# A Novel Neural Approximation Technique for integer formulation of the Asymmetric Travelling Salesman Problem

Alessandro Moschitti

University of Rome Tor Vergata
Department of Computer Science, Systems and Production
00133 Roma (Italy)

**Abstract**

In this paper a novel approximation algorithm for integer formulation of the Asymmetric Travelling Salesman Problem based on a sequence of Hopfield's networks is described. The sequence, in polynomial time, converges to a state representing a feasible solution for a given instance. The results, carried out with two different evaluation methods for the designed neural algorithm, show how this technique is interesting in order to make the neural network approximation approaches competitive with respect to the more classical ones.

## 1 Introduction

The Travelling Salesman Problem (TSP) can be defined as follows:
Given a set $V=\{c_1,..,c_n\}$ of cities and for each pair $\{c_i,c_j\}$ of distinct cities a cost $P(i,j)$, finding an ordering $h$ of the cities which minimises the quantity

$$P^*(h) = \sum_{i=1}^{m-1} P(h_i, h_{i+1}) + P(h_m, h_1) .$$

This quantity is referred to as the tour length, since it is the length of the tour a salesman would make when visiting the cities in the order specified by permutation and returning at the end to the initial city. In this article we will consider symmetric and asymmetric TSP even in its dual version (i.e. find out an ordering $h$ that maximises $P^*(h)$). Both TSP and dual TSP are NP-hard [Karp, 1972] and so any algorithms for finding optimal tours must have a worst case running time that grows faster than polynomial (assuming the widely believed conjecture that P≠NP).
Several attempts to build neural network algorithms have been carried out to solve TSP, but none of them have been proved competitive with more classical approaches [Potvin, 1993]. In particular, none can produce tours even as good as classical algorithms and most take substantially more time, at least on sequential machines. The set of neural algorithms for TSP can currently be divided in two main classes [Johnson and McGeoch, 1997]. In the former neurons are structured according to some formulation of the TSP as an integer program. The second is concerned with geometric instances of the problem and the neurons are viewed as points in space seeking out cities with which they are identified.
The neural algorithm that we propose referred to the integer programming formulation of the TSP and is based on discrete Hopfield's networks. Hopfield [Hopfield and Tank, 1985] designed the first application based on neural net approach for the TSP, but as it has been pointed out [Wilson and Pawley, 1988] that approach failed to find a feasible solution (i.e. some constraints were violated) for each instance. To overcome this, we propose an algorithm that uses the Recursive Hopfield Networks (RHN) technique [Bertoni et al, 1997] to find a feasible solution of a sub instance of the problem that we recursively extend in order to obtain the global solution. RHN technique consists of building a finite sequence of networks where the energy function of the ($k$+1)-th network is the energy function of the $k$-th network augmented by a penalty factor proportional to the number of unsatisfied constraints.
The algorithm has been experimented both on random instances of different sizes and on instances collected in the TSPLIB[1] [Reinelt, 1991] benchmark. The results show that our algorithm have good performance especially for dual TSP considering that asymmetric TSP does not allow to apply many performing classical approaches based on euclidean distance.
In Section 2 we present a novel way to characterise the TSP. In Section 3 we show how our technique can approximately solve the TSP problem and we prove the polynomial converging, in Section 4 we

[1] TSBLIB is a collection of benchmark for TSP problem.

present the algorithm performances and finally in Section 5 we discuss the results and future extensions.

## 2 TSP characterisation via Boolean 2-clauses

A TSP can be represented by means of a direct graph $G \equiv (V,E,P)$ where: $V=\{c_1,..,c_n\}$, $E=\{(i,j): c_i,c_j \in V\}$ and $P(i,j)$ is the function cost. Let $X_G = \{ x_{11},...,x_{1n}, x_{21},...,x_{2n}, \ldots, x_{n1},...,x_{nn} \}$ be a set of boolean variable associated with G. A *literal* over $X_G$ is either a variable $x_{ij}$ or its negation $\bar{x}_{ij}$, a *k-clause c* is a disjunction of *k* literal, and *k-term t* is a conjunction of *k literal.* By an assignment on $X_G$ we mean a function $\sigma: X_G \rightarrow \{0,1\}$ ; the literal $x_{ij}$ is said to be true (false) under the assignment $\sigma$ if $\sigma(x_{ij}) = 1$ ($\sigma(x_{ij}) = 0$) and the literal $\bar{x}_{ij}$ is said to be true (false) if $\sigma(x_{ij}) = 0$ ($\sigma(x_{ij}) = 1$). A clause *c* is said to be satisfied by the assignment $\sigma$ on $X_G$ if and only if at least one of its literals is true while a term *t* is said to be satisfied by the assignment $\sigma$ on $X_G$ if and only if all its literals are true.

Given a set $S \subseteq E$, let us define its characteristic vector $\chi^s$ as : $\chi_{ij}^S = \begin{cases} 1, \text{if } (i,j) \in S; \\ 0, \text{otherwise.} \end{cases}$

If we assign a variable $x_{ij}$ to each edge $(i,j) \in E$, an assignment $\sigma$ on $X_G$ uniquely identifies the subset *S* of *E* whose characteristic vector is $\chi^s = (\sigma(x_{11}),..,\sigma(x_{nn}))$. We say that $\chi^s$ induces the graph $G'(V,S,P)$. Figure 2 shows two graphs: the former is the original (complete of all edges) while the second is the induced one by an assignment.
Let's consider the following set of 2-clauses that constitute the set of constraints of the instance G:

1. $\bar{x}_{ik} \vee \bar{x}_{ih}$, $\forall\, i \in V\ \forall\, k,h \in \{p:(i,p) \in E', p \neq i\}$ with $k \neq h$,
2. $\bar{x}_{ki} \vee \bar{x}_{hi}$, $\forall\, i \in V\ \forall\, k,h \in \{p:(p,i) \in E,\ p \neq i\}$ with $k \neq h$,
3. $\bar{x}_{ij} \vee \bar{x}_{ji}$, $\forall\, i,j \in V$ with $j \neq i$.

The first constraint set allows a unique edge to go out of a node, the second bound set refers to a unique edge entering a node and the last constraint set imposes a unique edge between two nodes. Figure 1 graphically shows in which case the constraints are activated avoiding that both variable of 2-clauses are set to 1.

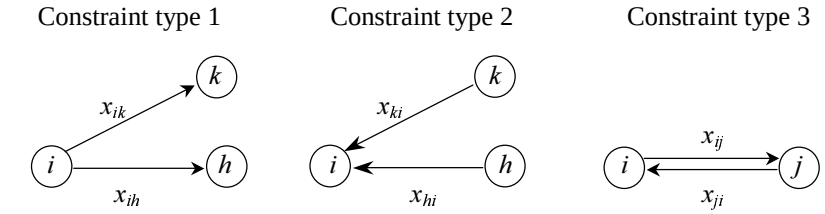
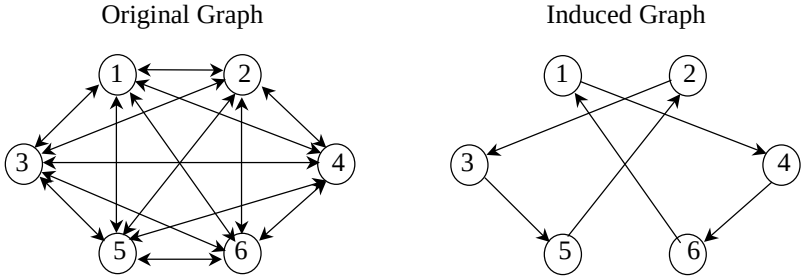


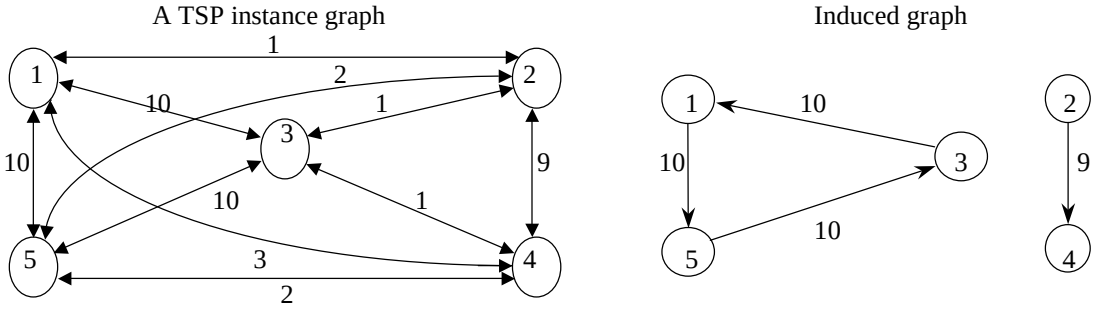

**Figure 1:** Graphic displaying of 2-clause relative to constraints 1, 2, and 3.




Assignment σ :

$x_{11}$=0 $x_{12}$=0 $x_{13}$=0 $\mathbf{x_{14}}$**=1** $x_{15}$=0 $x_{16}$=0, $x_{21}$=0 $x_{22}$=0 $\mathbf{x_{23}}$**=1** $x_{24}$=0 $x_{25}$=0 $x_{26}$=0, $x_{31}$=0 $x_{32}$=0 $x_{33}$=0 $x_{34}$=0 $\mathbf{x_{35}}$**=1** $x_{36}$=0
$x_{41}$=0 $x_{42}$=0 $x_{43}$=0 $x_{44}$=0 $x_{45}$=0 $\mathbf{x_{46}}$**=1,** $x_{51}$=0 $\mathbf{x_{52}}$**=1** $x_{53}$=0 $x_{54}$=0 $x_{55}$=0 $x_{56}$=0, $\mathbf{x_{61}}$**=1** $x_{62}$=0 $x_{63}$=0 $x_{64}$=0 $x_{65}$=0 $x_{66}$=0

**Figure 2**: Original complete graph associated to a TSP instance and the graph induced by the assignment σ over $X_G$.




Optimal assignment

$x_{11}$=0 $x_{12}$=0 $x_{13}$=0 $x_{14}$=0 $\boldsymbol{x_{15}}$**=1**, $x_{21}$=0 $x_{22}$=0 $x_{23}$=0 $\boldsymbol{x_{24}}$**=1** $x_{25}$=0, $\boldsymbol{x_{31}}$**=1** $x_{32}$=0 $x_{33}$=0 $x_{34}$=0 $x_{35}$=0
$x_{41}$=0 $x_{42}$=0 $x_{43}$=0 $x_{44}$=0 $x_{45}$=0, $x_{51}$=0 $x_{52}$=0 $\boldsymbol{x_{53}}$**=1** $x_{54}$=0 $x_{55}$=0.

**Figure 3:** The graph on the right represents a TSP instance with 5 cities while graph on the left represents a solution of the problem (1).

Let OR($\chi^S$) be the logic conjunction of the all 2-clauses set specified in point 1,2 and 3 subjected to the assignment $\chi^S$ and let's consider the following optimisation problem

$$\max_{S\subseteq E:\ \mathrm{OR}(\chi^S)=1} \sum_{(i,j)\in S} P(i,j)\cdot x_{ij} \qquad (1)$$

A set $S$ that solves the problem induces a graph containing at least a valid tour i.e. a set $T$ of edges so that if $(h_i,h_j)\in T$ then $(h_v,h_i)$ and $(h_j,h_w)$ belong to $T$. Figure 3 shows a graph representing a TSP instance and the relative induced graph by an assignment that solve the problem (1). It is worth noticing that at least one tour belongs to the induced graph, moreover there are no edges from (or to) nodes in set {1,3,5} to (or from) set {2,4}. This is due to constraints of type 1 and 2.
The following theorem proves that the problem described by the equation (1) constrains optimal solution ($\chi^s$ assignment) to induce a graph which has at least one tour.

***Theorem 1***. *Let $G\equiv(V,E,P)$ be a direct graph associated to a TSP, with $|V|>2$ and let S be a subset of E so that solves the equation (1), then $\chi^s$ induces a graph G′ that contains at least a tour.*

*Proof.* Suppose, by way of contradiction, that the induced graph by the optimal assignment $\chi^s$ does not contain a tour then we build a solution $\chi'^s$ better than optimum.
It is trivial to see that constraints specified in points 1,2 and 3 make the graph (induced by $\chi^s$) have nodes with only one outgoing edge (constraint 1), only one entering edge (constraint 2) and without a symmetric edge (constraint 3).
Let $\{s_1,\dots,s_r\}$ be a collection of subsets so that $\bigcup_{i=1,\dots,r} s_i = V$ and each $s_i$ satisfies the following property:

if $u,v\in s_i$ then $\exists\ u_1,..,u_k\in s_i : u=u_1, v=u_k$ and $(u_i,u_{i+1})\in S$ for each $i$=1,..,$k$-1 (2)
where $s_i\subseteq V$ is a set of nodes.

For each $s_i$ we built the *longest sequence* $ls(s_i)=\left\langle u_1^i,\dots,u_{n_i}^i\right\rangle$: $(u_k^i,u_{k+1}^i)\in S$ for $k=1,\dots,n_i$-1 and $u_k^i\in s_i$ for $k=1,\dots,n_i$.

There are two possibilities

*a)* $n_i=|V|$ (i.e. r = 1, there is only a set in the collection[2]): we built a new solution adding the edge $(u_{n_i}^i,u_1^i)=(u_{|V|}^1,u_1^1)$ to $S$. Such edge does not belong to $S$ on the basis of the theorem hypothesis (i.e. the solution does not contain any tour), moreover we can show that the new solution is feasible: constraint 3 is satisfied because $|V|>2$ on the basis of the theorem hypothesis while, constraint 1 and 2 are satisfied because they do not exist $v,w\in S$ so that $(v,u_1^1),(u_{|V|}^1,w)\in S$ otherwise $v$ would have two outgoing edges and $w$ would have two entering ones (note that $u,w$ necessarily belong to $s_1$).

*b)* $n_i\neq|V|$ (i.e. r > 1, there are at least two set in the collection): we add the edge $(u_{n_{i+1}}^{i+1},u_1^i)$.

[2] It is easy to prove that the collection $\{s_1,\dots,s_r\}$ constitutes a partition.

Constraint 3 is verified because nodes in different sequence are different. Constraint 1 is satisfied because $u_1^i$ has no any entering edge $(v,u_1^i)$ otherwise the sequence $\langle v,u_1^i,...,u_{n_i}^i \rangle$ would be longer that $ls(s_i)$. Similar reasons prove that constraint 3 is satisfied.

Recalling that P(*i,j*) > 0 (for each edge (*i,j*) in *G*), when we add a new edge to *S* we increment the function cost associated to the optimal solution, but this is a contradiction therefore there must be at least a tour so that the above construction can be avoided.

## 3 Designing a Recursive Neural Algorithm (RNA) for TSP

We have proved that solving (1) can lead to obtain a tour. In this section we shall introduce the RHN technique [Bertoni et al, 1997] to approximately solve (1) and than we shall show how an algorithm (RNA) can find a feasible tour by using RHN.

### 3.1 Recursive Hopfield Network (RHN)

Given a graph G, let's introduce, for each variable $x_i \in X'_G = \{\alpha_k : \alpha_k = x_{ij},\ k = i*n+j,\ x_{ij} \in X_G\}$ a variable $y_i = 2x_i - 1$ that assumes values on the set {-1,1}, where $n = |V|$. Moreover, let's define for each 2-term $x_i \wedge x_j$ the polynomial

$$f_{AND}(y_i, y_j) = y_i + y_j + y_i y_j,$$

and for each 2-clause $\bar{x}_i \vee \bar{x}_j$ the polynomial

$$f_{OR}(y_i, y_j) = -y_i - y_j - y_i y_j.$$

The mappings of the functions $f_{AND}$ and $f_{OR}$ is given in the Table 4.

| $I_1$ | $I_2$ | $f_{OR}$ | $f_{AND}$ |
|---|---|---|---|
| 1 | 1 | -3 | 3 |
| -1 | 1 | 1 | -1 |
| 1 | -1 | 1 | -1 |
| -1 | -1 | 1 | -1 |

**Figure 4**: Mapping of the functions $f_{AND}$ and $f_{OR}$.

Let's consider then the following quadratic functions $\psi : \{-1,1\}^n \to \mathbf{Z}$ and $\Omega : \{-1,1\}^n \to \mathbf{Z}$, defined as:

$$\Psi(y_1,\ldots y_n) = \sum_{x_i \wedge x_j \in W} f_{AND}(y_i, y_j) = \sum_{i \neq j} a_{ij} y_i y_j + \sum_i u_i y_j\ , \qquad (3)$$

and

$$\Omega(y_1,\ldots y_n) = \sum_{\bar{x}_i \vee x_j \in C_{1.2.3.}} f_{OR}(y_i, y_j) = \sum_{i \neq j} (a_{ij} - 1) y_i y_j + \sum_i v_i y_i\ , \qquad (4)$$

where W = { $y_i \vee z_r$: $r$ =1,…,$P(i)$} determines the weight associated with the edge *i*, by means of the dummy variables $z_r$ and $C_{1.2.3}$ is the set of constraints defined in point 1,2 and 3.

In this setting, solving the problem (1) is reduced to solving the following integer quadratic problem.

$$\text{Maximize } \Psi(\mathbf{y}) = \mathbf{y}^T A\mathbf{y} + \mathbf{u}\cdot\mathbf{y}$$
$$\text{Subject to } \Omega(\mathbf{y}) = |C_{1.2.3}|$$

A discrete sequence of Recursive Hopfield Networks $\{\Re_k\}_{k\geq 0}$ is a sequence of Hopfield's networks in which the neurons {1,…,*n*} represent the edges of G (they assume state in {-1,1}) and the energy function $\Phi_k$ of the specific network $\Re_k$ is obtained as follows:

$$\Phi_0(\mathbf{y}) = \Psi(\mathbf{y}) + \Omega(\mathbf{y});$$
$$\Phi_k(\mathbf{y}) = \Phi_{k-1}(\mathbf{y}) + \sum_{\bar{x}_i \vee \bar{x}_j \in C_{1.2.3}^{(k-1)}} f_{OR}(y_i, y_j), \qquad (5)$$

where $C_{1.2.3}^{(k-1)}$ is the set of 2-clauses in $C_{1.2.3}$ not satisfied by the assignment $\sigma_{\mathbf{y}^{(k-1)}}$ corresponding to $\mathbf{y}^{(k-1)}$; $\mathbf{y}^{(0)},\ldots,\mathbf{y}^{(k-1)}$ is the sequence of solution vectors of the $\Re_0,\ldots,\Re_{k-1}$ network sequence (it is usually

initialised with $y^{(0)}$ = [1, ..., 1]).

In [Bertoni et al, 1997] it has been proved that Recursive-Hopfield-Nets sequence is finite with a upper bound of $|X'_G|^2$ and it has been implemented by the RHN algorithm.

### 3.2 Recursive Neural Technique for dual TSP

The idea of RNA is to solve (approximately), using the RHN algorithm, problem (1) in order to obtain a solution containing at least a tour. All neurons belonging to the tours are fixed to 1 value (i.e. they belong to the network but can assume only the 1 value); only the neuron (belonging to a tours) associated with the edge having the lowest weight is fixed to –1 value. The new network is thus composed: the neurons fixed to 1, the neuron fixed to –1 and the remaining neurons (named free neurons). It converges to a new configuration that induces at least another tour. In this fashion it is guaranteed that each time at least one neuron is removed from the network, therefore the algorithm converges.

The network is built as follows:

- a neuron $y_k$ is associated to each variable in $x_{ij} \in X_G$, for each $i, j = 1,\ldots,n$ and $k=i*n+j$ where $n=|V|$.
- to each edge $y_k$ is assigned the weight $P'(k)=P(i,j)$ adding the set of 2-clauses $\{y_k \vee z_r: r=1,\ldots,P'(k)\}$, where $z_r$ are dummy boolean variables fixed to 1. In this way, setting the variable $y_k$ to 1 augments the network energy function $\Phi(\mathbf{y})$ of the quantity $P'(k)$.

### RNA algorithm for dual TSP

1. Let G(*V,E,P*) be a weighted graph representing a dual TSP, $m=0$ and $n=|V|$, so build a Hopfield network $\Re^0_m$ associating to each edge (*i,j*) of G a neuron $y_k$. The set of free neurons is $I_m = \bigcup_{(i,j)\in E}\{y_k\}$
2. RHN algorithm in finite time satisfies constraints 1,2 and 3. By doing this a sequence $\{\Re^v_m\}_{v\in N}$ of Hopfield networks is generated, It converges to the final state $\mathbf{y}=(y_1,\ldots,y_{n^2})$, with the classical transition function [Alberti et al, 1995]:
$$y_i(t+1) = sg\left(\sum_{k=1,\ldots,i-1}\omega_{ik}\,y_k(t+1) + \sum_{k=i,\ldots,n}\omega_{ik}\,y_k(t) - \lambda_i\right) \qquad (6)$$
3. where $\omega$ is the matrix associated to the networks $\Re^v_m$ (i.e. the matrix extracted from equation (3) plus equation (4)) and $i \in \{k : y_k \in I_m\}$ (i.e. only free neurons change state)
4. Let *Cn* be the set of neurons belonging to a tour of the graph induced by assignment $\sigma_y$:
   if $|Cn|<|V|$ then the new initial network $\Re^0_{m+1}$ has a free neuron set $I_{m+1} = (I_m - Cn)$, set $y_k = -1$ (with $P'(k) = \min_{y_r \in Cn} P'(r)$), do $m = m+1$ and go point 3
5. The solution is $\sigma_y$ and the tour cost is $\sum_{(i,j)\in E} P(i,j)\cdot x_{ij} = \sum_{k=1}^{n^2} P'(k)\cdot\frac{1+\sigma_y(y_k)}{2}$

### 3.3 RNA converging

To prove that RNA converges we prove that:

- *The RHN sequence converges to a state that induces at least a tour.*
- *A RHN containing fixed neurons maintains the above property .*

The following lemma shows that the energy function of the Hopfield's networks converges to local max and extends Theorem 1 application on local max solutions found by a RHN.

***Lemma 1.*** *Let G≡(V,E,P) be a direct graph associated to a TSP, with $|V|>2$ and let $S \subseteq E$ be the final state of an RHN sequence, then $\chi^s$ induces a graph G′ that contains at least a tour.*

*Proof.* To prove Theorem 1 we have shown that if an optimal solution does not contain a sub tour we

can add an edge that augments the function cost.
The same operation is carried out in case of a solution found by a neural network. Let us consider the network energy function below

$$\Phi_C(y_1,\ldots,y_n)=\frac{3}{4}|C|+\frac{1}{4}\left(\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\omega_{ik}\cdot y_i y_k+\sum_{i=1}^{n}I_i\cdot y_i\right)$$

and let $(y_1,\ldots,y_n)$ be the stable state (final state) associated to the set $S$, then adding an edge (i.e. set to 1 a neuron) can not augment the energy. In fact suppose, in way of contradiction, that setting $y_h$ to 1 would improve the energy then the following disequation would be true

$$\Phi_C(y_1,\ldots,y_{h-1},1,y_{h+1},\ldots,y_n)-\Phi_C(y_1,\ldots,y_{h-1},-1,y_{h+1},\ldots,y_n)>0$$

that is

$$\Phi_C(y_1,\ldots,y_{h-1},1,y_{h+1},\ldots,y_n)-\Phi_C(y_1,\ldots,y_{h-1},-1,y_{h+1},\ldots,y_n)=$$

$$=\sum_{i=1}^{n}(\omega_{ih}y_i+\omega_{ih}y_i)+I_h+I_h=$$

$$2\cdot\left(\sum_{i=1}^{n}(\omega_{ih}y_i)+I_h\right)>0$$

On one hand, the argument of the *sg* function in equation (6) is the above member between parenthesis (note that $S$ is a stable state therefore $y_r(t)=y_r(t+1)$ for each $r\neq h$), so it is positive. On the other hand, $y_h$ is set to -1 by equation (6) only if the $\sum_{i=1}^{n}(\omega_{ih}y_i)+I_h$ is negative, therefore we have a contradiction that is resolved only by assuming that swapping the neuron state from -1 to 1 (i.e. variable state from 0 to 1) can not improve the energy for a stable state.

As in Theorem 1 if an optimal solution does not contain a tour we can augment the function cost (i.e. the energy network) adding an edge (i.e. set to 1 a neuron) but we have just proved that this leads to a contradiction therefore the solution must contain a tour.

***Theorem 2.*** *Lemma 1 is valid even for networks containing fixed neurons.*

*Proof.* Let us reformulate case a) and b) of Theorem 1 in order to prove that the edges added in Lemma 1 are not associated to neurons fixed to -1.

a) we have a unique sequence $s_1=\langle u_1^1,\ldots,u_{|V|}^1\rangle$ that includes every graph node and we add the edge $(u_{|V|}^1,u_1^1)$ to augment the function cost. We show that the neuron associated to this edge could not be fixed to -1. In fact if RNA algorithm set to -1 the neuron associated to edge $(u_{|V|}^1,u_1^1)$ in a previous step, $u_1^1$ and $u_{|V|}^1$ must belong to the same tour *t,* therefore the outgoing edge of $u_1^1$ and the entering edge $u_{|V|}^1$ have been fixed to 1 (step 4 of RNA algorithm). Since $s_1$ has $u_1^1$ as the initial node of the sequence and it has $u_{|V|}^1$ as the final node, it can not have more node than *t* (since the edges of *t* are fixed the algorithm can only add to the head or to the tail of the sequence derived from *t*), therefore $t=|V|$ but this is a contradiction because the algorithm would have found a solution for TSP and it should have stopped before removing the edge $(u_{|V|}^1,u_1^1)$.

b) The neuron associated to the edge $(u_{n_{i+1}}^{i+1},u_1^i)$ could not be set to -1 because this implies that $u_{n_{i+1}}^{i+1},u_1^i$ have belonged to the same tour *t* in a previous step and since the nodes of *t* are fixed they must still belong to the same sequence. This is in contrast with the initial hypothesis that $u_{n_{i+1}}^{i+1},u_1^i$ belong to a different sequences ($s_{i+1}$ and $s_i$).

Since the edges that we add to the solution to prove Theorem 1 have not been fixed, we can repeat the same step of its proof, then apply the same steps of Lemma 1 in order to prove Theorem 2.

### 3.4 RNA algorithm for TSP

To solve TSP (minimisation problem) we transform it in a dual form as follows:

1) given a TSP instance graph *G(V,E,P)* compute $p = \max_{(i,j)\in E} P(i,j)$,
2) build $G'(V,E,P')$ associated with the dual TSP so that $P'(i,j) = p - P(i,j)\ \forall(i,j)\in E$.

It is easy to prove that *S* is a solution for G′ if and only if *S* is a solution for G, moreover the tour cost is computed by subtracting the tour cost of dual TSP to $|V|\cdot p$.

### 3.5 RNA Computational Complex

In [Bertoni et al, 1997], the time required by algorithms RHN has been shown to range from a few seconds for the smallest instances to some minutes for the largest ones. The analysis of the worst case time complexity for RHN is very hard because it depends on two factors that are not simple to calculate: the length of the sequence of the Hopfield networks and the number of the complete updating cycles that each network requires to evolve from the initial state to the equilibrium state. Since RHN is used in RNA we can not give an analytical estimation, but we observe that if RHN has a complexity $\alpha$ then RNA has a worst case time of $\alpha\cdot(n-1)(n)$ (i.e. the max number of cycles necessary to remove all edges not belonging to the solution) where *n* is the number of nodes.

## 4 Experimental evaluation

To evaluate RNA and dual RNA algorithm we use Rate Relative Rapport (RRR) for TSP (i.e. $\frac{\text{optimal solution cost}}{\text{RNA solution cost}}$), $(RRR)^{-1}$ for dual TSP and the execution time.
Both the first and second parameters are used to evaluate RNA approximation performance i.e. how much RNA solution is near to the optimum. The third parameter measures the RNA complexity, as it depends on the number of cities.
The above parameters have been computed in two different fashions: estimating them from a set of TSP instances random generated (with a number of cities varying between six and fourteen) and measured from benchmark whose solutions are available. The optimal solution of random generated instances has been calculated using a super-polynomial branch and bound algorithm; this limits the size of instances in terms of few cities.

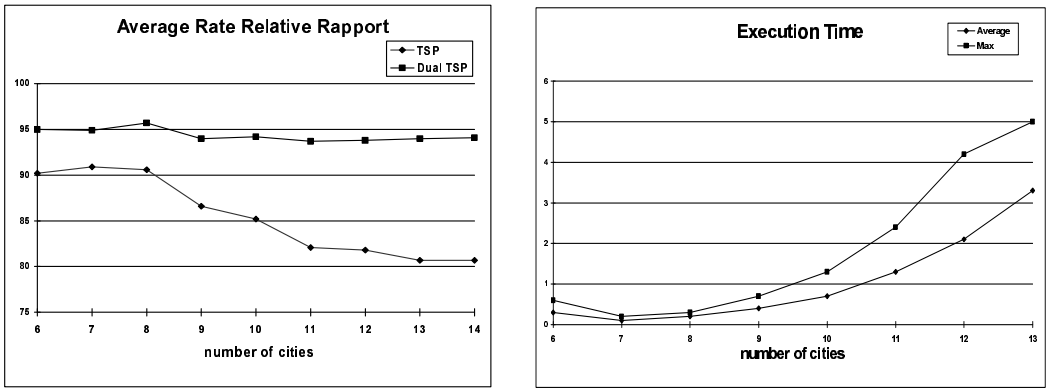



**Figure 4**: Rate Relative Rapport and Execution Time in function of the number of cities

In the graphic on the right of Figure 4 we observe that the average RNA solutions for dual TSP problem is constantly better than 94% of the optimal ones while for TSP the RRR decreases up to 81% with augmenting the number of cities. In graphic on the left of Figure 4 the execution time in function of the number of cities is shown: the average time ranges between 0.5 and 3 seconds while the max time reach 5 seconds for each problem that has a size of 13 cities. In Figure 6 we can see that RRR of dual TSP is over 95% for all instances and the average RRR for TSP among four instance is over 81%. Since these results are in agreement with the evaluation of the previous random instances we may attest that the approximation does not decrease even with instance sizes up to 26 cities.

| Benchmark instance name | GR17 | GR21 | GR24 | FRI26 |
|---|---|---|---|---|
| Number of cities | 17 | 21 | 24 | 26 |
| RRR TSP | 76.2 | 85.9 | 77.9 | 85.0 |
| RRR dual TSP | 93.8 | 97.1 | 95.5 | 97.4 |

**Figure 6**: RRR for TSP and dual TSP on instances of TSPLIB benchmark

## 5 Conclusions

In this paper an approximation technique for solving the TSP problem and its dual version based on a sequence Hopfield's networks has been described. We have proved analytically the polynomial converging of the derived algorithm RNA. Moreover two kinds of performance comparisons have been carried out in order to better understand the RNA performances: these regard both the benchmark and the branch and bound solution for TSP instances. The experiments have shown that the average solution for TSP is ever better than 81% of the optimum while for the average of dual TSP performances reach 95%. This results are good considering that asymmetric TSP does not allow to apply many performing classical approaches based on euclidean distance. The performances related to the execution time needs more experimentation to better assess the behaviour in function of the number of cities, at the moment we can see that the RNA algorithms is fast with respect to instances of small sizes.

Future work is needed to better attest the RNA performances: the classical approaches to TSP are usually measured with respect to Karp lower bound [Held and Karp, 1970] [Held and Karp, 1971]; an experimental set-up based on this optimum estimation makes a more direct comparison with all other approaches achievable. Further extensions of the algorithm are promising, for example a more complex heuristic for choosing the removing edge of a tour. An other possible extension is to consider each graph, induced by a solution, as a macro node and recursively to apply the RNA algorithm to the graph composed of macro nodes. Our feeling is that to take advantage of macro reduction should better improve both speed and approximation performances.

## References


[Alberti et al., 1995] M. A. Alberti, A. Bertoni, P. Campadelli, G. Grossi, e R. Posenato. A neural Circuit for the Maximum 2-Satisfability Problem. Mateo Valero e Antonio Gonzalez, editors, *Euromicro Workshop on Parallel and Distributed Processing*, 319-323, Los Alamitos, CA, gennaio 1995. EUROMICRO, IEEE Computer Society Press.

[Hopfield e Tank, 1985] Hopfield J. J. and Tank D. W. (1985), Neural Computation of decision in optimisation problems. *Biological Cybernetics*, (52):141-152.

[Wilson e Pawely, 1988] Wilson V. and Pawely G. (1988). On the stability of the TSP problem algorithm of Hopfield and Tank. *Biological Cybernetics*, (52):63-70.

[Karp, 1972] Karp R. (1972). Reducibility among Combinatorial Problem, pages 85-103. *Complexity of Computer Computation*. Plenum Press, New York.

[Bertoni, Campadelli e Grossi, 1997] A. Bertoni, P. Campadelli, G. Grossi. A discrete neural algorithms for the maximum clique problem: analysis and circuit implementation. *Workshop on Algorithms Engineering*, (1997),84-91.

[Potvin, 1993] J.V. Potvin, "The travelling salesman problem: a neural network perspective," *Orsa j. Comput*. 5 (1993), 328-347.

[Reinelt, 1991] G. Reinelt, "TSPLIB – A traveling salesman problem library", *Orsa j. Comput.* 4 (1992), 206-217

[Johnson and McGeoch, 1997] Johnson, D, and L. McGeoch, ``The Traveling Salesman Problem: A Case Study in Local Optimization", *in Local Search in Combinatorial Optimization*, E. H. Aarts and J. K. Lenstra (eds.), Wiley and Sons (1997).

[Held and Karp, 1970] Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Oper. Res.*, 18:1138--1162, 1970.

[Held and Karp, 1971] Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Math. Prog.*, 1:6--25, 1971.