2013 IEEE 6th International Workshop on Computational Intelligence and Applications
July 13, 2013, Hiroshima, Japan

A (μ, λ) Evolutionary and Particle Swarm Hybrid Algorithm, with an Application to Dinosaur Gait Optimization

Yoshiyuki Matsumura, Ayumu Kobayashi, Kiyotaka Sugiyama, Todd Pataky
Shinshu University
3-15-1, Tokida, Ueda, Nagano 386-8567, JAPAN
matsumu@shinshu-u.ac.jp

Bill Sellers
The University of Manchester
D1239 Michael Smith Building, Oxford Road,
Manchester, M13 9PT, UK

Toshiyuki Yasuda and Kazuhiro Ohkura
Hiroshima Universisty
Kagamiyama, Higashihiroshima, Hiroshima 739-8527, JAPAN

***Abstract*—A hybrid evolutionary algorithm based on (μ, λ) evolutionary algorithms and particle swarm optimization is proposed for the numerical optimization problems. In order to find out the performance of the hybrid, the computer experiment is tested on dinosaur's gait generation problem. Experimental results show that hybrid optimization finds maximum fitness and is faster in the first phase.**

***Index Terms*—A hybrid evolutionary algorithm, (μ, λ) evolutionary algorithms and particle swarm optimization, dinosaur's gait generation problem.**

I. Introduction

Evolution Strategies (ESs) [1] are a class of optimization algorithms which were inspired by the natural evolution theory. There have been several variants of ESs, such as $(1 + 1)$-ES, $(\mu + 1)$-ES, $(\mu + \lambda)$-ES, (μ, λ)-ES, and so forth. In this paper, we adopt (μ, λ)-ES for implementation. Mutation is the primary evolutionary operator in (μ, λ)-ES, where $\lambda > \mu \geq 1$. (μ, λ) means μ parents generate λ offspring through mutation for each generation. The best μ offspring are selected and enter the next generation. Classical (μ, λ)-ES (CES) uses Gaussian mutation. Yao and Liu proposed a Cauchy mutation in (μ, λ)-ES, called Fast Evolution Strategies (FES) [19]. Later, in order to address the problem of lower bound of the mutation factor, Robust Evolution Strategies (RES) were developed [14]. ES and variants are still being studied and applied in various areas [12].

In recent years, swarm intelligence is drawing more attentions in different research areas, especially in numerical optimization, multi-agent systems, and so on. Unlike natural evolution based algorithms, swarm intelligence makes use of information of the whole swarm and search through the information sharing [3]. Although no centralized control dictating the individuals, information sharing often cause a global pattern to emerge. Particle swarm optimization (PSO) [11] is one of the most well-known swarm intelligence algorithms in optimization research. It imitates the behavior of flying bird flock. In particle swarm optimization, individuals are called particles, which are moving in the search space, bearing locations and velocities. A global best location is shared by all the particles and every particle itself bears a local best location. Particle updates velocity according to the distance from these two locations. Particle swarm optimization is now applied widely in numerous optimization problems [6], [10].

Basically, natural evolutionary and swarm intelligence are two systems with different methodologies. If these two different systems hybrid together, what would happen? Would the hybrid system show some interesting properties? Hsieh et al. [7] developed a particle swarm guided evolution strategies. In their method, ES adopted a new mutation operator, called guided mutation, which is inspired by the particle swarm optimization. The experiments showed some good results. Mo et al. [13] introduced a new hybrid algorithm called particle swarm assisted incremental evolution strategies. In their study, search space is sliced by cutting planes and hyperplanes along different dimensions. PSO is used to globally adjust the planes, and ES is used to locally search the optima in those planes.

In our papers [4][5], we proposed a different hybrid method based on (μ, λ)-ES and particle swarm intelligence. ES is strong in generating new search individuals around the current ones, then we can consider it as a local search [8]. On the other hand, PSO focuses on the global information, and we can consider it as a global search [11]. In a generation, the best ones need some exploit, and the worst ones need some exploring. It is very essential to find a good balance between exploit and exploring. Since ES concentrates on local search, we can use it to guide the best ones doing exploit. PSO concentrates on global search, then we can use it to lead the worst ones to explore. Based on this methodology, we proposed our hybrid method. Then, this paper would show the results of (μ, λ) Evolutionary Algorithms and PSO hybrid on the test model of Allosaurus.

The rest of this paper is organized as follows. Section II briefly introduces (μ, λ)-Evolutionary Algorithms, particle swarm optimization and their implementations. In Section III, our hybrid algorithm is described. Numerical Experiments

978-1-4673-5726-5/13/$31.00 ©2013 IEEE

on the test model of Allosaurus are done in Section IV to test the performance of the hybrid algorithm, followed by some discussions comparing the results on the test model of Allosaurus. Finally, conclusions are drawn in Section V.

II. Evolution strategies and particle swarm optimization

In this section, we give a brief background of CES, FES and PSO, which is related to the proposed hybrid algorithm.

For a search space $S \subseteq R^n$ and a function $f : S \rightarrow R$, minimization problem is to find a solution $x_{min} \in S$, $\forall x \in S$, $f(x_{min}) \leq f(x)$.

A. (μ, λ)-ES

Similar to other evolutionary algorithms, ES has recombination, mutation and selection. The classical (μ, λ)-ES is usually implemented as follows [2]. Elitism and stochastic selection are not used.

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

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

where $x_i(j), x'_p(j), \eta_i(j)$ and $\eta'_p(j)$ denote the j-th component of the vectors $\boldsymbol{x}_i, \boldsymbol{x'}_p, \boldsymbol{\eta}_i$ 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 $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$.
4) Calculate the fitness of each offspring $(\boldsymbol{x'}_i, \boldsymbol{\eta'}_i)$, $\forall i \in \{1, \cdots, \lambda\}$, according to $f(\boldsymbol{x'}_i)$.
5) Sort offspring $(\boldsymbol{x'}_i, \boldsymbol{\eta'}_i)$, $\forall i \in \{1, \cdots, \lambda\}$ according to their fitness values, and select the μ best offspring out of λ to be parents of the next generation.
6) If stop condition is not reached, go back to step 3.

Yao and Liu [19] proposed the Fast ES (FES) algorithm 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 convergence trails of the Cauchy mutation operator [20]. In other words the Cauchy distribution has a higher probability than the Gaussian distribution of producing large mutations. Yao and Liu [19] conducted empirical experiments using a number of test functions, demonstrating an improvement in performance especially on multi-modal problems.

B. Particle swarm optimization

Particle swarm optimization is a population-based stochastic optimization algorithm. Particle moves towards two locations, one is the best location found by itself, and the other is the best location found by the whole swarm. At each generation, particle modifies its velocity and locations expecting to find the optimum. The procedure is usually implemented as follows [17].

1) Initialize the locations x_i, velocities v_i, local best locations x_{ibest} and global best location x_{gbest} of the particles.
2) The particles move in the search space according to:

$$v_i(k+1) = w * v_i(k) + c_1 * rand() * (x_{ibest} - x_i(k)) + c_2 * Rand() * (x_{gbest} - x_i(k)) \quad (4)$$
$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (5)$$

where x_{ibest} is the best found ever by i-th particle, x_{gbest} is the best found ever by the whole swarm, c_1,c_2 are positive constants, w is the inertia weight, $rand()$ and $Rand()$ are random functions in the range of $[0, 1]$. If $v_i < -VMax$, set $v_i = -VMax$, if $v_i > VMax$, set $v_i = VMax$.
3) Evaluate the fitness of the particles $f(x_i)$, and update x_{ibest} and x_{gbest}.
4) If stop condition is not reached, go back to step 2.

C. Empirical study on the search step size

In [21], the authors showed the relationship between the search step size and the probability of finding a global optimum. They pointed out that when the global optimum is sufficiently far away from the current search point, i.e., when the distance between the current point and the global optimum is larger than the step size of the search, large search step size is beneficial. In order to identify the differences of the search step size of CES and PSO, an simple empirical study based on benchmark functions is carried out.

1) Benchmark functions: In this paper, we adopted the first 13 high dimensional benchmark functions used in [19] to carry out numerical optimization experiments. Functions f_1 to f_7 are unimodal functions. Function f_6 is the step function. Function f_7 is a noisy quartic function, where *random*$[0, 1)$ is a uniformly distributed random number in the range of $[0, 1)$. Functions f_8 to f_{13} are multimodal functions, which have numerous local minima, thus relatively difficult to optimize.

2) Parameter settings: In our study, for CES, $\mu = 30$, $\lambda = 200$. For PSO, $c_1 = c_2 = 2.0$, $w = 0.8$. $V_{max} = X_{max}$. These parameters are selected based on [2] and [18]. For the convenience of comparing, we set $population = 200$ in PSO. The search stops after 2000 generations.

TABLE I
BENCHMARK FUNCTIONS

Test Function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i})^2$	30	$[-100, 100]^n$	0
$f_4(x) = \max\{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100, 100]^n$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^n$	0
$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^n$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^n$	0
$f_8(x) = \sum_{i=1}^{n} (-x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	$[-500, 500]^n$	-12569.5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10)]$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i) + 20 + e$	30	$[-32, 32]^n$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^n$	0
$f_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1)^2 + \sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin(\pi y_{i+1})^2] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + 0.25(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	30	$[-50, 50]^n$	0
$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1}] + (x_n - 1)[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0

Since the search space in these problems are all real number space, the Euclidean distance is a natural choice to measure the step size. Thus the step size of individual x between generation g and generation $g-1$ can be defined as follows.

$$d(g) = |x(g) - x(g-1)| = \sqrt{\sum_{i=1}^{n} (x_i(g) - x_i(g-1))^2} \quad (6)$$

The average step size of generation g is defined as

$$D(g) = \frac{\sum_{j=1}^{pop} d_j(g)}{pop} \quad (7)$$

3) Results on the search step size: The search stopped when it iterated 2000 generations, and the step sizes were calculated through (6) and (7). After analysis, we found that the algorithms almost converged after 1000 generations, i.e., the step size almost got down to 0. The average step size of the first 1000 generations were also calculated. The results are shown in Table II. The step size in the search process on f_1 (unimodal), f_6 (step function) and f_{13} (multimodal) are also illustrated in Fig. 1, Fig. 2 and Fig. 3.

From Table II and the figures, we can see that both in unimodal or multimodal function, the search step size of PSO are larger than CES. This characteristic helps PSO be more efficient in finding global optimum, especially in the beginning of the search, where PSO can very quickly find a near-optimum

TABLE II
AVERAGE STEP SIZE FOR THE FIRST 1000 GENERATIONS

	f_1	f_2	f_3	f_4	f_5
CES	5.3803	0.9370	8.6882	31.5077	3.2979
PSO	81.3812	8.8756	91.7065	85.0926	25.0895
	f_6	f_7	f_8	f_9	f_{10}
CES	7.6126	0.7838	34.1299	1.5194	3.2102
PSO	89.6132	1.5929	1283.6025	4.8418	25.8612
	f_{11}	f_{12}	f_{13}		
CES	25.9439	4.6539	6.9033		
PSO	472.5660	46.3836	41.5835		

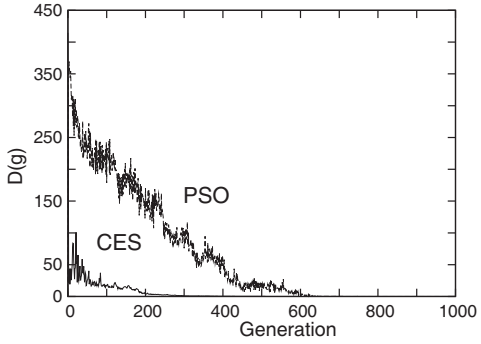

Fig. 1. Search step size on f_1

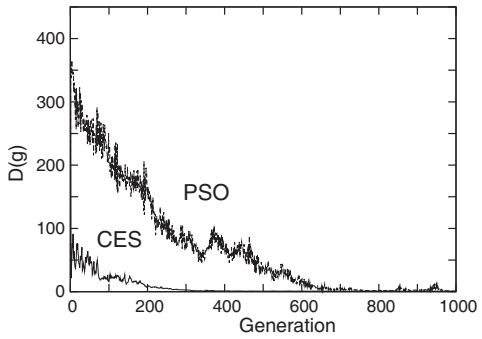

Fig. 2. Search step size on f_6

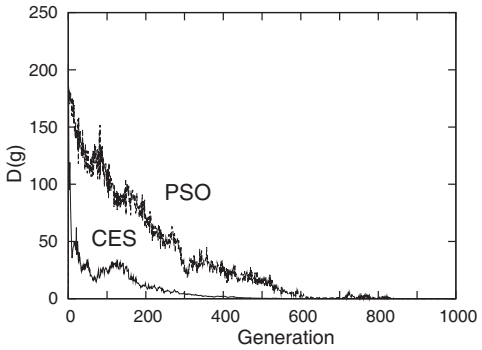

Fig. 3. Search step size in f_{13}

area. This is especially beneficial for the unimodal functions. While for multimodal functions, PSO might fall in some local optima, because the diversity of the population decreases very quickly due to the fast convergence. As for CES, the step size is smaller than in PSO. This is inferior when the individuals are sufficiently far away from the global optimum, but when the individuals come to the very near neighborhood of the global optimum, this becomes beneficial.

From the point view of exploration and exploitation, we can say that PSO is strong at exploring, while CES is strong at exploiting. In order to achieve a balance between exploration and exploitation, we can hybridize Evolutionary Algorithms with PSO. In the next section, hybrid algorithm will be described in detail.

III. Hybrid algorithm

There is a trade-off between exploring and exploit, to which a good optimization algorithm should give enough attention. Based on [8], [11], ES is good at exploit and PSO is powerful in exploring. As for a population of individuals in optimization algorithm, the best ones usually need exploit in the near areas to search for better points, and the worst ones need to explore to escape from the areas. Based on (μ, λ)-Evolutionary Algorithms and particle swarm optimization, our hybrid is implemented as follows.

1) Initialize the first generation, the individuals bear locations x_i, velocities v_i, mutation factor η_i, local best location x_{ibest}, global best location x_{gbest}.
2) Evaluate the fitness of the individuals according to $f(x_i)$, and sort the individuals by fitness.
3) Carry out classic particle swarm update for the last P_{pso} individuals to produce $C_{pso} = P_{pso}$ offspring, and maintain their η_i property.
4) Create $C_{cea} = \lambda - C_{pso} = \lambda - P_{pso}$ offspring out of $P_{cea} = \mu - P_{pso}$ parents. Carry out the (μ, λ)-Evolutionary Algorithms mutation to the offspring. Maintain their v_i.
5) Evaluate the fitness of the offspring, and sort the offspring by fitness. Select the best μ individuals into the next generation. Update the x_{ibest} and x_{gbest} of all these individuals according to their own locations and fitness.
6) If stop condition is not reached, go back to step 3.

Compared with Evolutionary Algorithms and PSO, hybrid algorithm does not increase any on computing complexity.

IV. Dinosaur's Gait Generation Problem

All carnivores chase prey and, in this regard, maximum gait speed may be an important factor to understand dinosaurs' evolutionary success. Since it is impossible to directly test extinct species, various researchers have proposed methods to estimate their gait speeds. One method is to generate a mechanical model of the animal using fossil-based morphological information, and then maximize that model's gait speed.

Maximizing gait speed is a tremendously complex, computational problem, because non-trivial, even-step-like nonlinearities exist in four different model components: (1) Muscle activation — there is a nonlinear relation, including a time delay between neural command intensity and muscle force. (2) Body configuration — changes in body configuration can drastically affect the motion that muscle forces cause, (3) Contact — foot impacts create very large, very brief forces, making dynamic control very difficult in the vicinity of foot contacts, and (4) Overload — muscles have maximum neural intensities, maximum tensions, maximum shortening velocities, maximum stretch, and bones have maximum strains that cannot be exceeded. Parameterized mathematical models govern the behavior of all four components.

However, a gait-speed optimizer can only adjust muscle activation at each instant in time (inputs). It then must attempt to map the highly nonlinear relation between muscle activation and gait speed (output). Due to a large number of local maxima traps, Sellers et. al. [15] proposed various evolutionary approaches to the maximization problem, including use of genetic algorithms (GA), and posted both their model and their best solution to the web [16].

This paper revisits the Allosaurus model of Sellers et al. [15], to determine whether hybrid evolutionary algorithms could outperform the GA approaches proposed by the authors; the Allosaurus species lived during the Upper Jurassic period (approximately 170-147 million years ago). In the Allosaurus model there are 240 parameters governing dynamic muscle activation, and the GaitSym dynamic engine [16] was used to drive model motion. Optimization parameters were set are listed as Table III.

TABLE III
PARAMETERS USED IN COMPUTER EXPERIMENTS

name	value
ω	0.95
c_1, c_2	0.95
evaluation time	< 7500
P_{ea}	3
C_{ea}	20
$P_{pso} = C_{pso}$	20
s	10^{-9}

TABLE IV
RESULTS OF EXPERIMENTS

	Averaged Best Fitness	Standard Deviations	Maximum Fitness
GA	47.2360	0.0229	47.2970
PSO	47.1752	0.0037	47.1838
(1+1)-EA	47.2816	0.0334	47.3604
(3,20)-EA	47.3102	0.0245	47.3630
(9,60)-EA	47.2761	0.0306	47.3534
(15,100)-EA	47.2514	0.0223	47.3052
Hybrid	47.2997	0.0336	47.3777

Table. IV shows the averaged best values, standard deviation and max values in 50 runs. In Table. IV, the best value of maximum fitness is from Hybrid. Fig. 4 shows the averaged best fitness on EAs, PSO, Hybrid and GA achieved from 50 runs. And, in Figure. 4, Hybrid's averaged best fitness rises faster than $(3, 20)$-EAs' at the first phase and $(3, 20)$-EA has higher averaged best fitness than others at 7500 evaluation time.

V. CONCLUSIONS

The hybrid algorithm based on (μ, λ)-EA and particle swarm optimization tested on dinosaur's gait generation problem. In our hybrid algorithm, individuals of each population are divided into two groups by fitness. The first (and better) group is dealt with (μ, λ) mutation, and the second group is dealt with particle swarm optimization. The hybrid algorithm is designed mainly to take the advantage of the balance between exploring and exploit. Experiments were done on the test model of Allosaurus. Experimental results indicate that hybrid algorithm converges faster in the beginning phase and finds maximum best fitness.

ACKNOWLEDGMENT

The authors acknowledge the basic research development achieved by the collaborator of Mr. T. Hamashima and Dr. C. Feng and the computational support in part through the cloud computing system at the Hokkaido university.

REFERENCES

[1] H. Beyer and H. Schwefel, "Evolution strategies: a comprehensive introduction," *Natural Computing*, Volume 1, pp. 3–52, 2002.
[2] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, 1(1), pp. 1–23, 1993.
[3] E. Bonabeau, M. Dorigo and G. Theraulaz, "Swarm Intelligence, From Natural to Artificial Systems," Oxford University Press, 1999.

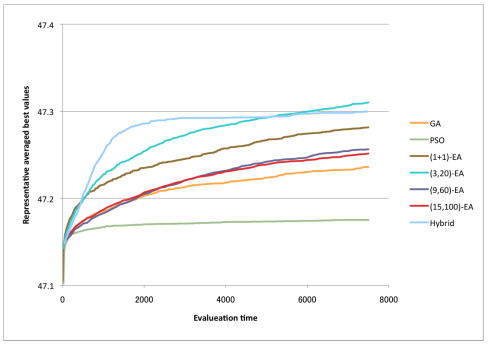

Fig. 4. Optimization efficiencies

[4] C. Feng , Y. Matsumura, T. Hamashima, K. Ohkura and S. Cong, "Hybrid optimization based on (μ, λ)-evolution strategies and particle swarm intelligence, " *Proc. of Joint 4th International Conference on Soft Computing and Intelligent Systems and 8th International Symposium on advanced Intelligent Systems*, pp. 849–854, 2008.
[5] C. Feng , Y. Matsumura, T. Hamashima, K. Ohkura and S. Cong, "Hybrid Optimization based on Fast Evolution Strategies and Particle Swarm Optimization, " *Proc. of the 3rd International Conference on Bioinspired Optimization Methods and their Applications*, pp. 29–39, 2008.
[6] Z-L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Transactions on Energy Conversion*, 19(2), pp. 384–391, 2004.
[7] C. Hsieh, C. Chen and Y. Chen, "Particle swarm guided evolution strategy," *Proc. of the 9th annual conference on genetic and evolutionary computation*, pp. 650–657, 2007.
[8] F. Hoffmeister and T. Bäck, "Genetic algorithms and evolution strategies: similarities and differences," *Lecture Notes in Computer Science*, Springer, pp.455–469, 1991.
[9] J. R. Hutchinson and M. Garcia, "Tyrannosaurus was not a fast runner," Nature, Vol. 415,pp. 1018-1021, 2002.
[10] W. Jatmiko, K. Sekiyama and T. Fukuda, "A PSO-based mobile sensor network for odor source localization in dynamic environment: theory, simulation and measurement," *Proc. of IEEE Congress on Evolutionary Computation*, pp. 3781–3788, 2006.
[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. of IEEE International Conference of Neural Networks.*, pp. 1942–1948, 1995.
[12] E. Mezura-Montes and C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, 9(1), pp. 1–17, 2005.
[13] W. Mo, S. Guan and S. Puthusserypady, "A novel hybrid algorithm for function optimization: particle swarm assisted incremental evolution strategy," *Studies in Computational Intelligence*, Springer Berlin, pp. 101–125, 2007.
[14] K. Ohkura, Y. Matsumura and K. Ueda, "Robust Evolution Strategies," *Applied Intelligence*, 15(3), pp. 153–169, 2001.
[15] W. I. Sellers and P. L. Manning, "Estimating dinosaur maximum running speeds using evolutionary robotics," Proc. R. Soc. B, Vol. 274, pp.2711-2716, 2007.
[16] W. I. Sellers et. al. , "Animal Simulation Laboratory," http://www.animalsimulation.org/
[17] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *Proc. of IEEE World Congress on Computational Intelligence*, pp. 69–73, 1998.
[18] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," *Lecture Notes in Computer Science*,Springer, pp.591–600, 1998.
[19] X. Yao and Y. Liu, "Fast Evolution Strategies," *Control and Cybernetics*, 26(3), pp. 467–496, 1997.
[20] X. Yao, G. Lin and Y. Liu, " An analysis of evolutionary algorithms based on neighborhood and step sizes", *Proc. of the 6th Annual Conference on Evolutionary Programming*, Lecture Notes in Computer Science Vol. 1213, pp.297–307, 1997.
[21] X. Yao, Y. Liu and G. Lin, Evolutionary Programming Made Faster, *IEEE Transactions on Evolutionary Computation*, 3(2), 1999, pp. 82–102.