Stochastic Functional Annealing as Optimization Technique: Application to the Traveling Salesman Problem with Recurrent Networks*

Domingo López-Rodríguez[1], Enrique Mérida-Casermeiro[1], Gloria Galán-Marín[2], and Juan M. Ortiz-de-Lazcano-Lobato[3]

[1] Department of Applied Mathematics, University of Málaga, Málaga, Spain
{dlopez,merida}@ctima.uma.es
[2] Department of Electronics and Electromechanical Engineering, University of Extremadura, Badajoz, Spain
gloriagm@unex.es
[3] Department of Computer Science and Artificial Intelligence, University of Málaga, Málaga, Spain
jmortiz@lcc.uma.es

Abstract. In this work, a new stochastic method for optimization problems is developed. Its theoretical bases guaranteeing the convergence of the method to a minimum of the objective function are presented, by using quite general hypotheses. Its application to recurrent discrete neural networks is also developed, focusing in the multivalued MREM model, a generalization of Hopfield's. In order to test the efficiency of this new method, we study the well-known Traveling Salesman Problem. Experimental results will show that this new model outperforms other techniques, achieving better results, even on average, than other methods.

1 The Neural Model MREM

A powerful generalization of Hopfield's model appears in the works [11,12], where the model MREM (*Multivalued REcurrent Model*) is presented. This model (in its discrete version) presents two important characteristics which make it very versatile and increase its applicability:

- The output of each neuron, s_i, is a value from the set $\mathcal{M}=\{m_1, m_2, \ldots, m_L\}$, which is not necessarily numerical. The state vector of the net is defined as $\boldsymbol{S} = (s_1, \ldots, s_N)$.
- A new concept is introduced: a function f that measures the similarity between the outputs of the neurons. This is the so-called similarity function. So, $f(s_i, s_j)$ represents the similarity between outputs of neurons i and j.

* This work has been partially supported by Junta de Andalucía project number P06-TIC-01615.

J. Hertzberg, M. Beetz, and R. Englert (Eds.): KI 2007, LNAI 4667, pp. 397–411, 2007.

Thus, the energy function of this model, which characterizes the behaviour of the net, is as follows:

$$E(\boldsymbol{S}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j} f(s_i, s_j) + \sum_{i=1}^{N} \theta_i(s_i) \tag{1}$$

where $W = (w_{i,j})$ is the synaptic weight matrix, representing the interconnection strength between each pair of neurons, f is the similarity function, and $\theta_i \colon \mathcal{M} \to \mathbb{R}$ is a generalization of the thresholds of each neuron.

These characteristics make that certain optimization problems (including the ETSP) have better representation in this model than in the case of the binary or the bipolar model developed by Hopfield, as well as in other multivalued models.

It is clear that MREM generalizes Hopfield's models (with outputs $\mathcal{M} = \{-1, 1\}$ as well as $\mathcal{M} = \{0, 1\}$), if we consider the similarity function given by the product $f(a, b) = ab$.

This model has been successfully applied to several (combinatorial) optimization problems, achieving very good results, which were better than the best known results in literature [13,14,15].

The aim of this work is to present a technique that helps MREM (as well as other optimization techniques) to escape from local minima, improving so its efficiency, by means of a randomized technique that we call Stochastic Functional Annealing.

2 Stochastic Functional Annealing

The Functional Annealing method is designed to help optimization techniques to avoid some local minima of the objective function. In this case, the objective function F (defined over a discrete set V) is substituted by a sequence of functions $\{F_n\}$ such that $\lim_{n\to\infty} F_n(x) = F(x)$ for all $x \in V$. Then, an optimization technique is used to minimize each of the *approximating* functions F_n.

The point used as initial guess to minimize F_{n+1} is the minimum of F_n, which will be denoted as $x_*^{(n)}$. We iterate by using a stochastic algorithm which, under certain hypotheses, will lead to the minimization of F, although the minimization of F_n is not guaranteed.

This stochastic algorithm is as follows:

1. Begin with $m = 1$, $n = 1$.
2. Choose an initial guess $x_1^{(n)}$, randomly (if $n = 1$), or by taking $x_1^{(n)} = x_*^{(n-1)}$ (if $n > 1$).
3. Select a point $z \in V$ according to the law of probability $\mathbb{P}_{\mathrm{s}}(z; x_m^{(n)})$.
4. The update $x_{m+1}^{(n)} = z$ will be accepted with probability $\mathbb{P}_{\mathrm{a}}(\Delta F_n)$, where $\Delta F_n = F_n(z) - F_n(x_m^{(n)})$.
5. Repeat steps 3 and 4 until the acceptance of an update $x_{m+1}^{(n)}$. Let $m = m+1$.
6. If $m \geq K$ (for a fixed K), let $x_*^{(n)} = x_K^{(n)}$, $m = 1$ and $n = n + 1$ and return to step 2.
7. If $m < N$, return to step 3.

Thus, we obtain a sequence $\{x_*^{(n)}\}$ of points in V which are approaches to the minima of the corresponding F_n. In the limit, we expect that this sequence approaches a minimum of F.

Note that we are not making an infinite number of iterations in order to minimize F_n. We make only a finite number of them, which implies, due to the stochastic nature of the algorithm, that we are not arriving at the minimum of F_n.

We must observe that there are several possible ways of sampling a point $z \in V$. It does not need to be chosen completely at random. This sampling can be made by taking into account the function values of all points in a certain neighborhood of $x_m^{(n)}$, and assigning to every point a probability which is proportional to its increase of the function value.

In the next section we will impose some conditions to guarantee the minimization of F.

2.1 Convergence Theorems

At least two hypotheses have to be imposed in order to obtain some results of convergence:

Condition of Probabilistic Monotonicity
The next equality must hold

$$\lim_{n\to\infty} \mathbb{P}\left(F_n(x_{m+1}^{(n)}) > F_n(x_m^{(n)})\right) = 0$$

for all $m \in \{1, \ldots, N-1\}$.

Condition of Acceptance of the Update $z = x_{m+1}^{(n)}$
We consider the probability of acceptance of $x_{m+1}^{(n)}$ *as*

$$\mathbb{P}_{\mathrm{a}}(z) = \begin{cases} 1, & \text{if } \Delta F_n < 0 \\ g_n(\Delta F_n) < 1, & \text{if } \Delta F_n \geq 0 \end{cases}$$

where $g_n \colon \mathbb{R}^+ \to [0,1)$ *and* $\Delta F_n = F_n(z) - F_n(x_m^{(n)})$.

In addition, to simplify the algorithm, let us suppose the following condition:

Condition of Sampling in Neighbourhoods
Given $x = x_m^{(n)}$, *we consider the probability of sampling a point* $z \in V$ *given by*

$$\mathbb{P}(\text{sampling } z) = \mathbb{P}_{\mathrm{s}}(z) = \begin{cases} 0, & \text{if } z \notin \mathcal{N}_x \\ a(z) > 0, & \text{if } z \in \mathcal{N}_x \end{cases}$$

where $\mathcal{N}_x$ *is a neighbourhood of* x.

We must note that if the functional sequence $\{g_n\}$ tends to 0 uniformly, then the condition of acceptance of $z = x_{m+1}^{(n)}$ implies that of probabilistic monotony.

With these hypotheses in mind, we can get some technical results that guarantee the convergence of our algorithm to a minimum of F. However, due to the limitation in the length of this paper, proofs for these results will be omitted.

The following technical results (lemmas, propositions and theorems) establish some quite general conditions to ensure the convergence of our method. These conditions are simplified in the particular case of discrete recurrent networks, in the next section.

Theorem 1. *With probability 1, there exists L such that*

$$L = \lim_{n\to\infty} F_n(x_*^{(n)}) = \lim_{n\to\infty} F(x_*^{(n)})$$

Corollary 1. *If ξ is an accumulation point of the sequence $\{x_*^{(n)}\}$, then, with probability 1, the next equality holds:*

$$F(\xi) = \lim_{n\to\infty} F_n(x_*^{(n)}) = \lim_{n\to\infty} F(x_*^{(n)})$$

We now proceed to establish the optimality of the accumulation points of $\{x_*^{(n)}\}$.

Theorem 2. *Let $z \in V$ with $F(z) < L$, where*

$$L = \lim_{n\to\infty} F_n(x_*^{(n)}) = \lim_{n\to\infty} F(x_*^{(n)})$$

Then, there exists $N \in \mathbb{N}$ such that if $n \geq N$ then

$$\mathbb{P}_{\mathrm{s}}(z) \cdot \mathbb{P}\left(\text{accept } z = x_2^{(n)}\right) = 0$$

This technical result gives two interesting corollaries, dealing with the optimality of the accumulation points of $\{x_*^{(n)}\}$, and the convergence of this sequence.

Proposition 1. *Let ξ be an accumulation point of $\{x_*^{(n)}\}$. Then, we have $F(\xi) \leq F(z)$ for all $z \in \mathcal{N}_\xi$. Therefore we can affirm that ξ is a local minimum of F.*

Let us study the time required by this algorithm to visit, at least once, the global optimum x_*.

Suppose that $g_n(\Delta F_n) \geq \delta > 0$ for all n and for all possible $\Delta F_n = F_n(y) - F_n(x)$ with $x, y \in V$ and that $\mathbb{P}_{\mathrm{S}}(z; x) \geq \rho > 0$ for all $z \in \mathcal{N}_x$, for all $x \in V$.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph given by $\mathcal{V} = V$ and the edge $(x, y) \in \mathcal{E}$ if, and only if, $x \in \mathcal{N}_y$.

Suppose that $\mathcal{G}$ is strongly connected, that is, given two nodes (points of V), there always exists a path (sequence of nodes) connecting them by means of edges in $\mathcal{E}$.

Let D be the diameter of the graph $\mathcal{G}$, that is, the longest path between to nodes of $\mathcal{G}$.

Lemma 1. *Under the above hypotheses, given a random initial guess $x_1^{(1)} \in V$, the expected number of steps to visit x_* is less than or equal to $D\rho^{-D}\delta^{1-D}$.*

Proposition 2. *For all $k > 0$, our algorithm visits the global optimum x_* in less than $kD\rho^{-D}\delta^{1-D}$ steps with probability greater than $1 - \frac{1}{k}$.*

But we can say more:

Proposition 3. *For all $k > 0$, our algorithm visits the global optimum x_* in less than $2kD\rho^{-D}\delta^{1-D}$ steps with probability greater than $1 - 2^{-k}$.*

These results are applicable when the algorithm can use some memory to store data, that is, the algorithm is able to remember the best solution up to the current iteration (the *best-solution-so-far*).

When the algorithm has no memory, we look for other results proving convergence.

For the next results, which prove the convergence of the sequence $\{x_*^{(n)}\}$, we need an additional hypothesis on $\{g_n\}$, the sequence that appears in the definition of the probability of acceptance.

Definition 1. *A sequence $\{\varphi_n \colon [0,\infty) \to \mathbb{R}\}$ of functions converges ε-uniformly to $\varphi \colon [0,\infty) \to \mathbb{R}$ if, for all $\varepsilon > 0$, the sequence $\{\varphi_n \big|_{[\varepsilon,\infty)}\}$ of functions restricted to the interval $[\varepsilon,\infty)$ converges uniformly to $\varphi \big|_{[\varepsilon,\infty)}$, that is, if*

$$\lim_{n\to\infty} \sup_{t\geq\varepsilon} |\varphi_n(t) - \varphi(t)| = 0$$

It is clear that if a sequence converges uniformly, it also converges ε-uniformly, therefore it is a less restrictive hypothesis.

Proposition 4. *If local minima of F are strict, $\{g_n\}$ converges ε-uniformly to 0, and ξ is an accumulation point of $\{x_*^{(n)}\}$, then the next equality holds:*

$$\lim_{n\to\infty} \mathbb{P}\left(x_*^{(n+1)} = \xi | x_*^{(n)} = \xi\right) = 1$$

But we can still specify a little more:

Corollary 2. *Let us suppose that local minima of F are strict and that $\{g_n\}$ converges ε-uniformly to 0. Let ξ be an accumulation point of $\{x_*^{(n)}\}$ and $\{x_*^{(n_k)}\}$ one subsequence of $\{x_*^{(n)}\}$ such that $\lim_{k\to\infty} x_*^{(n_k)} = \xi$. Then*

$$\lim_{k\to\infty} \mathbb{P}(x_*^{(n_k+1)} = \xi) = 1$$

From this result we can deduce the convergence of the sequence $\{x_*^{(n)}\}$.

Theorem 3. *Let us suppose that local minima of F are strict and that $\{g_n\}$ converges ε-uniformly to 0. Let ξ be an accumulation point of $\{x_*^{(n)}\}$. Then*

$$\xi = \lim_{n\to\infty} x_*^{(n)}$$

Note that we have arrived at the strong convergence of the sequence $\{x_*^{(n)}\}$, not only convergence in probability, to ξ, which is a minimum of F.

3 Application to Recurrent Networks

In this section, we will specify the previous results in the case in which the optimization algorithm used to minimize each F_n is a recurrent neural network. Therefore, we will consider energy functions such as the one given in Eq. (1) for the MREM model.

3.1 Stochastic Functional Annealing Applied to MREM

Now, in order to minimize the energy function E given by Eq. (1), we will consider a sequence of energy functions $\{E_n\}$, defined as in Eq. (2):

$$E_n(\boldsymbol{S}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}^{(n)} f^{(n)}(s_i, s_j) + \sum_{i=1}^{N} \theta_i^{(n)}(s_i) \tag{2}$$

where $\{W^{(n)} = (w_{i,j}^{(n)})\}$ is a sequence of synaptic weights matrices verifying that $\lim_{n\to\infty} W^{(n)} = W$, $\{f^{(n)} \colon \mathcal{M} \times \mathcal{M} \to \mathbb{R}\}$ is a functional sequence such that the limit $\lim_{n\to\infty} f^{(n)}(x, y) = f(x, y)$ for all $x, y \in \mathcal{M}$, and $\{\theta_i^{(n)} \colon \mathcal{M} \to \mathbb{R}\}_{n\geq 1}$ is a sequence of threshold (or bias) functions for neuron i, $i = 1, \ldots, N$, which converges punctually to θ_i, for all i.

Obviously, $E(\boldsymbol{S}) = \lim_{n\to\infty} E_n(\boldsymbol{S})$ for each state vector $\boldsymbol{S} \in \mathcal{M}^N$.

For every state vector $\boldsymbol{S}$ we define a neighbourhood $\mathcal{N}_{\boldsymbol{S}}$ such that the next state of the net will be chosen from this neighbourhood.

Let us suppose that, to minimize E_n, we iterate and obtain $\boldsymbol{S}_m^{(n)}$ and make a sampling in its neighbourhood to arrive at the state vector $\boldsymbol{S}$. We define the probability of acceptance of the update $\boldsymbol{S} = \boldsymbol{S}_{m+1}^{(n)}$ as follows (analogous to what was made in the previous section):

$$\mathbb{P}_{\mathrm{a}}(\boldsymbol{S}) = \begin{cases} 1, & \text{if } \Delta E_n < 0 \\ g_n(\Delta E_n) < 1, & \text{if } \Delta E_n \geq 0 \end{cases}$$

where $g_n \colon \mathbb{R}^+ \to [0, 1)$ and $\Delta E_n = E_n(\boldsymbol{S}) - E_n(\boldsymbol{S}_m^{(n)})$.

Thus, for each n, given $\boldsymbol{S}_1^{(n)}$, we will obtain a finite sequence of state vectors $\{\boldsymbol{S}_i^{(n)}\}_{i=1,\ldots,K}$, since we only iterate K times, to arrive at the state vector $\boldsymbol{S}_K^{(n)}$ which will be denoted $\boldsymbol{S}_*^{(n)}$, in order to keep the notation introduced in the previous sections.

So, as a result of applying the previous technical results, we can affirm that:

Theorem 4. *With the previous hypotheses, we have:*

- *There exists a value L which is the common limit of the sequences $\{E_n(\boldsymbol{S}_*^{(n)})\}$ and $\{E(\boldsymbol{S}_*^{(n)})\}$.*
- *If $\boldsymbol{S}_*$ is an accumulation point of $\{\boldsymbol{S}_*^{(n)}\}$, then*

$$E(\boldsymbol{S}_*) = \lim_{n\to\infty} E_n(\boldsymbol{S}_*^{(n)}) = \lim_{n\to\infty} E(\boldsymbol{S}_*^{(n)})$$

and, in addition, $E(\boldsymbol{S}_*) \leq E(\boldsymbol{S})$ *for all state vector* $\boldsymbol{S}$ *belonging to the neighborhood* $\mathcal{N}_{\boldsymbol{S}_*}$ *of* $\boldsymbol{S}_*$.

- *If* $\{g_n\}$ *converges* ε*-uniformly to zero, and local minima of* E *are strict, then*

$$\boldsymbol{S}_* = \lim_{n \to \infty} \boldsymbol{S}_*^{(n)}$$

3.2 A Particular Case: The Stochastic MREM Model

If we consider $E_n = E$ for all n, we arrive at a stochastic version of MREM.

This version, which will be called sMREM (*stochastic MREM*), will converge to a state of (local) minimal energy, since it is a particular case of the Stochastic Functional Annealing, and it verifies the conditions of Probabilistic Monotonicity, Acceptance of Updates and Sampling in Neighborhoods.

The main drawback of this model sMREM, with respect to its deterministic version, is the amount of computational time needed to achieve that convergence. However, this increase in the computational effort obtains its reward as an increase in the quality of the solution, as we will see in the experimental results.

4 The Euclidean Traveling Salesman Problem

The Euclidean Traveling Salesman Problem (ETSP) is a classical and very well-known issue of study in the field of Operations Research, as well as in Artificial Intelligence, since it has become one of the most popular benchmarks to test the efficiency of optimization-related methods.

We can define this problem as follows: given N cities in the Euclidean space $X_1, \ldots, X_N \in \mathbb{R}^2$ and distances $d_{i,j}$ between each pair of cities X_i and X_j, the objective is to find the shortest closed path that visits each city only once.

Real-life applications cover aspects such as automatic route planning for robots, and hole location in printed circuits design [1], as well as gas turbine checking, machine task scheduling or crystallographic analysis [2], among others.

Despite the simplicity of this definition, this problem is one of the most typical representatives of the complexity class of NP-hard problems, showing its high level of difficulty in its resolution. Thus, there is need of algorithms to achieve good approximations to the optimal solution with little time consumption.

For this reason, in addition to classical methods of Operations Research and Optimization, several different algorithms have been developed, including genetic algorithms [3], simulated annealing [4], taboo search [5], and neural networks [6].

Concerning neural networks, the main subject to deal with is to achieve a good representation or formulation of the problem such that its resolution arises as an energy function minimization problem.

In 1985, Hopfield and Tank [7] proposed the first neural network for the study of combinatorial optimization problems (Hopfield's analog model), which was precisely used to solve this problem.

This analog model has more ability to escape from local minima than the discrete model. Some deficiencies are present in both models, as it is the need to fine-tune a high number of parameters in the energy function, as mentioned by Wilson and Pawley [8].

Other approaches are entirely based on Kohonen's self-organizing maps [9], achieving the best results with the so-called KNIES network [10], in which some statistical measures were incorporated into the original model. This model also presents the drawback of the fine-tuning of a high number of parameters to achieve good results.

In the last few years, the multivalued model MREM has achieved very good results, outperforming KNIES, and presenting the advantage of not needing any adjustment of parameters, quite the opposite to KNIES.

5 The MREM Model for the Travelling Salesman Problem

In order to solve the ETSP with this neural model, two identifications must be made:

- **A network state must be identified to a solution of ETSP**
  A solution of ETSP can be represented as a permutation in the set of numbers $\{1,\ldots,N\}$, where N is the number of cities, since it represents the order in which cities are visited. For this reason, the net will be formed by N neurons, each of them taking a value in the set $\mathcal{M}=\{1,\ldots,N\}$, such that the state vector $\boldsymbol{S}=(s_1,\ldots,s_N)$ represents a permutation of $\{1,\ldots,N\}$ (a *feasible state*). With this representation, $s_i=k$ means that the k-th city will be visited in i-th place.
- **The energy function must be identified to the total distance of the tour**
  If we make, in Eq. (1), $f(x,y)=-2d_{x,y}$ and

$$w_{i,j}=\begin{cases}1 & \text{if } (j=i+1)\vee((i=N)\wedge(j=1))\\ 0 & \text{otherwise}\end{cases}$$

  then the energy function that we obtain is $E(\boldsymbol{S})=\sum_{i=1}^{N-1} d_{s_i,s_{i+1}}+d_{s_N,s_1}$, that is, the total distance of the tour represented by the state vector $\boldsymbol{S}$.

Computational dynamics is based in beginning with a random feasible initial state vector and updating neurons outputs such that the state vector of the net will always be feasible. To this end, in each iteration, a *2-opt* [16,17,18] update will be made over the current state vector. That is, every pair of neurons, p,q with $p>q+1$, is studied and the net checks for crosses between the segments $[s_p,s_{p+1}]$ and $[s_q,s_{q+1}]$. In that case, the next inequality holds:

$$d_{s_p,s_{p+1}}+d_{s_q,s_{q+1}}<d_{s_p,s_q}+d_{s_{p+1},s_{q+1}}$$

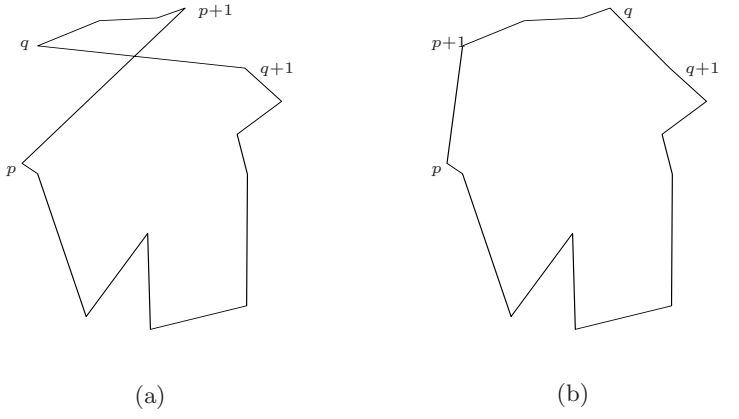

Fig. 1. An example of *2-opt* iteration: (a) Original tour. (b) Tour modified by 2-opt technique.

Then, the path from city s_{p+1} to s_q is reversed (see Fig. 1), that is, if $\boldsymbol{S}$ is the current state, the new state vector $\boldsymbol{S'}$ will be defined as:

$$s'_i = \begin{cases} s_{q+p+1-i} & \text{if } p+1 \leq i \leq q \\ s_i & \text{otherwise} \end{cases}$$

As an additional technique for improvement, *3-opt* updates have also been used: the tour is decomposed in 3 consecutive arcs, A, B and C, which are recombined in all possible ways: $\{ABC, ACB, A\hat{B}C, AB\hat{C}, A\hat{B}\hat{C}, A\hat{C}B, AC\hat{B}, A\hat{C}\hat{B}\}$, where $\hat{A}$, $\hat{B}$, $\hat{C}$ are the arcs corresponding to the inversion of A, B, and C, respectively. For example, if $A = (8, 9, 4, 6)$, $B = (1, 5, 3)$ and $C = (2, 7)$, then $\hat{A} = (6, 4, 9, 8)$, $\hat{B} = (3, 5, 1)$ and $\hat{C} = (7, 2)$, and the combination $A\hat{C}\hat{B} = (8, 9, 4, 6, 7, 2, 3, 5, 1)$. Note that $\{ABC, A\hat{B}C, AB\hat{C}, A\hat{C}\hat{B}\}$ are *2-opt* updates and there is no need to check them again.

The next state of the net will be the combination (chosen from the above list) which decreases most the value of the energy function. For more details, please refer to [12].

6 Functional Annealing for the Resolution of ETSP

In this section, several ways of applying Functional Annealing to ETSP are considered. In each of them we will try to introduce some kind of knowledge about the problem in its resolution, that is, this knowledge will be present in the construction of the sequence $\{E_n\}$.

We will represent this knowledge by means of a transformation of the distances between each pair of cities, that is, we will consider approximating energy functions as in Eq. (2), with $W^{(n)} = W$ for all n, and $f^{(n)}(x, y) = -2d^{(n)}_{x,y}$. The introduction of these functions $d^{(n)}_{x,y}$ will intensify the fact that cities close to

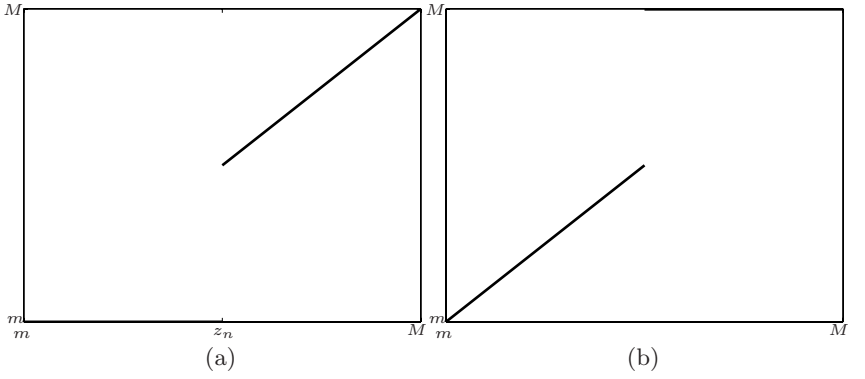

Fig. 2. Graphical representation of the proposed $d_{x,y}^{(n)}$: (a) for FA_1, (b) for FA_2

each other should be visited consecutively, and that cities which are far away from each other should not. Thus, energy function E_n defined in this way will represent the total distance of the tour associated to the corresponding state vector, but with different measures of the distances between cities.

For simplicity, only a finite number of E_n will be considered: $E_1, E_2, \ldots, E_L$, $E_{L+1} = E$. Let us define a pair of sequences $\{E_n\}$ (it suffices to define $d_{x,y}^{(n)}$) which introduce some of this knowledge in the resolution:

- The first approach will be called FA_1: In this case, $d^{(n)}$ is defined as

$$d_{x,y}^{(n)} = \begin{cases} d_{x,y}, \text{ if } d_{x,y} \leq \theta_n \\ M, \quad \text{otherwise} \end{cases}$$

  where $\theta_n = m + \dfrac{(n-1)(M-m)}{L}$, and m and M are, respectively, the minimum and maximum values of $d_{x,y}$ for $x \neq y$.
  This election of $d^{(n)}$ makes the algorithm order cities which are close to each other, giving no importance to cities that are far away. That is, in the first few iterations, a partial ordering is induced in the neighborhood of each city. As the neighborhood expands, this series of partial orderings becomes less partial, generating inductively a total ordering that usually represents a better tour in terms of total distance.
- The second approach (FA_2), consists in defining

$$d_{x,y}^{(n)} = \begin{cases} 0, & \text{if } x \in \mathcal{V}_{L-n+1}(y) \vee y \in \mathcal{V}_{L-n+1}(x) \\ d_{x,y}, & \text{otherwise} \end{cases}$$

  where $\mathcal{V}_m(X_k)$ is the set formed by X_k and the m nearest cities to X_k, such that $\mathcal{V}_0(X_k) = \{X_k\}$.
  This means that, in the first iterations, the model will try to build a global ordering, not focusing in cities that are close to each other, but achieving a rough ordering that can be refined in the following iterations, by taking into account distances between cities close to each other.

In fact, these techniques can help to avoid certain local minima of the energy function E, since the objective function to minimize in each step is E_n, allowing the net to increase temporarily the value of the original energy function.

A graphical representation of $d^{(n)}$ for these two techniques FA_1 and FA_2 can be found in Fig. 2.

7 Experimental Results

Our algorithm has been tested with a wide set of ETSP instances. These instances come from the well-known repository TSPLIB, available on the world wide web and created by Reinelt [19,20]. One of its most important features is that every instance has a record with the distance of the optimal tour, allowing to compare the relative efficiency of our algorithms with respect to MREM. It must be noted that MREM results are better than those of KNIES [12].

Experimental results are shown in Tables 1 and 2. 25 independent executions were made for each instance. The quality measure used in these tables is the percentage of error over the optimum, that is, to compare two results we use their respective relative errors, given by (in percentage):

$$\text{error} = \frac{\text{(Best or average) tour length} - \text{Opt}}{\text{Opt}} \cdot 100$$

We have tested the sMREM model and Functional Annealing with dynamics FA_1 and FA_2. The acceptance function g_n considered takes the following form: $g_n(\Delta) = \exp\left(\frac{-\Delta}{T_n}\right)$, where T_n is a 'temperature' parameter, converging to 0, what makes $\{g_n\}$ converge ε-uniformly to 0.

Since we only use a finite number of approximating energy functions, we can consider that the temperature parameter T_n decreases linearly from $T_1 = 1$ to $T_L = 0$ and $K = 20$ iterations for each value of the temperature.

The parameter L takes the following values:

- $L = 20$ for sMREM and FA_1, $L = 5$ for FA_2, whose results are shown in Table 1.
- $L = 40$ for sMREM and FA_1, $L = 10$ for FA_2, whose results are shown in Table 2.

We must also specify the way the next state of the net is sampled:

- The increase of energy corresponding to each of the states which are in the neighborhood of the current state, $\boldsymbol{S}$, is computed.
- The probability of sampling the state $\boldsymbol{S'} \in \mathcal{N}_{\boldsymbol{S}}$ is proportional to the exponential of the opposite of its increment of energy, divided by the temperature at that iteration: $\mathbb{P}_{\mathrm{s}}(\boldsymbol{S'}) \propto \exp(\frac{-\Delta_{\boldsymbol{S},\boldsymbol{S'}}}{T_n})$.

Once the next state is sampled, it is accepted or not depending on the probability of acceptance $\mathbb{P}_{\mathrm{a}}$, which is defined in terms of g_n.

Table 1. Comparative results (25 executions per instance) between MREM and the stochastic methods herein proposed for different instances of the Travelling Salesman Problem ($L = 20$ for sMREM and FA_1, $L = 5$ for FA_2)

Instance	Opt.	KNIES	MREM			sMREM			FA_1			FA_2		
		Best	Best	Average	t	Best	Average	t	Best	Average	t	Best	Average	t
eil51	426	2.86	0.23	2.43	3.12	0.23	1.42	11.91	0	1	12.14	0.23	1.67	3.73
st70	675	1.51	0	1.89	9.01	0	0.94	23.09	0	0.85	23.19	0	1.11	8.32
eil76	538	4.98	1.3	3.43	10.8	0.19	1.72	28.15	0	1.62	28.11	0	1.8	8.87
rd100	7910	2.09	0	3.02	61.7	0.62	2.99	57.66	0	2.16	52.44	0.43	2.9	23.33
eil101	629	4.66	1.43	3.51	27.76	0.48	1.78	54.44	0	2.14	53.71	0.16	1.38	24.81
lin105	14379	1.29	0	1.71	28.83	0.79	3.17	65.1	0.15	2.27	66.37	0	2.81	29.3
pr107	44303	0.42	0.15	0.82	49.79	0.83	1.57	76.12	0.2	0.64	71.95	0	0.45	26.59
pr124	59030	0.08	0	1.23	59.51	0.26	1.87	100.98	0	1.19	100.81	0.26	1.36	43.88
bier127	118282	2.76	0.42	2.06	66.29	1.22	3.78	112.43	0.65	2.44	112.18	0.73	2.26	61.15
kroA200	28568	5.71	3.49	6.70	318.44	4.79	7.30	474.51	3.22	4.97	377.43	4.23	6.27	260.74

Table 2. Comparative results (25 executions per instance) between MREM and the stochastic methods herein proposed for different instances of the Travelling Salesman Problem, second version ($L = 40$ for sMREM and FA_1, $L = 10$ for FA_2)

Instance	Opt.	KNIES	MREM			sMREM			FA_1			FA_2		
		Best	Best	Average	t	Best	Average	t	Best	Average	t	Best	Average	t
eil51	426	2.86	0.23	2.43	3.12	0	1.22	22.78	0	1.03	23.25	0.47	2.19	6.66
st70	675	1.51	0	1.89	9.01	0	0.59	43.46	0	0.58	44.1	0.59	1.43	14.26
eil76	538	4.98	1.3	3.43	10.8	0	1.46	52.43	0.19	1.07	53.03	0.93	2.45	16.07
rd100	7910	2.09	0	3.02	61.7	1.31	4.38	103.44	0.59	2.52	97.09	0.43	2.5	36.28
eil101	629	4.66	1.43	3.51	27.76	0	1.35	94.6	0.16	1.49	96.75	0.16	2.17	35.92
lin105	14379	1.29	0	1.71	28.83	0.77	3.66	110.14	0.38	1.9	110.83	0	1.41	42.29
pr107	44303	0.42	0.15	0.82	49.79	0.3	1.06	132.94	0.5	1.18	117.33	0.12	0.78	44.38
pr124	59030	0.08	0	1.23	59.51	0.08	1.66	166.77	0.38	1.76	163.29	0	1.22	63.97
bier127	118282	2.76	0.42	2.06	66.29	1.96	3.18	195.07	0.72	2.41	186.19	0.91	1.93	83.68
kroA200	28568	5.71	3.49	6.7	318.44	4.38	7.04	648.64	3.47	5.9	558.65	3.45	6.02	273.34

When $n = L$, the next iteration ($n = L+1$) does not use a stochastic dynamics, so it is the *2-opt* (or *3-opt*) mentioned before.

Regarding solution quality, we can observe in Tables 1 and 2 that the formulation that gets the best results is FA_1, achieving the best average results in almost every test instance, showing its ability to escape from local minima. This fact can also be observed by comparing the optimal results of these methods.

It must be noted that results obtained by KNIES were achieved by means of a trial-error process, since this algorithm has to fine-tune a high number of parameters.

In columns labeled 't' in these tables, we can check that the time consumption is not a drawback in this case, since there are instances in which FA_2 is, at least, as fast as MREM and FA_1 does not represent a high increase of time consumption. This kind of 'acceleration' comes from the fact that Functional Annealing gets good solutions in the first iterations. Then, since F_n is very close to F in the last iterations, these good solutions are actually good approximations of the final solution, and the algorithm hardly iterates.

8 Conclusions

In this work we have studied an optimization technique, Functional Annealing, based on stochastic searches that can help to avoid certain local minima of the objective function. This technique also allows us to introduce some knowledge about the problem in its resolution.

We have proposed the theoretical results on which the algorithm is based. These results prove the convergence of Functional Annealing to a local minimum of the objective function.

We have used the proposed techniques to solve the well-known Traveling Salesman Problem. With these methods, we eliminate long paths in the tour, at the same time that crosses are avoided. Not eliminating those long paths is the main cause of the local minima in which an optimization algorithm may get trapped.

The proposed algorithms outperform, in most cases, results obtained by MREM, without a great increase of time consumption.

References

1. Reinelt, G.: The Travelling Salesman. In: Computational Solutions for TSP Applications, Springer, Heidelberg (1994)
2. Bland, R., Shallcross, D.F.: Large traveling salesman problem arising from experiments in x-ray crystallography: a preliminary report on computation. Technical Report No. 730, School of OR/IE, Cornell University, Ithaca, New York (1987)
3. Potvin, J.: Genetic algorithms for the traveling salesman problem. Annals of Operations Research 63, 339–370 (1996)
4. Aarts, E., Korst, J., Laarhoven, P.: A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem. J. Stats. Phys. 50, 189–206 (1988)

5. Fiechter, C.: A parallel tabu search algorithm for large scale traveling salesman problems. Technical Report 90/1, Department of Mathematics, Ecole Polytechnique Federale de Lausanne, Switzerland (1990)
6. Potvin, J.: The traveling salesman problem: A neural network perspective. INFORMS Journal on Computing 5, 328–348 (1993)
7. Hopfield, J., Tank, D.: Neural computation of decisions in optimization problems. Biological Cybernetics 52, 141–152 (1985)
8. Wilson, V., Pawley, G.: On the stability of the TSP problem algorithm of Hopfield and Tank. Biological Cybernetics 58, 63–70 (1988)
9. Kohonen, T.: Self-organizing Maps. Springer, Heidelberg (1995)
10. Aras, N., Oomen, B.J., Altinel, I.: The Kohonen network incorporating explicit statistics and its application to the Travelling Salesman Problem. Neural Networks 12, 1273–1284 (1999)
11. Mérida-Casermeiro, E.: Red Neuronal recurrente multivaluada para el reconocimiento de patrones y la optimización combinatoria. PhD thesis, Universidad de Málaga, Spain (2000)
12. Mérida-Casermeiro, E., Galán-Marín, G., Muñoz Pérez, J.: An efficient multivalued Hopfield network for the travelling salesman problem. Neural Processing Letters 14, 203–216 (2001)
13. Mérida-Casermeiro, E., Muñoz Pérez, J., Domínguez-Merino, E.: An n-parallel multivalued network: Applications to the Travelling Salesman Problem. In: Mira, J.M., Álvarez, J.R. (eds.) IWANN 2003. LNCS, vol. 2686, pp. 406–413. Springer, Heidelberg (2003)
14. Mérida-Casermeiro, E., López-Rodríguez, D.: Graph partitioning via recurrent multivalued neural networks. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 1149–1156. Springer, Heidelberg (2005)
15. López-Rodríguez, D., Mérida-Casermeiro, E., Ortiz-de Lazcano-Lobato, J.O., López-Rubio, E.: Image compression by vector quantization with recurrent discrete networks. In: Kollias, S., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4132, pp. 595–605. Springer, Heidelberg (2006)
16. Lin, S., Kernigham, B.W.: An effective heuristic algorithm for the Traveling Salesman Problem. Operations Research 21, 498–516 (1973)
17. Johnson, D.S., McGeoch, L.A.: The Traveling Salesman Problem: A Case Study in Local Optimization. In: Local Search in Combinatorial Optimization, John Wiley, Chichester (1997)
18. Hoos, H.H., Stuetzle, T.: Traveling Salesman Problems. In: Stochastic Local Search, Morgan Kaufman (2004)
19. Reinelt, G.: TSPLIB - a Travelling Salesman Problem library. ORSA Journal of Computing 3, 376–384 (1991)
20. Bixby, B., Reinelt, G.: Travelling Salesman Problem library (1999), `http://www.crpc.rice.edu/softlib/tsplib.html`