K-Pages Graph Drawing with Multivalued Neural Networks*

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

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

Abstract. In this paper, the K-pages graph layout problem is solved by a new neural model. This model consists of two neural networks performing jointly in order to minimize the same energy function. The neural technique applied to this problem allows to reduce the energy function by changing outputs from both networks –outputs of first network representing location of nodes in the nodes line, while the outputs of the second one meaning the page where the edges are drawn.

A detailed description of the model is presented, and the technique to minimize an energy function is fully described. It has proved to be a very competitive and efficient algorithm, in terms of quality of solutions and computational time, when compared to the state-of-the-art heuristic methods specifically designed for this problem. Some simulation results are presented in this paper, to show the comparative efficiency of the methods.

1 Introduction

In the last few years, several graph representation problems have been studied in the literature. Most of them are related to the linear graph layout problem, in which the vertices of a graph are placed along a horizontal "node line", or "spine" (where K half-planes or *pages* intersect) and then edges are added to this representation as specified by the adjacency matrix. The objective of this problem is to minimize the total number of crossings (adding over all K pages) produced by such a layout.

Some examples of problems associated to this linear graph layout problem (or K pages crossing number problem) are the bandwidth problem [1], the book thickness problem [2], the pagenumber problem [3,4], the boundary VLSI layout problem [5] and the single-row routing problem [6] and automated graph drawing [7]. Another important application is the design of printed circuit boards [8], since, for the case of non-insulated wires, overlapping wires between electrical components may cause short circuits and thus may be avoided as much as possible.

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

J. Marques de Sá et al. (Eds.): ICANN 2007, Part II, LNCS 4669, pp. 816–825, 2007.

Several authors study a restricted version of this problem in which the vertex order is predetermined and fixed along the node line, and edges are drawn as arcs in one of the pages [9]. Other authors are more interested in the variant in which the node order is not fixed [10]. For this variant, it has been considered necessary to first find an optimal ordering of the vertices in order to compute the layout.

This problem is NP-hard [11,12]. So, many researchers have focused on finding efficient algorithms (some of them specially designed for the case of certain families of graphs) to solve the graph layout problem.

A comparison of several heuristics for this problem is presented in [9,13], including greedy, maximal planar, one-page, bisection and a neural heuristic, among others. Concretely, the neural model developed in [14] (and based on Takefuji and Lee's work [15,16]) was tested and obtained very good results, although authors indicate the possibility of non-convergence of this method. Due to the use of binary neurons, the model needs $2M$ neurons to represent the solution for a graph of M edges. In addition, this model is only intended to solve the 2-pages graph layout problem, and needed to preprocess the graph in order to obtain a good node ordering.

In this work we present a neural model designed to solve this problem. One of the differences of our model with the algorithms developed in literature is that there is no need of assigning a good ordering of the vertices at a preprocessing step. This optimal node order is computed by the model, as well as the relative position of the arcs.

Our model is a variant of the multivalued MREM model which has obtained very good results when applied to other combinatorial optimization problems [17,18,19,20], guaranteeing the convergence to local minima of the energy function.

2 Formal Description of the Problem

Let $G = (V, E)$ be an undirected graph where $V = \{v_i\}$ is the set of vertices and $E = (e_{i,j})$ is a symmetric binary matrix where $e_{i,j} = 1$ if edge (v_i, v_j) exists.

The K-pages book Crossing Number Problem consists in placing graph nodes on a horizontal "node line" in the plane. Every edge can be drawn as an arc in one of the half-planes (pages), which intersect that line, see Fig. 1. The objective is to minimize the number of edge crossings. This problem belongs to the class of NP-hard optimization problems, even if node order is fixed and the number of pages is $K = 2$.

An example of linear embedding of the complete graph K_7 in 4 pages, with 0 crossings, is drawn in Fig. 1. In this figure, we can observe the representation of pages 1 and 2 in the left side and pages 3 and 4 in the right hand side. First and third pages are represented as half-planes above the node line, while the second and fourth pages are under the node line.

Crossings Detection

Let us consider 4 positions in the node line verifying $1 \leq i < k < j < l \leq N$, where i, j, k and l are assigned to nodes v_i, v_j, v_k and v_l. Then, edges (v_i, v_j) and (v_k, v_l) cross each other if, and only if, both are represented (drawn) in the same page.

In Fig. 2, we can observe that edges (v_i, v_j) and (v_k, v_l), represented in the node line and with endpoints $i < k < j < l$, produce a crossing, whereas if $i < j < k < l$ they do not, when both are represented in the same half-plane.

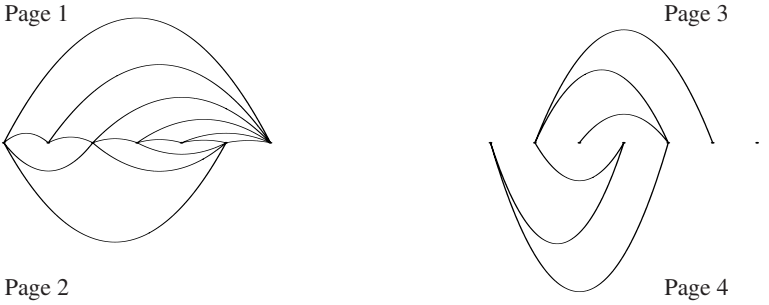

Fig. 1. Optimal linear layout for K_7 in 4 pages

Fig. 2. Crossing condition $i < k < j < l$

It seems reasonable to define $V_{v_a,v_b} = k$ to indicate that the edge (v_a, v_b) will be represented in the k-th page or half-plane. If the edge does not exist, we will denote $V_{v_a,v_b} = 0$ for simplicity.

These definitions allow us to define the number of crossings by means of the cost function:

$$C = \sum_{i} \sum_{k>i} \sum_{j>k} \sum_{l>j} \delta(V_{v_i,v_j}, V_{v_k,v_l})(1 - \delta(V_{v_i,v_j}, 0)) \tag{1}$$

where $\delta(x, y) = 1$ if $x = y$, otherwise it equals 0 (Krönecker delta function).

In Eq. (1), the term $\delta(V_{v_i,v_j}, V_{v_k,v_l})$ expresses that edges (v_i, v_j) and (v_k, v_l) will be drawn in the same page, whereas $(1 - \delta(V_{v_i,v_j}, 0))$ indicates that the edge exists.

3 Previous Heuristics

Cimikowski [9,13] presented a comparison of some heuristic approaches to solve the 2-pages graph drawing problem. All of them, except the neural one developed therein, can be extended to solve the K-pages problem. We make here a brief summary of them:

- *Edge-length heuristic* (e-len): This heuristic initially orders all edges by their length (the length of edge (a, b) is $|b - a|$). Intuitively, longer edges are most likely to produce a big number of crossings than shorter edges and hence should be embedded first in the layout. So, each edge is sequentially added to the page of smallest increase in the number of crossings.
- *One-page heuristic* (1-page): This heuristic initially embeds all edges in the first page. After this, a "local improvement" phase is carried out, in which each edge is moved to the page with the smallest number of new crossings. Edges are selected for movement in order of non-increasing local crossing number, that is, the number of crossings involving an edge.

– *Greedy heuristic* (greedy): In this method, edges are sorted according to the node index of its ends, that is, first, all edges $(1, i)$ (in increasing order of i), then $(2, i)$, etc. Edges are added, in this order, to the page which results in the smallest increase in the number of crossings.

4 The Neural Model MREM

It consists in a series of multivalued neurons, where the state of i-th neuron is characterized by its output (v_i) that can take any value in any finite set $\mathcal{M}$. This set can be a non numerical one, but, in this paper, the neuron outputs only take value in $\mathcal{M} \subset \mathbb{N}$.

The state vector $\boldsymbol{V} = (v_1, v_2, \ldots, v_N) \in \mathcal{M}^N$ describes the network state at any time, where N is the number of neurons in the net. Associated with any state vector, there is an energy function $E : \mathcal{M}^N \to \mathbb{R}$, defined by the expression:

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

where $W = (w_{i,j})$ is a matrix, $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is usually a similarity function since it measures the similarity between the outputs of neurons i and j, and $\theta_i : \mathcal{M} \to \mathbb{R}$ is a threshold function. At each step, the state vector will be evolving to decrease the value of the energy function.

The cost function (number of crossings in the graph given by Eq. (1)), must be identified with the energy function of Eq. (2). As a result, we obtain $w_{i,j} = 1$ if $i < j$ and 0 otherwise. The similarity function $f(v_i, v_j)$ and the threshold θ_i can be expressed as:

$$f(v_i, v_j) = -2\sum_{k}\sum_{l>k} \delta(V_{v_i,v_k}, V_{v_j,v_l})(1 - \delta(V_{v_i,v_k}, 0))$$

$$\theta_i(v_i) = -\sum_{j} w_{i,j} \sum_{k\leq j}\sum_{l>k} \delta(V_{v_i,v_k}, V_{v_j,v_l})(1 - \delta(V_{v_i,v_k}, 0))$$

To solve our problem we have considered two MREM neural networks:

– The first network (the 'vertices' net) will be formed by N neurons, being N the number of nodes in the graph. Neurons output (the state vector) indicate the node ordering in the line. Thus, $v_i = k$ will be interpreted as the k-th node being placed in the i-th position in the node line. Hence, the output of each neuron can take value in the set $\mathcal{M}_1 = \{1, 2, \ldots N\}$.
– The second network (the 'edges' net) will be formed by as many neurons as edges in the graph, M. The output of each neuron will belong to the set $\mathcal{M}_2 = \{1, 2, \ldots, K\}$. As mentioned before, for the arc (v_i, v_j), $V_{v_i,v_j} = k$ will indicate that (v_i, v_j) will be embedded in the k-th page. For simplicity, let us denote the absence of edge (v_i, v_j) as $V_{v_i,v_j} = 0$.

Initially, the state of the 'vertices' net is randomly selected as a permutation of $\{1, 2, \ldots, N\}$, and the initial state of the 'edges' net is a random element from $\mathcal{M}_2^M =$

$\{1, 2, \ldots, K\}^M$. At any time, the net is looking for a better solution than the current one, in terms of minimizing the energy function.

In this paper, we study the permutation of two nodes and the change in the location of an edge. These produce the energy increment given in the next subsections. As an additional technique for improvement, we have also considered changes in the position of four edges ('change-4'), since our studies have demonstrated that this dynamics is able to undo some crossings which can not be broken just by changing one edge position.

4.1 Permutation of Two Nodes

When two vertices v_a and v_b permute their order a and b in the node line, we should take into account that the unique edges changing their position (and therefore changing the number of crossings) are those that have exactly one endpoint in $\{v_a, v_b\}$.

Let us study the increase in the number of crossings depending on the relative positions of the endpoints.

Consider a position x in the line and let us see how the number of crossings with the edge (v_x, v_a) is modified when it becomes the edge (v_x, v_b) since nodes a and b permute their positions. Hence, the arc represented with endpoints (x, a) will be drawn, after the update, with endpoints (x, b), and the unique edges modifying the number of crossings due to the change must be in the same page and must have an endpoint v_s represented between a and b ($a < s < b$) and the other, v_t, outside that interval (($t < a$) or ($t > b$)). Some cases, depending on the position of x, are considered:

1. Case $x < a < s < b$: As shown in Fig. 3 (1), if $t < x < a < s < b$ the number of crossings is increased in one unit, since the edge (t_1, s) crosses the arc (x, b), but not (x, a). If $x < t < a < s < b$, a crossing disappears (the arc (t_2, s) cuts (x, a) but not (x, b)) and, at last, if $x < a < s < b < t$, the number of crossings will be increased in 1 unit (analize the arc (s, t_3)).
2. Case $a < x < b$: As shown in Fig. 3 (2), if $t < a < x < s < b$, or $a < x < s < b < t$, a new crossing is introduced (represented by the cuts of arcs (s_2, t_1) and (s_2, t_2) with the new edge (x, b)), whereas if $t < a < s < x < b$ or $a < s < x < b < t$ the number of crossings is reduced since crossings of (s_1, t_1) and (s_1, t_2) with (a, x) disappear.
3. Case $a < s < b < x$: A crossing is introduced if $a < s < b < t < x$ (arc (s, t_2)) and will be erased if $t < a < s < b < x$, or, $a < s < b < x < t$ (arcs (t_1, s) and (s, t_3)), as shown in Fig. 3 (3).

We must also take into account the change in the number of crossings with edges (v_x, v_b). Its study is similar to the already made for (v_x, v_a), it suffices to permute the literals a and b and to change the sense of the inequalities.

Finally, let us consider changes in the number of crossings produced between edges (v_a, v_x) and (v_b, v_y). All possible changes are shown in Fig. 4. There are different cases:

1. Case $a < y < x < b$:
   - Edges (a, x) and (y, b) (Fig. 4 (1)) are transformed into (x, b) and (y, a) (Fig. 4 (2)), vanishing the existing crossing.
   - Edges (a, y) and (x, b) (Fig. 4 (2)) are transformed into (b, y) and (x, a) (Fig. 4 (1)), causing the apparition of a crossing.

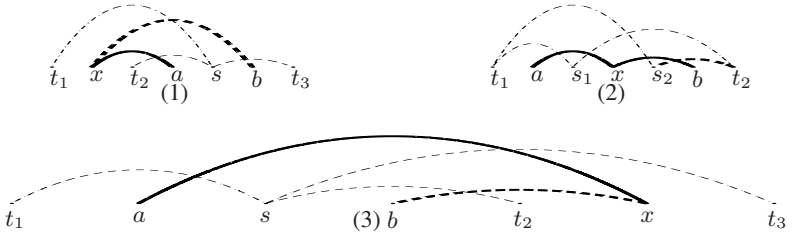

Fig. 3. Changes in the number of crossings when permuting nodes v_a and v_b, represented at positions a and b. An edge represented by the arc (a, x) will be transformed into the arc (b, x).

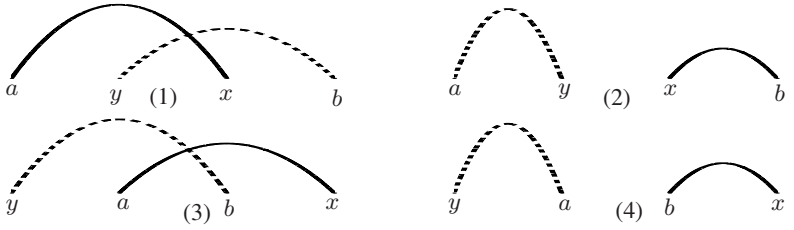

Fig. 4. Changes in the number of crossings when permuting nodes v_a and v_b. Edges represented by arcs (a, x) and (y, b) are transformed into arcs (b, x) and (y, a).

2. Case $y < a < b < x$:
   - When edges (a, x) and (y, b) (Fig. 4 (3)) are transformed into (x, b) and (y, a) (Fig. 4 (4)), a new crossing is formed.
   - Arcs (a, y) and (x, b) (Fig. 4 (4)), are transformed into (b, y) and (x, a) (Fig. 4 (3)), and a crossing is eliminated.

We can derive an explicit formula for the increase of energy related to all these cases, just by considering that Eq. (2) (the number of crossings) can be rewritten as:

$$E = \sum_i \sum_j w_{i,j} \sum_k w_{j,k} \sum_l w_{k,l} \delta(V_{v_i,v_k}, V_{v_j,v_l})(1 - \delta(V_{v_i,v_k}, 0))$$

and by denoting $g(x, y, s, t) = \delta(V_{x,s}, V_{y,t})(1-\delta(V_{x,s}, 0))$, then the increase of energy caused by the permutation of nodes a and b is given by:

$$\begin{aligned}
\Delta E = & \sum_{i\in\{a,b\}} \sum_j w_{i,j} \sum_k w_{j,k} \sum_l w_{k,l} \left(g(v_i, v_j, v_k, v_l) - g(v'_i, v_j, v_k, v_l)\right) \\
& + \sum_i \sum_{j\in\{a,b\}} w_{i,j} \sum_k w_{j,k} \sum_l w_{k,l} \left(g(v_i, v_j, v_k, v_l) - g(v_i, v'_j, v_k, v_l)\right) \\
& + \sum_i \sum_j w_{i,j} \sum_{k\in\{a,b\}} w_{j,k} \sum_l w_{k,l} \left(g(v_i, v_j, v_k, v_l) - g(v_i, v_j, v'_k, v_l)\right) \\
& + \sum_i \sum_j w_{i,j} \sum_k w_{j,k} \sum_{l\in\{a,b\}} w_{k,l} \left(g(v_i, v_j, v_k, v_l) - g(v_i, v_j, v_k, v'_l)\right) \qquad (3)
\end{aligned}$$

where $v'_s = v_a$ if $v_s = v_b$; $v'_s = v_b$, if $v_s = v_a$; otherwise $v'_s = v_s$.

4.2 Change of the Position of an Edge

When the edge with endpoints v_a, v_b is represented in a given page and its location changes from its current page to the k-th page, an increase (or decrease) of the energy function (number of crossings) is produced and is given by

$$\Delta E(k) = (1 - \delta(V_{v_a,v_b}, 0)) \sum_{a<s<b} \sum_{(t<a)\vee(t>b)} (2\delta(V_{v_a,v_b}, V_{v_s,v_t}) - 1) \cdot \\ \cdot (1 - \delta(V_{v_s,v_t}, 0)) \cdot \max\{\delta(V_{v_s,v_t}, V_{v_a,v_b}), \delta(V_{v_s,v_t}, k)\} \quad (4)$$

This expression can be obtained from Eq. (2), by simplifying the difference between the number of crossings before and after the possible change, since edges (v_x, v_y) with both endpoints located between a and b in the node line ($a < x < b$, $a < y < b$), or both placed outside the interval $[a, b]$, i. e., $x, y \notin [a, b]$, do not contribute to modify the number of crossings with (v_a, v_b), which is the changed edge.

In the improvement technique, since four edge positions are to be changed, the procedure to compute the energy increment consists in analysing independently the increase produced by the assignment of each individual edge to each of the K pages. Then, it must be taken into account the possible crossings between the selected edges. This produces a hyper-matrix $\boldsymbol{\Delta E} = (\Delta E(k_1, k_2, k_3, k_4))_{k_1,k_2,k_3,k_4}$ which represents the increase of energy the i-th considered edge is moved to the k_i-th page.

5 Implementation for K-Pages Book Graph Layout Problem

Several dynamics can be used to solve this problem with our model, but we have chosen the following one due to its simplicity and efficiency:

1. Initialization: Given a graph with N nodes and M edges, a random feasible initial configuration $\boldsymbol{V}_0$ is selected for the location of the nodes. This initial state will be a permutation of the set of node indices $\{1, 2, \ldots, N\}$.

   For the edges set, the initial state vector $\boldsymbol{W}_0$ will be a random element of the set $\{1, 2, \ldots, K\}^M$. As usual, the output $V_{v_i,v_j} = k$ means that the arc (i, j) will be embedded in the k-th page.
2. Two positions a and b in the node line are selected in all possible ways.

   Firstly, the net studies the increase of energy when vertices v_a and v_b are permuted by using Eq. (3). If the energy is reduced, the net permutes the vertices: $v_a(t + 1) = v_b(t)$, $v_b(t + 1) = v_a(t)$. Secondly, the net studies to change the page in which the edge (v_a, v_b) is located. To this end, the increase of energy given by Eq. (4) is computed and the change $V_{v_a,v_b}(t + 1) = k_0$ is done, where $\Delta E(k_0) = \min_{1\le k\le K} \Delta E(k)$.
3. If all possible combinations of two nodes positions and all possible edge locations have been considered and no change has been made, both networks have converged to a local minimum of the energy function (the cost function) and state vectors represent the obtained solution.

   In this case, in order to break some crossings, we apply the additional improvement technique ‘change-4’. For each possible combination of 4 edges, the

net studies the hyper-matrix $\boldsymbol{\Delta E}$, computes $\min_{k_1,k_2,k_3,k_4} \Delta E(k_1,k_2,k_3,k_4)$ and the corresponding update is performed.

6 Simulation Results

In this Section we test the performance of our model and compare it with some of the heuristic methods proposed in [9,13] for a test set formed by graphs belonging to well-known graph families. Concretely:

- Complete graphs K_n, where all n nodes are interconnected (no self-connections). the graph K_n has $\frac{n(n+1)}{2}$ edges.
- Circulant graph $C_n(a_1,\ldots,a_k)$, where $0 < a_1 < \ldots < a_k < \frac{n+1}{2}$ is a graph with n vertices such that vertex i is adjacent to vertices $i \pm a_1, \ldots, i \pm a_k (\mathrm{mod}(n))$. The circulant graph $C_n(a_1,\ldots,a_k)$ has $n \cdot k$ edges.

In order to test the efficiency of out method, we use the heuristics given in [9]. Concretely, those named *e-len*, *1-page* and *greedy*, described above. In addition, comparative results of Cimikowski's neural model (CN) are presented in the case $K = 2$.

We must note that CN is a neural model which needs the fine-tuning of some parameters, and our model does not. Also, CN needs of a preprocessing step in which graph nodes are ordered and then remain fixed along the iterations. Our model is able to dynamically obtain a very good ordering.

For every graph, 10 independent executions of our model were performed.

For the case of $K = 2$ pages, results are shown in Table 1. It can be observed that, although CN obtains very good results, our proposal is able to achieve better solutions. Considered heuristics obtain good solutions, but do not perform better than our proposal.

Table 1. Comparative results of our model for the 2-pages graph problem, for the considered test graphs. Numbers between parentheses indicate the average number of crossings in all the 10 executions of our model, if this average differs from the minimum obtained.

Graph	$\|V\|$	$\|E\|$	Prop.	CN	e-len	1-page	greedy
K_6	6	21	3	3	3	4	5
K_7	7	28	9	9	11	9	13
K_8	8	36	18	18	18	30	27
K_9	9	45	36	36	42	50	50
K_{10}	10	55	60	60	80	92	84
$C_{20}(1,2)$	20	40	0 (3.8)	2	0	0	0
$C_{20}(1,2,3)$	20	60	19 (24.2)	24	36	48	40
$C_{20}(1,2,3,4)$	20	80	74 (79.3)	74	90	118	108
$C_{22}(1,2,3)$	22	66	22 (26.9)	26	40	54	44
$C_{22}(1,3,5,7)$	22	88	198 (226.5)	200	306	294	286
$C_{24}(1,3)$	24	48	11 (19.4)	14	22	16	22
$C_{26}(1,3)$	26	52	11 (18.8)	16	24	16	24
$C_{28}(1,3,5)$	28	84	80 (98.0)	86	138	138	130
$C_{30}(1,3,5)$	30	90	92 (113)	96	148	150	140

Table 2. Results for $K = 3$ and $K = 4$ pages problems. Numbers between parentheses indicate the average number of crossings in all the 10 executions of our model, if this average differs from the minimum obtained.

	$K = 3$				$K = 4$			
Graph	Prop.	e-len	1-page	greedy	Prop.	e-len	1-page	greedy
K_6	0	1	1	1	0	0	0	0
K_7	2	4	3	4	0	1	1	1
K_8	5	6	8	12	0	4	4	4
K_9	9	15	23	19	3	7	8	8
K_{10}	20	27	49	40	7	12	22	19
$C_{20}(1,2)$	0 (0.9)	0	0	0	0 (0.1)	0	0	0
$C_{20}(1,2,3)$	4 (8.5)	2	10	2	0 (1.4)	1	2	1
$C_{20}(1,2,3,4)$	21 (27.5)	60	71	45	11 (15.3)	6	15	4
$C_{22}(1,2,3)$	5 (9.4)	5	12	4	0 (3.7)	1	1	1
$C_{22}(1,3,5,7)$	96 (106.7)	141	184	166	38 (49.4)	85	101	99
$C_{24}(1,3)$	2 (7.3)	0	12	0	0 (3.6)	0	0	0
$C_{26}(1,3)$	5 (7.7)	2	14	2	0 (2.4)	1	1	1
$C_{28}(1,3,5)$	31 (44.4)	70	71	55	11 (20.8)	29	31	32
$C_{30}(1,3,5)$	43 (53.6)	59	73	61	19 (29.1)	26	30	37

For larger problem sizes, $K = 3$ and $K = 4$, we can observe the same fact as above, since our proposal outperforms the heuristic methods in most cases. This means that our method is a significant improvement for this problem, since these heuristics are well-known to perform very well [13].

Regarding computational time, it must be noted that, although spending more time in obtaining a solution, our method is able to achieve good solutions on average, while the heuristics herein considered always achieve the same solution, no matter the number of times they are executed.

7 Conclusions and Future Work

In this work we have presented a neural model especially designed to solve some kinds of combinatorial optimization problems. This model is a variant of the multivalued model MREM formed by two networks. The dynamics of each of these networks depends on the outputs of the other network, and they are updated alternatively, to reach an equilibrium state corresponding to a local minimum of the common energy function.

We have tested our model with the well-known K-pages graph linear layout problem from graph theory. The proposed model avoids some of the drawbacks of other models in specialized literature, like the absence of convergence guarantees or the fine-tuning of parameters. In addition, it does not need a pre-processing stage to obtain a good node ordering, since it can be achieved dynamically.

By using some test instances, we have observed that our model is, at least, comparable to the other models, and it is able to achieve, in many cases, better solutions.

Future research lines cover aspects such as developing new dynamics for the model which could help to achieve better results. This model is also applicable when the node line is not a straight line, it can be a circle, or another geometry.

References

1. Chinn, P.Z., Chvátalová, L., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices – a survey. J. Graph Theory 6, 223–253 (1982)
2. Kainen, P.C.: The book thickness of a graph, ii. Congr. Numer. 71, 127–132 (1990)
3. Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: a layout problem with applications to vlsi design. SIAM J. Alg. Disc. Meth. 8, 33–58 (1987)
4. Malitz, S.M.: On the page number of graphs. J. Algorithms 17(1), 71–84 (1994)
5. Ullman, J.D.: Computational Aspects of VLSI. Computer Science Press (1984)
6. Raghavan, R., Sahni, S.: Single row routing. IEEE Trans. Comput C-32(3), 209–220 (1983)
7. Tamassia, R., Di Battista, G., Batini, C.: Automatic graph drawing and readability of diagrams. IEEE Trans. Syst. Man. Cybern. SMC-18, 61–79 (1988)
8. Sinden, F.W.: Topology of thin films circuit. Bell Syst. Tech. Jour. XLV, 1639–1666 (1966)
9. Cimikowski, R.: Algorithms for the fixed linear crossing number problem. Discrete Applied Mathematics 122, 93–115 (2002)
10. Nicholson, T.A.J.: Permutation procedure for minimising the number of crossings in a network. Proc. IEE 115(1), 21–26 (1968)
11. Garey, M.R., Johnson, D.S.: Crossing number is np-complete. SIAM J. Alg. Disc. Methods 4(3), 312–316 (1983)
12. Masuda, S., Nakajima, K., Kashiwabara, T., Fujisawa, T.: Crossing minimization in linear embeddings of graphs. IEEE Trans. Comput. 39(1), 124–127 (1990)
13. Cimikowski, R.: An analysis of some linear graph layout heuristics. J. Heuristics 12, 143–153 (2006)
14. Cimikowski, R., Shope, P.: A neural-network algorithm for a graph layout problem. IEEE Transactions on Neural Networks 7(2), 341–345 (1996)
15. Takefuji, Y.: Design of parallel distributed cauchy machines. In: Proc. Int. Joint Conf. Neural Networks, pp. 529–536 (1989)
16. Takefuji, Y.: Neural Network Parallel Computing. Kluwer Academic Publishers, Dordrecht (1992)
17. Mérida-Casermeiro, E., Galán-Marín, G., MuñozPérez, J.: An efficient multivalued hopfield network for the travelling salesman problem. Neural Processing Letters 14, 203–216 (2001)
18. 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)
19. 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)
20. López-Rodríguez, D., Mérida-Casermeiro, E., Ortiz-de Lazcano-Lobato, J.M., 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)