The 1998 IEEE International Joint Conference on Neural Networks Proceedings

(V.2) (V.B)

The 1998 IEEE International Joint Conference on Neural Networks Proceedings

IEEE World Congress on Computational Intelligence

May 4 - May 9, 1998
Anchorage, Alaska, USA

Volume 2 Page 803 - 1687

The 1998 IEEE International Joint Conference on Neural Networks Proceedings

IEEE Catalog Number 98CH36227
ISBN 0-7803-4859-1 (softbound)
ISBN 0-7803-4860-5 (casebound)
ISBN 0-7803-4861-3 (microfiche)
ISBN 0-7803-4862-1 (CD-ROM)
ISSN 1098-7576

Additional copies of this publication are available from

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
1-800-678-IEEE or 1-732-981-0060

Coordinated and produced by

ALTEC
Technology Driven Print Solutions

in cooperation with

DOCUMATION

Contact ALTEC at 800.997.9921 x 1334

Author's Index

A Complex Valued Hebbian Learning Algorithm

Maria Cristina Felippetto De Castro, Fernando César C. De Castro, José Nelson Amaral, Paulo Roberto G. Franco

***Abstract*— We present a new training rule for a single-layered linear network with complex valued weights and activation levels. This novel network can be used to extract the principal components of a complex valued data set. We also introduce a new training method that reduces the training time of the complex valued as well as of the real valued network. The use of the new network and training algorithm is illustrated with a problem of compressing images represented in the spectral domain.**

***Keywords*— Generalized Hebbian Learning, Principal Component Analysis, Complex Domain Neural Networks.**

I. Introduction

In this work we develop a complex valued version of the Generalized Hebbian Algorithm (GHA) proposed by Sanger[14]. GHA combines the Gram-Schmidt orthonormalization [1] with the single linear neuron model introduced by Oja [11].

A complex valued GHA finds application in the extraction of principal components of a complex data set, such as those encountered in radar and sonar systems or communication systems[6], [7].

We use a complex valued artificial neural network with a single layer of linear neurons trained according to a Hebbian learning rule to perform Principal Components Analysis (PCA) [2], [3], [8], [10], [15]. The learning rule has been extended to accommodate complex values. The data and the synapse weights are also complex valued. After the convergence of the Complex Generalized Hebbian Algorithm (CGHA), each neuron of the network yields the following: i) A set of complex synapse weights that corresponds to the components of the eigenvector associated with an eigenvalue of the input data set correlation matrix; and ii) An output value that is the principal component which represents the projection of the complex input vector over the associated complex eigenvector.

The set of eigenvectors of the correlation matrix forms an orthonormal basis for the modes of variation of the input data set [1], [10]. The eigenvalue associated with each eigenvector is equal to the variance of the projection of the data set in the direction of that eigenvector. Also, the variance of the data set projections are local maxima in the directions of the eigenvectors [13], [12].

Principal Component Analysis (PCA) in the complex domain follows similar rules as those for PCA in the real domain. Assume that we have a complex data set X composed of a set of zero-mean, complex vectors. The correlation matrix is $C_X = \langle XX^H \rangle$, where $\langle\cdot\rangle$ is the expectation over the set operator and X^H denotes the conjugate transpose of X. C_X represents the average energy over all possible combinations of two data elements in X. Because the correlation matrix of any complex data set is Hermitian all its eigenvalues are real [1].

II. The Complex Training Rule

Consider a complex data set composed of a set of zero-mean, complex vectors. This data set constitutes the neural network training set. The goal is to extract the principal components of the data set.

Figure 1 shows our neural network composed of p complex input nodes and a single output layer containing m complex linear neurons. Let $X(n)$ represents the n-th vector of the training set. The presentation of the vector $X(n)$ to the network constitutes the iteration n. The presentation of one complete training set constitutes one epoch.

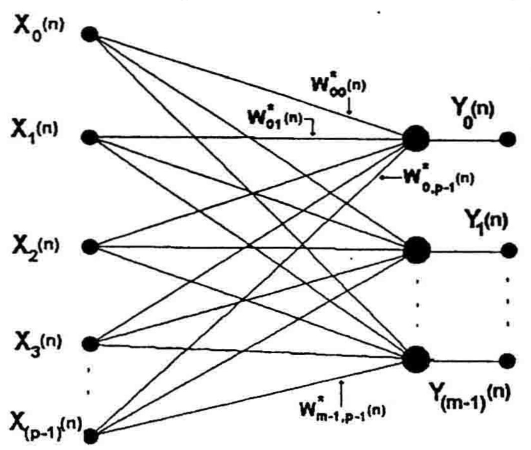

Fig. 1. The Complex Valued Neural Network Architecture.

For the n-th complex vector $X(n)$ presented to the neural network, the complex output value $Y_j(n)$ of neuron j is given by equation (1).

$$Y_j(n) = \sum_{i=0}^{p-1} W^*_{ji}(n) X_i(n) \tag{1}$$

$$j = 0, 1, \ldots, m-1$$

Electrical Engineering Department, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), 90619-900 - Porto Alegre - RS - Brazil. Emails: cristina@ee.pucrs.br, decastro@ee.pucrs.br, amaral@ee.pucrs.br, and pfranco@ee.pucrs.br. This work was in part supported by a grant from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and by PUCRS.

0-7803-4859-1/98 $10.00©1998 IEEE

where $W_{ji}(n)$ is the complex weight of the synapse that connects the i-th input node to the j-th output neuron at iteration n and * denotes the complex conjugate operator.

The synapse weight vector $W_j(n)$, associated with neuron N_j, is updated according to

$$W_{ji}(n+1) = W_{ji}(n) + \eta Y_j^*(n)[X_i(n) - \sum_{k=0}^{j} W_{ki}(n)Y_k(n)] \quad (2)$$

After achieving convergence for all neurons, we have the m eigenvectors represented by the synapse vectors $\mathbf{W}_j$ and the m neuron output values $Y_j(n)$ that represents the principal components. Observe that the weight update (equation 2) contains operations that cannot be directly performed with the connections shown in the network representation of Fig. 1.

III. Properties of the CGHA

To understand how the single layer linear neural network trained by our complex valued version of the Generalized Hebbian Algorithm performs principal component extraction, consider the architecture shown in Figure 1. For simplicity, assume that the neural network is formed by a single neuron N_0. The pre-synaptic signal $X(n)$, the post-synaptic signal $Y_0(n)$ and the synapse weight $W_0(n)$ are complex valued. The neuron output $Y_0(n)$ for the iteration n, due to the input vector $X(n)$ is given by

$$Y_0(n) = X^T(n)W_0^*(n) = W_0^*(n)X^T(n) \quad (3)$$

As in the Hebbian rule, in the CGHA the synapse weight $W_0(n)$ is modified based on the correlation between the pre-synaptic signal $X(n)$ and the post-synaptic signal $Y_0(n)$. From equation (2), and with $\Delta W_0(n) = W_0(n+1) - W_0(n)$, the synapse weight update is given by

$$\Delta W_0(n) = \eta \left\{ Y_0^*(n)X(n) - |Y_0(n)|^2 W_0(n) \right\} \quad (4)$$

where the positive constant η determines the learning rate. The term $|Y_0(n)|^2 W_0(n)$ is the complex equivalent to the Oja deflation term proposed to stabilize the algorithm [2], [11].

At convergence the expected change in the synapse weights is zero, i.e., $\langle \Delta W_0(n) \rangle = 0$. Thus, taking the expected value of both sides of equation (4), and using equation (3) we obtain

$$\langle \eta[X(n)X^H(n)W_0(n) - W_0^H(n)X(n)X^H(n)W_0(n)W_0(n)] \rangle = 0 \quad (5)$$

Notice that in equation (5) η is deterministic, the vectors $X(n)$ and $W_0(n)$ are statistically independent and $\langle X(n)X^H(n) \rangle$ is the covariance matrix C_x. Thus, we can rewrite equation (5) as

$$C_x \mathbf{q}_0 = (\mathbf{q}_0^H C_x \mathbf{q}_0)\mathbf{q}_0 \quad (6)$$

where $\mathbf{q}_0 = \langle W_0(n) \rangle$ is a constant vector. Now, let $(\mathbf{q}_0^H C_x \mathbf{q}_0) = \lambda_0$, where λ_0 is a scalar. Therefore,

$$C_x \mathbf{q}_0 = \lambda_0 \mathbf{q}_0 \quad (7)$$

From equation (7), $\mathbf{q}_0$ is an eigenvector of the correlation matrix C_x and λ_0 is the associated eigenvalue. In the case of several neurons, due to the deflation of the input data X (second term of equation 2), the synapses vector of each neuron will converge to the respective eigenvector of C_x. The synapse vector of the p-th neuron converges to the eigenvector associated with the p-th highest eigenvalue of C_x. Thus, after the convergence of the CGHA, the synapse weight vectors of the neural network represent the complex eigenvectors of the input data set correlation matrix C_x.

IV. The Training Window

When the complex valued algorithm presented in section II is applied to the network of Figure 1 to perform principal component analysis, a neuron N_j enters into the final stage of the convergence process towards the associated eigenvector only after the convergence of the neuron N_{j-1} [2], [10]. After the synapses of a particular neuron have converged to the respective eigenvector, the algorithm should not perform any further synapse updates on that neuron. Because only few neurons, those immediately adjacent to the converging neuron N_c, will be near the convergence point, the updating of the synapse weights of the whole set of m neurons leads to unnecessary computations.

To avoid updating the synapses of all neurons that are not ready for convergence yet, we propose a new algorithm that we call Training Window Algorithm. The goal of this algorithm is to reduce the computational cost of the CGHA training[1]. This goal is achieved by applying the CGHA training only to $W_s < m$ neurons, where W_s is the training window size. For instance, if neuron N_c is the first neuron in the training window, the CGHA is applied only to neurons $N_c, N_{c+1}, \ldots, N_{c+W_s-1}$. Once the neuron N_c has converged, the window is slid down by incrementing the value N_c. This process goes on until all m neurons have converged.

In [4] we give an expression for the reduction of the computational complexity of the training when the Training Window Algorithm (TWA) is used. This reduction of complexity γ is defined as the ratio of the CGHA complexity using TWA to the CGHA complexity without TWA. γ is a function of the window size W_s, the number of neurons m and the number of inputs p. Table I shows the complexity reduction γ for different window sizes used to train a neural network with 32 neurons and 64 input nodes.

When the weight vector W_j of neuron N_j have converged to the eigenvector $\mathbf{q}_j$, additional training will not change

[1] Notice that this training method is also suited to the real valued GHA

W_s	32	16	8	4
γ	1	0.726	0.414	0.219

TABLE I

Training Window Algorithm Complexity Reduction Factor γ as a function of W_s, $p = 64$ and $m = 32$.

the norm $||W_j||$ [10]. In our experiments we assume that the weight vector of a given neuron N_j has converged if the change in $||W_j||$ averaged over three epochs is less than 0.1 %.

V. Experimental Results

In this section we present experimental results from the application of our complex learning rule to image compression [2], [3]. The training set is obtained from the spectrum of the image "Lenna", without the redundant conjugate spectral components. Discarding the principal components and complex eigenvectors associated to the smallest eigenvalues and storing those associated with the largest eigenvalues we can achieve data compression with reduced information loss [2], [13], [12], [10].

We start with a 128×128 pixel image with 256 levels of grey. First the pixel values are normalized to the interval $[0, 1.0]$, and then the image is converted to the spectral domain through a two-dimensional Discrete Fourier Transform. We discard the conjugate spectral components and divide the remaining half spectrum into 128 small 8×8 frames. Each frame is read from left to right and top to bottom, resulting in one 1×64 training vector. The complex data set mean vector $\overline{X}$ is subtracted from each training set vector before the training process.

We use a neural network with $p = 64$ input nodes and $m = 32$ neurons. Both the real and imaginary parts of the synapse weights of the neural network are initialized with random numbers generated uniformly in the interval $[-1.0, 1.0]$. The initial learning rate η is set to 1×10^{-9}. At the end of each epoch we update η according to the inverse of the largest eigenvalue [2], [3], [5]. The neural network training is performed by presenting the training set for several epochs until the synapses converge to the eigenvector. It is important to note that the training set is shuffled at the end of each epoch.

After achieving convergence for all 32 neurons, we store the compressed image spectrum defined by the 64×32 matrix Q formed by the complex eigenvectors, the 1×64 complex mean vector $\overline{X}$ and the 128×32 matrix Y formed by the neural network output to each training vector. The compression ratio obtained is 0.76 with no additional techniques, such as entropy coding. To decompress the image we obtain the estimated half spectrum $\widehat{X} = YQ^H$, add $\overline{X}$ to each 1×64 vector of $\widehat{X}$, restore the conjugate spectral components and apply the inverse two dimensional Discrete Fourier Transform.

Figure 2 is the original image and Figure 4 is the respective decompressed images using the new complex valued algorithm. In Figure 3 we present the eigenvalues obtained with the real valued algorithm (identified as GHA in the figure), and with the complex valued algorithm (identified as CGHA). The real valued Generalized Hebbian Algorithm is applied to the original image represented in the space domain. Notice that the eigenvalue decreasing ratio is higher for the complex valued algorithm than for the real valued one, which implies that for the image in Figure 2, the complex valued algorithm concentrates more energy in the first eigenvalues than the real valued one. We might infer that the new algorithm will retain more information than the original Generalized Hebbian Algorithm for the same number of principal components stored.

For images with small details and high contrast regions the new complex valued algorithm yields a higher Peak Signal to Noise Rate than the original real valued algorithm. Such images have a broad and smooth spectrum [9], which implies in a high correlation between the frames in the spectral representation of the image. In images with less contrast, such as "Lenna," both algorithms yield similar Peak Signal to Noise Rates. Another advantage of the complex valued algorithm proposed for image compression is that the use of frames in the frequency domain prevents the blocking effect, often present in decompressed images[9], [10].

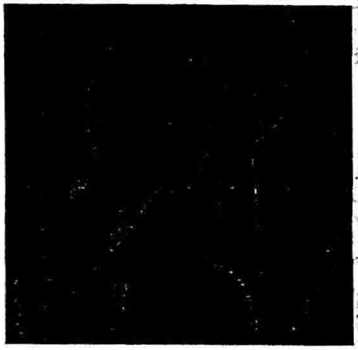

Fig. 2. .Original image Lenna.

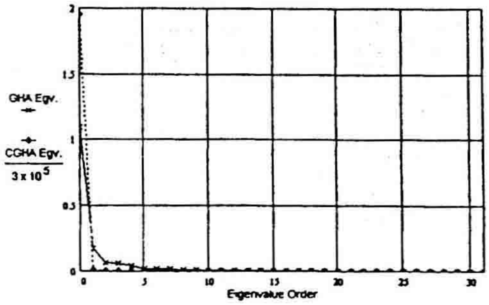

Fig. 3. GHA and CGHA eigenvalues distribution of the image in Fig. 2

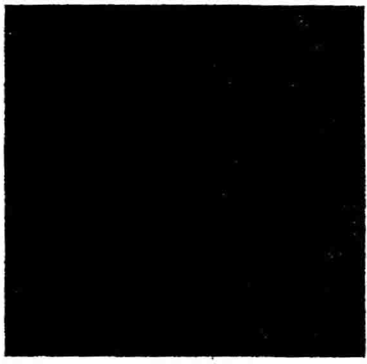

Fig. 4. Decompressed image of Fig. 2 using CGHA (32 eigenvectors). The PSNR of this image is 37.8dB. The PSNR of the GHA decompressed image is 37.3dB.

VI. Conclusion

Our Complex Valued Generalized Hebbian Algorithm can be used to extract the principal components of a complex valued data set. Although we demonstrated our method with an image compression application, complex PCA can be applied to other classes of signal processing problems such as seismic analysis, radar and sonar signal processing and communication systems. We also present a new training method for the single layer linear network that can be used in both the complex valued and the real valued Hebbian Learning Algorithm.

References

[1] R. Bronson. *Matrix Methods: An Introduction.* Academic Press Inc, 1991.

[2] M. C. F. De Castro. Algoritmo hebbiano generalizado para extração dos componentes principais de um conjunto de dados no domínio complexo. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, RS, BRAZIL, June 1996.

[3] M. C. F. De Castro, F. C. C. De Castro, J. N. Amaral, and P. R. G. Franco. Uma formulação complexa para o algoritmo hebbiano generalizado aplicada à compressão de imagens. In *III Simpósio Brasileiro de Redes Neurais*, pages 55–62, Recife, PE, BRAZIL, November 1996.

[4] M. C. F. De Castro, F. C. C. De Castro, J. N. Amaral, and P. R. G. Franco. A new training algorithm to reduce the computational complexity of principal component analysis by hebbian learning. In *III Congresso Brasileiro de Redes Neurais*, pages 7–11, Florianopolis, SC, BRAZIL, July 1997.

[5] L. H. Chen and S. Chang. An adaptive learning algorithm for principal component analysis. *IEEE Trans. Neural Net.*, 6(5), 1995.

[6] S. Chen, S. McLaughlin, and B.Mulgrew. Complex-valued radial basis function network, part i: Network architecture and learning algorithms. *Signal Processing*, 35:19–31, 1994.

[7] S. Chen, S. McLaughlin, and B.Mulgrew. Complex-valued radial basis function network, part ii: Application to digital communications channel equalization. *Signal Processing*, 36:175–188, 1994.

[8] S. Bannour e M. R. Azimi-Sadjadi. Principal component extraction using recursive least squares learning. *IEEE Trans. Neural Net.*, 6(2), 1995.

[9] R. Gonzales and R. E. Woods. *Digital Image Processing.* Addison Wesley, 1993.

[10] S. Haykin. *Neural Networks.* Macmillan College, New York, NY, 1994.

[11] E. Oja. A simplified neuron model as a principal component analyzer. *Mathematical Biology*, 1982.

[12] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 92.

[13] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 85.

[14] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 12:459–473, 1989.

[15] L. Xu and A. L. Yulle. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. Neural Net.*, 6(6), 1995.

Rescaled Simulated Annealing

L. Hérault, CEA-LETI, Grenoble, FRANCE

Abstract—**This paper presents a new metaheuristics called rescaled simulated annealing (RSA). It is based on a generic modification of the Metropolis procedure inside the simulated annealing (SA) algorithm. This modification consists in rescaling, before applying the Metropolis criterion, the energies of the states candidate to a transition . The direct consequence is an acceleration of convergence, by avoiding to dive and escape from high energy local minima. Asymptotic results are established and favorably compared to the famous ones due to Mitra and al. for SA [13]. Some practical implementations are presented for the Traveling Salesman Problem and results are compared to those obtained with SA. Less transitions need to be tested with RSA to obtain results of similar quality. As a corollary, within a limited computational effort, RSA provides better quality solutions than SA.**

Keywords—**Combinatorial optimization, meta-heuristics, rescaled simulated annealing, simulated annealing, Metropolis criterion, asymptotic results, recursive neural networks, traveling salesman problem.**

I. Introduction

SEVERAL metaheuristics have been developed over the past 15 years to tackle hard combinatorial optimization problems or learning in neural networks. The *simulated annealing* (SA) is one of the most popular because of its ease of use and of the asymptotic results of convergence to optimal solutions. It is extensively described in [2]. Unfortunately, it is still too slow to be convenient in many applications. Some current attempts to practically overcome these limitations are presented in [6].

Some involve the use of a modified cooling schedule. The most popular ones are the geometric reduction of the temperature and a cooling schedule due to Lundy and al. [12], but more elaborate cooling schedules have been proposed, as reported by Aarts and al. [2],

Other attempts are based on parallel processing [1], [4], [15], [8].

Some other uses a modified Metropolis procedure inside SA. Anily and al. [3] have proposed the use of a biased version of the Metropolis procedure due to Hastings [10], to improve the computational performances while keeping desired asymptotic properties. This algorithm has been used in some applications but needs a good a priori knowledge of the energy landscape [5].

We propose in this paper a new algorithm, also based on a modified Metropolis procedure, called *rescaled simulated annealing* (RSA), which needs less computational efforts than SA and no a priori knowledge of the energy landscape. Asymptotic results are established and favorably compared to the ones developed by Mitra and al. concerning SA [13].

The idea of RSA is to modify the Metropolis criterion in order to be "patient" during the search. In fact, instead of diving into low energy states as soon as possible as SA does (and thus loosing a lot of transitions to escape from high energy local minima), at each temperature step, the local search evolves in an energy "slice", around a target energy. A Monte-Carlo procedure let the system evolve freely towards a stationary distribution such that the most probable states are around a target energy.

The paper is organized as follows. Section II describes RSA. Section III presents asymptotic results of the algorithm. Section IV presents experimental results and comparisons between SA and RSA, applied to the traveling salesman problem.

II. Description of the algorithm

A. Definitions

The following definitions are used in this paper:

- Υ: set of states
- For any state $i \in \Upsilon$, its associated energy E_i is supposed positive.
- E_{opt}: the least energy; $E_{opt} \triangleq \min_{j\in\Upsilon} E_j$
- Υ_{opt}: subset of Υ containing the optimal states.
- $P_{ij}(k-1,k)$: probability that the outcome of the k-th trial is j, given that the outcome of the $k-1$-th trial is i. Also called *transition probability*. $P = (P_{ij})$ is the *transition matrix*.
- $\tau_1(P)$: coefficient of ergodicity of a matrix P:

$$\tau_1(P) \triangleq \frac{1}{2}\max_{i,j}\left\{\sum_{l=1}^{n}|P_{il}-P_{jl}|\right\} \qquad (1)$$

Obviously, $\tau_1(P) < 1$.

- Let c_k be the temperature at the k-th trial. The transition probability is defined by:

$$P_{ij}(k-1,k) \triangleq \begin{cases} G_{ij}(c_k)A_{ij}(c_k) & \forall j \neq i \\ 1-\sum_{l\neq i} G_{il}(c_k)A_{il}(c_k) & j = i \end{cases} \qquad (2)$$

where $G_{ij}(c_k)$ is the *generation probability* of state j from state i, and $A_{ij}(c_k)$ is the *acceptance probability* of state j, once it has been generated from i. If c_k, $(k = 0, 1, 2, \ldots)$ is kept constant, the corresponding Markov chain is homogeneous. Otherwise, it is in-homogeneous.

- Υ_i: neighborhood of state i; $\Upsilon_i \triangleq \{j \in \Upsilon | G_{ij} \neq 0\}$

B. Rescaled Simulated Annealing

Let us define the following criterion to decide whether a transition from state i to state j is accepted.

This criterion is the Metropolis criterion where the energy E_i of any state i has been rescaled:

$$E_i \leftarrow \left(\sqrt{E_i} - \sqrt{E_{target}}\right)^2 \qquad (3)$$

Address of the author: CEA-LETI (Systems Dpt.), CEA-G, 17 rue des Martyrs, F38054 Grenoble Cedex 9. E-mail: Laurent.Herault@cea.fr

0-7803-4859-1/98 $10.00©1998 IEEE

Algorithm 1 Metropolis criterion with rescaled energies around E_{target}

```
Compute
    $\Delta E_{i\rightarrow j} = (\sqrt{E_j} - \sqrt{E_{target}})^2 - (\sqrt{E_i} - \sqrt{E_{target}})^2$
    $A_{ij}(c) = \begin{cases} 1 & \text{if } \Delta E_{ij} \leq 0 \\ \exp(-\Delta E_{i\rightarrow j}/c) & \text{otherwise} \end{cases}$
```

To illustrate the rescaling of the energy landscape, figure 1 gives a trivial example of a 1-D function to be minimized and figure 2 shows the deformation of the energy landscape as a function of the target energy E_{target}. When the energy target is 0, the rescaled energy landscape is the original one. At high target energies, the minima of the rescaled energy landscape correspond to the maxima of the original function. Thus, if initially E_{target} is high, the most probable states at the beginning of the search are the crests of the energy landscape. As E_{target} gets smaller and smaller, the rescaled energy landscape converges towards the original energy landscape. More precisely, $\forall(i,j) \in \Upsilon^2$:

$$\left\{E_i < E_j \text{ and } E_{target} < \min_{i\in\Upsilon} E_i\right\}$$
$$\Rightarrow \left(\sqrt{E_i} - \sqrt{E_{target}}\right)^2 < \left(\sqrt{E_j} - \sqrt{E_{target}}\right)^2$$

As a consequence, if the target energy is smaller than the least energy, then the minima of the rescaled energy landscape are also minima in the original energy landscape and the optimal states of the problem with rescaled energies are also optimal in the original energy landscape. Moreover, one notices that the local minima of the rescaled energy landscape are shallower than in the original landscape. Thus, it is less costly to escape from the local minima of the rescaled energy.

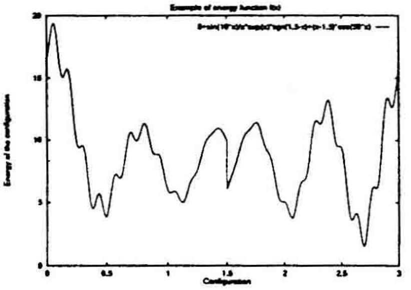

Fig. 1. Example of energy function to be minimized: $f(x) = 8 + \frac{\sin(x)}{x}\exp(x)\text{sign}(1.5 - x) + (x - 1.5)\cos(50x)$.

The energy E_{target} is defined as a function of the parameter c that decreases as c decreases. Choosing E_{target} as proportional to the square of the c parameter leads to have interesting asymptotic properties. In other words:

$$E_{target}(c) \triangleq a.c^2 \tag{4}$$

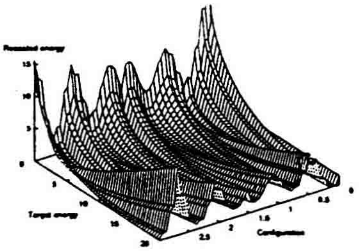

Fig. 2. Deformation of the f(x) as a function of the target energy.

The parameter a can be experimentally determined from the initial temperature step.

The rescaled SA algorithm can be defined as the SA algorithm where the energies of the states are rescaled before the evaluation of the transition energies (see algorithm 2).

Algorithm 2 Rescaled Simulated Annealing

```
Let $c = c_0$,
Let $a \approx \frac{\sum_{i\in\Upsilon} E_i}{|\Upsilon|c_0^2}$.
while "stopping criterion" is not satisfied do
    while "inner loop criterion" is not satisfied do
        If $i$ is the current state, generate a state $j$ in the neighborhood of $i$.
        Compute $\Delta E = E_j - E_i$.
        Rescale of the energy variation:
            $\Delta E := \Delta E - 2\sqrt{ac}\left(\sqrt{E_i + \Delta E} - \sqrt{E_i}\right)$
        Accept the transition with the probability $A_{ij}$
            $A_{ij}(c) = \begin{cases} 1 & \text{if } \Delta E \leq 0 \\ \exp(-\Delta E/c_m) & \text{otherwise} \end{cases}$
    end while
    Update the temperature parameter
        $c_{m+1} := Update(c_m)$
    $m := m + 1$
end while
```

The "inner loop criterion" determines how many steps are taken by the algorithm at a given temperature. The parameter a can be chosen so that the initial target energy is near the mean energy of the states. Practically, we have verified in section IV that this value leads to good results and that the performances do not depend a lot on a fine tuning of a.

All the following results hold for any update function of the parameter c (in algorithm 2) in which:

$$\forall m \geq 0, \quad c_m > c_{m+1} \tag{5}$$

$$\lim_{m\to\infty} c_m = 0 \tag{6}$$

III. Theoretical considerations

As with SA, two formulations of the algorithm can be distinguished:

- an *homogeneous algorithm*: the algorithm is described by a sequence of homogeneous Markov chains. Each Markov chain is generated at a fixed value of the temperature parameter and this parameter is decreased in between subsequent Markov chains, and
- an *in-homogeneous algorithm*: the algorithm is described by a single in-homogeneous Markov chain. The value of the temperature parameter is decreased in between subsequent transitions.

We can establish asymptotic properties similar to SA. They are extensively detailed in [11] and only the main results are outlined in this section.

A. The homogeneous RSA

The statistical properties of any homogeneous Markov chain are the ones of the homogeneous SA, but the energies of the states have been rescaled according to equation 3. As a consequence, the Markov chain associated with the probabilities $P_{ij}(c)$ is homogeneous, irreducible and aperiodic [2]. Thus, from [7], it converges towards a stationary distribution $\mathbf{q}$ uniquely given by:

$$\forall i \in \Upsilon, \quad q_i(c) = \frac{\exp(-(\sqrt{E_i} - \sqrt{E_{target}})^2/c)}{\sum_j \exp(-(\sqrt{E_j} - \sqrt{E_{target}})^2/c)} \qquad (7)$$

In other words, this criterion let the system evolve towards a stationary distribution where the most probable states are those with energy close to E_{target}.

It can be established that if the update function of c satisfies equation 6. Then, $\lim_{c \to 0} \mathbf{q}(c) = \pi$, where:

$$\forall i \in \Upsilon, \quad \pi_i = \begin{cases} |\Upsilon_{opt}|^{-1} & \text{if } i \in \Upsilon_{opt} \\ 0 & \text{elsewhere} \end{cases}$$

This result can be interpreted as the convergence of the algorithm to an optimum state provided that an infinite number of transitions are taken at each value of m, so that the stationary distribution is reached.

A.1 Monotonicity of the stationary probabilities

The main results obtained in the standard homogeneous SA [13] can be extended to this new algorithm with only slight differences. Moreover, the following theoretical results are used to establish asymptotic results of the in-homogeneous algorithm.

To summarize, if the target energy is proportional to the square of the temperature, then the stationary probabilities of least-energy states monotonically increase with decreasing temperature and the increasing rate is higher than with SA. For states of energy greater than $\bar{E}$, where $\bar{E} = \frac{\sum_j E_j \exp(2\sqrt{aE_j})}{\sum_j \exp(2\sqrt{aE_j})}$, the opposite is true. Each state i with an energy greater than least-energy and smaller than $\bar{E}$ has an associated "critical temperature" $\hat{c}_i$; while the temperature c is greater than $\hat{c}_i$, the state's stationary probability increases with decreasing temperature, and for temperatures lower than $\hat{c}_i$, the opposite is true. Furthermore, the critical temperature is an increasing function of energy.

A.2 Analysis of the quasi-homogeneous distributions

Practically, a finite number of transitions are tested at each temperature step and a quasi-homogeneous algorithm is often implemented. We estimate the speed of convergence of the current probability state vector to the stationary distribution $\mathbf{q}$ at each temperature and compare it with the one computed for SA.

Let $\nu(c,m) \triangleq [\nu_1(c,m), \nu_2(c,m), \cdots, \nu_{|\Upsilon|}(c,m)]$ be the state probability vector after m transitions of the Markov chain at temperature c. We have established that after m transitions at temperature c:

- in the case of RSA: $\|\nu(c,m) - \mathbf{q}(c)\| = O({e_{RSA}}^m)$
- in the case of SA: $\|\nu(c,m) - \mathbf{q}(c)\|_{a=0} = O({e_{SA}}^m)$

where e_{RSA} (resp. e_{SA}) is the coefficient of ergodicity of the transition probability matrix at temperature c (equation 1). Moreover, if $c \leq \min_{i \in \Upsilon}\{\sqrt{E_i}/\sqrt{a}\}$ (i.e. if the target energy is smaller than the least energy), then $e_{RSA} < e_{SA}$.

This result implies that the convergence to stationary distribution at any small enough temperature is estimated as faster with RSA than with SA. Thus, the error made, at each temperature, on the stationarity of the distribution after a finite number of transitions is estimated smaller with RSA than with SA as soon as the target energy is smaller than the least energy. Thus, within a limited number of tested transitions at each temperature, RSA reaches a quasi-stationary distribution estimated closer to the stationary distribution than SA does. This justifies a best quality of solution with RSA within a limited number of tested transitions, as experimentally verified in section IV.

B. The in-homogeneous RSA

We now consider that the algorithm can be described by a single in-homogeneous Markov chain, whose transition probability matrix $P(k-1,k)$ ($k = 1, 2, \ldots$) is defined by equation 2.

By analyzing our algorithm with tools similar to those used in [13], sufficient conditions for convergence to an optimal state can be established and compared with the ones proposed for SA in [13]. In this paper, authors define r as:

$$r = \min_{i \in \Upsilon \setminus \Upsilon_{max}} \max_{j \in \Upsilon} d(i,j) \qquad (8)$$

r is the *radius* of the neighborhood graph, where $d(i,j)$ is the minimal number of transitions to reach j from i and Υ_{max} is the set of locally maximal states. They have established a cooling schedule which ensures the convergence to optimal configurations. It is given by, for $k = 0, 1, 2, \ldots$:

$$c_k \geq \frac{\max_{i \in \Upsilon, j \in \Upsilon_i}(r|E_i - E_j|)}{\log(k + m_0)} \qquad (9)$$

With RSA, if the annealing schedule satisfies, for any

$k = 0, 1, 2, \ldots$

$$c_k \geq \max_{i\in\Upsilon, j\in\Upsilon_i} \left\{ \frac{r|E_i - E_j|}{\log(k + m_0) + 2\sqrt{a}r|\sqrt{E_i} - \sqrt{E_j}|} \right\} \tag{10}$$

where m_0 is any parameter satisfying $2 \leq m_0 < \infty$, then, similarly to SA:

$$\forall m, \forall i, j \in \Upsilon \quad \lim_{k\to\infty} P_{ij}(m, k) = \pi_j. \tag{11}$$

One notices that equations 9 and 10 are identical when $\sqrt{a} = 0$ (i.e., when RSA=SA).

We can compare the cooling schedules between SA and RSA. The lower bound obtained for temperature annealing in RSA (equation 10) is smaller than the bound given in [13] (equation 9).

An immediate consequence of this theorem is that the decreasing of the temperature parameter can be faster with RSA than with SA.

IV. Experiments

The algorithm has been tested on several combinatorial optimization problems. All the conclusions are similar to the ones obtained from tests on the Traveling Salesman Problem (TSP). This problem is a benchmark in literature and has been extensively studied in [14]. In order to highlight the main properties of RSA, we have compared the performances of a standard SA with a standard RSA algorithm. A RSA algorithm is obviously derived from SA by modifying the Metropolis criterion, as given in algorithm 1.

In the algorithms, the state transition that is used is defined as follows:

- Pick at random two cities.
- Exchange their positions in the current tour.

A. Performance evaluation

The algorithms have been evaluated and compared in terms of:

- Quality of the solution, i.e. smallest found energy in a run.
- Number of tested transitions to reach the smallest found energy in a run.
- Quality of the solution in a limited number of tested transitions.
- The solution quality's sensitivity to the tuning of the parameters.
- Computational effort.

We have chosen to compare a standard quasi-homogeneous SA with a simple cooling schedule with its RSA counterpart. Practically, for real applications, more elaborate cooling schedules could be used (see [2]). Thus, a quasi-homogeneous RSA with a geometric reduction of the temperature has been extensively tested. The algorithm is given in algorithm 3. The tested SA is exactly the same algorithm as RSA except two lines (underlined in algorithm 3) corresponding to the initialization of the parameter a and to a rescaling of the energy variation.

Except when specified, the parameters of SA and RSA in the following are exactly the same: $p = 90\%$, $dec_c = 0.99$.

Algorithm 3 Quasi-homogeneous RSA for the TSP

```
Set $c = c_0$ such that $p\%$ of the tested transitions are accepted.
Set $Nb_{maxtest}$ and $dec_c$.
Let $a \approx \frac{\sum_{i\in\Upsilon} E_i}{|\Upsilon|c_0^2}$.
Generate a random tour.
Compute the length $L$ of the tour.
Let $Nb_{accepted} = 1$
while $Nb_{accepted} \neq 0$ do
  $Nb_{tested} = 0$, $Nb_{accepted} = 0$
  while $Nb_{tested} < Nb_{maxtest}$ do
    $Nb_{tested} := Nb_{tested} + 1$
    Pick at random two different towns.
    Compute the energy variation $\Delta E$ associated with an exchange of their positions on the tour.
    Rescale the energy variation:
        $$\Delta E := \Delta E - 2\sqrt{ac}\left(\sqrt{L + \Delta E} - \sqrt{L}\right)$$
    if $\Delta E < 0$ then
      Exchange their positions on the tour.
      $L := L + \Delta E$
      $Nb_{accepted} := Nb_{accepted} + 1$
    else
      Generate a random number $rand \in [0, 1]$.
      if $rand < \exp\left(-\frac{\Delta E}{c}\right)$ then
        Exchange their positions on the tour.
        $L := L + \Delta E$
        $Nb_{accepted} := Nb_{accepted} + 1$
      end if
    end if
  end while
  Update the temperature parameter $c := dec_c.c$.
end while
```

In all the tests, with each set of parameters, the algorithms have been rerun 100 times with different initializations of the random generator.

Let us for instance consider the 101 cities TSP problem *eil101* referenced in [14]. Figure 3 visualizes the optimal tour of this problem. The optimum tour has a length of 629. Figures 4 and 5 show, for SA and RSA:

- the evolution of the quality of the current smallest found energy as a function of the number of tested transitions, with SA and RSA ($Nb_{tested} = 500000$).
- the mean energy of the visited states on each temperature step (computed on $Nb_{maxtest}$ tested transitions), and its standard deviation. The mean energy is computed in the following way:

$$< E >= \frac{\sum_i E_i.\exp\left(-(\sqrt{E_i} - \sqrt{ac})^2/c\right)}{\sum_i \exp\left(-(\sqrt{E_i} - \sqrt{ac})^2/c\right)}$$

- the target energy in RSA.

The mean energy of the visited states with RSA follows the target energy till the target energy is smaller than the least energy. One also notice that the number of transitions to reach the best found state is much smaller with RSA.

B. Less transitions are needed to obtain a good solution

Figure 6 shows the number of transitions needed to reach the best found state of the TSP problem "eil101", as a func-

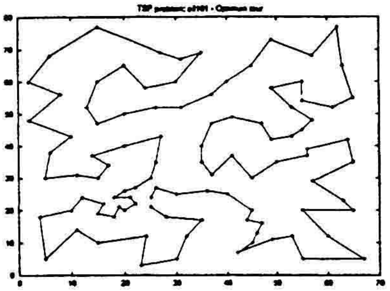

Fig. 3. "eil101" TSP problem: an optimum state. The associated energy is 629.

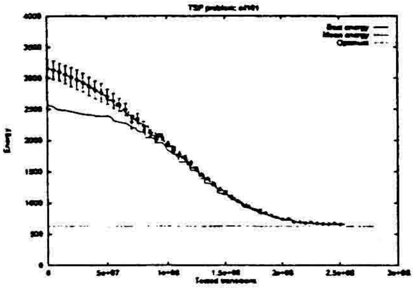

Fig. 4. SA: Evolution of the smallest current energy and of the mean energy. In this experiment, $dec_c = 0.99$, $Nb_{maxtest} = 500000$, $p = 90\%$. The best found state has an energy of 659,48 after 254,2 10^6 tested transitions, corresponding to 3692 seconds on a SUN Ultrasparc.

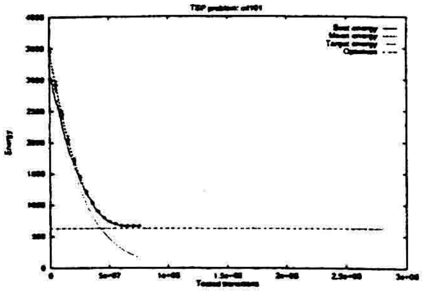

Fig. 5. RSA: Evolution of the smallest current energy and of the mean energy. In this experiment, $dec_c = 0.99$, $Nb_{maxtest} = 500000$, $p = 90\%$. The target energy is also visualized. The best found state has an energy of 665, 406 after 66, 5 10^6 tested transitions, corresponding to 1209 seconds on a SUN Ultrasparc.

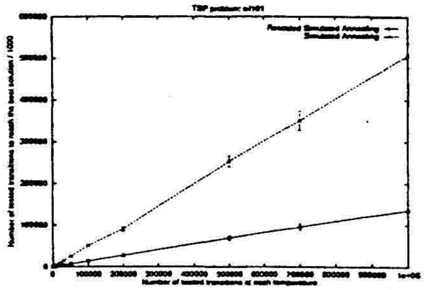

Fig. 6. "eil101" TSP problem. Log-Log diagram of the number of tested transitions to reach the best found energy as a function of $Nb_{maxtest}$, for SA (upper curve) and RSA (lower curve).

tion of the number of tested transitions on each temperature step ($Nb_{maxtest}$), when dec_c is fixed ($dec_c = 0.99$). Figure 7 shows the CPU time on a standard Sun Ultrasparc as a function of the number of tested transitions on each temperature step. Clearly, the computational effort needed by RSA is far smaller than the one needed by SA.

We experimentally verified that many fewer transitions need to be tested to reach the best found state when using RSA. This property of RSA should be nuanced by the extra computational cost induced by the rescaling of the energy variation. Nevertheless, on our SUN Ultrasparc Station, when considering the CPU time needed to reach the best found state, this extra computational cost is still within the computational benefit due to the small number of transitions needed.

Figures 8 and 9 show the gain of performance of RSA compared to SA. Less transitions (and less computational effort) is needed to obtain a given quality of solution.

C. Influence of a in equation 4

We have tested the influence of the a parameter in the rescaling process (equation 4), i.e. the influence of the initial target energy. If $a = 0$, then RSA is exactly SA. By approximating the distribution of the energies of the valid states by a gaussian distribution with a mean $\hat{E}$ and a standard deviation σ, one derives a good value of a:

$$a = \frac{\sum_{i \in \Upsilon} E_i}{|\Upsilon| c_0^2} \qquad (12)$$

where c_0 is the initial temperature, chosen high enough so that most of the tested transitions are accepted. In other words, the target energy is initially chosen around the mean energy of the states.

We have tested the sensitivity of the algorithm to the parameter a, i.e. to the initial target energy. Figures 10 and 11 visualizes the mean value of the smallest found energy as a function of the computational effort in the cases where the initial target energy is $\hat{E}$, $\hat{E} - 3\sigma$ and $\hat{E} + 3\sigma$. Clearly, the results are not sensitive to a fine tuning of a, as given by equation 12.

V. Conclusion and perspectives

This paper has presented a new metaheuristics, called rescaled simulated annealing (RSA), based on a generic modification of the Metropolis criterion. Some asymptotic properties are presented, using results obtained for discrete-time in-homogeneous Markov chains. This properties favorably compare with the ones established [13]. They express that within a limited computational effort, RSA provides better solutions than SA. This has been experimentally verified in this paper on the traveling salesman problem.