Estimating PCB Assembly Times Using Neural Networks

Frans Vainio, Timo Knuutila, Michael Maier, Esa Alhoniemi, Mika Johnsson, Olli S. Nevalainen

Estimating PCB Assembly Times Using Neural Networks

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0093.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	19-Sep-2008
Complete List of Authors:	Vainio, Frans; University of Turku, Department of Computer science Knuutila, Timo; University of Turku, Department of Information Technology Maier, Michael; University of Turku, Department of Computer science Alhoniemi, Esa; University of Turku, Department of Computer science Johnsson, Mika; Valor Computerized Systems Nevalainen, Olli
Keywords:	PCB ASSEMBLY, PRODUCTION PLANNING, NEURAL NETWORK APPLICATIONS, PROBABILISTIC MODELS
Keywords (user):	placement machines, assembly time

Estimating PCB assembly times using neural networks

Frans Vainio[a], Michael Maier[a], Timo Knuutila[a*], Esa Alhoniemi[a], Mika Johnsson[b], Olli S. Nevalainen[a]

[a]*Computer Science and Turku Centre for Computer Science (TUCS), Turku University, Finland;*
[b]*Valor Computerized Systems, Turku, Finland*

Several production planning tasks in printed circuit board (PCB) assembly industry involve the estimation of the component placement times for different PCB types and placement machines. This kind of task is, for example, scheduling of jobs or line balancing for single or multiple jobs. The simplest approach to time estimation is to let the production time to be a linear function of the number of the components to be placed. To achieve more accurate results, the model should include more parameters (e.g. the number of different component types, the number of different component shapes, the dimensions of the PCBs etc.). In this study we train multilayer neural networks (MLPs) to approximate assembly times of two different types of assembly machines based on several parameter combinations. It turns out that conventional learning methods are prone to overfitting when he number of hidden units of the network is large in relation to the number of training cases. To avoid this and complicated training and testing, we use Bayesian regularisation to achieve efficient learning and good accuracy automatically.

Keywords: PCB assembly, placement machines, assembly time, production planning, neural networks, multilayer perceptron, Bayesian framework.

1. Introduction

Automated placement of electronics components on printed circuit boards (PCBs) is performed by various placement machines of different types. Each machine type is able to handle a certain set of component types. The components have different characteristics (e.g. dimensions) and this is reflected in the design of placement machines. Different mechanical solutions lead to differed nominal operation speeds, and the same machine can be run at different speeds according to the handled components types.

Production of PCBs is performed mainly in assembly lines consisting of several placement machines of different types connected to each other by the means of PCB conveyors. In addition to placement machines, a production line includes several

*Email: timo.knuutila@it.utu.fi

other production machines, like glue dispenser, reflow-oven and special type component feeding systems. Assembly lines of such a high technology are very expensive and need to be utilized as efficiently as possible.

The aim of line balancing is to optimize the operations of the machines so that the number of PCBs processed in a time unit (throughput) is maximized, see Crama *et al*. (2002) and Smed *et al*. (2003). This task is usually accomplished by maximizing the throughput of the bottleneck machine, i.e. the machine consuming the longest production time per PCB. Optimization of whole assembly lines is too complex task to be performed using one-level approaches as presented by Smed *et al.* (2003). However, the complex optimization problems are connected to each other in such a way that they can be divided in a hierarchy of simpler ones. Since the single machine optimization is at the lowest level of this hierarchy, it has to be solved each time (for each different PCB type or for each group of PCBs of similar type) when higher level optimizations are made. Consequently rapid estimations, instead of time consuming optimization calculations of PCB assembly times, are needed to form the basis for higher level optimizations.

As first approach, the production time can be estimated from the nominal (empirical) component placement time. The total production time is the sum of a constant term for feeding to and fixing the next PCB on the placement position and from a term which is a product of the number of components and the average placement time per component. Benefits of this method are simplicity and immediate results. On the other hand, its accuracy is modest and it suits only for finding an initial approximate solution for line balance optimization. When a control software (i.e., result of the optimization) for an assembly robot has been implemented and tested in production, it becomes soon clear that there are situations where the above "rule of thumb" time estimate is not accurate enough. The optimization based on nominal component times may generate a low value for the objective (time) function but the observed real assembly times can be far from that and from optimal.

To get more detailed and accurate results, assembly engineering optimizer software is in broad use. Commercial software packages for assembly engineering optimization (direct from machine producers or from third party providers) are creating assembly lines of specific characteristics and choosing machines for selected component groups from a large collection of machine types. Optimizers obtain the particular assembly times from a built-in simulator for each particular machine type. The simulators use, among other things, information about the placement sequence, operation sequence, machine timings, component coordinates, and machine geometry.

Current simulators for assembly machines are relatively accurate; the usual error marginal of the placement machine simulators is less than 2 %. On the downside, a simulator does complex computations which take a substantial running time if excessive (feeder allocation and component placement sequence) optimization is needed. This limits the use of optimization in (multi-product) line balancing and production scheduling situations where results are needed in a short time. On the other hand, the accuracy of the estimates of the placement times is vital for the success of line balancing and scheduling; even the best possible optimizer is useless if its objective function is wrong or inaccurate.

Above considerations lead to the following question: How can we get rapid and accurate estimates on PCB assembly times? As a compromise between the two above methods for time estimates, multivariate linear regression model was derived for a rotary turret machine in Laakso *et al.* (2002). This model is based on the number of component types, the total number of placements and the dimensions of the PCBs.

The model achieved good results in spite of its underlying hypothesis of linear dependency of time on the parameters. It is therefore tempting to consider the power of non-linear time estimates to these data.

In this paper, we consider multilayer feedforward networks of type MLP for estimating the assembly times. We train MLPs to approximate the objective (assembly time) function using Bayesian regulation with two very different machine types. For a turret type machine we use the same simulated data as in Laakso *et al.* (2002). The training data for gantry type machine is based on real production obtained from two different PCB assembly factories.

We begin in Section 2. with literature review. Section 3 contains a short description of rotary turret type machines and gantry type machines. Two commercial optimizer-simulators Valor Trilogy (2005) and Siemens SIPLACE Pro (2007) which were used to evaluate the "true" assembly times for MLP training and testing are reviewed in Section 4. Non-linear estimation of assembly times by using MLP networks is presented in Section 5. Experiments with the MLP network are reported in Section 6. Concluding remarks appear in Section 7.

2. Literature review

Over the recent years, a rich body of papers have been published dealing with problems of optimizing manufacturing processes and scheduling operations in PCB assembly. For a comprehensive literature review on general manufacturing operations, see for instance Pinedo, M.L. (2005) and specially on PCB assembly operations, see Crama *et al*. (2002) and Smed *et al*. (2003). Optimization of the feeder setup and component pick-and-place sequence is very important for the efficient utilization of placement machines and lines. The component pick-and-place sequencing problem is a travelling salesman problem (TSP), which is strongly NP-hard. As a result og this, most practical problem instances can not be solved to optimality in a reasonable time. Therefore, a common practice is to operate with suboptimal machine control programs as produced by suitable heuristic methods. These heuristic algorithms can generate good solutions at a reasonable computational time, but their drawback is that they are not able to guarantee optimality. For a comprehensive literature review on heuristic methods used in assembly optimization, see for instance Reeves, C.R., (1995), Ayob, M., (2005), Ayob *et al*. (2002 and 2008).

In the everyday industrial practice planning of assembly processes is made using commercial assembly planning and optimization software, like Valor Trilogy (2005) or Siemens Siplace Pro (2007). This kind of proprietary software use heuristic algorithms to solve the general assembly problem. But, there is no information available about how these software operate and which heuristic algorithms are used in them. It is only known that the optimizer software use separate programs tuned for each particular placement machine type. The running times for the optimization procedures are typically from minutes until several hours. In the general case the optimization includes both pick-and-place sequencing and component-to-feeder assignment which make the PCB assembly optimization still more complex to solve, see Leipälä, T. and Nevalainen, O. (1989).

When considering possible methods for estimating operation times of manufacturing machines one can trust on statistical techniques. Common non-linear regression methods are well known in classical statistics literature, see for instance Hald, A. (1967). In recent years, neural networks have gained much popularity due to their flexibility, versality and simplicity in usage. There are many different types of neural

networks supporting non-linear regression e.g. MLP (multi-layer perceptron), RBF (radial basis function) and SVM (support vector machine), for a comprehensive study of this topic, see for instance Bishop, C.M. (1995). Each one of these methods has its own strengths and weaknesses. It turns out that conventional learning methods are prone to overfitting when the number of hidden units of the network is large in relation to the number of training cases. To avoid this and complicated training and testing, one can use Bayesian regularisation in MLP to achieve efficient learning and good accuracy automatically, see for instance MacKay, D.J.C. (1992a and 1992b) and Neal, R.M. (2004). The number of the neurons in the hidden layer determines the flexibility (the number of free parameters) of the model. Hornik *et al.* (1989) have showed that "feedforward networks with as few as one hidden layer are capable of approximating any Borel measurable function from a finite dimensional space to another to any desired degree of accuracy, provided *sufficiently* many hidden units are available".

When working with neural networks, there are two different operation phases; constructing the optimal layout and parameters of the network on the basis of the training data, and using the constructed network to determine the output for a new input. The training phase may take a long time (many hours) but it is done off-line. More importantly the use of a given neural network is very fast which is a necessity in the context of line balancing where one has to determine the assembly time of numerous different subsets of the components on the machines of the production line. The neural network model has also the further advantage of being adaptive; one can use the measured true assembly times to quickly retrain the model parameters.

3. Operation principles of placement machines

While different placement machine types normally include parts with rather similar functionality, many details of the machine design cause that each type should be analyzed separately. Here we concentrate on two rather different, well known and widely used machine types and discuss their design on the level which is necessary for understanding their main time factors.

3.1. Turret type placement machine

Turret-type placement machines (Figure 1) have been used in particular for high volume production. In the following, the discussion refers to the high-speed Universal HSP 4795/96 placement machine. The main body of this older machine type includes a conveyor belt, a PCB holding table and a rotary turret placement head for holding the vacuum nozzles. The feeder unit moves horizontally in the x-direction on the plane of the PCB holding table and comprises the required component reels. The number of component reels in the feeder unit is limited and each reel occupies a certain number of storage places, called feeder slots. This number depends on the width of the particular feeder reel. The task of the feeder unit is to bring the proper reel to the pick-up position where the placement head picks up a component with the appropriate vacuum nozzle. After that, the turret rotates stepwise around the vertical axis until the nozzle is at the placing position (which is 180° from the pick-up position). The intermediate positions are for inspection, rotation etc. of the components. Because the rotary turret contains several placement heads, there are several components ready in the turret and the feeder movement is performed a given

number of turret rotation steps prior to the actual placement operation. Because the printing position is fixed, the table holding the PCB must

---------> INSERT FIGURE 1 HERE <----------

Figure 1. Main components of a turret type placement machine (top view).

be moved by two independent step motors to the proper location for placing the component. Usually for heavy components the speed of the holding table must be limited so that the components will stay in their proper places during the table movements. These components are usually placed last so that the limited speed does not need to be used with the smaller components.

The turret rotates on steps, and each step lasts a fixed time during which the feeder unit and the holding table can move freely (i.e., their movements are simultaneous to the rotation step and do not increase the total time for the assembly task). Mathematically speaking, we have the maximum or Chebyshev metric (L_4), where the largest time of component rotation, x-transfer, y-transfer and feeder transfer gives the final time of a particular placement operation. The control program tries to minimize the assembly time and, consequently, arranges the feeder tapes and the pick-up and placement operations so that

1. the feeder and PCB movements are short enough to be carried out during the rotation steps, and
2. if this cannot be realized, the overall length of the movements is short and long jumps occur parallel in both the feeder unit and the holding table.

Obviously, we can get accurate time estimates of the placement operations either by performing and observing the operations in the device in question itself or by simulating the operations using an optimizer-simulator. High time costs often rule out both of these methods in everyday practise and that is why there is a need for rapid and accurate estimators.

3.2. Gantry type placement machines

Nowadays, turret machines have been frequently replaced with collect-and-place machines. These are equiped with (one or more) gantries which carry rotating heads that collect the components from feeders and place them to the PCB. Collect-and-place machines can usually manipulate also bigger components that are not accepted by turret machines. The two-gantry placement machines of Siplace S-Series (Fig. 2) are typical examples of this design. Each gantry carries a collect-and-place revolver head with either 12 or 6 nozzles according to the component heights (max. component height for 12-nozzle-head is 6 mm and for 6-nozzle-head 8,5 mm). Both head types can collect a number of components and then place them sequentially on the PCB which is motionless during the placement operation. The placement heads operate in parallel on the two gantries and have their own stationary feeder slots and

own nozzle trays for both changeover and storage of nozzles. The heads can be equipped with the same or different type nozzles to handle different component types.

---------> INSERT FIGURE 2 HERE <----------

Figure 2. Main components of the Siplace S-Series two-gantry placement machine (1: gantries, 2: feeders, 3: revolver heads, 4: nozzle, 5: PCB).

4. Optimizing simulators for assembly engineering

To get an idea about how much time is needed to simulate and optimize the assembly operations of a single placement machine, we take a closer look on two commercial assembly engineering software packages. In both software packages the optimization process is similar. The process is complex and time consuming in practice but simple in theory; initial information on the assembly equipment configuration (e.g., assembly lines, placement machines, heads, nozzle changers, etc.) functions as raw data. This information is given as an input to the optimizer software which optimizes the configuration of the production line (feeders, nozzles etc.) and generates the placement programs/rules for the machines.

4.1. Optimizing simulator for rotary turret machine

The "Line Engineering Package" is a part of the "Trilogy" assembly engineering software of Valor Corporation Ltd, see Valor Trilogy (2005). The software package contains machine specific optimizers, also for our case, the Universal HSP 4795/96 turret machine. The Line Engineering Package operates on three optimization levels. At level 1 the optimizer generates a quick-and-dirty control program for the placement machine. This option suits for testing prototypes when the amounts of PCBs are small or the result of this optimization level is used as an initial solution for higher level optimization. The running time is typically *tens of seconds* and it is limited to one minute. At level 2 the optimizer is suited for normal production situations when the PCB batches are of a reasonable size. The running times are typically *few minutes* and they are restricted to one hour. The optimizer determines four reasonable feeder setups and chooses among them the most promising one. Level 3 is for the optimization of mass production and it takes typically *one working day*. The optimizer performs a full-scale optimization of the feeder setup. The difference of the value in objective function in comparison to level 2 is typically one percent. The training dataset has been generated by Laakso *et al.* (2002) using the level 2 optimization (includes reasonable component-to-feeder allocation) and the optimization time has been restricted to one hour per PCB job.

The line engineering package bases its operation on a machine simulator tuned for each particular machine type. When the solutions found by the package are carried

out in a real placement machine, the observed time differs slightly from the simulator (usually less than two percent) but the difference is stochastic with a zero mean. Nevertheless, the simulator assumes that the machine operates in ideal conditions (e.g. all components are accepted by the optical quality check and component reels in the feeder are in the sequence they are needed) and there are no production interrupts caused by some real world phenomena.

4.2. Optimizing simulator for gantry type machines

To simulate and optimize the assembly times for Siemens gantry type Siplace placement machines we used real data from PCB factories and Siemens' own, in-house optimization software, see Siemens Siplace Pro (2007). Siplace Pro assembly line control and optimization software converts basic CAD data into placement programs on Siemens Siplace placement machines and lines. Siplace Pro includes an optimizer as a separate program having another data model to optimize the placement programs at the machine and line level. There are various optimization scenarios: optimize setups for a single product, multiple products in a single family setup, or multiple products in multiple family setups. There are, similar to Valor's Trilogy package, also 3 different optimization levels. The running time for the optimization procedure can be limited by giving the number of the optimization steps or by giving a maximum running time. The training datasets for gantry type machines has been generated by Siplace Pro using the level 3 optimization (which includes also feeder allocation optimization) and the optimization time has been limited to ten minutes per PCB job.

5. Non-linear regression estimation of assembly times

Even though the PCB assembly time can be roughly estimated by using linear models, it is actually a non-linear function of the input parameters. As can be seen in Figure 8 (left panel), the input-output mapping seems to be rather smooth and continuous. However, it is also obvious that a linear estimator of the processing time is not sufficient. Therefore we now turn our attention into non-linear regression models.

Artificial neuron (perceptron) networks with linear activation function are known to be closely related to linear regression models. The need to be able to handle more complex dependencies has resulted in development of the perceptron models that include more layers and apply a non-linear function to make the activation of the perceptrons non-linear, scaled and differentiable.

The perceptron computes a single output from multiple real valued inputs by forming a linear combination according to its input weights and then putting the output through some nonlinear activation function, see Figure 3. The operation of the perceptron can be mathematically described as $y = N(\sum w_i x_i + b) = N(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$, where $\boldsymbol{w}$ denotes the weight vector, $\boldsymbol{x}$ is the vector of inputs, b is the bias and N the activation function.

---------> INSERT FIGURE 3HERE <----------

Figure 3. Artificial neuron (perceptron).

In this study, we evaluate the feasibility of a standard neural network based approach to the estimation of assembly times by using an MLP network in the regression. Our multilayer perceptron network (Figure 4) consist of three layers, there is one input layer (k input variables = k neurons), one hidden layer having n neurons and one output layer (1 output variable = 1 neuron). The number of the neurons in the hidden layer determines the flexibility (the number of free parameters) in the model. The three layer model is selected because Hornik *et al.* (1989) has showed that feedforward networks with as few as one hidden layer are capable of approximating any Borel measurable function from a finite dimensional space to another to any desired degree of accuracy, provided *sufficiently* many hidden units are available". We calculate later on in Section 6.2.2 how many hidden units are sufficiently.

---------> INSERT FIGURE 4 HERE <----------

Figure 4. MLP network having one hidden layer.

Input values of the input layer are fed as such into the layer upstream, into the hidden layer. Once the neurons for the hidden layer are computed, their activations are fed upstream to the output neuron. In a fully connected multilayer feedforward network, each neuron in one layer is connected by a weight to every neuron in the layer upstream. A bias is also associated with each of these weighted sums. The activation functions of the hidden layer are sigmoids whereas the activation function of the output neuron is linear.

When the network structure is fixed, there still exists a large variety of different algorithms and heuristics that can be used in the training of the neural network. The problems that are faced in the learning, i.e. estimation of the network parameters, include overfitting to the training data (the network learns the characteristics of the training data but is not able to generalize), slow learning rate and getting stuck in local minima of the cost function that the learning algorithm is minimizing (sum of squared errors).

One known way to avoid the overfitting problem is to use Bayesian framework having automated regularisation in practical MLP network calculations. The Bayesian framework presents uncertainties in the values of the parameters as probability density functions. Before presenting the data, the parameters are described by *a prior* probability density, which is typically quite broad to reflect the fact that we have little idea of what values the parameters should take. Once the data are presented, we can compute the corresponding *posterior* probability density. Since some values of the parameters are more consistent with the data to be presented than others, we find that the posterior distribution is narrower than the prior distribution. This phenomenon is known as Bayesian learning and is illustrated in Figure 5.

--------> INSERT FIGURE 5 HERE <----------

Figure 5. A prior distribution of weights p(w) and the posterior distribution p($w \mid D$). The most probable weight vector w_{MP} corresponds to the maximum of the posterior distribution.

In our experiments, we used the Bayesian regularisation of the network parameters, which is roughly equivalent to a technique called weight decay. The regularisation aims at keeping the weights small, and therefore prevents overfitting. As the training algorithm we used a scaled conjugate gradient method which is a faster substitute for the classical gradient descent algorithm. The classical gradient descent algorithm adjusts the weights in the steepest descent direction (negative of the gradient), the direction in which the error function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions.

The number of neurons in the hidden layer should be selected so that the model has good generalization properties, that is, its prediction error for testing data (not used in training of the model) is as low as possible. We estimate the suitable number of neurons in Section 6.2.2. Because our data sets are quite small, we were forced to use the leave-one-out (LOO) validation of the generalization error in our experiments.

6. Experiments

Because the two machine types of Section 3 are suited for different kinds of PCB assembly tasks, we had tree different sets of training data for placement time estimations. For the turret type machine the data set was taken from an earlier study Laakso et al. (2002) and for the gantry type machine two sets of real production data were obtained from two different PCB assembly factories.

6.1 Training data

The data we use in the context of *rotary turret machine* were originally generated by Laakso *et al.* (2002) for studying multivariate linear regression estimators. The data were formed by creating artificial PCBs for placing n components having c different component types on a square-like PCB with a side length of l millimetres. The locations of the components on the PCB were chosen by sampling the x- and y-coordinates independently and uniformly from the interval [0, l]. The parameter values were $n \in \{50,100, 400, 700, 1000\}$; $c \in \{1, 30, 100\}$; $l \in \{0, 50, 200, 500\}$. The generated assembly tasks were then simulated using the level 2 optimizer of the "Line Engineering Package" for each instance of the 70 PCB jobs. The simulation results were considered as the "true" placement times, which we use to train MLPs to estimate the times.

The two training data sets for *gantry type machine* were real proprietary production data, obtained from two different PCB assembly factories. The first

dataset (dataset 1) contained 63 different PCB jobs that were rather homogeneous since there were a lot of different versions of the same product. The second dataset (dataset 2) contained only 10 jobs which were from a rather different type of electronic product, in comparison to the jobs in the first dataset. Dataset 2 was not used alone since it is too small for the MLP to be used as training material. Each of these jobs was then optimized using the level 3 optimization of the Siplace Pro software to get the objective function ("true" placing times). The benefit of using this kind of real world assembly data is that the MLP can be trained using realistic inputs with corresponding simulated placing times.

6.2 Estimators for rotary turret type machine

Both linear regression models (univariate and multivariate) and the training data were taken directly from the earlier study made by Laakso *et al.* (2002) to be compared with our non-linear MLP estimator.

6.2.1 Estimators based on linear regression

Two linear regression estimators for the HSP 4795/96 turret type assembly machine were presented in the study by Laakso et al. (2002). Both of these were based on Valor Line Engineering Package simulations. The univariate linear regression model for the assembly time (t) is of the form

$$t = 0.117 \cdot n + 6.991, \quad (1)$$

where n is number of components to be assembled. The model explains 85.1 percent of the time variation. A multivariate linear regression model is of the form

$$t = 0.115 \cdot n + 0.163 \cdot c + 0.000536 \cdot l - 6.467, \quad (2)$$

where c is number of different component types and l the side length of the PCB. This model explains 92.2 percent of the time variation. Graphical results of this multivariate linear regression (green line) compared with simulated "true values" (blue line) and with absolute error values (red line) are shown in Figure 6.

---------> INSERT FIGURE 6 HERE <----------

Figure 6. Multivariate linear regressions on assembly times for rotary turret machine (HSP 4795) by formula (2).

6.2.2 Estimator based on non-linear regression

In our experiments with non-linear estimators, we first tested how many hidden neurons our MLP model should use for non-linear regression of the PCB assembly times. The input features of the MLP were the same as in the multivariate linear regression in Laakso *et al.* (2002). In the training of the MLPs we used the *Netlab toolbox for Matlab functions* software created by Nabley, I., (2005).

--------> INSERT FIGURE 7 HERE <----------

Figure 7. The effect of the network complexity (number of hidden neurons) on the error magnitude of the estimation. Sum of the squared errors (SSE) is given for 16 network models. The solid line (with dots) corresponds to the mean of the predictive distribution. Dashed lines represent the standard deviation of the predictive distribution around the mean.

We estimated a suitable network complexity (number of neurons in the hidden layer) by using the leave-one-out (LOO) validation method. As the learning of the network is stochastic, we repeated the validation 10 times for each network structure. The results of the model complexity estimation measured in terms of the sum of the squared errors (SSE) are summarized in Figure 7. The model which gives the best results has complexity of 12 hidden neurons. The errors here are the differences between the "true" assembly times and estimated assembly times measured in seconds. To get more information about the behaviour of the 12 hidden neurons model we calculated also some other statistical characteristics of the models MSE (Mean squared error), MAE (Mean absolute error), RSE (Relative squared error) and RAE (Relative absolute error) for this model, see Table 1.

Table 1. Comparison of statistical characteristics for multivariate linear regression (2) and the MLP model for rotary turret machine (seconds).

--------> INSERT TABLE 1 HERE <----------

Assembly times of the input data, the MLP model with 12 hidden neurons and the multivariate linear model are shown in Figure 8 for different number of component types. As it can be observed, the dependency between the input variables and the output variable is non-linear. The amount of non-linearity increases as the number of different component types in the assembly grows; if there is only one component type, the assembly time is in practice linear with respect to the two other input variables.

--------> INSERT FIGURE 8 HERE <----------

Figure 8. The input-output mappings of the test data of the MLP network for rotary turret machine. The columns of panels show the dependency of the assembly times on different combinations of system parameters (number of components, number of component types and PCB side length) for input data, MLP network having 12 hidden neurons and the multivariate linear model (2).

---------> INSERT FIGURE 9 HERE <----------

Figure 9. Results of the MLP model having 12 hidden neurons for rotary turret machine compared to the true values.

In Figure 9 the results of MLP having 12 hidden neurons are presented on the same visual manner as earlier for the multivariate linear model (Figure 6). The model having 12 neurons was found in Figure 7 to have the smallest generalization error. The graphs of the true values (blue) and the MLP approximations (green) are so closely one on the other that they can not be seen separately.

These results show that we can make apparently good assembly time estimations using MLP networks. However, one should note here that our data set from Laakso *et al.* (2002) was artificially created for a particular machine type as described in Section 6.1.

6.3 Estimators for gantry type placement machines

Preliminary test showed that the use of a combination of datasets 1 and 2 as a training data resulted in less accurate results than the use of dataset 1. That is why we decided to do two sets of tests, one for dataset 1 and the other for the combination of dataset 1 and 2. Dataset 2 was not used alone since it is all too small for the MLP to be used as training material. We tested the multivariate linear regression and MLP techniques with more parameters and multiple input parameter combinations to find the best combination.

Table 2. Input parameters used for the gantry type machine calculations.

---------> INSERT TABLE 2 HERE <----------

The accuracy of the estimation was measured with three error values; MSE, MAE and MAX. MAX is the maximum value of the estimation error which is the absolute difference in seconds between the correct value and the estimated value.

6.3.1 Estimators based on linear regression

Some key results from the multivariate linear regression with different input parameter combinations from Table 2 are shown in Table 3. It is natural that the accuracy of the linear estimators decreases with the number of different job types since presumably the variance of the jobs will also increases due to the use of different component shapes. That is especially the case when dataset 2 is used in conjunction with dataset 1. However, the same effect can be later seen also with the non-linear MLP.

Table 3. MSE, MAE, MAX and the square of Pearson product moment correlation (r^2) of the estimated placing time for different input parameter combinations in the multivariate linear regression. Symbols *a* to *f* refer to the parameters of Table 2.

---------> INSERT TABLE 3 HERE <----------

As it can be seen from Table 3, the parameter combination *a, b, d, e, f* seems to be the best (at least when looking at the MSE value) for both datasets. The best parameter combinations include the inputs *e* or *f,* so it is rather obvious that the variance in the component sizes affects the placement speed. That is probably because the usage of the nozzles may not be optimal when there are a lot of different sized components that require nozzles that can be used to place only a small subset of all components. The differences between some of the input parameter combinations are marginal and it is difficult to say which combination would give the smallest errors with a greater amount of data. The dependency between the number of components and placing time is shown in Figure 10. It is clear that the total number of components heavily dominates in the regression of the total placing time but improvements can be achieved by selecting additional input parameters.

---------> INSERT FIGURE 10 HERE <----------

Figure 10. Dependency between the number of components and the placement time (asterisk = dataset 1 and square = dataset 2).

6.3.2 Estimator based on non-linear regression

MLP (one hidden layer with 12 neurons) was trained and tested in the same way as in Section 6.2.2. Table 4 shows some of the results obtained with the MLP.

Table 4. MSE, MAE and MAX of the estimated placing time for different input parameter combinations in the MLP. Symbols *a* to *f* refer to the features of Table 2.

---------> INSERT TABLE 4 HERE <----------

The results for MLP are interesting since for most input combinations the accuracy decreases a lot when the dataset 2 is included in the training set. Also the best input combination is different when the dataset 2 is included. It must be noted that for dataset 1+2 the linear regression performs generally better than the MLP. The reason for the bad performance of the MLP may derive from the sparseness of the training set. Also the data may be so intense in certain areas that the MLP is distorted by those "clusters".

The Bayesian regularisation of the network parameters in the combination with the scaled conjugate gradient turned out to be somewhat problematic in our case due to the high number of inputs in comparison to the size of the training data. For high number of inputs the number of the regularisation cycles was reduced from three to two which decreases the accuracy of the estimation.

The two data sets included a number of cases where there was a basic PCB type and its minor variants. We therefore clustered the data in order to remove the possible effect of intense data spots in MLP training. When doing this, the number of clusters should be rather high so that only the most intense data spots would be identified as a single cluster. Clustering was done so that each identified cluster was replaced by a single data point which was the arithmetic mean of the points belonging to the clusters.

---------> INSERT FIGURE 11 HERE <----------

Figure 11. The effect of clustering to the error values in MLP with datasets 1+2 for gantry type machine.

We tested the MLP with different numbers of clusters using the input combination *a, b, e, f* (this combination had the best average accuracy for both input datasets), see Figure 11 for a summary of results. It appears that clustering gives no actual benefits since there is no remarkable improvement on the estimation accuracy while the cluster count is high. When 73 clusters are used the case is the same as without clustering because there are total 73 objects in dataset 1+2. Significant decrease of

the amount of clusters will naturally decrease the accuracy since objects that have a rather large distance from each other are placed into the same cluster.

7 Conclusions

Production control in fast-paced electronic assembly industry requires proper solutions for production scheduling and line balancing to ensure that the promised dead lines are met and expensive machines are used efficiently. To solve these problems in a short time a fast single machine placement time estimation scheme is needed. Not only that the estimation must be fast enough it must also be accurate since better estimation leads to better line balancing. Methods presented in this paper are directed for factory scheduling where heavy machine level optimizations are all too slow or need information that is not yet available.

We introduced the use of multi-layer perceptron neural network for the placement time estimation in PCB component assembly. The method was compared to multivariate non-linear regression for two rather different machine types. For the rotary turret the new method was clearly superior to the multivariate linear regression (Table 1). This mirrors the non-linearity of the processing times as demonstrated in Figure 8. For gantry type machines the results show the importance of a proper training set. When having a relatively homogeneous training set (dataset 1) the MLP neural network works again efficiently (Table 3 compared to Table 4). With very heterogeneous data (datasets 1+2) multivariate non-linear regression gives somewhat smaller MSE. On the other hand, this kind of data is an extreme case which is not very suited for modelling any how.

When considering the practical usability of MLP neural networks it is observed that the application of the technique includes four steps; collecting a training data set, selecting a proper network layout and optimizing the network weights, using the network in an application (like line balancing or cost accounting), and adapting the network to new observations. Out of these the first step, collecting a proper training set, has been most challenging to us. While there are plenty of machine data available from producers, creating a well-founded set of observations (PCB assembly definition along with observed manufacturing time) has to be done with care. Of course, this note holds also for the competitors of the neural network approach. The second step, selecting the layout and optimizing network weights, is standard one and when the layout has been chosen the same general structure can (hopefully) be used longer time. The step takes several hours but this is again true for other techniques, too. The third step, using the network, is fast as the application of the ready network can be written as a function. This even makes it possible to use the method in conjunction of exact mathematical optimization software; research on exact line balancing is currently going on in our group. A further advantage of the neural network approach is the possibility to retrain (step four) the network weights in an easy manner when more production data have been collected. This naturally presupposes the preservation of the general layout of the network.

As a topic of further studies one should consider the modelling of so-called fast and slow components in the same neural network. For turret machines one may have to slow down the machine speed in order to avoid unwanted displacements of some heavy components due to high accelerations of the PCB holding board. Modelling of two operation speeds with the same network seems to be possible but needs some care. An other open question is the consideration of the capacities of the placement heads (to hold different nozzle types) in relation to the number and kind of component types of

PCBs. In the current version the head capacity is fixed but one might let it be a parameter in order to generalise the model.

References

Ayob, M., 2005. *Optimization of Surface Mount Placement Machine in Printed Circuit Board Assembly.* Thesis (PhD), University of Nottingham, The School of Computer Science.

Ayob, M., Cowling, P. and Kendall, G., 2002. Optimization for Surface Mount Placement Machines. *IEEE International Conference on Industrial Technonology*, 1, 498-503.

Ayob, M., Kendal, G., 2008. A survey of surface mount device placement machine optimization: Machine classification. *European Journal of Operation Research*, 186, 893-914.

Bishop, C.M., 1995. *Neural Networks for Pattern Recognition*, Oxford: Clarendon Press.

Crama, Y., van de Klundert, J., and Spieksma, F. C. R., 2002. Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics*, 123, 339–361.

Crama, Y., Oerlemans, A. and Spieksma, F., 1994. *Production planning in automated manufacturing*. Lecture notes in Economics and Mathematical Systems, volume 414, Springer-Verlag.

Hald, A., 1967. *Statistical theory with engineering applications*, New York: Wiley

Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, 2, 359-366.

Johnsson, M., 1999. *Operational and Tactical Level Optimization in Printed Circuit Board Assembly*. Thesis (PhD), University of Turku.

Klomp, C., van de Klundert, J., Spieksma, F. and Voogt, S., 2000. The feeder rack assignment problem in PCB assembly: A case study. *International Journal of Production Economics,* 64, 399-407.

Knuutila, T., Pyöttiälä, S., Nevalainen, O.S., 2007. Minimizing the number of pickups on a multi-head placement machine, *Journal of the Operational Research Society*, 58,115-121.

Kodek, D.M. and Krisper, M., 2004. Optimal algorithm for minimizing production cycle time of a printed circuit board assembly line. *International Journal of Production Research*, vol.42, no.23, 5031-5048.

Laakso, T., Johnsson, M., Johtela, T., Smed, J. and Nevalainen, O., 2002. Estimating the Production Times in PCB Assembly, *Journal of Electronics Manufacturing,* 11 (2), 161-170.

Leipälä, T. and Nevalainen, O., 1989. Optimization of the movements of a component placement machine. *European Journal of Operational Research*, 38, 167-177.

MacKay, D.J.C., 1992a. Bayesian Methods for Adaptive Models, Thesis (PhD). California Institute of Technology.

MacKay, D.J.C., 1992b. A Practical Bayesian Framework for Backprop Networks, *Neural Computation* 4 (3), 448-604.

MacKay, D.J.C., 1995a. Bayesian Methods for Neural Networks, *Neural Networks Summer School*, University of Cambridge.

MacKay, D.J.C., 1995b. Bayesian Non-Linear Modeling for the Prediction Competition, Cavendish Laboratory, University of Cambridge.

MathWorks, 2007. *Matlab & Simulink technical computing software* [online]. The MathWorks, Inc., Massachusets. Available from: http://www.mathworks.com/academia/student_version/ [Accessed 2 November 2007]

Nabley, I., 2005. *Netlab toolbox of Matlab functions* [online]. Aston University, Information Engineering. Available from: http://www.ncrg.aston.ac.uk/netlab/ [Accessed 2 November 2007]

Neal, R.M., 2001. Survival Analysis Using a Bayesian Neural Network, *Slides for Joint Statistical Meetings*, University of Toronto.

Neal, R.M., 2004. Bayesian Methods for Machine Learning. *NIPS Tutorial*, University of Toronto.

Neammanee, P. and Randhawa, S.U., 2003. Integrated methodology for board assignment and component allocation in printed circuit board assembly. *International Journal of Production Research*, 41, 919-937.

Pinedo, M.L., 2005. *Planning and scheduling in manufacturing and services*, New York: Springer Science+Business Media, Inc.

Reewes, C.R., 1995. *Modern heuristic techniques for combinatorial problems*, London: McGraw-Hill

Rogers, P. and Warrington, R., 2004. Production planning for surface mount technology lines. *International Journal of Production Research*, 42, 2693-2718.

Siemens SIPLACE Pro, 2007. *Commercial assembly engineering software package*. Siemens AG, Germany. Information available from: http://ea.automation.siemens.com. [Accessed 2 November 2007]

Smed, J., Johnsson, M., Johtela, T., and Nevalainen, O. S., 2003. Techniques and applications of production planning in electronics manufacturing systems, *Computer Aided and Integrated Manufacturing Systems*, 5, 1–48.

Sze, M.T., Ji, P. and Lee, W., 2001. Modelling the component assignment problem in PCB assembly. *Assembly Automation,* 20, 55-60.

Valor Trilogy, 2005. *Commercial assembly engineering software package*. Valor Computerized Systems Ltd, Israel. Information available from: http://www.valor.com/en/Trilogy.aspx [Accessed 2 November 2007]

Wilhelm, W., Arambula, I., Choudry, N.N.D, 2006. Optimizing Picking Operations on Dual-Head Placement Machines. *IEEE Transactions on Automation Science and Engineering*, Volume 3, Number 1.

Yuan, P., Hu, Y., Liu, H., Gao, H., 2006. Feeder assignment optimization algorithm for multi-head mounter. *Journal of Control Theory and Applications,* Volume 4, Number 3, pages 223-228.

Figures and tables for "Estimating PCB assembly times using neural networks"

Figure_1.tif

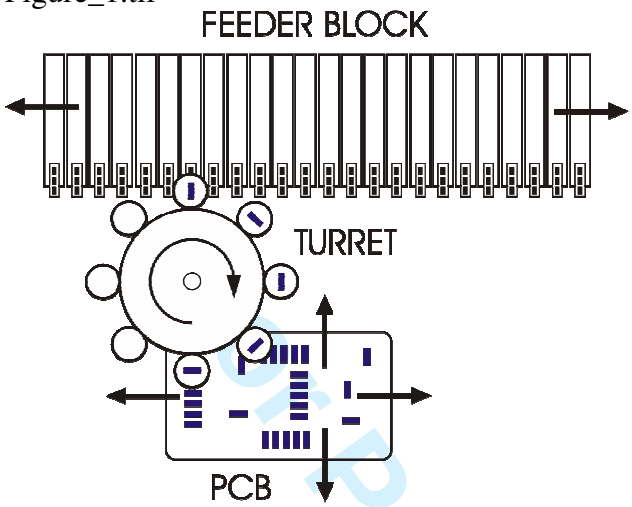

Figure_2.tif

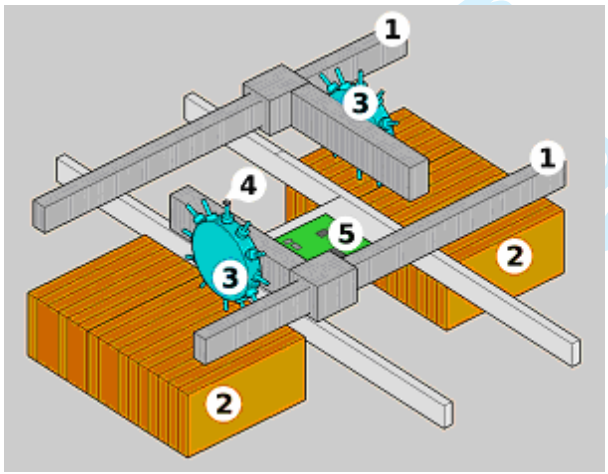

Figure_3.tif

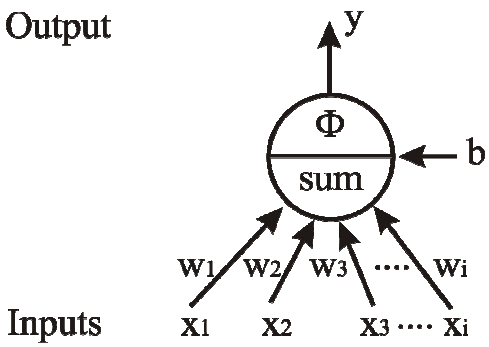

Figure_4.tif

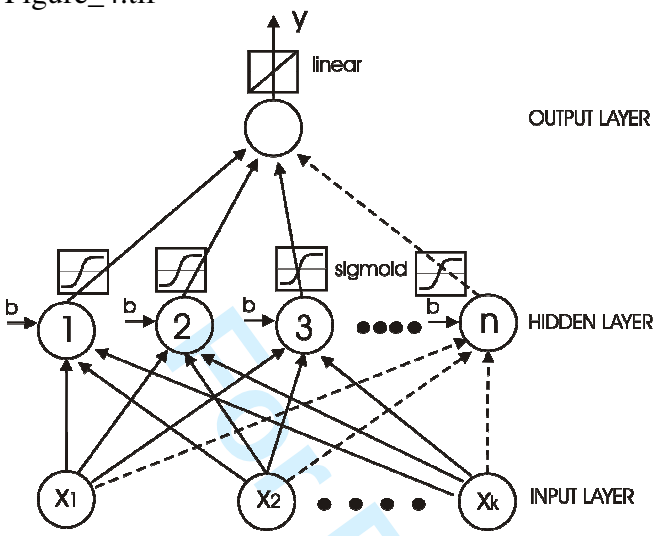

Figure_5.tif

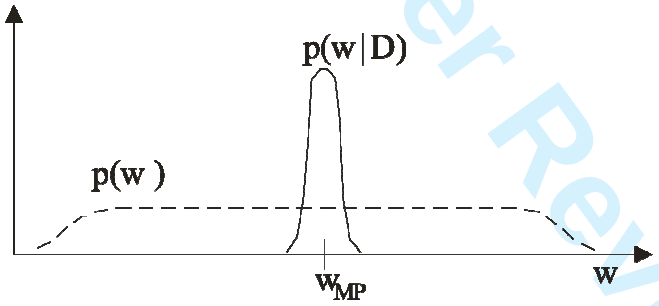

Figure_6.tif

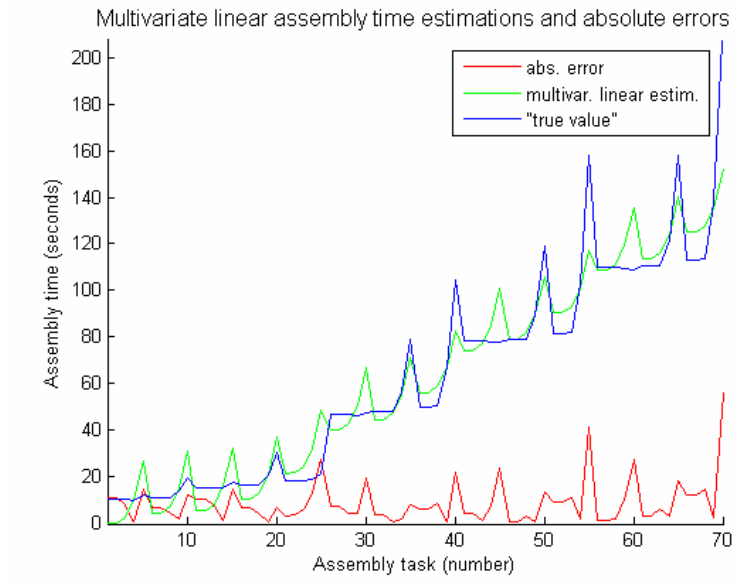

Figure_7.tif

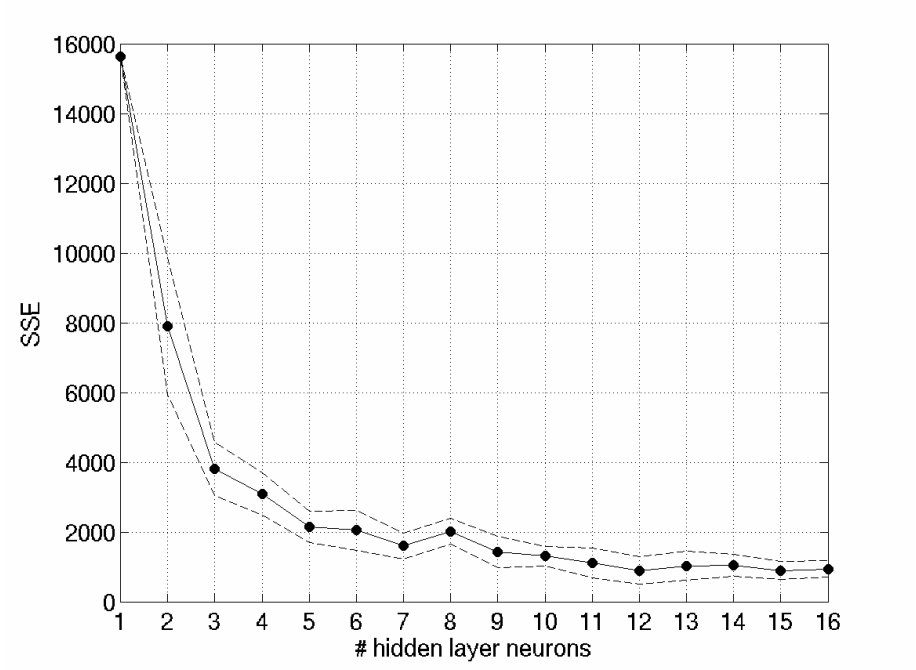

Figure_8.tif

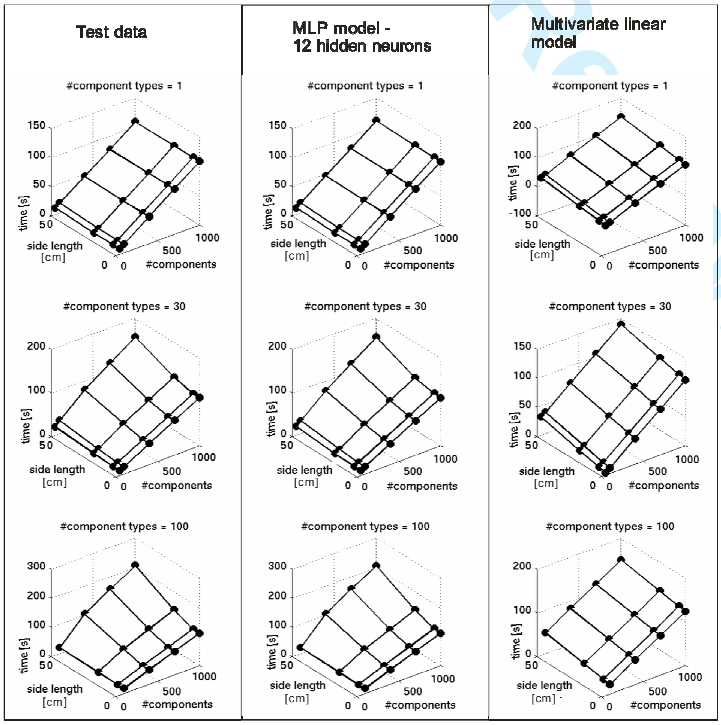

Figure_9.tif

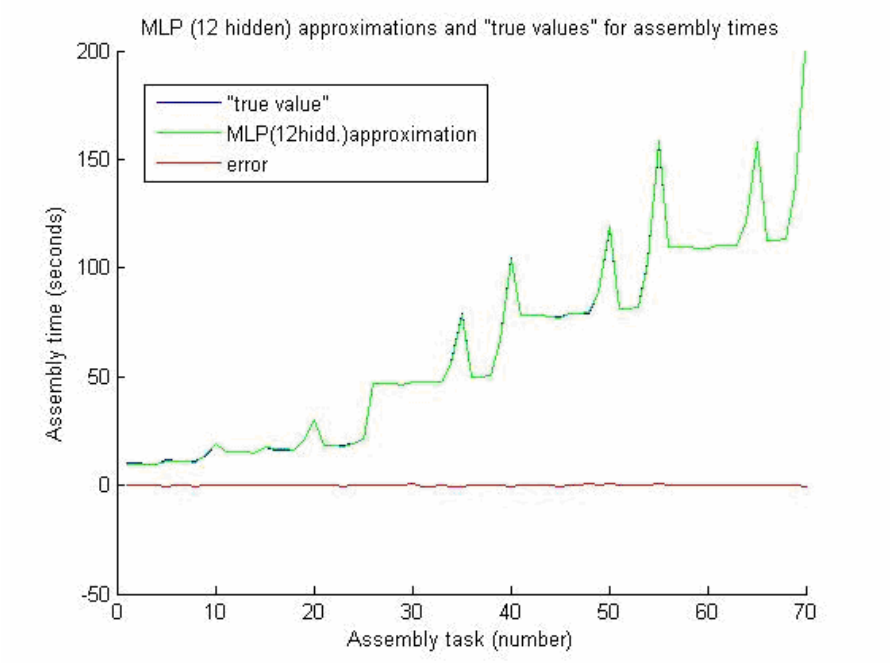

Figure_10.tif

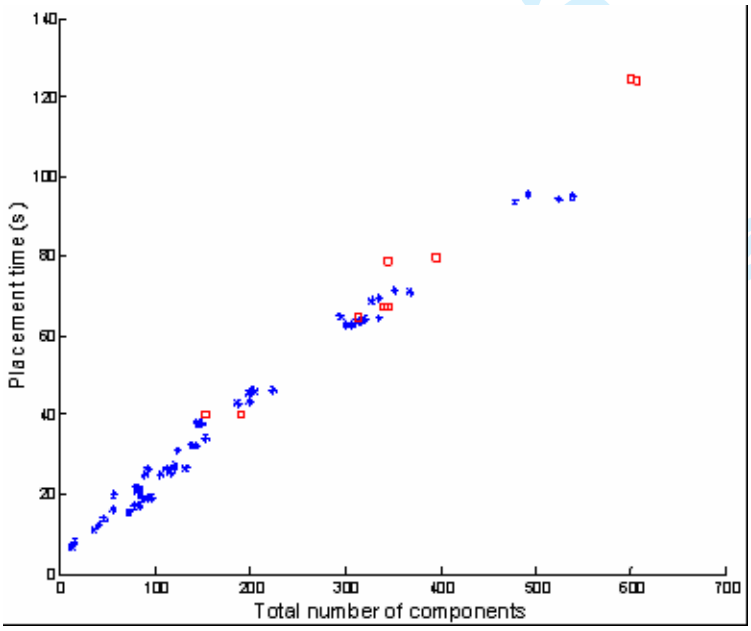

Figure_11.tif

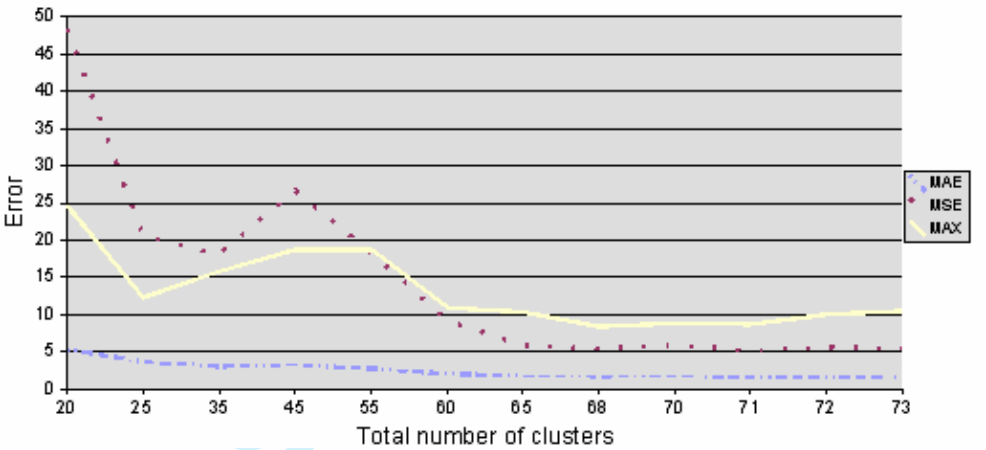

Table_1.tif

	multivariate linear model	12 hidden neurons
MSE (mean squared error)	194.302	10.413
MAE (mean absolute error)	9.130	1.166
RSE (relative squared error)	0.095	0.005
RAE (relative absolute error)	0.235	0.030

Table_2.tif

a = number of components
b = number of component types
c = side of the PCB (max. side length in mm)
d = number of different component shapes
e = average size of component area (area in mm^2)
f = component size variance

Table_3.tif

Datasets 1+2		*a*	*a,b,c*	*a,d,f*	*a,b,d,f*	*a,b,d,e,f*	*a,b,c,d,f*
	MSE	10,81	6,89	4,33	4,79	3,98	4,09
	MAE	2,41	2,14	1,52	1,60	1,46	1,43
	MAX	10,80	6,21	5,90	6,22	5,84	5,99
	r^2	0,985	0,991	0,994	0,994	0,995	0,994
Dataset 1		*a*	*a,b,c*	*a,d,f*	*a,b,d,f*	*a,b,d,e,f*	*a,b,c,d,f*
	MSE	9,39	5,92	3,00	2,93	2,63	2,68
	MAE	2,26	2,02	1,30	1,27	1,22	1,18
	MAX	10,80	5,82	4,99	4,86	5,06	5,34
	r^2	0,984	0,989	0,995	0,995	0,995	0,995

Table_4.tif

Datasets 1+2		*a*	*a,b,c*	*a,d,f*	*a,b,d,f*	*a,b,d,e,f*	*a,b,c,d,f*
	MSE	10,81	6,89	4,33	4,79	3,98	4,09
	MAE	2,41	2,14	1,52	1,60	1,46	1,43
	MAX	10,80	6,21	5,90	6,22	5,84	5,99
	r^2	0,985	0,991	0,994	0,994	0,995	0,994
Dataset 1		*a*	*a,b,c*	*a,d,f*	*a,b,d,f*	*a,b,d,e,f*	*a,b,c,d,f*
	MSE	9,39	5,92	3,00	2,93	2,63	2,68
	MAE	2,26	2,02	1,30	1,27	1,22	1,18
	MAX	10,80	5,82	4,99	4,86	5,06	5,34
	r^2	0,984	0,989	0,995	0,995	0,995	0,995