A Multivalued Recurrent Neural Network for the Quadratic Assignment Problem

Gracián Triviño, José Muñoz, Enrique Domínguez

A multivalued recurrent neural network for the Quadratic Assignment Problem.

Gracián Triviño[1] José Muñoz[1] Enrique Domínguez[1]

[1]E.T.S. Ingeniería Informática
Universidad de Málaga
{gracian,nmunozp, enriqued}@ lcc.uma.es

Abstract. The Quadratic Assignment Problem (QAP) is an NP-complete problem. Different algorithms have been proposed using different methods. In this paper, the problem is formulated as a minimizing problem of a quadratic function with restrictions incorporated to the computational dynamics and variables $S_i \in \{1,2,..., n\}$. To solve this problem a recurrent neural network multivalued (RNNM) is proposed. We present four computational dynamics and we demonstrate that the energy of the neuron network decreases or remains constant according to the Computer Dynamic defined.

Keywords: Quadratic assignment problem, neural network, combinatorial optimization.

1 Introduction

The Quadratic Assignment Problem (QAP) is a problem formed by Koopmans and Beckmann in 1957 [1] which gives facilities to localizations. With the goal to minimize the cost function associated to the distance between localization and the flow between the facilities along with the importance of localizing the facility in certain localization. Three matrices of size n x n denominated $F=(f_{ij})$, $D=(d_{kl})$, $C=(c_{im})$, f_{ij} being the flow between the i facility and j facility; d_{kl} is the distance between the localization k and the localization l; c_{im} is the importance of localizing the facility *i* to the localization *m* . Moreover they define a variable matrix $n \times n$ size, denominated X, with the following interpretation for an element i ,m:

$$x_{i,m} = \begin{cases} 1 \text{ if the facility } i \text{ is asigned to the localization m} \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

Taking into account the previous definitions, the QAP is formulated:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{n}\sum_{l=1}^{n} f_{ij} d_{ml} x_{im} x_{jl} + \sum_{i=1}^{n}\sum_{m=1}^{n} c_{im} x_{im} \quad (\mathbf{2})$$

Subject to:

$$\sum_{m=1}^{n} x_{im} = 1 \quad i = 1, 2, ..., n \quad (\mathbf{3})$$

$$\sum_{i=1}^{n} x_{im} = 1 \quad m = 1, 2, ..., n \quad (\mathbf{4})$$

The restrictions (3) and (4) prevent assigning a facility more than one localization or that localization is given more than one facility. Therefore the matrix X can be represented as a vector φ of n size in which each *i* position is given a value k with $k = 1, 2, ..., n$ (that is $\varphi(i) = k$) which we can interpret as the variable $x_{ik} = 1$. The vector φ corresponds to a permutation of the possible solutions to the problem Γ_n being the group of aforesaid permutations. With the previous ideas the QAP is reformulated:

$$\min_{\varphi \in \Gamma_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\varphi(i)\varphi(j)} x_{i\varphi(i)} x_{j\varphi(j)} + \sum_{i=1}^{n} c_{i\varphi(i)} x_{i\varphi(i)} \quad (\mathbf{5})$$

The number of permutations of φ vector is n! Increasing n the number of permutations increases nonlinearly. Therefore we are before an NP-complete problem, as Sahni and Gonzalez have demonstrated [2].

The term $\sum_{i=1}^{n} c_{i\varphi(i)} V_{i\varphi(i)}$ of (5) is eliminated adding the costs to the matrix f_{ij} with $i = j$ .The objective function can be written as:

$$\min_{\varphi \in \Gamma_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\varphi(i)\varphi(j)} x_{i\varphi(i)} x_{j\varphi(j)} \quad (\mathbf{6})$$

QAP has been solved by different techniques among which standout branch and bound, construction methods, improvement methods and , tabu search , simulated annealing, genetic algorithms, greedy randomized adaptive search procedure , ant system and Hopfield neural network .

There are a series of well known combinatorial optimization problems which can be formulated like the QAP: graph partitioning, maximum clique, the traveling salesman problem [4], the linear arrangement problem, the minimum weight feedback arc set problem, placement of electronic modules, balancing hydraulic turbine.

RNNM is free adjustment parametric and incorporate the restrictions of the problem to the computational dynamics. The energy of the network decrease in each iteration. The results between executions of a problem have a deviation little with time of executing minor that other techniques of the literature.

The paper is organized as follows. In section 2, a new formulation for the QAP is proposed. In section 3 we design four computational dynamic inspired in the

Hopfield's synaptic potential (section 3.1) and the increment of Energy (section 3.2, 3.3 and 3.4). Experimental results comparing the performance of the simulated neural approach provided in section 4. Finally, conclusions are presented in the last section 5.

2 Multivaluated Recurrent Networks.

In this work a multivalued recurrent network is proposed where the state of neuron *i* is given by the variable S_i. So, $S_i = m$ if the facility *i* is assigned to location $m \in \{1,2,..., n\}$. For the k iteration the state of the network is defined by the vector of state $\boldsymbol{S} = \{S_1(t), S_2(t), ..., S_n(t)\}$ which represents the state of each of the neurons. Associated to each state of the net, an energy function is defined:

$$E = -1/2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} g(S_i, S_j) + \sum_{i=1}^{n} \theta_i(S_i) \qquad (7)$$

where $W = (w_{ij})$ is an *n*x*n* matrix, denominated synoptic weight matrix; $\theta_i(S_i)$ is the threshold function and *g* is a similarity function. With the computational dynamics described in [5], the network evolves decreasing its energy function until stabilizing, reaching a configuration of the vector state $\boldsymbol{S}$, so that whatever modification in one of its components, will produce an increase of the energy function or at least a state with equal energy.

The energy function defined in (7) corresponds to the QAP formulation described in (6) when $w_{ij}=-2f_{ij}$, $g(S_i, S_j) = d_{S_i S_j}$ and $\theta(S_i) = c_{iS_i}$ ..

Therefore, the decrease of the energy function of the network $E(k+1) \leq E(k)$ is guaranteed according to the computational dynamic defined in [5].

3 Computational Dynamics

The Computer Dynamics mentioned beforehand is generic and here we are going to set out 4 dynamics adapted to the QAP. We start from a vector of initial state **S** (0) which satisfies the restrictions (3) and (4) which we call pseudo feasible initial solution. The computer dynamic in each repetition must satisfy these restrictions, allowing them to be eliminated from the objective function. The first dynamic uses the concept of synoptic potential to establish if a pair of neurons should change state while the rest use the increase of energy.

3.1 Dynamic based on synaptic potential u_i

The potential synoptic is defined as

$$u_i(t) = \sum_{j=1}^{n} f_{ij} d_{S_i(t) S_j(t)} \qquad (8)$$

The dynamic of the network $\forall i \in M$ and choosing two neurons with index a and b is

$$S_a(t+1) = \begin{cases} S_a(t) \text{ if } u_a(t+1) + u_b(t+1) \geq u_a(t) + u_b(t) \\ S_b(t) \text{ if } u_a(t+1) + u_b(t+1) < u_a(t) + u_b(t) \end{cases}$$
$$S_b(t+1) = \begin{cases} S_b(t) \text{ if } u_a(t+1) + u_b(t+1) \geq u_a(t) + u_b(t) \\ S_a(t) \text{ if } u_a(t+1) + u_b(t+1) < u_a(t) + u_b(t) \end{cases} \quad (9)$$

Theorem I. The energy of the neuron network decreases or remains constant according to the Computer Dynamic defined by the expression (9).

Proof. The energy of the neuron network in the k iteration is

$$E(t) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{S_i(t) S_j(t)} = \sum_{j=1}^{n} f_{aj} d_{S_a(t) S_j(t)} + \sum_{\substack{i=1 \\ i \neq a}}^{n} f_{ia} d_{S_i(t) Sa(t)} +$$
$$+ \sum_{j=1}^{n} f_{bj} d_{S_b(t) S_j(t)} + \sum_{\substack{i=1 \\ i \neq b}}^{n} f_{ib} d_{S_i(t) S_b(t)} + \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} d_{S_i(t) S_j(t)} =$$
$$= u_a(t) + u_b(t) + \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} (f_{ia} d_{S_i(t) Sa_b(t)} + f_{ib} d_{S_i(t) S_b(t)}) + \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} d_{S_i(t) S_j(} \quad (10)$$

Given that

$$\sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} (f_{ia} d_{S_i(t) Sa_b(t)} + f_{ib} d_{S_i(t) S_b(t)}) = \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} (f_{ai} d_{S_a(t) S_i(t)} + f_{bi} d_{S_b(t) S_i(t)}) = u_a + u_b \quad (11)$$

because f y d are symmetric functions. So, we can write the energy of the neuron network.

$$E(t) = 2(u_a(t) + u_b(t)) + \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} d_{S_i(t) S_j(t)} \quad (12)$$

The energy of the neuron network in the (t+1) iteration is

$$E(t+1) = 2(u_a(t+1) + u_b(t+1)) + \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} * d_{S_i(k+1) S_j(k+1)} \quad (13)$$

We observe that:

$$\sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} d_{S_i(t) S_j(t)} = \sum_{\substack{i=1 \\ i \neq a \\ i \neq b}}^{n} \sum_{\substack{j=1 \\ j \neq a \\ j \neq b}}^{n} f_{ij} * d_{S_i(t+1) S_j(t+1)} \quad (14)$$

The difference of the computation energy between the iteration k+1 and k is

$$E(t+1)-E(t)=2(u_a(t+1)+u_b(t+1)-\left(u_a(t)+u_b(t)\right)) \qquad (15)$$

.According to the dynamics of computing (9) $u_a(t+1)+u_b(t+1)<u_a(t)+u_b(t)$ and therefore $E(t+1)<E(t)$ ▪

3.2 Dynamic based on increment of the Energy $\Delta E_{(a,b)}$

The energy increase ΔE is defined between two configurations of the vector state **S**(t+1) and **S**(t) as $\Delta E = E(t+1)\text{-}E(t)$. If $\Delta E \le 0$, the energy reduces or remains constant and **S**(t+1) has an energy less than **S**(t) associated. So that $E(t+1) \le E(t)$.

Let be two neurons *a* and *b* so that $S_a(t)=l$ y $S_b(t)=m$ are the state of these neurons.

The following state of the net will be $S_a(t+1)=m, S_b(t+1)=l$ y $S_i(t+1)=S_i(t), \forall i \notin G$ if $E(t+1) \le E(t)$. Otherwise $S_i(t+1)=S_i(t), \forall i \in M$.

The contribution to the energy of the network of two neurons *a* and ***b*** in the t-th step and vector of state **S**(t) is defined by

$$H_{a,b}^{(t)}=\sum_{j=1}^{n} f_{aj} d_{S_a(t)S_j(t)}+\sum_{\substack{i=1\\ i\neq a}}^{n} f_{ia} d_{S_i(t)Sa(t)}+\sum_{j=1}^{n} f_{bj} d_{S_b(t)S_j(t)}+\sum_{\substack{i=1\\ i\neq b}}^{n} f_{ib} d_{S_i(t)S_b(t)} \qquad (16)$$

The difference between these contributions $H_{a,b}^{(t+1)}-H_{a,b}^{(t)}$, can be interpreted as the energy difference between the state **S**(t+1) with $S_a(t+1)=m$ and $S_b(t+1)=l$ and **S**(t) with $S_a(t)=l$ and $S_b(t)=m$.

So we can write the expression $E(t+1)-E(t)=H_{a,b}^{(t+1)}-H_{a,b}^{(t)}=\Delta E_{(a,b)}$.

If, $E(t+1) \le E(t)$ the vector state **S**(t+1) is accepted as an entrance to the following repetition. Otherwise the vector state **S**(t+1)is rejected and **S**(t) is kept as the entrance to the following iteration.

The dynamic of the network $\forall i \in M$ and choosing two neurons with index *a* and *b* is:

$$\begin{aligned} S_a\ (t+1) &= \begin{cases} S_a\ (t)\ if\ \ \Delta E_{(a,b)} > 0 \\ S_b\ (t)\ \text{if}\ \Delta E_{(a,b)} \le 0 \end{cases} \\ S_b\ (t+1) &= \begin{cases} S_b\ (t)\ \text{if}\ \Delta E_{(a,b)} \le 0 \\ S_a\ (t)\ \text{if}\ \ \Delta E_{(a,b)} > 0 \end{cases} \end{aligned} \qquad (17)$$

Theorem II: The energy of the neuron network decreases or remains constant according to the Computer Dynamic defined by the expression (17).

Proof: Let be

$E(t)=\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{S_i(t)\ S_j(t)}=\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t)}$ and

$E(t+1)=\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{S_i(t+1)\ S_j(t+1)}=\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t+1)}$. The increase of energy can be written

$$E(t+1)-E(t)=\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t+1)}-\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t)} \tag{18}$$

If we update the neurons a–th and b-th so that $G=\{a,b\}\subseteq M$, the permutations **S**(t+1) and **S**(t) will coincide in the rest of the index, meaning $S_i(t+1)=S_i(t)\ \forall i\in M-G$. So we can write:

$$E(t+1)-E(t)=\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t+1)}-\sum_{i=1}^{n}\sum_{\substack{j=1\\(i\neq j)}}^{n}H_{ij}^{(t)}=H_{ab}^{(t+1)}-H_{ab}^{(t)}=\Delta E_{(a,b)} \tag{19}$$

Because $H_{ij}^{(t)}=H_{ij}^{(t+1)}\ \forall i,j\in M\wedge((i=a\wedge j\neq b)\vee(i\neq a\wedge j=b))$. If an update has been produced in the state vector it is because $\Delta E_G\leq 0$ therefore $E(t+1)\leq E(t)$. ▪

3.3 Dynamic based on maximum increment of a neuron ΔE_{Min_e}

The dynamic obtains the maximum decrease between two consecutive updates setting a ***c*** neuron**.**

Let be $c\in M$ the c-th neuron of the randomly chosen recurrent neuronal network and let be $e\in M-\{c\}$ the e-th neuron of the network obtained from the rest of the indexes.

The minimum energy increase of the update group {c, e} is $\Delta E_{Min_{c,e}}=\min_{e\in M-\{c\}}\{H_{ce}^{(t+1)}-H_{ce}^{(t)}\}$ which corresponds to the maximum energy $\left|\Delta E_{Min_{c,e}}\right|$ with which the networks can decrease exchanging the c-th neuron with the e-th.

The dynamic of the network and choosing a neuron with $c\in M$ index will be:

$$S_i\ (t+1) = \begin{cases} S_e\ (t) \text{ if } \Delta E_{Min_{c,e}} \le 0 \wedge (i = c) \\ S_c\ (t) \text{ if } \Delta E_{Min_{c,e}} \le 0 \wedge (i = e) \\ S_i\ (t) \text{ otherwise} \end{cases} \tag{20}$$

Theorem III: i) The energy of the neuronal network decreases or remains permanent according to the dynamics of the computation defined by the expression (20).

ii) The slope of the energy E is maximum for the neuron which is taken as reference in each iteration.

Proof:

i) Similar to demonstration theorem I

ii) Let be *t*-th iteration and *c* the c-th neuron which we take as reference. So we can write:

$$\frac{E(t+1) - E(t)}{\Delta t} = \min_{e \in M - \{c\}} \{H_{ce}^{(t+1)} - H_{ce}^{(t)}\} = \Delta E_{Min_{c,e}} \ge \Delta E_{(a,b)} \tag{21}$$

▪

3.4 Dynamic based on maximum increment of two configurations of the state vector ΔE_{Min}

The dynamic obtains the maximum decrease between the two consecutive updates given whatever configuration of vector state **S**.

The minimum energy increase for a configuration of vector state S is $\Delta E_{MinMax} = \min_{c \in M \wedge e \in M - \{c\}} \{H_{ce}^{(t+1)} - H_{ce}^{(t)}\}$ which corresponds with the maximum energy $\left|\Delta E_{MinMax}\right|$ with which the network can decrease exchanging two neurons.

The dynamic of the $\forall i \in M$ network will be:

$$S_i\ (t+1) = \begin{cases} S_e\ (t) \text{ if } \Delta E_{Min} \le 0 \wedge (i = c) \\ S_c\ (t) \text{ if } \Delta E_{Min} \le 0 \wedge (i = e) \\ S_i\ (t) \text{ otherwise} \end{cases} \tag{22}$$

Theorem IV:

i) The energy of the neuronal network decreases or remains constant according to the dynamics of the computation defined by the expression (22).

ii) The slope of the E energy is the maximum in each iteration.

Proof:

i) Similar to theorem I demonstration.

ii) Similar to the demonstration of Theorem II section II

4 Experimental results

The figures 1-2 show the evolution of energy for two classical problems in the literature [7] called *nug30* and *sko100a*. The running time is limited while the neural network converges to a local minimum before this deadline. For example, in Figure 1 ends at 0.36759 seconds after 1000 iterations. The minimum reached using the computational dynamic defined in the section 3.1 is 6382 which compared with the best literature solution is 6124 to get a gap = 4.2129.

In the table 1, summarize the description above for the rest of dynamics and the problems *nug30* and *sko100a.* The dynamic 3.1 and 3.2 obtain similar results but are conceptually different. The dynamic 3.3 and 3.4 are based on 3.2 and how we show in the Fig. 5 to 8 the slope of the energy is locally maximum for a neuron in iteration or the energy is locally maximum in each iteration, respectively.

The results (table 3) were compared with the results obtained by Ji et al. [6], Ahuja et al [3] and Drezner [8]. So, the gaps obtained by the RNNM are worst for the two problems. But, our algorithm reached solutions near the better solution of the literature in less time than the genetic algorithm defined in this work. By example, the problem denominate *nug30* finish in 21.6 s. with Ji´s algorithm versus 11.8443 s. with the RNMM.

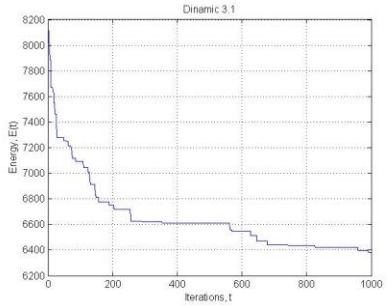
This difference is major with *sko100a*: 10447.8 s. with Ji´s algorithm versus 203.4079 with the RNMM. These results are obtained in all the QAPLIB' problems. Table 2, shows the gap and the time for problem with $n \geq 90$ and we once again deduct the same conclusion: the relations between the solution reached and the time that RNNM needed to reach is the best of the literature.

Table 1. Comparation result's betwen the differents dynamic with the problem *nug30* and *sko100a*.

Dynamic	Problem	Energy Initial	Energy Final	Gap (%)	Iterations	Time (s)
Dynamic 3.1	nug30	8170	6382	4.2129	1000	0.36759
Dynamic 3.2	nug30	8002	6374	4.0823	2000	1.2107
Dynamic 3.3	nug30	8426	6272	2.4167	1000	11.8443
Dynamic 3.4	nug30	7940	6240	1.8942	35	5.9817
Dynamic 3.1	sko100a	177946	161082	5.9736	1000	5.0183
Dynamic 3.2	sko100a	176472	159684	5.0539	3001	24.0721
Dynamic 3.3	sko100a	177904	177904	3.0671	151	20.0509
Dynamic 3.4	sko100a	178534	160904	5.8565	32	203.4079

Table 2. Gap and the time for problem with $n \geq 90$.

Problem	Gap (%)	Time (s)
wil100	1.9902	22.7115
tai256c	4.9068	123.032
tho150	7.0377	50.9777
esc128	3.125	129.6653



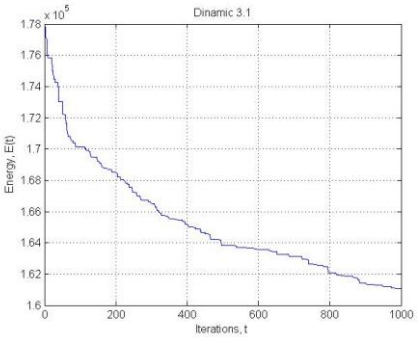


Fig. 1. Evolution of the energy for *nug30*.

Fig. 2. Evolution of the energy for *sko100a*.

Table 3. Comparative times RNMM versus other algorithm

Problem	JI	Ahuja	Drezner	RNMM
nug30	21.6	177.1	22,2	0.36759
Sko100a	10447.8	8304. 1	2013	5.0183

7 Conclusions.

In this article, we discussed a multi-valued neural network to solve the quadratic assignment problem (QAP) using four dynamic starting from pseudo feasible solutions in relation to the constraints of the problem and updating two neurons reaches finest solutions and execution times lower than those obtained with other methods. For small values of n the possibility that the network reaches a configuration which needs more than two changes in the values of the neurons increases and the network falls into a local minimum. However, for large n the possibility of reaching state configurations requiring more than two changes is less or may not occur. In the next work we will study the update of more than two neurons to avoid falling into local minimum.

In addition, the neural network is free of adjustment parameters. It is recommended for quadratic problems where finding an acceptable solution quickly is more appropriate than otherwise, or as a method to find an initial solution for further optimization with another method.

References

1. T. C. Koopmans and M. J. Beckmann, Assignment problems and the location of economic activities, Econometrica 25, 1957, 53-76.
2. M. S. Bazaraa and O. Kirca, Branch and bound based heuristic for solving the quadratic assignment problem, Naval Research Logistics Quarterly 30, 1983, 287-304.
3. R. K. Ahuja, J.B. Orlin and A. Tivari, A greedy genetic algorithm for the quadratic assignment problem, Working paper 3826-95, Sloam School of Management, MIT, 1995.
4. E. Merida Casermeiro, G. Galan Marin and J. Muñoz Perez, An Efficient Multivalued Hopfield Network for the Travelling Salesman Problem, Neural Processing Letters 14, 203-216, 2001.
5. E. Domínguez, R. Benítez, E. Mérida, y J. Muñoz Pérez, Redes Neuronales Recurrentes para la resolución de Problemas de Optimización,
6. P. Ji , Yongzhong Wu and Haozhao Liu, A Solution Method for the QuadraticAssignment Problem (QAP), The Sixth International Symposium on Operations Research and Its Applications (ISORA'06), Xinjiang, China, August 8–12, 2006.
7.] R.E. Berkard, S.E. Karisch and F. Rendl, "QAPLIB – a quadratic assignment problem library," http://www.opt.math.tu-graz.ac.at/qaplib
8. Z. Drezner. A new genetic algorithm for the quadratic assignment problem. INFORMS Journal on Computing, 115, 320–330, 2003.