Elucidating the Benefits of A Self-Adaptive Pareto EMO Approach for Evolving Legged Locomotion in Artificial Creatures

Jason Teo
School of Engineering and Information Technology,
Universiti Malaysia Sabah,
Kota Kinabalu, Sabah, Malaysia.
j.teo@ums.edu.my

Hussein A. Abbass
Artificial Life and Adaptive Robotics (A.L.A.R.) Lab,
School of Information Technology and Electrical
Engineering, University of New South Wales @
Australian Defence Force Academy, Canberra, Australia.
h.abbass@adfa.edu.au

Abstract- A self-adaptive Pareto *Evolutionary Multi-objective Optimization* (EMO) algorithm based on differential evolution is proposed for evolving locomotion controllers in an artificially embodied legged creature. The objective of this paper is to demonstrate the trade-off between quality of solutions and computational cost. We show empirically that evolving controllers using the proposed algorithm incurs significantly less computational cost compared to a self-adaptive weighted sum EMO algorithm, a self-adaptive single-objective evolutionary algorithm and a hand-tuned Pareto EMO algorithm. The main contribution of the self-adaptive Pareto EMO approach is its ability to produce sufficiently good controllers with different locomotion capabilities in a single run, thereby reducing the evolutionary computational cost dramatically. Moreover, the performance of our proposed Pareto EMO algorithm was found to be comparable against a current state-of-the-art Pareto EMO algorithm, the NSGA-II algorithm, for evolving legged locomotion controllers.

1 Introduction

The automatic synthesis of embodied and situated creatures through artificial evolution has become a key area of research in robotics [Lipson and Pollack, 2000], artificial life [Taylor and Massey, 2001] and the cognitive sciences [Nolfi and Floreano, 2002]. This concept stresses the importance of studying systems that have a body and are situated in a physical environment. It also emphasizes the utilization of artificial evolution as the primary mechanism for driving the self-organization process. This approach enables artificial creatures to autonomously develop intelligent behavior through the dynamic interactions between its body, nervous system and environment.

However, research into evolving artificial creatures have focused mainly on generating the desired behavior using single-objective fitness functions. These evaluation functions typically consist only of a single term for assigning the fitness of individuals generated [Sims, 1994, Lipson and Pollack, 2000, Bongard and Pfeifer, 2002] or a combination of multiple terms into a single weighted objective when the desired behavior cannot be achieved with simpler single-termed functions [Floreano and Mondada, 1998, Hornby and Pollack, 2001, Reil and Husbands, 2002]. It is highly surprising that a true multi-objective optimization approach involving optimization of explicitly distinct objectives has remained largely unexplored thus far for artificial creature evolution. Such explorations may very well reveal significant advantages over standard single-objective EAs in terms of the evolutionary optimization process itself in addition to the possibility of generating greater varieties of creature morphologies and behaviors.

2 The Pareto EMO Approach

EMO combines the fields of evolutionary computation with multiple criteria decision-making for solving multi-objective optimization problems [Deb, 2001, Coello Coello et al., 2002]. A multi-objective optimization problem gives rise to a number of optimal solutions, known as Pareto optimal solutions, of which none can be said to be better than the others with respect to all objectives. EAs are particularly suited for tackling multi-objective optimization problems by virtue of their population-based nature that allows for the generation of multiple solutions of the Pareto set within a single run [Deb, 2001, Coello Coello et al., 2002].

The use of EMO for the evolution of embodied artificial creatures allows for the consideration of different objectives simultaneously. Although there have been some studies that appear to have multi-objectivity present in the evolutionary system, such as predator-prey simulations [Nolfi and Floreano, 2000], coevolution [Hornby and Pollack, 2001] and evolution by physical competition [Sims, 1994], they do not explicitly impose the evolutionary search on distinctly different optimization criteria. Consequently, the resulting artificial creatures cannot exhibit clear trade-offs in terms of their different evolutionary goals.

An EMO approach has been previously used in a robotics design problem although the experiment involved only a non-autonomous subject in the form of an attached robotic manipulator arm [Coello Coello et al., 1998]. EMO has also been used for the design of autonomous robots [Leger, 1999] although the focus was for optimizing the physical configurations of modular robotic components rather than for the generation of autonomous robotic controllers. The use of EMO has also been reported for solving navigational problems in simulated 2D mobile agents [Gacogne, 1999, Kim and Hallam, 2002].

We have previously demonstrated the use of EMO for evolving locomotion controllers in fully embodied and situated creatures that minimized the hidden layer size of the controller and maximized locomotion capability achieved [Teo and Abbass, 2002] as well as for characterizing the complexity of artificially evolved creatures [Teo et al., 2003]. Our experiments were aimed at generating legged locomotion in 3 dimensions rather than wheeled or mobile locomotion behaviors that are restricted to 2 dimensions. In this paper, our main aim is to compare the trade-off between using a self-adaptive Pareto EMO approach against more conventional EA methodologies in terms of solution quality and computational cost. The objectives of these comparisons are firstly to elucidate the effectiveness of using these conventional algorithms for generating high quality locomotion controllers and secondly whether the advantages of the self-adaptive Pareto approach are truly beneficial against these more common methods of evolutionary optimization. Finally, a comparison of our proposed algorithm is made against NSGA-II [Deb et al., 2002], a well-known and state-of-the-art hand-tuned Pareto EMO algorithm.

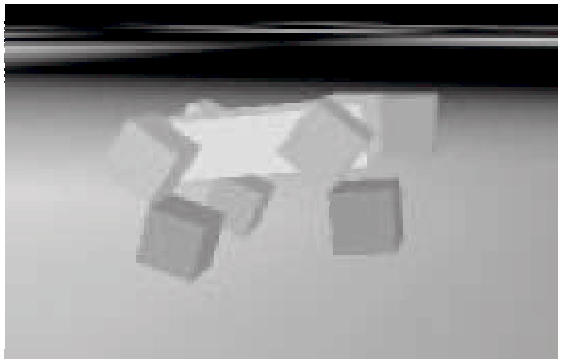

Figure 1: Screen dump of the simulated quadruped.

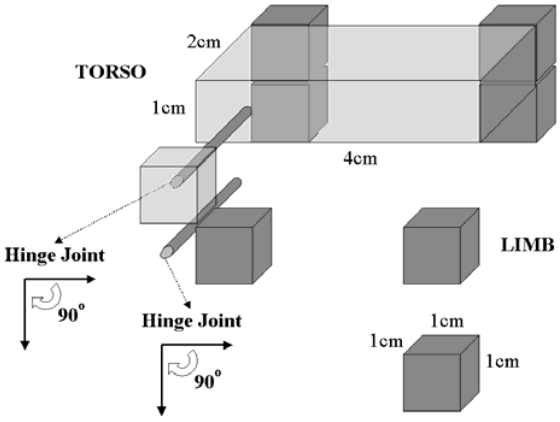

Figure 2: Geometric description of the creature.

3 The Virtual World

3.1 Physics Engine

The Vortex physics engine [CM Labs, 2002] was employed to generate the physically realistic artificial creature and its simulation environment. Vortex is a commercial-off-the-shelf (COTS) simulation toolkit which consists of a set of C++ routines for robust rigid-body dynamics, collision detection, contact creation, and collision response. However, as Vortex is a constraint-based simulation, it naturally suffers from increasingly higher computational requirements as the number of objects being simulated in the world increases. As such, the design of the artificial creature and its world are kept relatively simple in order to maintain a reasonable run time, especially when conducting the evolutionary experiments. The experiments were carried out using a Pentium IV 2.4GHz PC with 256MB RAM. A single typical evolutionary run would take on average 4 hours to complete using this hardware configuration.

3.2 Creature Morphology

The artificial creature is a basic quadruped with 4 short legs. Each leg consists of an upper limb connected to a lower limb via a hinge (one degree-of-freedom) joint and is in turn connected to the torso via another hinge joint. The hinge joints are allowed to rotate between 0 to 1.57 radians. Each of the hinge joints is actuated by a motor that generates a torque producing rotation of the connected body parts about that hinge joint. The mass of the torso is 1g and each of the limbs is 0.5g. The torso has dimensions of 4 x 1 x 2cm and each of the limbs has dimensions of 1 x 1 x 1cm. The artificial creature has 12 sensors and 8 actuators. The 12 sensors consist of 8 joint angle sensors corresponding to each of the hinge joints and 4 touch sensors corresponding to each of the 4 lower limbs of each leg. The joint angle sensors return continuous values in radians whereas the touch sensors return discrete values, 0 if no contact and 1 if contact is made.

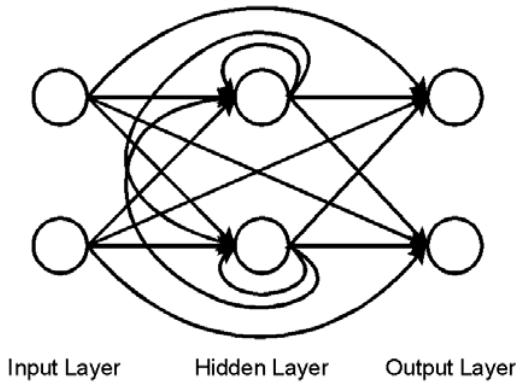

Figure 3: Artificial neural network architecture.

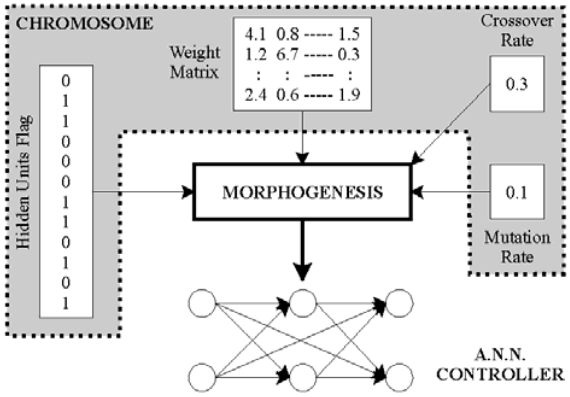

Figure 4: Morphogenesis of genotype into ANN controller.

The 8 actuators represent the motors that control each of the 8 articulated joints of the creature. These motors are controlled via outputs generated from the ANN controller which are then used to set the desired target velocity of rotation of the connected body parts about that joint. The accelerations of the hinge motors required to reach the target angular velocities are automatically calculated by Vortex.

3.3 Controller Architecture

The ANN architecture used in this study is a fully-connected feed-forward network with recurrent connections on the hidden units as well as direct input-output connections. Recurrent connections were included to allow the creature's controller to learn time-dependent dynamics of the system. Direct input-output connections were also included in the controller's architecture to allow for direct sensor-motor mappings to evolve that do not require hidden layer transformations. Bias is incorporated in the calculation of the activation of the hidden as well as output layers.

3.4 Genotype Representation

Let I be the number of inputs, H the number of hidden units, and O the number of outputs. The genotype is a class with an $(I + H) \times (H + O)$ dimension matrix Ω, an H dimension vector ρ, δ as the crossover rate and η as the mutation rate. $\omega_{ij} \in \Omega$ is the weight connecting unit i with unit j, where $i = 0, \ldots, (I - 1)$ is the input unit i, $i = I, \ldots, (I + H - 1)$ is the hidden unit $(i - I)$, $j = 0, \ldots, (H - 1)$ is the hidden unit j, and $j = H, \ldots, (H+O-1)$ is the output unit $(j-H)$. $\rho_h \in \rho$ is a binary value used to indicate the existence of hidden unit h. $\sum_{h=0}^{H} \rho_h$, represents the actual number of hidden units in a network. ρ allows a hidden node to evolve even if it is not active during a generation.

4 Experimental Setup

First, we outline our proposed self-adaptive Pareto EMO algorithm. Then we explain each of the other four EAs. The evolutionary parameters were set as follows in all experiments: 1000 generations, 30 individuals, maximum of 15 hidden units, 500 timesteps and 10 repeated runs for each setup. The objectives were to (1) maximize the locomotion distance achieved, and (2) minimize usage of hidden units in the ANN controller.

Algorithm 1: SPANN – A Self-Adaptive Pareto EMO Algorithm. Recently, the SPDE algorithm [Abbass, 2002] was combined with the MPANN algorithm [Abbass, 2001] for evolving artificial neural networks called the *Self-adaptive Pareto Artificial Neural Network* (SPANN) algorithm, implementation details of which can be found in [Abbass, 2003]. The evolutionary mechanism of SPANN is based on differential evolution [Storn and Price, 2995] but self-adapts the crossover and mutation rates. In this paper, we propose a modified version of SPANN for controller evolution. There are two major differences between this proposed version and the original version of SPANN. Firstly, the original version uses back-propagation whereas this modified version purely uses evolutionary adaptation. Secondly, the repair function used in the original algorithm for evolving the crossover and mutation rates (which truncates the whole number portion leaving only the decimal portion), though useful for the data mining task, was found to cause premature convergence of these rates to the lower boundary of 0 when evolving controllers. Consequently, the evolutionary optimization process would also prematurely stagnate due to the lack of crossover and mutation during reproduction. The new repair function, which adds (if rates < 0) or substracts (if rates > 0) a random number between 0 and 1, is proposed in this modified version.

Algorithm 2: A Self-Adaptive Weighted Sum EMO Algorithm. Here, we used an EMO algorithm with a single-objective that combined the two objectives using a weighted sum. Apart from the change to the manner in which the objectives are evaluated, the weighted sum EMO algorithm is otherwise similar to SPANN in all other respects. 10 different values were used for the relative weight which was controlled using the γ parameter. For example, when $\gamma = 10\%$, the weight put on maximizing locomotion is set to 10% while the weight put on minimizing the number of hidden units is set to 90%. A $(\lambda + \mu)$ strategy is used where the 15 best individuals of the population are carried over to the next generation intact.

Algorithm 3: A Self-Adaptive Single-Objective EA. Next, we used a conventional EA which optimizes only one objective of maximizing the locomotion distance achieved by the ANN controller while keeping the hidden layer size fixed. As in the weighted sum EMO algorithm, the $(\lambda + \mu)$ strategy is used in this single-objective EA. Sixteen separate sets of evolutionary runs were conducted corresponding to each one of the different number of hidden units ranging from 0 to 15, which is the range allowed in the multi-objective runs.

Algorithm 4: A Hand-Tuned Pareto EMO Algorithm. In this set of experiments, an EMO algorithm identical to SPANN is used but with the exception that the crossover and mutation rates are user-defined rather than self-adaptive. Apart from the non-self-adapting crossover and mutation rates, the hand-tuned EMO algorithm is otherwise similar to SPANN in all other respects. The following crossover and mutation rate combinations were used respectively — Setup 1: 50%, 50%; Setup 2: 50%, 90%; Setup 3: 90%, 50%.

Algorithm 5: NSGA-II – A Hand-Tuned Pareto EMO Algorithm. NSGA-II requires a number of parameters to be set by the user including the non-self-adaptive crossover and mutation rates. Recently, the authors of NSGA-II conducted a comprehensive comparative study of NSGA-II against other EMO algorithms [Deb et al., 2002]. Hence, in the first setup, these user-defined parameters were set according to those used in the above-mentioned comparative study as follows: crossover rate 90%, mutation rate for real-coded variables 0.1553% (representing the reciprocal of the number of real-coded variables), and mutation rate for binary-coded variables 6.6667% (representing the reciprocal of the number of binary-coded variables), distribution index for crossover operator 20, distribution index for mutation operator 20, and single-point crossover. As with the setup used in the hand-tuned Pareto EMO experiments, the following crossover and mutation rate combinations were used respectively — Setup 2: 50%, 50%; Setup 3: 50%, 90%; Setup 4: 90%, 50%.

5 Results and Discussion

The results obtained from SPANN are first contrasted against the weighted sum EMO algorithm, the single-objective EA and the hand-tuned EMO algorithm. A comparison of the trade-off between quality of solutions obtained and the associated computational cost involved with these evolutionary runs is given. Finally, the results of SPANN is compared against NSGA-II.

5.1 SPANN Against a Weighted Sum EMO Algorithm

Algorithm	Average Best Locomotion Distance $\pm$ Standard Deviation	t-value (against SPANN)	No. of Hidden Units
SPANN	13.9626 ± 1.7033	-	4.9 ± 2.6
γ=10%	9.8571 ± 1.4277	(5.97)	0.0 ± 0.0
γ=20%	10.4613 ± 2.6883	(3.19)	0.1 ± 0.3
γ=30%	8.4306 ± 1.3288	(7.96)	0.1 ± 0.3
γ=40%	9.4011 ± 2.1017	(4.46)	0.5 ± 0.8
γ=50%	11.3924 ± 3.0330	(2.15)	0.8 ± 0.9
γ=60%	12.1794 ± 2.9865	(1.86)	1.6 ± 1.0
γ=70%	13.7448 ± 2.4376	(0.33)	3.3 ± 1.9
γ=80%	14.0521 ± 2.3034	0.09	4.1 ± 1.5
γ=90%	15.1119 ± 1.9977	1.76	5.6 ± 1.9
γ=100%	15.2829 ± 3.6578	1.04	8.1 ± 1.5

Table 1: Comparison of best locomotion distance for Pareto/best solutions obtained over 10 independent runs with SPANN and the weighted sum EMO algorithm which utilizes the γ parameter.

In Table 1, we compare the weighted sum EMO against the Pareto SPANN algorithm. Results comparable to those obtained using the SPANN algorithm are achieved only with $\gamma = 70\%$. Although slightly higher locomotion distances were achieved using higher values of γ, which places more emphasis on the locomotion component of the weighted objective function, in all cases the standard deviation of the solutions were higher for the average best fitness for locomotion distance. Also, the case where $\gamma = 100\%$, which does not put any pressure whatsoever towards optimizing the size of the hidden layer, results in a very high average of hidden units used in the evolved controllers. This suggests that a significant amount of redundancy may be present in these networks, given that a t-test showed none of these weighted sum solutions were significantly better than those obtained with SPANN at both the $\alpha = 0.05$ and $\alpha = 0.01$ significance levels. Conversely, three of the weight combinations resulted in solutions significantly worse than SPANN at the $\alpha = 0.01$ significance level ($\gamma = 10\%, 30\%, 40\%$) and one weight combination worse than SPANN at the $\alpha = 0.05$ significance level ($\gamma = 20\%$). Therefore, obtaining good solutions when using a weighted sum method critically de-

pends on the choice of weights used on the respective objective functions and to find this right combination of weights would require multiple evolutionary runs to be conducted. Hence, the Pareto approach adopted in our SPANN algorithm is preferable from a computational cost point of view over a weighted sum method since it is able to proceed with the evolutionary optimization process without any tuning of weights and is still able to produce highly competitive results.

5.2 SPANN Against a Single-Objective EA

Algorithm	No. of Hidden Units	Average Best Locomotion Distance $\pm$ Standard Deviation	t-value (against SPANN)
SPANN	4.9 ± 2.6	13.9626 ± 1.7033	-
SO-EA	0	15.7516 ± 2.9721	1.53
SO-EA	1	15.1441 ± 2.0260	1.33
SO-EA	2	16.3236 ± 2.7242	2.43
SO-EA	3	15.1532 ± 3.1696	1.05
SO-EA	4	15.2088 ± 2.2106	1.61
SO-EA	5	15.1562 ± 2.8741	1.32
SO-EA	6	16.0317 ± 2.0719	2.86
SO-EA	7	15.8033 ± 1.6159	2.07
SO-EA	8	17.4358 ± 3.2508	2.89
SO-EA	9	15.7375 ± 2.6430	1.53
SO-EA	10	16.1514 ± 2.1318	2.49
SO-EA	11	15.0614 ± 3.5612	0.86
SO-EA	12	15.4287 ± 3.0020	1.41
SO-EA	13	15.0359 ± 1.8909	1.19
SO-EA	14	16.6273 ± 2.8095	2.70
SO-EA	15	15.6150 ± 2.4605	2.58

Table 2: Comparison of best locomotion distance for Pareto/best solutions obtained over 10 independent runs with SPANN and single-objective EA (SO-EA). Number of hidden units is fixed in the single-objective EA.

In Table 2, we compare the single-objective EA against the multi-objective SPANN algorithm. In all the single-objective runs, higher locomotion distances were achieved by the evolved controllers in terms of the mean of the best solutions compared to SPANN. This is expected since all the evolutionary optimization pressure is focused only on the one objective of maximizing locomotion distance whereas this pressure is halved in the EMO case, where it is being shared with the objective of minimizing the hidden layer size. However, none of these results were significantly different at the $\alpha = 0.01$ significance level compared to SPANN while only six out of the sixteen different setups were significantly different at the $\alpha = 0.05$ significance level (number of hidden units = 2, 6, 8, 10, 14 & 15). However, it should be noted that the standard deviations in 15 out of the 16 different setups in the single-objective EA were higher than SPANN which suggests that even though the search space is much larger in the EMO case, the SPANN algorithm is still more stable in terms of its optimization results. It should also be remembered that 150 more evolutionary runs (15 setups $\times$ 10 repeats) were required in the single-objective case simply to investigate the effects of the hidden layer size on the evolution of these controllers. This would be a serious limitation for such investigations if the different number of setups required increases in magnitude (for example, consider the case where 100 or 1000 hidden units are allowed) or if the additional factors to be investigated are not discrete in nature (for example variations in the morphological parameters of the artificial creature). Moreover, in obtaining these better locomotion capabilities, there is a significant trade-off since the overall computational costs in terms of evaluating the ANN during evolution is much higher for the single-objective EA compared to SPANN (see Table 4).

5.3 SPANN Against a Hand-Tuned EMO Algorithm

Algorithm	Average Best Locomotion Distance $\pm$ Standard Deviation	t-value (against SPANN)	No. of Hidden Units
SPANN	13.9626 ± 1.7033	-	4.9 ± 2.6
c=50%,**m**=50%	15.3819 ± 2.3195	1.39	6.5 ± 1.7
c=50%,**m**=90%	13.1881 ± 1.4715	(1.03)	7.7 ± 2.5
c=90%,**m**=50%	13.1978 ± 1.6447	(1.42)	6.1 ± 1.4

Table 3: Comparison of best locomotion distance for Pareto solutions obtained over 10 independent runs with SPANN and the hand-tuned EMO algorithm which requires the user to set the crossover rate **c** and the mutation rate **m**.

As shown in Table 3, the use of hand-tuned crossover and mutation rates did not provide any significant advantage over the SPANN algorithm in terms of the average best locomotion distance achieved by the evolved controllers. A t-test showed no significant differences at the $\alpha = 0.05$ and $\alpha = 0.01$ significance levels. One combination of the hand-tuned EMO algorithm gave a marginally better result over the 10 runs in terms of the average best solution obtained while two other combinations performed worse than the SPANN algorithm. In terms of the number of hidden units used, the three combinations of the hand-tuned EMO algorithm used an average of 1.8 nodes more than SPANN. As such, the self-adaptive SPANN algorithm is beneficial compared to a hand-tuned EMO algorithm in that it reduces the computational runs required while still being able to maintain the same quality of solutions generated both in terms of the locomotion distance achieved as well as in the number of hidden units required in the controller.

Algorithm	Best Locomotion Distance	$+/-\%$ of SPANN	Total Computational Cost	$+/-\%$ of SPANN
SPANN	17.6994	-	909,520,500	-
Weighted Sum EMO	21.8228	$+23.3\%$	3,073,867,500	$+238.0\%$
Single-Objective EA	22.4069	$+26.5\%$	1,441,441,000,000	$+15748.4\%$
Hand-Tuned EMO	19.5051	$+15.9\%$	2,509,938,000	$+176.0\%$

Table 4: Comparison of overall best locomotion controller obtained and corresponding computational cost using SPANN against the weighted sum EMO algorithm, the single-objective EA and the hand-tuned EMO algorithm.

5.4 Trading-Off Solution Quality Against Computational Cost

Table 4 compares the overall best solution found by the different algorithms against the overall computation cost involved in discovering these solutions. The computational cost is estimated using the total number of hidden unit activations registered during the search process for each algorithm. This is a reasonable estimate since most of the computational time involved in conducting these experiments is spent on the evaluation of different ANN controllers by way of physically simulating the creature as guided by each newly generated controller within the Vortex physics-based world (see Section 3.1). Therefore, the computational cost (C) will differ between different algorithms as a function of the number of hidden unit activations required to evaluate the fitness of each newly generated genotype (A), the number of new genotypes generated per evolutionary run (G) and the number of evolutionary runs per algorithm (R), as described by the following equation:

$$C = A \times G \times R \tag{1}$$

The best overall solution in terms of locomotion distance was obtained using the single-objective EA where the improvement over SPANN was 26.5%. However, the corresponding computational cost was a staggering 15,748% more than SPANN. This was mainly due to the number of repeated runs required for each different hidden layer size as well as the high usage of hidden units in the runs involving larger hidden layer sizes, which cannot be changed within each evolutionary run. Again, although better locomotion capabilities were obtained using the hand-tuned EMO and weighted sum EMO, dramatically higher computational costs were also associated with the use of the hand-tuned (176%) as well as the weighted sum methods (238%) compared to SPANN. Again these increases can be attributed to the need for repeated evolutionary runs required to test different weight assignments and crossover/mutation rates respectively in these algorithms. However, the inclusion of another objective in these two algorithms, which allowed for the minimization of the hidden layer size, did reduce the computational cost significantly compared to the single-objective EA.

In summary, although better controllers were evolved for locomotion distance using the single-objective EA, the corresponding trade-off in terms of overall computational cost was dramatically and unfavorably large. The trade-off between obtaining better locomotion capabilities and computational cost was again significantly and unfavorably large using the hand-tuned and weighted sum EMO algorithms. Hence, the SPANN algorithm has been shown to provide reasonably good results in terms of evolving locomotion controllers while at the same time providing the lowest overall computational cost compared to three other evolutionary methodologies investigated.

5.5 SPANN Against NSGA-II

Algorithm	Overall Best Locomotion Distance	Average Best Locomotion Distance $\pm$ Standard Deviation	t-statistic (against SPANN)
SPANN	17.6994	13.9626 ± 1.7033	-
Setup 1	15.5452	11.7421 ± 2.0497	(3.78)
Setup 2	18.3941	16.2022 ± 1.5860	2.85
Setup 3	20.4144	17.8635 ± 1.9744	4.54
Setup 4	20.9806	16.2667 ± 2.1868	2.54

Table 5: Comparison of best locomotion distance for Pareto solutions obtained over 10 independent runs with SPANN and NSGA-II which utilizes setups 1 through 4.

Table 5 lists the best Pareto solutions for locomotion distance obtained using the NSGA-II algorithm and compares them against those obtained using the SPANN algorithm. The best solutions obtained using the first setup for NSGA-II produced controllers that used no hidden units in all 10 runs. The overall best locomotion distance achieved was lower than that obtained using SPANN. The very small mutation rate used in this setup most probably caused the evolutionary search to prematurely converge to local optima

centered around controllers which did not use any hidden units. A t-test showed that the results obtained using NSGA-II were significantly worse than SPANN at the $\alpha = 0.01$ significance level for this setup. Also, the overall best controller from SPANN achieved over 2 units of distance more than the overall best controller obtained from this setup of NSGA-II (representing a decrease of 12.2% compared to the overall best locomotion distance achieved by SPANN).

To overcome the inferior results obtained using the setup reported in [Deb et al., 2002], a second experiment utilizing the best combination of crossover and mutation rates obtained from the hand-tuned EMO was conducted. This second setup used crossover and mutation rates of 50% with all other parameters unchanged. Much better results were obtained in this second setup, where the overall best controller in terms of locomotion achieved a higher distance than that obtained using SPANN by just under 0.7 units (representing a 3.9% improvement over the best locomotion distance achieved by SPANN). A t-test showed that the solutions obtained using NSGA-II with the second setup were significantly better than those obtained using SPANN at the $\alpha = 0.05$ significance level.

A third experiment using a setup with an even higher mutation rate of 90% while maintaining the crossover rate at 50% was conducted. However, a t-test comparing the results from this third setup against the second setup for NSGA-II showed no significant improvements. A fourth experiment was conducted using a setup with crossover rate of 90% and maintaining the mutation rate at 50%. Again, a t-test showed no significant improvements in the results obtained with the fourth setup compared to the second setup of NSGA-II. The solutions obtained with third and fourth setup for NSGA-II were significantly better than those obtained with SPANN at the $\alpha = 0.01$ and $\alpha = 0.05$ levels respectively. The best solutions obtained from the third setup of NSGA-II used an average of 8.4 hidden units, which is almost double the number used by the best solutions obtained using SPANN, while the fourth setup used an average of 7.7 hidden units.

Figure 5 plots the global Pareto-front for all the different algorithms used in evolving ANN controllers for the artificial creature. It can be seen that the Pareto-front generated through 10 runs of SPANN is comparable though dominated by the Pareto-front generated through 40 runs of NSGA-II (10 runs each in Setup 1–4). The solution with 0 hidden units of the NSGA-II global Pareto-front was contributed by the first setup of NSGA-II while the remaining 8 other solutions on the global Pareto-front were contributed by the other three setups. It is clear that the performance of NSGA-II is also sensitive to the parameters used. Although the Pareto-front of SPANN was not as optimal as the other algorithms, the controller with the highest locomotion distance discovered by each algorithm had the smallest hidden layer size for SPANN (4 nodes) compared to NSGA-II (9 nodes), the hand-tuned EMO algorithm (9 nodes), the weighted sum EMO algorithm (7 nodes) and the single-objective EA (14 nodes) but more importantly, the computational cost incurred was much higher in all of the other evolutionary algorithms compared to the SPANN algorithm in finding the right parameter settings to generate these more optimal Pareto fronts.

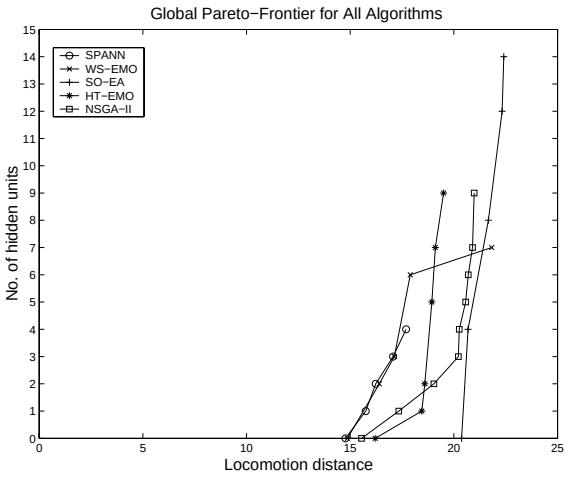

Figure 5: Global Pareto-front of controllers obtained using SPANN, weighted sum (WS-EMO), single-objective (SO-EA), hand-tuned (HT-EMO) and NSGA-II algorithms. X-axis: Locomotion distance, Y-axis: No. of hidden units.

6 Conclusion

The self-adaptive Pareto SPANN algorithm was compared against EMO algorithms that utilized hand-tuning of crossover/mutation rates and weighted sum approaches as well as a single-objective EA. It was found that SPANN discovered reasonably good quality controllers but most importantly required significantly less overall computational costs. Although better solutions were found using the single-objective EA, the weighted sum and hand-tuned EMO algorithms, the trade-off in terms of computational costs was extremely high in comparison to SPANN. The performance of SPANN was also found to be comparable to that of a current state-of-the-art benchmark Pareto EMO algorithm, NSGA-II. Therefore, the self-adaptive Pareto SPANN algorithm has been shown to be a highly beneficial EMO algorithm to use for evolving artificial creature controllers compared to other more conventional evolutionary optimization algorithms. As future work, it would be interesting to implement a self-adaptive version of NSGA-II for a direct comparison against the self-adaptive SPANN.

Bibliography

[Abbass, 2001] Abbass, H. A. (2001). A memetic Pareto evolutionary approach to artificial neural networks. *14th International Joint Conference on Artificial Intelligence*, LNAI 2256, pp. 1–12. Springer-Verlag, Berlin.

[Abbass, 2002] Abbass, H. A. (2002). The self-adaptive Pareto differential evolution algorithm. *2002 Congress on Evolutionary Computation*, pp. 831–836. IEEE Press, Piscataway, NJ.

[Abbass, 2003] Abbass, H. A. (2003). Speeding up back-propagation using multiobjective evolutionary algorithms. Accepted to appear in *Neural Computation*.

[Bongard and Pfeifer, 2002] Bongard, J. C. and Pfeifer, R. (2002). A method for isolating morphological effects on evolved behavior. *Seventh International Conference on the Simulation of Adaptive Behavior*, pp. 305–311. MIT Press, Cambridge, MA.

[CM Labs, 2002] CM Labs (2002). Vortex Toolkit [online]. http://www.cm-labs.com.

[Coello Coello et al., 1998] Coello Coello, C. A., Christiansen, A. D., and Aguirre, A. H. (1998). Using a new GA-based multiobjective optimization technique for the design of robot arms. *Robotica*, 16:401–414.

[Coello Coello et al., 2002] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York.

[Deb, 2001] Deb, K. (2001). *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chicester, UK.

[Deb et al., 2002] Deb, K., Pratab, A., Agrawal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Floreano and Mondada, 1998] Floreano, D. and Mondada, F. (1998). Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11:1461–1478.

[Gacogne, 1999] Gacogne, L. (1999). Multiple objective optimization of fuzzy rules for obstacles avoiding by an evolution algorithm with adaptative operators. *Fifth International Mendel Conference on Soft Computing*, pp. 236–242, Brno, Czech Republic.

[Hornby and Pollack, 2001] Hornby, G. S. and Pollack, J. B. (2001). Body-brain coevolution using L-systems as a generative encoding. *2001 Genetic and Evolutionary Computation Conference*, pp. 868–875. Morgan Kaufmann, San Francisco.

[Kim and Hallam, 2002] Kim, D.-E. and Hallam, J. (2002). An evolutionary approach to quantify internal states needed for the Woods problem. *Seventh International Conference on the Simulation of Adaptive Behavior*, pp. 312–322. MIT Press, Cambridge, MA.

[Leger, 1999] Leger, P. C. (1999). *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach*. PhD thesis, Carnegie Mellon University.

[Lipson and Pollack, 2000] Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978.

[Nolfi and Floreano, 2000] Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

[Nolfi and Floreano, 2002] Nolfi, S. and Floreano, D. (2002). Synthesis of autonomous robots through evolution. *Trends in Cognitive Science*, 6(1):31–36.

[Reil and Husbands, 2002] Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168.

[Sims, 1994] Sims, K. (1994). Evolving virtual creatures. *21st Annual Conference on Computer Graphics and Interactive Techniques*, pp. 15–22. ACM Press, New York.

[Storn and Price, 2995] Storn, R. and Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, 1995.

[Taylor and Massey, 2001] Taylor, T. and Massey, C. (2001). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial Life*, 7(1):77–87.

[Teo and Abbass, 2002] Teo, J. and Abbass, H. A. (2002). Multi-objectivity for brain-behavior evolution of a physically-embodied organism. *8th International Conference on Artificial Life*, pp. 312–318. MIT Press, Cambridge, MA.

[Teo et al., 2003] Teo, J., Nguyen, M. H., and Abbass, H. A. (2003). Multi-objectivity as a tool for constructing hierarchical complexity. *2003 Genetic and Evolutionary Computation Conference*, LNCS 2723, pp. 483–494. Springer-Verlag, Berlin.