Nedbalek Jakub
VŠB – Technical University of Ostrava, Ostrava Poruba, Czech Republic

New type of neural networks for rendering graph points

Keywords

description, gantt, graph, RBF 2 neural network

Abstract

The paper demonstrates new approach of rendering the graph point series called gantts. The gantts are placed in the two dimensional graph which contains the information about available production sources in the real manufacturing process. The gantt is defined as one dimensional coloured dash which has unique position. To have the interaction with a user, gantts are accompanied with the description text giving detailed information about each gantt. All gantt descriptions must be displayed without overlapping with each other. To optimize this task, the modified version of the RBF neural network with biases is applied. With respect to the similarity to the RBF structure, the new type of neural network is named RBF 2. We also give the picture of positive and negative attributes of the solution based on the neural network architecture.

1. Introduction

In the K2® ERP (Enterprise resource planning) system, the human operator can see the screen with the tree structure and graph. The tree structure content includes the items of the production process. These items differ in its kind. The superior element of the production items is operation, which belongs to the specific document called job card.

The two dimensional graph contains the inferior elements of the production item called the *source*. These items have subordinate items, that are called the *calendars*, describe the capacitive availability of the *source* item in the specific time interval. Generally, the *x*-axis depicts the constant time interval selected in advance and the *y*-axis shows the sources names, which depends on the topically chosen tree item. If we choose the superior operation or the job card type item in the tree, the graph must show all sources subordinated to the specifically chosen operation or job card.

Each calendar is depicted as a *gantt*. Gantt is the special chart series type; it's a dash with predefined color. For gantt series type, we ignore its *y* size and take into account only length in *x*. Gantts cannot overlap with each other; every single gantt has its unique position [1].

User can adjust the type of gantt description by clicking on the graph menu option. It is allowed to have the number and name of the superior job card or name of the final product. This gets us to the main point of problem.

2. Description

In the usual practice, user almost always wants to have some type of gantt description to be pictured. The reason to do this is that the gantts accompanied with the text give us better lay out of the planned calendars and their assignment to the specified job card. Text labeled gantts stop being "anonymous" and enable user to have convenient feedback.

If the number of gantts in a graph is small enough, maximally up to 100, there is no problem to develop the algorithm of displaying the gantt descriptions without overlapping each other. If the number of gantts is small enough, we do not have to take into account the elapsed computation time, because the computation takes negligible amount of time.

Problems with inadequate long computation time occur in the real world usually. If there is around 10^3 gantts in the graph, the computation can take up to one minute (for the pc, where test took place) in the case of badly optimized algorithm. Too long time makes the screen "stack" without any logical response to the operator. It is necessary to know, that the procedure of displaying gantt descriptions is called not only if the user choose the tree item, but also in the arbitrary operations in the graph itself. User operations in the graph comprise zooming,

scrolling and other similar actions that urge to refresh the position of gantt descriptions because these actions also change the position of gantts themselves.
There is also another problem with a large number of gantts in the graph. If we have too many gantts in the graph, there is a high chance of descriptions overlapping. That provides an uneasy survey for the user.
The target is to create the algorithm, which correctly assigns the description to the gantt without any overlapping of descriptions themselves. In the algorithm, there should be also taken into account the time necessary to compute the result. This means that the algorithm should be written with regard to the operator who needs to obtain results as quickly as possible. It is also suitable to depict as many non-overlapping gantt descriptions in the graph as possible to give the operator completed information about available items.
The paper is derived from [2] but corrects the potential mistakes in the previous solution and shows new prospects of the RBF 2 structure.
Our solution is applied in the *Production* module of the K2® ERP system. With respect to the computer language in which the software itself is written, the neural network is implemented in Delphi code.

2. The solution proposal

The main task of our neural network is to classify, which gantt belongs to the input point according to the criteria of least distance between the point and gantt. The point equals to the last point of the gantt description to which we need to find either a) the nearest non – colliding gantt with higher *x* coordinate (forward direction) or b) the nearest gantt description on a lower *y* coordinate (backward direction). Result of the forward check is the gantt (with higher *x* coordinate), which description we want to display (the gantt is not covered with previously displayed description). The result of the backward check is the gantt (with lower *y* coordinate), which description could collide with the already displayed description that is obtained by the forward check. Whether the description will be shown in the graph depends on the result of the geometry comparison of border points.
To tackle the problem of descriptions overlapping, it is possible to apply the RBF 2 neural network. Its architecture was evolved from the well known RBF [3], [4]. See the *Figure 1.*

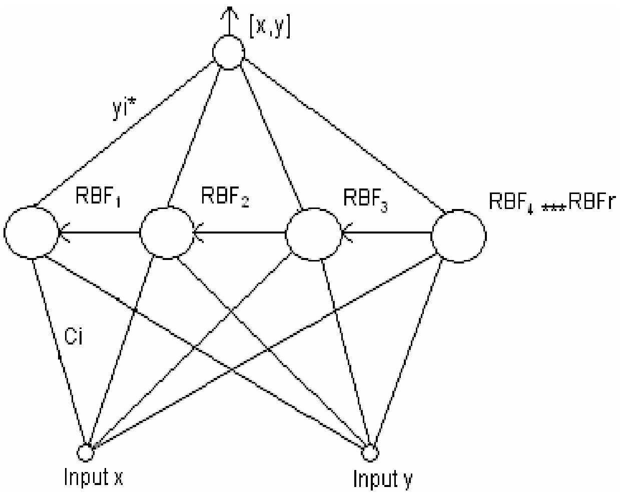

Figure 1. The RBF 2 neural network.

This network consists of three layers – distributive, hidden and output one.

2.1. Distributive layer

The distributive layer serves to transfer the signal from the input to the hidden layer. The ordered pair of numbers [x_i, y_i] (gantt is placed in two dimensional graph) is distributed to the i-th neuron. If the input vector has two coordinates, the distributive layer will have two inputs.

2.2. Hidden layer

The hidden layer contains the neurons with the activation function. The possible activation function can be

$$y^* = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2} \quad [5] \qquad (1)$$

which describes distance between the $\bar{c}$ center of a neuron and the arbitrary $\bar{x}$ input [3], [4]. The $\bar{c}$ center denotes to the vector, to which the neuron is trained and describes the pattern that is compared with an input. Practically, the $\bar{c}$ center refers to the gantt position.
Relation (1) is possible solution for the hidden layer of a standard RBF network. Finally, with respect to the time spare, the altered relation (1) was applied. The modification consists in the fact, that the square can be neglected in the case, if we know when the $\bar{x}$ input is less than $\bar{c}$ center (we know that because the gantts have ascending order according to their position). The sum across *i* is also ignored as we take into account only one coordinate (distance is counted separately for each coordinate – the collision can occur for *x* and *y* independently).

Our modified network activation function for the *x* axis equals

$$\varphi = x - c \ \text{ for } x \geq c\,, \tag{2}$$

$$\varphi = Grafx_{\max} - Grafx_{\min} + 1 \ \text{ for } \quad c > x\,, \tag{3}$$

for the backward and

$$\varphi = c - x \ \text{ for } c \geq x\,, \tag{4}$$

$$\varphi = Grafx_{\max} - Grafx_{\min} + 1 \ \text{ for } x > c\,, \tag{5}$$

for the forward check. $Grafx_{max}$ is the maximum and $Grafx_{min}$ is the minimum of *x*-axis. Maximum and minimum of x-axis are date and time values, that are optionally set by user. For instance, the minimum can be the beginning and the maximum can be the end of a calendar year. The maximal φ is greater than the *x* size of the graph that is always greater than the absolute distance of *c*-*x*.
To check the *y* coordinate, we will use relations:

$$\varphi = 0 \qquad \text{for } h > c - y \tag{6}$$

and

$$\varphi = 1 \text{ for } h < c - y\,, \tag{7}$$

where *h* is the constant height of gantt description. Activation function for *y* coordinate is consequently included only in the backward check. It is because the forward check is used to find the first gantt, which does not collide with the last displayed description on the same *y* line. If there is no such gantt, the first gantt on the nearest subsequent *y* line will be used.
The idea of relation (1) was applied in [2]. If we want to take into account the neuron biases, we can simply add to our activation function

$$y^* = \varphi + \Theta \tag{8}$$

where Θ stands for the bias. The amendment in (8) merely means that the result of activation function will be ignored (neuron will stay in a passive state) in case that the bias is not set to a predefined value.
The next difference of the RBF 2 is the presence of unidirectional bind between two neighbouring neurons. It means, that the neuron memorizes the $\bar{c}$ center [x,y] of the previous neuron with less position (not vice versa). This improvement is used in the training of the neural output layer.
Neurons of the hidden layer are implemented by a memory table (similar to database one, but stores all data in the memory, not physically on the hard drive) containing the index (sequence number) of neurons, their $\bar{c}$ [x,y] position, the bias flag, the position of a previous neuron and its bias. Each neuron defines directly the position of a gantt that is also the first point of description.
It is necessary to train the neural network to set the weights of a hidden layer.

2.3. The output layer

We solve a classification problem that is why the output layer has task to choose the appropriate neuron from the hidden layer. The neuron is chosen to reflect the criteria of the forward or backward check.
In the output layer, there is only single neuron to which all other neurons from the hidden layer are connected.
The output layer function is following:

$$y = f\left(\sum_{i=1}^{n} w_i y_i^*\right) \tag{9}$$

where y^* is output of the hidden layer, w_i is weight and *y* is the network output. The *f* function assigns the distance between the $\bar{c}$ center and an arbitrary $\bar{x}$ input to the value of $\bar{c}$, which currently defines the position of gantt. The RBF 2 network returns value of $\bar{c}$, which is the first (left) point of gantt description.
Setting weights of the output layer relates to the training process.

2.4. Training of the RBF 2 network

The training of the RBF 2 can be categorized in two parts – training of the hidden and the output layer.
Training of the hidden layer can be realized the way a pattern (gantt) is chosen from the set and is defined directly as a prototype. Its position describes exactly the $\bar{c}$ center of the neuron. This simple method provides the fastest way of training. Another advantage of this method consists in reflecting the data position within the input space. The hidden layer is trained backwardly according to the formulas (2), (3), (6) and (7) and forwardly according to the (4) – (7). The training process is executed during the neural network creation when the input data are disposed. The RBF 2 is forced to be trained again whenever the mutual position of

gantts in the graph is actualized (i.e. operator chose another item in the tree, etc.)

When we train the output layer, we need to set the weight according to the hidden layer. If the hidden layer neuron has the least distance between its $\bar{c}$ center and an arbitrary $\bar{x}$ input, then the weight from this input will equal one. In all other cases, the weight will be set to zero. This process of seeking the minimal distance is activated whenever there is any need to call for the forward or backward check.

Finding the minimal distance and the appropriate gantt, defined by the $\bar{c}$ center position, is optimized for time. This is enabled by the mentioned unidirectional connection between neighboring neurons. Due to that, the output neuron in each cycle across the memory table is able to "see" two neurons of hidden layer in one moment.

In the previous variant [2] of RBF 2, we neglect the network biases. In the present solution, the neuron bias is taken into account. This practically means that in the output layer training phase (looking for the minimal distance between the $\bar{c}$ center an arbitrary $\bar{x}$ input) and only for the backward check, we skip all neurons which do not have their description presence flag set – its value equals zero. The flag is additionally activated (set to nonzero value) for the neurons that successfully passed their collision (forward and backward) test and were defined as appropriate to hold and show the gantt description. Acquired information about presence of the description is later used in the backward check for another gantt in the sequence, when we need to decide, whether it is possible to display the description. The reason, why to use biases only in backward check is that in the forward check (from our gantt forth), there are no descriptions yet.

Adding the description presence flag as a neuron bias makes our output layer training phase faster and gives the user better feedback.

3. Results

We obtained all results from the actions that user typically performs – selecting tree nodes of *source* and *operation* type (some of them repeatedly). This is followed by automatic repainting of the graph surface.

Table 1. contains the results for each of the three methods. The first one, that is the original method, implements the forward and backward testing without a time optimisation, which is provided by the RBF 2. On contrary to the RBF 2 (a), the (b) version uses the neuron biases.

Table 1. The results for all methods.

User action:	Origin. method [s]	RBF 2 (a) [s]	RBF 2 (b) [s]
1. Sources aver.	30,8095	18,5220	9,5889
std.dev.	8,2622	0,1459	2,2745
2. Operations aver.	27,5607	19,4707	9,1866
std.dev.	5,1690	2,9958	1,9079
3. Operations 2x aver.	57,4603	42,6339	18,8508
std.dev.	11,5381	8,2758	3,3455
4. Operations 3x aver.	100,8735	68,2888	28,3871
std.dev.	21,3945	12,2965	7,0934
5. Main node aver.	31,1623	29,6636	5,3722
std.dev.	7,6225	5,1430	1,2236

The abbreviations in the first column from left denote: aver.- average, std.dev. – standard deviation; sources – user clicked on all nodes of the *source* type in the tree component; operations – user clicked on all nodes of the *operation* type; 2x (3x) – user clicked on the specific node type two times (three times).

Numbers in the *Table 1.* means the time necessary to compute all possible collisions among the gantt descriptions summed with the time necessary to depict the gantts with their descriptions on the graph surface. Simply expressed, it is the time which the graph needs to respond on the user action. All numbers in the second and third column of the *Table 2.* (see later) have the same meaning.

Making all these comparison tests makes sense only in case the set of displayed gantt descriptions is the same for all methods. This assumption was verified and successfully fulfilled.

*Figure 2.*shows the benefit of both versions of the RBF 2 which are compared to the original method without time optimisation.

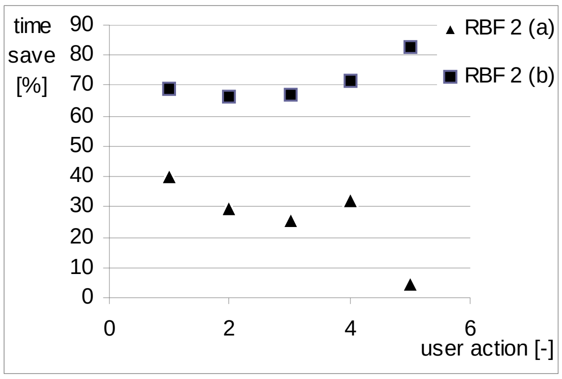

Figure 2. Time saving for the modifications of RBF 2 (a- no biases, b-with biases) with respect to the original method (see *Table 1.*)

Figure 2. demonstrates the time saving of computing time in percentual ratio (y axis) of the RBF 2 method referred to the former (original) method. The RBF 2 is depicted for both versions (a – no biases, b – with biases). The x axis includes the number of the user action (see *Table 1.*, first column from left). As we can see, the RBF 2 with the neural biases gives us better results. For instance, if we apply the RBF 2 ver. (b), we can achieve the result in computation of all possible gantt description collisions with approximately 70% of time saving with respect to the original method. The RBF 2 ver. (a) for the same computation enables typically 30% of time saving.
Finally, let us compare solely RBF 2 modifications. *Figure 3.* shows the time saving of computing time in percentual ratio (y axis) of the RBF 2 with biases referred to the RBF 2 without biases. The x axis includes the number of the same user action as in the *Figure 2.*

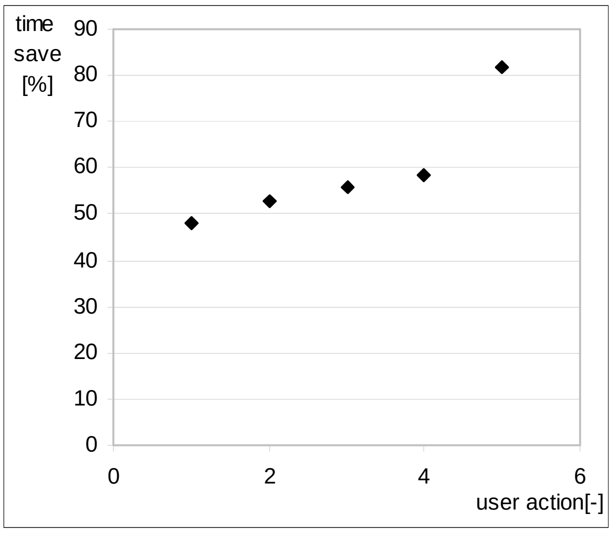

Figure 3. Time saving for the RBF 2 with biases with respect to the architecture without biases (see *Table 1.*)

In the *Figure 3*, we can see, that time saving for the version with biases is obvious – we can spare from 50% up to 80% of computing time with regard to the version without biases.

4. Conclusion

The paper demonstrates the new enhancement of the RBF 2 neural network – based method of displaying the data in a manufacturing graph. The enhancement consists in applying the neural biases, which were omitted in the previous version of the RBF 2 [2]. This method improves the contemporary state of the RBF 2 architecture, which requires more computation time to display all gantt descriptions. The RBF 2 structure itself was derived from the well known RBF, to optimize our task.
All time speed tests were realized on the Microsoft virtual computer with 2.2 GHz dual core processor and 1.5 GB of allocated RAM memory. All applications that had no reference to the tested calculations were halted during tests. Each user action was repeated more times for the later statistical handling.
The results were measured on the database, which was obtained from the real manufacturing process. This activity indirectly ensured that the graph should contain the high number of gantts. This is required because the RBF 2 must pass the stress tests in order to be applied in the practice. In other words, working with the real data provides us a good picture about the real conditions. Although we dispose of convictive results of the RBF 2 behavior, it still undergoes the stress tests presently.
The only obvious disadvantage of the RBF 2 neural network is a lack of generality of the output layer training function. That is caused by the fact that the RBF 2 neural network was developed for time optimization of gantt descriptions overlapping. To make the RBF 2 more universal, the output layer training should be altered. However, we can presume that the price for universality will be a partial loss of the time optimization ability.

5. Acknowledgements

The author of this paper would like to thank for the financial support on behalf of the research and development project num. 1M06047 (CQR), which is sponsored by the Ministry of Education of the Czech Republic.

References

[1] Gantt, H.L. (1910).Work, Wages and Profit. *The Engineering magazine*, New York, 1910. http://en.wikipedia.org/wiki/Gantt
[2] Nedbálek, J. (2010). The quality of graphical data display. Proc. ESREL 2009 Conference, Reliability, Risk & Safety: Theory and Applications, ISBN 978-0-415-55509-8, Taylor & Francis Group, pg. 2041-2044.
[3] Chen, S., Cowan, C.F.N. & Grant, P. M. (1991). Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks*, Vol. 2(2): 302-309.
[4] Yee, P.V. & Haykin, S. (2001). *Regularized Radial Basis Function Networks: Theory and Applications*. John Wiley.
[5] Šnorek, M. (2004). *Neuronové sítě a neuropočítače*. ČVUT, Praha.