time variance and various behaviours even in small networks [14][15]. While EAs have been widely

recognised as a robust approach to various kinds of optimisation problems, there are some main streams in this field, i.e., Evolution Strategies (ES) [1][13], Evolutionary Programming, Genetic Algorithms and so on. Following Harvey [6], Salomon [12] and others, we believe we should more frequently discuss which kinds of artificial evolution are better for designing ANNs, since designing ANNs is one of the hardest and the most essential problems in ER.

This paper concentrates on which kind of ES algorithm is better for designing CTRNNs, and in particular we look at how ES algorithms search the solution space. In general, for the purpose of designing high performance EAs, it is important to investigate their evolutionary behaviour on both simple test functions and real engineering problems, since the knowledge gained contributes to developing new, improved algorithms. Therefore, this paper investigates not only which types of ES algorithm give the highest performance, but also discusses the differences in the search strategies employed on an artificial test problem and a problem in ER.

First, experiments were conducted on a test function optimisation problem and a test ER problem based on the Sussex approach[7]. Then, we statistically analyse the evolutionary behaviour of the ES algorithms employed in order to find the differences in their search strategies. Among the many formulations of ES, we test two types: Classical-ES (CES), i.e. ES using Gaussian mutation; and Robust-ES (RES), i.e. our extended ES using Cauchy mutation.

The rest of this paper is organised as follows. Sec-

tion 2 briefly introduces the computational procedures of CES and RES. Section 3 shows the setup of computer experiments and the results. Section 4 analyses and discusses the statistical evolutionary behaviour of CES and RES. Finally, the conclusions are given in section 5.

2 Evolution Strategies

ES can take several forms [1][13], but the form adopted in this paper is (μ, λ) -ES, where $\lambda > \mu \geq 1$. (μ, λ) means that μ parents generate λ offspring through mutation in each generation. The best μ offspring are selected deterministically from the λ offspring and replace the current parents. Elitism and stochastic selection are not used.

2.1 Classical Evolution Strategies

The Classical ES (CES) algorithm adopted in this paper is described as follows:

1. Generate an initial population of μ individuals, and set $g = 1$. Each individual is taken as a pair of real-valued vectors $(\mathbf{x}_i, \boldsymbol{\eta}_i), \forall i \in \{1, \dots, \mu\}$, where \mathbf{x}_i and $\boldsymbol{\eta}_i$ are the i -th coordinate value in the object and the strategy parameters (larger than zero), respectively.
2. Evaluate the objective value for each individual $(\mathbf{x}_i, \boldsymbol{\eta}_i), \forall i \in \{1, \dots, \mu\}$ in the population, based on the objective function $f(\mathbf{x}_i)$.
3. Each parent $(\mathbf{x}_i, \boldsymbol{\eta}_i), i = 1, \dots, \mu$, creates λ/μ offspring on average, so that a total of λ offspring are generated. The offspring are generated as follows: for $i = 1, \dots, \mu, j = 1, \dots, n$, and $p = 1, \dots, \lambda$,

$$\begin{aligned} \eta'_p(j) &= \eta_i(j) \exp\{\tau' N(0, 1) + \tau N_j(0, 1)\} (1) \\ x'_p(j) &= x_i(j) + \eta'_p(j) N_j(0, 1) \end{aligned} \quad (2)$$

where $x_i(j), x'_p(j), \eta_i(j)$ and $\eta'_p(j)$ denote the j -th component of the vectors $\mathbf{x}_i, \mathbf{x}'_p, \boldsymbol{\eta}_i$ and $\boldsymbol{\eta}'_p$, respectively. $N(0, 1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0, 1)$ indicates that the random number is generated anew for each value of j . The factors τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$. Various types of recombination operators can also be applied before calculating equations (1) and (2).

4. Calculate the fitness of each offspring $(\mathbf{x}'_i, \boldsymbol{\eta}'_i), \forall i \in \{1, \dots, \lambda\}$, according to $f(\mathbf{x}'_i)$.

5. Sort offspring $(\mathbf{x}'_i, \boldsymbol{\eta}'_i), \forall i \in \{1, \dots, \lambda\}$ according to their fitness values, and select the μ best offspring out of λ to be parents of the next generation.
6. Stop if the halting criterion is satisfied; otherwise, $g = g + 1$ and go to step 3.

2.2 Fast Evolution Strategies

Yao and Liu [17] proposed the Fast ES (FES) algorithm, a variant of the (μ, λ) -ES. In FES, the Gaussian mutation (step 3 above) is replaced by Cauchy mutation, using the following Cauchy distribution function:

$$F_t(x) = 1/2 + (1/\pi) \arctan(x/t) \quad (3)$$

where $t = 1$. The success of FES is explained as a result of a larger probability of escaping from local optima, due to the fatter tails of the Cauchy mutation operator. In other words the Cauchy distribution has a higher probability than the Gaussian distribution of producing large mutations. Yao and Liu [17] conducted empirical experiments using a number of test functions, demonstrating an improvement in performance, especially on multi-modal problems.

2.3 Robust Evolution Strategies

When ESs are applied to an optimisation problem successfully, the observed evolutionary dynamics show qualitatively similar behaviour to that of other evolutionary algorithms: over generations the focus of the search shifts from global regions to smaller local regions. This arises from the gradual convergence of the population due to the direct effects of natural selection. Associated with this, the strategy parameters $\boldsymbol{\eta}_i$ tend to zero. This is the process of “self-adaptation”, which is considered to be one of the most attractive features of ES. Although this may be useful for uni-modal functions, in many multi-modal functions, ES are often trapped in local optima.

Robust-ES (RES) [11] was designed to avoid this problem of entrapment. The key idea of RES is to utilise selectively neutral mutations (Kimura, 1983) on strategy parameters so that the algorithm is capable of rapidly increasing or decreasing strategy parameters, irrespective of natural selection. RES follows the same procedure as CES or FES except for the following two points (See [11] in detail):

- A different individual representation is used, incorporating redundant strategy parameters, i.e. inactive strategy parameters, which have no effect on the selection process.

- Extra stochastic mutation mechanisms are used to change the original strategy parameters. These mutations replace, swap or copy active strategy parameters with “inactive” strategy parameters.

2.3.1 Individual Representation in RES

An individual \mathbf{X}_i is represented as follows, assuming that $i = 1, 2, \dots, \mu$, $j = 1, 2, \dots, n$, $k = 0, 1, \dots, m$ and $\eta_{ik}(j) \in R^+$:

$$\mathbf{X}_i = [\mathbf{x}_i, (\boldsymbol{\eta}_{i0}, \dots, \boldsymbol{\eta}_{ik}, \dots, \boldsymbol{\eta}_{im})] \quad (4)$$

$${}^t\mathbf{x}_i = (x_i(1), \dots, x_i(j), \dots, x_i(n)) \quad (5)$$

$${}^t\boldsymbol{\eta}_{ik} = (\eta_{ik}(1), \dots, \eta_{ik}(j), \dots, \eta_{ik}(n)) \quad (6)$$

where $x_i(j)$ and $\eta_{ik}(j)$ denote the j -th component of the vectors \mathbf{x}_i and $\boldsymbol{\eta}_{ik}$, respectively. The symbol t means transposed matrix. Note that each $x_i(j)$ has $(m + 1)$ strategy parameters [11].

2.3.2 Mutation Mechanisms for Strategy Parameters in RES

We define D as the mutation mechanism given in equation(1). In addition, $\boldsymbol{\eta}_{ik}$ is modified stochastically, according to the following new mutation operators:

- O_{dup} retains $\eta_{i0}(j)$, shifts all of $\eta_{ik}(j)$ into the adjacent position of $(k + 1)$ and removes $\eta_{im}(j)$ from the list. Then, O_{dup} mutates all $\boldsymbol{\eta}_{ik}$ with D .
- O_{del} discards $\eta_{i0}(j)$ and moves $\eta_{ik}(j)$ to the adjacent position of $(k - 1)$. At the m -th position η_L is calculated as the smaller value of η_{max} and $\sum_{p=1}^{m-1} \eta_{ip}(j)$. Then, O_{del} mutates all $\boldsymbol{\eta}_{ik}$ with D .
- O_{inv} swaps $\eta_{i0}(j)$ with one of $\eta_{ik}(j)$, $k = 1, \dots, m$ and mutates $\eta_{i0}(j)$ and $\eta_{ik}(j)$ with D .

When the probabilities of O_{dup} , O_{del} and O_{inv} are set at 1.0, 0.0 and 0.0, respectively RES is equivalent to FES. Thus FES can be seen as a special case of RES.

3 Computer Simulations

3.1 Test Functions

We conduct a series of computer simulations using the test functions given by Yao and Liu [17]. Due to limited space, only the results for sphere function are presented. This function defines a 30 dimensional problem ($n=30$) where the global minimum is 0. This function is the basic unimodal function as follows:

$$f(x) = \sum_{i=1}^n x_i^2 \quad (-100 \leq x_i \leq 100) \quad (7)$$

The experimental setup is based on Yao and Liu [17]: $(\mu, \lambda) = (30, 200)$ with Gaussian or Cauchy mutation, no recombination, and no correlated mutations. CES and RES use the same initial populations. All simulations are independently repeated for 50 runs. The upper bound of the strategy parameters η_{max} is set to 3.0. In RES, the number of inactive strategy parameters m for each variable is set to 5. O_{dup} , O_{del} and O_{inv} are applied with the probabilities of 0.6, 0.3 and 0.1, respectively. The main purpose of our computer simulations is to investigate the differences between the evolutionary behaviour of the two algorithms. Thus, the parameters are not fully tuned. They are not meant to be optimal.

Figure 1 shows the averaged best results on sphere function. With CES the average improves until about generation 900 after which there is no further practical improvement of the average values. In contrast, with RES the average continually improves up until the last generation.

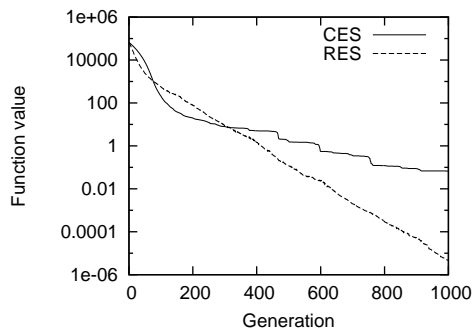


Figure 1: Averaged best results on Sphere function.

3.2 Evolutionary Artificial Neural Networks

Based on the Sussex approach [5][7], we construct the minimal simulator for Khepera robot (in Fig.2). In this test simulation, the Khepera robot is able to move around, avoiding obstacles and seeking light source. The Khepera robot is equipped with 8 active infrared(IR) sensors, 2 light sensors and 2 motor-driven wheels. The Khepera robot itself is 5.5cm in diameter and the sensor range is approximately 5.2cm, note that the field size is 60×60cm. The IR sensors give the robot distal information (i.e a noisy and non-linear indication of the presence of an object).

The Khepera robot is controlled by CTRNNs (in Fig.3). Fully connected neurons are governed by fol-

lowing state equation:

$$\begin{aligned} \tau_i \dot{\gamma}_i &= -\gamma_i + \sum_{j=1}^N w_{ji} \sigma(g_j(\gamma_j + \theta_j)) \\ &+ \sum_{j=1}^S S_{ji} I_j \quad (i = 1, \dots, N) \end{aligned} \quad (8)$$

where γ is the state of the neuron, τ is the time constant of the neuron, N is the total number of neurons, w_{ij} gives the strength of the connection from the j th to the i th neuron, σ is the standard sigmoidal activation function, g is a gain factor, θ is a bias term, S is the number of sensory inputs, S_{ji} is the strength of the connection from the j th sensor to i th neuron, and I_j is the sensory input with noise. Note that every neuron receives an input from every sensor. The sensory input consists of 11 inputs ($S=11$) (8 IR sensors, 2 light sensors and the bias). The initial state of all neurons is set to zero and the system of equations is integrated using the forward Euler method with a step-size of 0.1. Time constants are in the range $(0,2]$, biases and gains are in the range $[-5,5]$ and connection weights are in the range $[-5,5]$. The sensor reading ($x'_i: i=1, \dots, S$) is of type integer and the values are in the range $[0,1023]$. Based on notations in [9], the sensory input (I_i) is calculated as follows:

$$I_i = x'_i + N(0, 1) \quad (9)$$

Also, speed values ($Y_j: j=1,2$) are calculated as follows:

$$Y_j = 10.0 \times \gamma_j + 5.0 \times N(0, 1) \quad (10)$$

$(-10 \leq Y_j \leq 10)$

The Khepera robot is placed at the starting point with its heading direction oriented randomly. The fitness value (F_i) is calculated as follows:

$$F_i = 1000 - T \quad (11)$$

where T is the number of time-steps when the Khepera robot reaches the goal (light source), note that maximum time-step is 1000. The time constant of the neuron, the gain factor, the bias term, the connection weights and the number of neurons are optimised by the ES algorithm. The experimental setup of ES algorithms is same as that on the test function.

The averaged best fitness values for 50 runs are shown in Fig.4. With CES the average increases until about generation 100 after which there is no further practical improvement of the average values and with RES the average increases until about generation 300. These results suggest that RES improves the performance of CES.

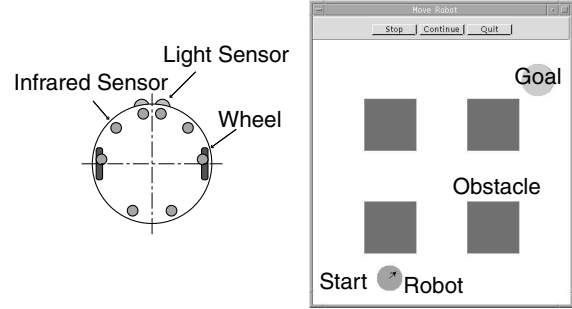


Figure 2: Simulation models of Khepera robot and environment.

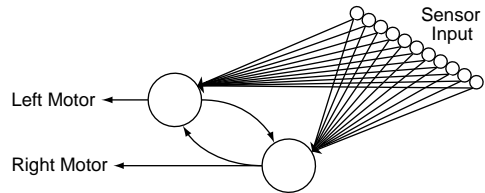


Figure 3: The topology of CTRNNs.

4 Statistical Analyses

In order to discuss the differences in the search strategies employed between the artificial test problem and the robot problem, we analyse the statistical evolutionary behaviour of two types of ES algorithms. Then, we use Hotelling's T^2 which is a statistical measure of the multivariate distance of each observation from the centre of a data set. In an evolutionary population, individuals located far from the centre of the population can be said to be engaged in exploration, while those located close to the centre are engaged in exploitation. Hotelling's T^2 is useful because it gives a measure of the ratio of population individuals engaged in local search (exploitation) to those engaged in global search (exploration). Hotelling's T^2 is calculated as follows:

$$\mathbf{T}^2 = \lambda(\lambda - 1)(\bar{\mathbf{x}} - \boldsymbol{\delta})' \mathbf{S}^{-1}(\bar{\mathbf{x}} - \boldsymbol{\delta}) \quad (12)$$

$$\bar{\mathbf{x}} = \sum_{i=1}^{\lambda} \frac{\mathbf{x}_i}{\lambda} \quad (13)$$

$$(\mathbf{x}_i \in N(\boldsymbol{\delta}, \boldsymbol{\sigma}_\zeta))$$

$$\mathbf{S} = \sum_{i=1}^{\lambda} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})' \quad (14)$$

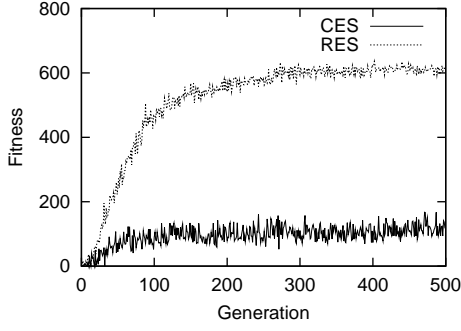


Figure 4: Averaged best results on EANN.

where λ is the number of data points. Assume that \mathbf{x}_i follow $N(\boldsymbol{\delta}, \boldsymbol{\sigma}_\zeta)$. $N(\boldsymbol{\delta}, \boldsymbol{\sigma}_\zeta)$ means the N-dimensional normal distribution in which $\boldsymbol{\delta}$ and $\boldsymbol{\sigma}_\zeta$ are the average vector and the covariance, respectively. If \mathbf{x}_i has the probability density function $f(\mathbf{x})$, $\boldsymbol{\delta}$ is calculated as follows:

$$f(\mathbf{x}) = (2\pi)^{-n/2} |\boldsymbol{\sigma}_\zeta|^{-1/2} \times \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\delta})' \boldsymbol{\sigma}_\zeta^{-1} (\mathbf{x} - \boldsymbol{\delta})\right] \quad (15)$$

$$E(\mathbf{x}_i) = \boldsymbol{\delta} \quad (16)$$

4.1 Test Functions

Hotelling's T^2 on sphere function is shown in Fig.5 and Fig.6 when the generation is 100, 500 and 1000. These results clearly demonstrate that Hotelling's T^2 on CES is different from that on RES. In Fig.5, there is no individual who is more than 150 of Hotelling's T^2 value and less than 10 of Hotelling's T^2 value. This means that no individual is located far from the centre of the population and located close to the centre.

In Fig.6, the number of individuals which are more than 150 of Hotelling's T^2 value increases from ultimately 20 to ultimately 30 as the runs progress. The number of individuals which are less than 10 of Hotelling's T^2 value increases from ultimately 40 to ultimately 70 as the runs progress. This means that the distribution changes in such a way that ultimately 10% of individuals are located far from the centre of the population, while 20% of individuals are located close to the centre. Then, RES gradually shifted toward a strategy composed of broad exploration (20%) and fine exploitation (35%).

4.2 Evolutionary Artificial Neural Networks

Hotelling's T^2 on EANN is shown in Fig.7 and Fig.8 when the generation is 100, 250 and 500. These re-

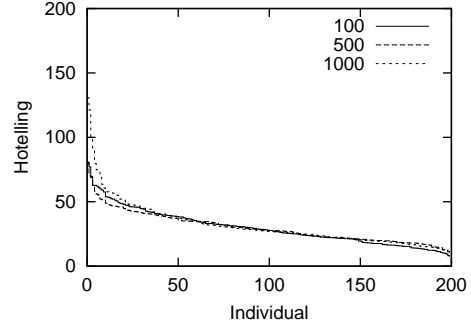


Figure 5: Hotelling's T^2 for CES on Sphere function.

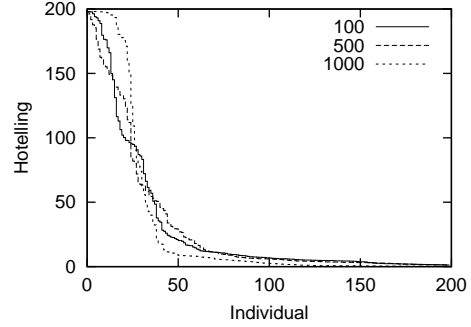


Figure 6: Hotelling's T^2 for RES on Sphere function.

sults clearly demonstrate that Hotelling's T^2 on CES is different from that on RES. In Fig.7, there is no individual who is more than 150 of Hotelling's T^2 value and less than 10 of Hotelling's T^2 value. This means that no individual is located far from the centre of the population and located close to the centre.

In Fig.8, the number of individuals which are more than 150 of Hotelling's T^2 value increases from ultimately 5 to ultimately 15 as the runs progress. There is no individual who is less than 10 of Hotelling's T^2 value. This means that the distribution changes in such a way that ultimately 2.5% of individuals are located far from the centre of the population. Then, RES gradually shifted toward a strategy of broad exploration (7.5%).

4.3 Summaries

To summarise our analyses, in the case of CES, there is no difference in search strategies between test problems of sphere function and EANN. CES on each problem keeps the same search strategy over generations. However, RES has different search strategies

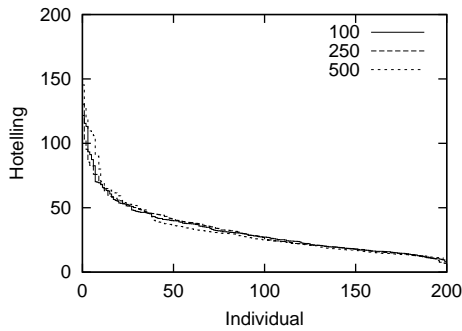


Figure 7: Hotelling's T^2 for CES on EANN.

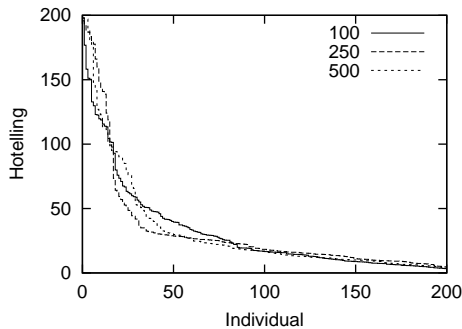


Figure 8: Hotelling's T^2 for RES on EANN.

on the sphere function and on the ER problem. In the case of the sphere function, RES changes the local search strategy by shifting toward a fine grained exploitation over several generations. On the contrary, in the case of EANN, RES retains the local search strategy over the generations. This is good evidence that RES can adapt its search strategy to the features of a problem.

5 Conclusions

This paper empirically investigated the ES algorithms with two test problems: the sphere function and that of evolving EANNs in evolutionary robotics. Computer simulations were conducted to compare the performance of two types of ES algorithms. We saw that the performance of the RES algorithm improved on that of CES algorithm. Statistical analyses show that the CES algorithm keeps the same search strategy for both, while the RES algorithm adapts its search strategies to the features of each problem.

References

- [1] T. Bäck (1996), *Evolutionary Algorithms in Theory and Practice*, Oxford University Press.
- [2] R. D. Beer, and J.C. Gallagher (1992), "Evolving Dynamical Neural Networks for Adaptive Behavior", *Adaptive Behavior* Vol.1, No.1, pp.91-122.
- [3] R. D. Beer (1995), "On the Dynamics of Small Continuous-Time Recurrent Neural Networks", *Adaptive Behavior*, Vol.3, No.4, pp.471-511.
- [4] D. Floreano and F. Mattiussi (2001) "Evolution of Spiking Neural Controllers for Autonomous Vision-based Robot", *Evolutionary Robotics IV*, Lecture Notes in Computer Science, Vol. 2217, pp. 38-61, Springer.
- [5] I. Harvey, P. Husbands and D. Clif (1993), "Issues in Evolutionary Robotics", *From animals to animats 2: Proc. of the second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pp. 364-373, MIT Press.
- [6] I. Harvey (1994), "Evolutionary Robotics and SAGA: the Case for Hill Climbing and Tournament Selection", *Artificial Life III*, pp. 299-326, Addison Wesley.
- [7] I. Harvey, P. Husbands, D. Clif, A. Thompson, and N. Jakobi (1997), "Evolutionary Robotics: the Sussex Approach", *Robotics and autonomous systems*, Vol. 20, pp. 205-224.
- [8] P. Husbands, T.M.C Smith, N. Jakobi and M. O'Shea (1998), "Better Living Through Chemistry: Evolving Gas-Nets for Robot Control", *Connection Science*, 10(3-4), pp. 185-210.
- [9] N. Jacobi, P. Husbands and I. Harvey (1995), "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics", *Third European Conference on Artificial Life (ECAL95)*, pp.704-720, Springer.
- [10] M. Kimura (1983), *The Neutral Theory of Molecular Evolution*, Cambridge University Press.
- [11] K. Ohkura, Y. Matsumura and K. Ueda (2001), "Robust Evolution Strategies", *Applied Intelligence*, Vol. 15, No. 3, pp.153-169, Kluwer Academic Publishers.
- [12] R. Salomon (1996), "Increasing Adaptivity through Evolution Strategies", *From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, pp.411-420, MIT Press.
- [13] H.-P. Schwefel (1995), *Evolution and Optimum Seeking*, John Wiley & Sons.
- [14] E. Tuci, I. Harvey and M. Quinn (2002), "Evolving Integrated Controllers for Autonomous Learning Robots using Dynamic Neural Networks", *Proc. of The Seventh International Conference on the Simulation of Adaptive Behavior (SAB'02)*, pp.282-291, MIT Press.
- [15] E. Tuci, I. Harvey and P.M. Todd (2002), "Using a Net to Catch a Mate: Evolving CTRNNs for the Dowry Problem", *Proc. of The Seventh International Conference on the Simulation of Adaptive Behavior (SAB'02)*, pp.292-302, MIT Press.
- [16] X. Yao (1993), "A Review of Evolutionary Artificial Neural Networks", *International Journal of Intelligent Systems* Vol.8, pp. 539-567.
- [17] X. Yao and Y. Liu (1997), "Fast Evolution Strategies", *Control and Cybernetics*, Vol. 26, No. 3, pp.467-496.
- [18] X. Yao (1999), "Evolving Artificial Neural Networks", *Proc. of the IEEE*, Vol. 87 No. 9, pp.1423-1447.