Fitness Expectation Maximization

Daan Wierstra[1], Tom Schaul[1], Jan Peters[3] and Jürgen Schmidhuber[1,2]

[1] IDSIA, Galleria 2, 6298 Manno-Lugano, Switzerland
[2] TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany
[3] Max Planck Institute for Biological Cybernetics, Tübingen, Germany
{daan, tom, juergen}@idsia.ch, mail@jan-peters.net

Abstract. We present Fitness Expectation Maximization (FEM), a novel method for performing 'black box' function optimization. FEM searches the fitness landscape of an objective function using an instantiation of the well-known Expectation Maximization algorithm, producing search points to match the sample distribution weighted according to higher expected fitness. FEM updates both candidate solution parameters and the search policy, which is represented as a multinormal distribution. Inheriting EM's stability and strong guarantees, the method is both elegant and competitive with some of the best heuristic search methods in the field, and performs well on a number of unimodal and multimodal benchmark tasks. To illustrate the potential practical applications of the approach, we also show experiments on finding the parameters for a controller of the challenging non-Markovian double pole balancing task.

1 Introduction

Real-valued 'black box' function optimization is one of the major topics in modern applied machine learning research (e.g. see [1]). It concerns itself with optimizing the continuous parameters of an unknown (black box) objective fitness function, the exact analytical structure of which is assumed to be unknown or unspecified. Specific function measurements can be performed, however. The goal is to find a reasonably high-fitness candidate solution while keeping the number of function measurements limited. The black box optimization framework is crucial for many real-world domains, since often the precise structure of a problem is either not available to the engineer, or too expensive to model or simulate.

Now, since exhaustively searching the entire space of solution parameters is considered to be infeasible, and since we do not assume we have access to a precise model of our fitness function, we are forced to settle for trying to find a reasonably good solution that satisfies certain pre-specified constraints. This, inevitably, involves using a sufficiently intelligent heuristic approach, since in practice it is important to find the right domain-specific trade-off on issues such as convergence speed, expected quality of the solutions found and the algorithm's sensitivity to local suboptima on the fitness landscape.

A variety of algorithms has been developed within this framework, including methods such as Simulated Annealing [2], Simultaneous Perturbation Stochastic

Approximation [3], the Cross-Entropy method [4, 5], and evolutionary methods such as Covariance Matrix Adaption (CMA) [6] and the class of Estimation of Distribution Algorithms (EDAs) [7].

In this paper, we postulate the similarity and actual equivalence of black box function optimization and one-step reinforcement learning. In our attempt to create a viable optimization technique based on reinforcement learning, we fall back onto a classical goal of reinforcement learning (RL), i.e., we search for a way to reduce the reinforcement learning problem to a supervised learning problem. In order to do so, we re-evaluate the recent result in machine learning, that reinforcement learning can be reduced onto *reward-weighted regression* [8] which is a novel algorithm derived from Dayan & Hinton's [9] expectation maximization (EM) perspective on RL. We show that this approach generalizes from reinforcement learning to fitness maximization to form Fitness Expectation Maximization (FEM), a relatively well-founded instantiation of EDAs which relates to other (EM-inspired) methods for optimization (e.g. see [10, 11]).

This algorithm is tested on a set of unimodal and multimodal benchmark functions, and is shown to exhibit excellent performance on both unimodal and multimodal benchmarks. A defining feature of FEM is its adaptive *search policy*, which takes the form of a multinormal distribution that produces *correlated* search points in search space. Its covariance matrix makes the algorithm invariant across rotations in the search space, and enables the algorithm to fine-tune its search appropriately, resulting in arbitrarily high-precision solutions. Furthermore, using the stability properties of the EM algorithm, the algorithm seeks to avoid catastrophically greedy updates on the search policy, thus preventing premature convergence in some cases.

The paper is organized as follows. The next section provides a quick overview of the general problem framework of real-valued black box function optimization. The ensuing sections describe the derivation of the EM-based algorithm, the concept of 'fitness shaping', and the online instantiation of our algorithm. The experiments section shows initial results with a number of unimodal and multimodal benchmark problems. Furthermore, results with the non-Markovian double pole balancing problem are discussed. The paper concludes with a discussion on the advantages and problems of the method, and points to some possible directions for future extensions.

2 Algorithm Framework

First let us introduce the algorithm framework and the corresponding notation. The objective is to optimize the n-dimensional continuous vector of objective parameters $\mathbf{x}$ for an unknown fitness function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The function is unknown or 'black box', in that the only information accessible to the algorithm consists of function measurements selected by the algorithm. The goal is to optimize $f(\mathbf{x})$, while keeping the number of function evaluations – which are considered costly – as low as possible. This is done by successively evaluating batches of a number $1 \dots N$ of separate search points $\mathbf{z}_1 \dots \mathbf{z}_N$ on the fitness function, while using the information extracted from fitness evaluations $f(\mathbf{z}_1)\ldots f(\mathbf{z}_N)$ to adjust both the current candidate solution $\mathbf{x}$ and the search policy defined as a Gaussian with mean $\mathbf{x}$ and covariance matrix $\mathbf{\Sigma}$.

3 Expectation Maximization for Black Box Function Optimization

At every point in time while running the algorithm, we want to optimize the expected fitness $J = \mathbf{E}_{\mathbf{z}}[f(\mathbf{z})]$ of the next batch, given the current batch of search samples. We assume that every batch g is generated by search policy $\pi^{(g)}$ parameterized by $\theta = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$, representing the current candidate solution $\mathbf{x}$ and covariance matrix $\mathbf{\Sigma}$.

In order to adjust parameters $\theta = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$ towards solutions with higher associated fitness, we *match* the search distribution to the actual sample points, but weighted by their utilities. Now let $f(\mathbf{z})$ be the fitness at a particular search point $\mathbf{z}$, and, utilizing the familiar multivariate normal distribution, let $\pi(\mathbf{z}|\theta) = \mathcal{N}(\mathbf{z}|\mathbf{x}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{z}-\mathbf{x})^{\mathbf{T}}\mathbf{\Sigma}^{-1}(\mathbf{z}-\mathbf{x})\right]$ denote the probability density of search point $\mathbf{z}$ given the current search policy π. The expectation

$$J = \mathbf{E}_{\mathbf{z}}[f(\mathbf{z})] = \int \pi(\mathbf{z}|\theta) f(\mathbf{z}) d\mathbf{z}.$$

indicates the expected fitness over all possible sample points, weighted by their respective probabilities under policy π.

3.1 Optimizing Utility-transformed Fitness

While an objective function such as the above is sufficient in theory, algorithms which simply optimize it have major disadvantages. They might be too aggressive when little experience – few sample points – is available, and converge prematurely to the best solution they have seen so far. On the opposite extreme, they might prove to be too passive and be biased by less fortunate experiences. Trading off such problems has been a long-standing challenge in reinforcement learning. However, in decision theory, such problems are surprisingly well-understood [12]. In that framework it is common to introduce a so-called utility transformation $u(f(z))$ which has to fulfill the requirement that it scales monotonically with f, is semi-positive and integrates to a constant. Once a utility transformation is inserted, we obtain an expected utility function given by

$$J_u(\theta) = \int p(\mathbf{z}|\theta) u(f(\mathbf{z})) d\mathbf{z}. \tag{1}$$

The utility function $u(f)$ is an adjustment for the aggressiveness of the decision making algorithms, e.g., if it is concave, it's attitude is risk-averse while if it is convex, it will be more likely to consider a fitness more than a coincidence.

Obviously, it is of essential importance that this risk function is properly set in accordance with the expected fitness landscape, and should be regarded as a metaparameter of the algorithm. Notice the similarity to the selection operator in evolutionary methods.

We have empirically found that rank-based shaping functions (rank-based selection) work best for various problems, also because they circumvent the problem of extreme fitness values disproportionately distorting the estimation of the search distribution, making careful adaptation of the forget factor during search unnecessary even for problems with wildly fluctuating fitness. In this paper, we will consider a simple rank-based utility transformation function, the piecewise linear $u_k = u(f(\mathbf{z}_k)|f(\mathbf{z}_{k-1}), \ldots, f(\mathbf{z}_{k-N}))$ which first ranks all samples $k-N, \ldots, k$ based on fitness value, then assigns zero to the $N-m$ worst ones and assigns values linearly from $0 \ldots 1$ to the m best samples.

3.2 Fitness Expectation Maximization

Analogously as in [8, 9], we can establish the lower bound

$$\begin{aligned}
\log J_u(\theta) &= \log \int q(\mathbf{z}) \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{q(\mathbf{z})} d\mathbf{z} && (2)\\
&\geq \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{q(\mathbf{z})} d\mathbf{z} && (3)\\
&= \int q(\mathbf{z}) \left[\log p(\mathbf{z}|\theta) + \log u(f(\mathbf{z})) - \log q(\mathbf{z})\right] d\mathbf{z} && (4)\\
&:= \mathcal{F}(q, \theta), && (5)
\end{aligned}$$

due to Jensen's inequality with the additional constraint $0 = \int q(\mathbf{z}) d\mathbf{z} - 1$. This points us to the following EM algorithm:

Proposition 1. *An Expectation Maximization algorithm for both optimizing expected utility and the raw expected fitness is given by*

$$\textit{E-Step:} \quad q_{g+1}(\mathbf{z}) = \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{\int p(\tilde{\mathbf{z}}|\theta)u(f(\tilde{\mathbf{z}}))d\tilde{\mathbf{z}}}, \tag{6}$$

$$\textit{M-Step Policy:} \quad \theta_{g+1} = \arg\max_{\theta} \int q_{g+1}(\mathbf{z}) \log p(\mathbf{z}|\theta) d\mathbf{z}. \tag{7}$$

Proof. The E-Step is given by $q = \mathrm{argmax}_q \mathcal{F}(q, \theta)$ while fulfilling the constraint $0 = \int q(\mathbf{z}) d\mathbf{z} - 1$. Thus, we have a Lagrangian $L(\lambda, q) = \mathcal{F}(q, \theta) - \lambda$. When differentiating $L(\lambda, q)$ with respect to q and setting the derivative to zero, we obtain $q^*(\mathbf{z}) = p(\mathbf{z}|\theta)u(f(\mathbf{z})) \exp(\lambda - 1)$. We insert this back into the Lagrangian obtaining the dual function $L(\lambda, q^*) = \int q^*(\mathbf{z}) d\mathbf{z} - \lambda$. Thus, by setting $dL(\lambda, q^*)/d\lambda = 0$, we obtain $\lambda = 1 - \log \int p(\mathbf{z}|\theta)u(f(\mathbf{z}))d\mathbf{z}$, and solving for q^* implies Eq (6). The M-steps compute $\theta_{g+1} = \mathrm{argmax}_\theta \mathcal{F}(q_{g+1}, \theta)$.

In practice, when using a Gaussian search distribution parameterized by $\theta^{(k)} = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$, the EM process comes down to simply fitting the samples in every batch to the Gaussian, weighted by the utilities.

4 Online Fitness Expectation Maximization

In order to speed up convergence, the algorithm can be executed *online*, that is, sample by sample, instead of batch by batch. The online version of the algorithm can yield superior performance since updates to the policy can be made at every sample instead of just once per batch, and because doing so tends to preserve sample diversity better than by using the batch version of the algorithm. Crucial is that a *forget factor* α is now introduced to modulate the speed at which the search policy adapts to the current sample. Batch size N is now only used for utility ranking function u which ranks the current sample among the N last seen samples. The resulting FEM algorithm pseudocode can be found in Algorithm 1.

Algorithm 1 Fitness Expectation Maximization

```
use shaping function $u$, batch size $N$, forget factor $\alpha$
$k \leftarrow 1$
initialize search parameters $\theta^{(k)} = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$
repeat
    draw sample $\mathbf{z}_k \sim \pi(\mathbf{x}, \mathbf{\Sigma})$
    evaluate fitness $f(\mathbf{z}_k)$
    compute rank-based fitness shaping $u_k = u(f(\mathbf{z}_k) | f(\mathbf{z}_{k-1}), \ldots, f(\mathbf{z}_{k-N}))$
    $\mathbf{x} \leftarrow (1 - \alpha u_k)\mathbf{x} + \alpha u_k \mathbf{x}$
    $\mathbf{\Sigma} \leftarrow (1 - \alpha u_k)\mathbf{\Sigma} + \alpha u_k (\mathbf{x} - \mathbf{z}_k)(\mathbf{x} - \mathbf{z}_k)^{\mathbf{T}}$
    $k \leftarrow k + 1$
until stopping criterion is met
```

5 Experiments

5.1 Standard benchmark functions

Good test functions should be easy to interpret, but scale up with n. They must be highly nonlinear, non-separable, largely resistant to hill-climbing, and preferably contain deceptive local suboptima. To test the performance of the algorithm, we chose 6 unimodal functions (Sphere, Schwefel, Tablet, Cigar, Different-Powers, Ellipsoid) and 4 multimodal functions (Ackley, Rastrigin, Weierstrass and Griewank) from a set of benchmark functions from [13] and [6] that are typically used in the literature, for comparison purposes and for competitions. As those functions are designed to be minimized, we take the fitness to be the negative function value. The multimodal functions were tested with both FEM and the Covariance Matrix Adaptation (CMA) [6] algorithm – widely regarded as one of the premier algorithms in this field – for comparison purposes.

In order to prevent potentially biased results, and to avoid trivial optima (e.g. at the origin), we follow [13] and consistently transform (by a combined rotation and translation) the functions' inputs in order to make the variables

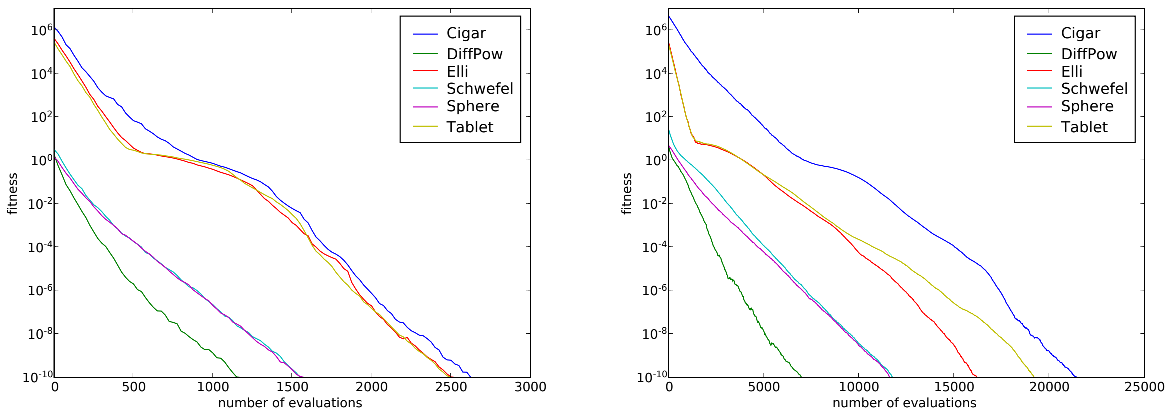

Fig. 1. Results for experiments on the unimodal benchmark functions. Left: dimensionality 5, right: dimensionality 15.

non-separable. This immediately renders many direct search method virtually useless, since they cannot cope with correlated search directions, unlike FEM and CMA.

The tunable parameters of the FEM algorithm are comprised of batch size N, the fitness shaping function u applied on the fitness function f and forget factor α. The parameters should be chosen by the expert to fit the expected ruggedness of the fitness landscape. The forget factor must be low enough such that it does not too quickly forget earlier successful search points. The shaping function must be chosen such that enough randomness is preserved in the search policy after every update, which entails including the lesser samples in utility attribution. For all experiments, comprising both the benchmark unimodal/multimodal functions and the non-Markovian double pole balancing task, initial $\mathbf{\Sigma}$ was set to the identity matrix $\mathbf{\Sigma} = \mathbf{I}$ and $\mathbf{x}$ was always randomly initialized as $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We ran FEM on the set of unimodal benchmark functions with dimensions 5 and 15 using a target precision of 10^{-10}. Figure 1 shows the average number of evaluations until success over 20 runs on the unimodal functions. The parameter settings for dimensionality 5 were identical in all runs: $\alpha = 0.1$ and $N = 50$, parameter m for selecting the shaping function's top m samples was set at $m = 5$. The parameter settings for all runs in dimensionality 15 were: $\alpha = 0.02$, $N = 25$ and $m = 10$. All runs converged. The number of evaluations was roughly equal to that of CMA on the small dimensionality, and for most problems not more than a factor 3 slower, even with dimensionality 15 [6].

On the multimodal benchmark functions we performed experiments while varying the distance of the initial guess to the optimum between 1 and 100. As with the unimodal functions, the problems were appropriately translated and rotated, while the initial $\mathbf{x}$ was randomly initialized on the surface of the hypersphere with radius 1, 10 or 100 and the optimum at its center. Those runs were performed on dimension 2 with a target precision of 0.01, since here the focus was on avoiding local maxima. The parameter settings for the multimodal runs were: $\alpha = 0.02$, $N = 25$ and $m = 10$. Table 1 shows, for all multimodal functions, the percentage of runs where FEM found the global optimum (as opposed to it getting stuck in a local suboptimum) depending on the distance from the initial guess to the optimum. The percentages are computed over 100 runs. For comparison purposes we included the results for the CMA implementation of [6], although it must be said that in all likelihood better results can be achieved for CMA using population sizes that are larger than standard for that algorithm.

One additional, linear benchmark function $f(\mathbf{z}) = \sum_j z_j$ was tested to verify the expected premature convergence of the algorithm. Indeed, FEM converges prematurely like EDAs typically do (e.g. [14]), while CMA performed well (see e.g. [15]). This suggests the approach might not be applicable to all domains and that it might benefit from a mutative approach modeling mutations instead of weighted sample distributions.

Lastly, we performed experiments using a batch-based version of the algorithm instead of the online version. We found the standard benchmark problems could only be solved using large batch sizes (1000 and up), slowing down the algorithm considerably. This might be due to the reduced sample diversity using small batch sizes, which is ameliorated using an online update rule which only gradually adjusts $\mathbf{\Sigma}$ values.

To summarize, our experiments on these standard black box optimization benchmarks indicate that FEM is competitive with other high-performance algorithms. The premature convergence on the simple linear test function was expected and it remains to be seen whether this will affect the long-term viability of the approach. Last, the superior performance of the online version of this algorithm might indicate that the problem of diversity maintenance could prove to be an important topic of future research on FEM and EDAs in general.

Table 1. Results for the multimodal benchmark functions. Shown are percentages of runs that found the global optimum, for both FEM and CMA, for varying starting distances.

	FEM			**CMA**		
Distance	1	10	100	1	10	100
Rastrigin	91%	87%	64%	13%	11%	14%
Ackley	100%	100%	0%	89%	70%	3%
Weierstrass	19%	9%	19%	90%	92%	92%
Griewank	100%	2%	0%	100%	2%	0%

5.2 Non-markovian Double Pole Balancing

Non-Markovian double pole balancing [16] can be considered a difficult benchmark task for control optimization. We use the implementation as found in [17]. The FEM algorithm optimizes the parameters of the controller, which is implemented as a simple neural network with three inputs, three hidden sigmoid neurons, and one output neuron.

The algorithm's parameters were set as follows: piecewise linear shaping function with $m = 5$ (top 5 selection), forget factor $\alpha = 0.05$ and batch size $N = 50$. A run was considered a *success* when the poles did not fall over for $100,000$ time steps. The results on a total of 200 runs are, on average, 2099 evaluations until success (standard deviation: 1505). Not included in these statistics are 49/200 runs that did not reach success within the limit of 10000 evaluations, which compares badly with both CoSyNE and CMA which (almost) always converge. Table 2 shows results of other premier algorithms applied to this task, including CMA. All methods optimized the same type of recurrent neural network, albeit with differing numbers of hidden neurons. FEM, when it converges, outperforms all other methods except CoSyNE. Since our algorithm performs well on this relatively hard control benchmark, we expect the algorithm to do well on future real-world experiments.

Table 2. Results for non-Markovian double pole balancing. The table shows the average number of evaluations for SANE [18], ESP [17], NEAT [19], CMA [20, 6], CoSyNE [21] and FEM.

Method	SANE	ESP	NEAT	CMA	CoSyNE	FEM
Evaluations	$262,700$	$7,374$	$6,929$	$3,521$	$1,249$	$2,099$

6 Discussion

Fitness Expectation Maximization constitutes a simple, principled approach to real-valued black box function optimization with a rather clean derivation from first principles. Its theoretical relationship to the field of reinforcement learning and in particular reward-weighted regression should be clear to any reader familiar with both fields. We anticipate that rephrasing the black box optimization problem as a reinforcement learning problem solvable by RL methods will spawn a whole series of additional new algorithms exploiting this connection.

The experiments show that, on the unimodal and multimodal benchmarks, FEM is competitive with respect to its the main 'competitor' algorithm CMA, at least on lower dimensional problems. Taking into account the good results on the pole balancing tasks, we envision that FEM might become a serious competitor in the field of black box function optimization, especially for neuroevolution.

Future work on FEM will include a systematic study that must determine whether it can be made to outperform other search methods consistently on other typical benchmarks and real-world tasks. It remains to be seen how the method scales up with increased dimensionality, especially compared to CMA. We suggest extending the algorithm from a single multinormal distribution as search policy representation to a mixture of Gaussians (which is a common procedure for 'vanilla' EM), thus further reducing its sensitivity to local suboptima. Other pressing work includes a theoretical analysis of the shaping (selection) function, which should ideally be made to adapt automatically based on the data instead of tuned manually.

The premature convergence on the linear test function is worrisome. Future work will determine whether this phenomenon affects the practical applicability to real-world problems such as neurocontrol. Alternatively, we must investigate whether the introduction of a more mutative approach like CMA might be beneficial.

7 Conclusion

We introduced Fitness Expectation Maximization to tackle the important class of real-valued 'black box' function optimization problems. Reframing black box optimization as a one-step reinforcement learning problem, we developed a method similar in spirit to expectation maximization. Using a search policy which matches samples weighted by their utilities, the algorithm performs competitively on a standard benchmark set of unimodal and multimodal functions and non-Markovian double pole balancing control.

Acknowledgments

This research was funded by SNF grants 200021-111968/1 and 200021-113364/1.

References

1. Spall, J., Hill, S., Stark, D.: Theoretical framework for comparing several stochastic optimization approaches. Probabilistic and Randomized Methods for Design under Uncertainty (2006) 99–117
2. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science, Number 4598, 13 May 1983 **220, 4598** (1983) 671–680
3. Spall, J.C.: Stochastic optimization and the simultaneous perturbation method. In: WSC '99: Proceedings of the 31st conference on Winter simulation, New York, NY, USA, ACM (1999) 101–109
4. Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning. Springer-Verlag (2004)
5. De Boer, P., Kroese, D., Mannor, S., Rubinstein, R.: A tutorial on the cross-entropy method. In: Annals of Operations Research. Volume 134. (2004) 19–67

6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2) (2001) 159–195
7. Larraanaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Norwell, MA, USA (2001)
8. Peters, J., Schaal, S.: Reinforcement learning by reward-weighted regression for operational space control. In: Proceedings of the International Conference on Machine Learning (ICML). (2007)
9. Dayan, P., Hinton, G.E.: Using expectation-maximization for reinforcement learning. Neural Computation **9**(2) (1997) 271–278
10. Wolpert, D.H., Rajnarayan, D.G.: Parametric Learning and Monte Carlo Optimization. ArXiv e-prints **704** (April 2007)
11. Gallagher, M., Frean, M., Downs, T.: Real-valued evolutionary optimization using a flexible probability density estimator. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 1., Orlando, Florida, USA, Morgan Kaufmann (13-17 1999) 840–846
12. Chernoff, H., Moses, L.E.: Elementary Decision Theory. Dover Publications (1987)
13. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore (2005)
14. C. Gonzalez, J. A. Lozano, P.L.: Mathematical modelling of umdac algorithm with tournament selection. behaviour on linear and quadratic functions. International Journal of Approximate Reasoning **31**(3) (2002) 313–340
15. Hansen, N.: An analysis of mutative σ-self-adaptation on linear fitness functions. Evolutionary Computation **14**(3) (2006) 255–275
16. Wieland, A.: Evolving neural network controllers for unstable systems. In: Proceedings of the International Joint Conference on Neural Networks (Seattle, WA), Piscataway, NJ: IEEE (1991) 667–673
17. Gomez, F.J., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior **5** (1997) 317–342
18. Moriarty, D.E., Miikkulainen, R.: Efficient reinforcement learning through symbiotic evolution. Machine Learning **22** (1996) 11–32
19. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
20. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Congress on Evolutionary Computation (CEC 2003). Volume 4., IEEE Press (2003) 2588–2595
21. Faustino Gomez, J.S., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: Proceedings of the 16th European Conference on Machine Learning (ECML 2006). (2006)