Boltzmann Machine for Population-Based Incremental Learning

Arnaud Berny[1]

Abstract. We propose to apply the Boltzmann machine (BM) to population-based incremental learning (PBIL). We will replace the statistical model used in PBIL, which assumes that the binary variables of the optimisation problem are independent, with that of a BM. From the logarithm of the expectation of the function to maximise, we derive specific learning rules for the BM. These learning rules involve expectations with respect to the distribution of the BM and also to its selected version, in the spirit of PBIL or genetic algorithms. New populations are sampled from the BM using traditional Gibbs sampling. The proposed BM-PBIL algorithm alternates Gibbs sampling, selection, and update of the parameters. We evaluate BM-PBIL with different classes of functions, compare it to the original PBIL, and identify classes for which it is superior to PBIL, in particular functions with jumps and quadratic functions.

Keywords: genetic algorithms, neural networks, search, black-box optimisation, boltzmann machine, Gibbs sampling, population-based incremental learning.

1 INTRODUCTION

Population-based incremental learning (PBIL) [1] is an adaptive algorithm for the optimisation of functions defined on fixed-length binary vectors. It belongs to the class of black-box optimisation algorithms, like genetic algorithms, which means that it can only gain some knowledge about the function to maximise by evaluating it on any point in the search space.

PBIL maintains a probability vector of the same dimension as that of binary vectors, whose components represent the probabilities of sampling a "1". The main assumption is that the components are statistically independent. At each iteration, PBIL samples a population of binary vectors from the probability vector. Then, it selects the best two ones with respect to the function to maximise and averages them. Finally, it moves the probability vector in the direction of the average using a linear equation.

Despite its apparent limitations, PBIL has been reported to perform competitively on different problems, in particular with respect to genetic algorithms. However, it seems promising to consider underlying statistical models more complex than product probabilities, and still keep the idea of an adaptive algorithm, in contrast to Distribution Estimation Algorithms which mainly use directed graphical models [2]. Such studies have already been initiated. In the connectionist community, Williams and Peng [5] have proposed a feedforward network of logistic Bernoulli units. Their conclusion was negative on the approach compared to independent units. We suspect the feedforward architecture to be responsible for this failure. This has motivated our interest in Boltzmann machines which are fully connected neural networks. In the evolutionary computation community, Zhang and Shin [7] have proposed to use a Helmholtz machine which is a two-layered bidirectional neural network. One objection to this approach is the lack of some energy function from which to derive update rules. This is what we provide in this paper for a Boltzmann machine. We define a precise statistical criterion whose gradient gives rise to update rules involving a selection operator, in the spirit of genetic algorithms and PBIL.

Sec. 2 provides an introduction to the statistical approach in optimisation. Sec. 3 presents the statistical models of PBIL and Boltzmann machines. Sec. 4 contains the derivation of specific update rules for a Boltzmann machine applied to an optimisation problem. Sec. 5 presents the stochastic algorithm BM-PBIL. In Sec. 6, we report some results obtained with PBIL and BM-PBIL on a set of functions from different classes, both deterministic and randomly generated ones. We are aware of the "No Free Lunch" theorem [6] which states that the average performance of any algorithm across all possible functions is conserved (considering functions taking values in a finite set). As a consequence, there exists no universal black-box optimisation algorithm, but only specialized ones. This is what we have observed for PBIL and BM-PBIL applied to a limited, yet diverse, set of functions.

2 STATISTICAL OPTIMISATION

Let $f : E \to \mathbf{R}$ be some function to maximise, also called fitness function, on the set $E = \{0,1\}^n$ of binary vectors of length n. Statistical optimisation consists in replacing the maximisation of f on E with that of a statistical criterion defined on the set of probabilities on E. One such criterion is the expectation of f with respect to a probability. Let π be some probability on E and define

$$J(\pi) = \mathbf{E}_\pi(f) = \sum_{x \in E} f(x)\pi(x)\,.$$

We also use the two logarithmic criteria defined as $\log J(\pi) = \log \mathbf{E}_\pi(f)$ and $\log \mathbf{E}_\pi(e^{\beta f})$, $\beta > 0$, in order to introduce fitness proportionate and Boltzmann selection (do not confuse with Boltzmann machine).

Such a transformation of the optimisation problem does not reduce the complexity of the search. In order to do this, we have to restrict the search to some family of probabilities.

3 STATISTICAL MODELS

3.1 Product probabilities

Product probabilities are characterized by the fact that the components of their random vectors are statistically independent. We call

[1] arnaud.berny@free.fr

such a model linear, for it is completely determined by n parameters. This is the model used by PBIL.

A product probability μ on E is defined by

$$\mu(x) = \prod_{i=1}^{n} \phi(x_i,\, v_i)\,,$$

where $x = (x_i)$, $1 \le i \le n$, and ϕ is given by

$$\phi(x_i,\, v_i) = \begin{cases} v_i & \text{if } x_i = 1, \\ 1 - v_i & \text{if } x_i = 0. \end{cases}$$

Sampling μ reduces to performing n independent and biased coin tosses.

3.2 Boltzmann machines

In order to overcome the limitations of the linear model, we propose to use a Boltzmann machine, which is equivalent to a Gibbs probability with a quadratic energy function.

Let Q be a symmetric nn matrix and m a vector in $\mathbf{R}^n$. We define the opposite of the energy function of the Boltzmann machine as

$$h(x) = \langle x - m \mid Q \mid x - m \rangle\,,$$

for all $x \in E$, where $\langle x|$ denotes the transposed vector of x which we write $|x\rangle$ in this context.

The Gibbs probability related to h is defined as

$$\pi(x) = e^{h(x)}/Z\,,$$

for all $x \in E$, where Z is the partition function (normalisation of probabilities)

$$Z = \sum_{x \in E} e^{h(x)}\,.$$

4 GRADIENT SYSTEMS

In order to maximise the criterion J, we use a gradient system on the parameters of the model which takes the form of a differential equation

$$\mathrm{d}\,\theta / \mathrm{d}\,t = \nabla_\theta J\,,$$

where ∇ is the gradient operator, and $\theta = m$ or Q.

The key observation is that the gradient of the criterion can be expressed as an expectation, which suggests a stochastic algorithm. More precisely,

$$\begin{aligned} \nabla J &= \sum_{x \in E} f(x) \nabla \pi(x) \\ &= \sum_{x \in E} f(x) \nabla \log \pi(x) \pi(x) \\ &= \mathbf{E}_\pi (f \nabla \log \pi)\,. \end{aligned}$$

We have

$$\log \pi(x) = h(x) - \log Z\,.$$

The gradient of the logarithm of the partition function is

$$\begin{aligned} \nabla \log Z &= \nabla Z / Z \\ &= \sum_{x \in E} \nabla h(x) e^{h(x)} / Z \\ &= \mathbf{E}_\pi (\nabla h)\,. \end{aligned}$$

Then

$$\nabla \log \pi(x) = \nabla h - \mathbf{E}_\pi (\nabla h)\,,$$

and the gradient of the criterion can be written

$$\nabla J = \mathbf{E}_\pi (f \nabla h) - \mathbf{E}_\pi (\nabla h)\, \mathbf{E}_\pi (f)\,.$$

We now compute the gradient of the logarithmic criterion in order to introduce fitness proportionate selection.

$$\begin{aligned} \nabla \log J &= \mathbf{E}_\pi \left(\nabla h \frac{f}{\mathbf{E}_\pi (f)} \right) - \mathbf{E}_\pi (\nabla h) \\ &= \mathbf{E}_\sigma (\nabla h) - \mathbf{E}_\pi (\nabla h)\,, \end{aligned}$$

where σ is the probability resulting from the selection of π:

$$\pi(x) \xrightarrow{\text{selection}} \sigma(x) = \frac{f(x) \pi(x)}{\mathbf{E}_\pi (f)}$$

Using the same reasoning, the gradient of the criterion $\log \mathbf{E}_\pi (e^{\beta f})$ is obtained replacing fitness proportionate selection with Boltzmann selection:

$$\pi(x) \xrightarrow{\text{selection}} \sigma(x) = \frac{e^{\beta f(x)} \pi(x)}{\mathbf{E}_\pi (e^{\beta f})}$$

The gradient of the energy function with respect to each parameter is

$$\begin{aligned} \nabla_m h(x) &= 2Q(m - x) \\ \nabla_Q h(x) &= |x - m\rangle \langle x - m|\,. \end{aligned}$$

The gradient of the logarithmic criterion itself with respect to each parameter is

$$\begin{aligned} \nabla_m \log J &= 2Q\big(\mathbf{E}_\pi (x) - \mathbf{E}_\sigma (x)\big) \\ \nabla_Q \log J &= \mathbf{E}_\sigma \big(|x - m\rangle \langle x - m|\big) - \mathbf{E}_\pi \big(|x - m\rangle \langle x - m|\big)\,. \end{aligned}$$

Initial conditions for the gradient system are

$$\begin{aligned} m(0) &= (1/2,\, \ldots\,,\, 1/2) \\ Q(0) &= 0\,. \end{aligned}$$

5 STOCHASTIC ALGORITHMS

5.1 Gibbs sampling

The above gradients require the computation of expectations with respect to the Gibbs probability π or its selected version σ. As in Monte-Carlo integration, we propose to replace expectations with averages over samples from both probabilities. Gibbs sampling provides samples from an approximation of the Gibbs probability, and the selection operator provides samples from an approximation of its selected version.

Let us briefly recall the principles of Gibbs sampling which is a Markov chain on the search space. At each iteration, a component i of the current state x is randomly and uniformly chosen, and its conditional probability $p = \pi(x_i = 1 \mid x_j, j \neq i) = \pi(x^1)/\pi(x^1) + \pi(x^0)$ is computed, where x^1 is the current state x in which x_i has been replaced by 1 (by 0 for x^0).The next value of x_i is then sampled using a biased coin toss, and another component is considered.

From the expression of the Gibbs probability, we have $p = 1/1 + e^{-\Delta}$, where $\Delta = h(x^1) - h(x^0)$, that is the variation of h when the value of the ith component is switched from 0 to 1, the other ones being kept constant. From the expression of h, we find

$$\Delta = (1 - 2m_i)q_{ii} + 2\sum_{j\neq i}(x_j - m_j)q_{ij} \,,$$

which can be calculated in time linear in n.

We use Gibbs sampling to build a finite population of N individuals denoted by (x^i), $1 \leq i \leq N$. For each $1 \leq i \leq N$, we run the Gibbs sampler for T iterations and set $x^i = x$. Thus, the Hamming distance between two consecutive individuals is bounded by T (and n). We do not reset the state x between consecutive individuals or consecutive populations. The initial state is uniformly sampled. The complexity of building a population is proportional to NTn.

Contrary to common practice in the field of learning in Boltzmann machines, we cannot avoid Gibbs sampling, for example using mean field approximation, for only samples can be evaluated and provide some knowledge on the fitness function. Gibbs sampling can be seen in this context as a controlled mutation operator.

5.2 Update rule for BM-PBIL

Bringing together Gibbs sampling, gradient expressions, and selection yields the following population-based incremental learning algorithm:

1. Sample a population (x^i), $1 \leq i \leq N$, of binary vectors using Gibbs sampling;
2. Apply a selection operator to it to produce a new population, (y^i), $1 \leq i \leq P$. In the case of Bolzmann selection, the probability of being selected for an individual x^i is

$$e^{\beta f(x^i)} \Big/ \sum_{j=1}^{N} e^{\beta f(x^j)} \,,$$

   where $\beta > 0$ controls the selective pressure;
3. Compute averages:

$$m_\pi = 1/N \sum_{i=1}^{N} x^i \quad m_\sigma = 1/P \sum_{i=1}^{P} y^i$$

$$Q_\pi = 1/N \sum_{i=1}^{N} |x^i - m\rangle\langle x^i - m|$$

$$Q_\sigma = 1/P \sum_{i=1}^{P} |y^i - m\rangle\langle y^i - m| \,;$$

4. Update parameters using

$$\begin{aligned} m_{t+1} &= m_t + \alpha \times Q_t(m_\pi - m_\sigma) \\ Q_{t+1} &= Q_t + \alpha \times (Q_\sigma - Q_\pi) \,, \end{aligned}$$

   where $\alpha > 0$ is the learning gain.

The complexity of update rules is proportional to n^2.

5.3 Update rule for PBIL

We briefly recall the update rule of PBIL which will serve as a reference for BM-PBIL. The underlying statistical model of PBIL has been described in Sec. 3.1. PBIL updates a probability vector v which is directly used to sample a population of N individuals at each iteration. The probability vector v is updated with the equation

$$v_{t+1} = v_t + \alpha \times (m_\sigma - v_t) \,,$$

where m_σ is the average of selected individuals.

6 EXPERIMENTS

6.1 Test functions

We have applied both BM-PBIL and PBIL to six different classes of fitness functions, both deterministic and random ones:

1. *"Big Jump"* is the *"One Max"* function whose optimum has been isolated. Let x be some binary vector of length n and m be its number of "1", then "Big Jump" is defined as

$$\begin{cases} n, & \text{if } m = n, \\ 0, & \text{if } n - t < m < n, \\ m, & \text{otherwise,} \end{cases}$$

   where t controls the isolation of the global optimum. We have considered the cases $t =$ 10, 15, and 20.
2. The *"Four Peaks"* problem has been introduced in [1]. Its fitness function has four optima, two local, and two global. Just as in the case of "Big Jump", a threshold controls isolation of global optima. The fitness function is defined as

$$\max(o(x), z(x)) + r(x) \,,$$

   where $o(x)$ is the number of leading "1", $z(x)$ the number of trailing "0", and $r(x)$ is a conditional reward such that

$$r(x) = \begin{cases} n, & \text{if } o(x) > t \text{ and } z(x) > t \\ 0, & \text{otherwise.} \end{cases}$$

   We have considered the cases $t =$ 10, 15, and 20.
3. The problem of *summation cancellation* has been defined in [2]. It consists in cancelling the sum

$$\sum_{i=1}^{n/9} |b_i| \,,$$

   where $b_i = b_{i-1} + a_i$, $b_1 = a_1$, and each a_i is a fixed-point number represented with nine bits between -2.56 and 2.55. In this problem, each binary vector of length n is first converted to $n/9$ fixed-point numbers a_i and evaluated. The fitness function to be maximised is defined as the opposite of the sum. We have also considered a variation of the summation cancellation problem defined in [4] where $b_i = \sin b_{i-1} + a_i$.
4. The problem of *equal products* has been defined in [2]. Given a set of real numbers (a_i), $1 \leq i \leq n$, it consists in finding a partition, represented as a binary vector x of length n, such that

$$\prod_{i=1}^{n} a_i^{x_i} = \prod_{i=1}^{n} a_i^{1-x_i} \,.$$

   If no such partition exists, the distance between both products should be minimised or its opposite maximised. We have generated five random instances of this problem where each a_i has been sampled uniformly on the interval [0, 4].

5. *NK landscapes* [3] are a model of the complex interactions between an individual's genes and its environment. The two structural parameters of a family of NK landscapes are the length n of binary vectors and the order k of interactions between their components. Each fitness function is defined as the sum of n partial functions, each of which depends on $k + 1$ components. To build such a fitness function, we have sampled a set of k neighbours for each component, and for each of their 2^{k+1} combinations, we have sampled a value from the normal distribution for the corresponding partial function. We have generated five functions for $k = 2$ and five others for $k = 4$.
6. Finally, we have considered the problem of *boolean quadratic programming*, which is equivalent to finding the ground state of a spin system in the context of statistical physics. This is an NP-complete problem. Fitness functions for this problem are equivalent to energy functions of Bolzmann machines and are of the form

$$\sum_{i<j}(2x_i - 1)(2x_j - 1)a_{ij} + \sum_i (2x_i - 1)b_i \, .$$

We have generated five such functions whose coefficients have been sampled from the normal distribution.

6.2 Parameters

The length of binary vectors is $n = 80$, except for the two summation cancellation problems for which it is set to 90. Both algorithms have been allowed 10000 iterations. The sampled population size is $N = 50$. The total number of function evaluations is then equal to 500000. To give an order of magnitude, the ratio of the size of the sampled search space to the size of the whole search space is less than 10^{-18}. In the case of BM-PBIL, the length of the Markov chain between two individuals is $T = 20$. At each iteration, only the best individual is selected from the sampled population. This is equivalent to Bolzmann selection with $\beta = \infty$. There is a tradeoff between the quality of the solutions and the speed of convergence which is controlled by the learning gain. BM-PBIL and PBIL have different dynamics and do not achieve the best results at the same learning gain. We have considered $\alpha = 0.001$ and $\alpha = 0.005$ for both, but $\alpha = 0.005$ proved to be too high for BM-PBIL. Each algorithm has been run 30 times, and for each run, we have recorded the best fitness at each iteration.

6.3 Results

The results are shown on Tables 1–3. For each algorithm, we have computed the average (over 30 runs) of the best fitness found at the end of the simulation, as well as the average iteration at which the algorithm has reached it for the first time. We have also computed the standard deviation of these two statistics. For each test function, the best fitness has been typeset in bold.

BM-PBIL dominates PBIL on all but two jump functions, "Big Jump" with a threshold $t = 20$ and "Four Peaks" with a threshold $t = 10$. The former function is a hard instance, in the sense that both algorithms always get stuck at a local maximum. The latter function is an easy one, in the sense that both algorithms always find one of the two global maxima. However, for both functions, PBIL with $\alpha = 0.005$ is faster than BM-PBIL.

BM-PBIL clearly dominates PBIL on the spin functions. This last result is not a surprise since BM-PBIL has been designed to take into account some level of interaction between the binary variables of the problem. BM-PBIL slightly dominates PBIL on the problem of equal products, where it finds the best result in 3 out of 5 instances.

BM-PBIL is in turn dominated on summation cancellation problems and on NK landscapes, either of order 2 or 4. One may observe that NK landscapes of order 2 cannot be represented as quadratic functions. So neither BM-PBIL nor PBIL are able to capture the structure of these landscapes. Improvement in the statistical model has been of little help to BM-PBIL. We conjecture that the reason for this failure lies in the randomness of NK landscapes rather than in their structure. For such highly random functions, PBIL is to be preferred to BM-PBIL (and indeed, for completely random functions, one should consider random search).

Table 1. Results for BM-PBIL ($\alpha = 0.001$). `bj`, `ep`, and `fp` respectively refer to "Big Jump", "equal products", and "Four Peaks". The dimension of the search space is $n = 80$, except for `cancel` and `cancel-sin` for which $n = 90$.

	BM-PBIL			
	Max		Iteration	
Function	Mean	Dev.	Mean	Dev.
`bj-10`	**80.00**	0	1215	139
`bj-15`	**68.00**	6.0	922	268
`bj-20`	**60.00**	0	636	150
`cancel`	-1.053e-01	5.7e-02	9319	521
`cancel-sin`	-7.668e-02	3.6e-02	7902	614
`ep-0`	**-6.17**	6.4	4601	3194
`ep-1`	**-434.35**	448.1	5656	2761
`ep-2`	-20.93	19.7	5747	3291
`ep-3`	-11.16	12.8	5889	2554
`ep-4`	**-5.557e-01**	4.7e-01	6497	2570
`fp-10`	**149.00**	0	5216	1044
`fp-15`	**144.00**	0	4269	421
`fp-20`	**137.97**	4.1	3905	322
`nk-2-0`	**64.89**	1.1e-01	2725	461
`nk-2-1`	62.67	5.0e-01	2578	283
`nk-2-2`	61.44	2.9e-01	2530	277
`nk-2-3`	66.17	5.0e-01	3551	1004
`nk-2-4`	68.09	2.4e-01	2924	355
`nk-4-0`	69.39	2.3	5054	943
`nk-4-1`	76.02	1.4	4707	792
`nk-4-2`	72.79	3.2	5071	1003
`nk-4-3`	70.79	1.8	4668	868
`nk-4-4`	72.89	2.7	5242	1424
`spin-0`	**489.65**	13.2	4513	376
`spin-1`	**496.29**	12.9	4892	369
`spin-2`	**527.58**	14.5	4459	428
`spin-3`	**497.72**	9.1	4529	439
`spin-4`	**512.36**	8.5	4574	499

Tab. 4 shows the computation time[2] for both algorithms on two functions, `fp`, which is computed in a time linear in n, and `spin`, which is computed in a time quadratic in n. Despite its additional complexity, BM-PBIL is interesting when the function itself has a quadratic complexity. One can scale the length T of the Markov chain so as to obtain a quadratic complexity, which is acceptable.

7 CONCLUSION

We have proposed an extension of population-based incremental learning which relies on a Bolzmann machine. We have derived specific update rules, involving a selection operator, from a precise statistical criterion. Their complexity can be kept quadratic in the length of the binary vectors. We have compared the resulting BM-PBIL algorithm to PBIL on different test functions. BM-PBIL was found to

[2] The algorithms have been implemented in C++ and run on a Athlon XP 1800 processor.

Table 2. Results for PBIL ($\alpha = 0.001$).

	PBIL			
	Max		Iteration	
Function	Mean	Dev.	Mean	Dev.
bj-10	79.00	3.0	5580	1823
bj-15	65.00	0	1096	104
bj-20	**60.00**	0	460	125
cancel	-1.12	3.1e-01	9563	566
cancel-sin	-4.217e-01	1.7e-01	9787	195
ep-0	-7.35	7.8	4671	2756
ep-1	-534.67	542.1	5240	2895
ep-2	-22.49	22.1	4957	2783
ep-3	-18.47	17.3	4750	2922
ep-4	-7.790e-01	7.1e-01	4984	2642
fp-10	144.30	2.3	9595	423
fp-15	141.47	2.3	9721	249
fp-20	129.00	7.2	8784	1197
nk-2-0	64.86	6.6e-02	6770	356
nk-2-1	63.08	3.4e-01	6736	688
nk-2-2	61.55	1.8e-01	7879	942
nk-2-3	**66.21**	4.9e-01	8093	1171
nk-2-4	**68.33**	6.2e-02	7415	660
nk-4-0	**72.06**	1.2	9565	246
nk-4-1	76.28	9.3e-01	8980	706
nk-4-2	75.06	2.5	9422	552
nk-4-3	71.15	1.5	9531	301
nk-4-4	**76.41**	2.0	9576	357
spin-0	481.90	6.9	8964	671
spin-1	475.87	10.4	9449	507
spin-2	519.29	7.8	8933	725
spin-3	493.84	5.0	9134	799
spin-4	490.36	14.6	9351	606

Table 3. Results for PBIL ($\alpha = 0.005$).

	PBIL			
	Max		Iteration	
Function	Mean	Dev.	Mean	Dev.
bj-10	79.00	3.0	2033	1293
bj-15	65.00	0	259	26
bj-20	**60.00**	0	126	32
cancel	**-4.867e-02**	7.6e-03	3372	215
cancel-sin	**-4.833e-02**	7.8e-03	2929	181
ep-0	-7.31	8.6	5184	2874
ep-1	-539.61	465.9	5284	2994
ep-2	**-15.03**	14.1	5445	2516
ep-3	**-10.55**	11.5	5017	2556
ep-4	-7.432e-01	7.4e-01	4678	2921
fp-10	**149.00**	0	2194	92
fp-15	137.30	12.7	2020	226
fp-20	89.90	18.6	2256	417
nk-2-0	64.89	9.4e-02	1460	128
nk-2-1	**63.08**	3.6e-01	1495	251
nk-2-2	**61.59**	3.3e-01	1590	165
nk-2-3	66.10	5.1e-01	1718	230
nk-2-4	68.28	9.8e-02	1544	111
nk-4-0	71.85	1.8	2208	311
nk-4-1	**77.07**	1.4	1952	244
nk-4-2	**75.24**	3.2	2130	291
nk-4-3	**73.11**	2.1	2125	270
nk-4-4	76.39	1.9	2061	366
spin-0	476.29	16.2	2036	325
spin-1	479.95	14.2	2165	399
spin-2	519.61	10.6	1957	364
spin-3	484.23	13.6	2057	358
spin-4	491.93	21.3	2169	450

Table 4. Comparison of computation times (in seconds).

Function	PBIL	BM-PBIL
fp	3.6	37.4
spin	17.6	51.7

dominate PBIL on jump functions, on quadratic functions, and on the problem of equal products. It was found in turn to be dominated on NK landscapes and on the problem of summation cancellation. These promising results should motivate the further development of adaptive algorithms for black-box optimisation taking into account the correlations between the variables of the problem, in parallel with the work done with directed graphical models.

REFERENCES

[1] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proceedings of the 12th Annual Conference on Machine Learning*, pages 38–46, 1995.

[2] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. Technical Report CMU-CS-97-107, Carnegie-Mellon University, January 1997.

[3] S. A. Kauffman. *The origins of order: self-organisation and selection in evolution*. Oxford University Press, 1993.

[4] M. Sebag and M. Schoenauer. A society of hill-climbers. In *Proc. IEEE Int. Conf. on Evolutionary Computation*, pages 319–324, Indianapolis, April 1997.

[5] R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[6] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation*, 1(1):67–82, April 1997.

[7] B.-T. Zhang and S.-Y. Shin. Bayesian evolutionary optimization using Helmholtz machines. In M. Schoenauer et al., editors, *Parallel Problem Solving from Nature VI*, Lecture Notes in Computer Science, pages 827–836, Paris, September 2000. Springer-Verlag.