
Production Systems and Information Engineering
Volume 6 (2013), pp. 3-14

IDENTIFICATION OF A COMPLEX DRIVE CHAIN BASED ON LOCAL LINEAR MODEL TREE

VIKTOR FÜVESI
University of Miskolc, Hungary
Research Institute of Applied Earth Science
Department of Research Instrumentation and Informatics
fuvesi@afki.hu

ERNŐ KOVÁCS
University of Miskolc, Hungary
Department of Electrical and Electronic Engineering
elkke@uni-miskolc.hu

CSABA VÖRÖS
University of Miskolc, Hungary
Research Institute of Applied Earth Science
Department of Research Instrumentation and Informatics
voros@afki.hu

[Received November 2011 and accepted September 2012]

Abstract. This paper uses the local linear model tree (LOLIMOT) method for modeling the angular velocity of a complex nonlinear system called Gamma-log. The drive chain of the Gamma-log contains nonlinear parts such as an AC servo drive or a worm gear drive. The drive chain is modeled with LOLIMOT algorithm. An experiment was conducted to collect data from the original system and to simulate the kinematics of the track. The best model was selected from ARX and FIR models using a correlation coefficient based performance index.

Keywords: LOLIMOT, AC servo, electromechanical drive chain

1. Introduction

State-of-the-art industrial applications frequently use different kinds of actuators with electromechanical kinematic chains (EKC). A wide range of these cannot be controled only with a single motor but different kinds of gears and gearboxes are applied. The drive chains thus developed include a great number of different nonlinear components, such as motors, gears and bearings. The nonlinearities of the components of the chain make it hard to control the system precisely. It is a common requirement to supervise or monitor such systems, therefore modeling and simulation are important.

There can be found several references where complex drive chains are modeled with different methods. One common method is when the system components are separately modelled with differential equations. An example [1] introduces this method where a PMSM AC servo with its drive is included. Erdogan models an electric drive system with differential equations combined with object-oriented technics and verifies also the simulation model with an experiment [2]. The disadvantage of these methods is the need for a deep knowledge of the system components, which is regularly not available. After the simulation process a validation procedure follows when the real parameters are identified.

Systems can also be modeled with the 'black box' method. Inputs of the analysed process are matched to the inputs of the black box and the process should be carried out with the outputs as well. The black box can be e.g. a neural network structure or a locally linear neuro-fuzzy structure (LLNF). During the training process, these kinds of mathematical structures can find any nonlinear relation between the inputs and outputs if any level of correlation exists. So this method can powerfully assist as a general function approximator [3].

Neuro-fuzzy modeling is used in a wide range of applications. E.g. modeling and identification of a vehicle suspension was carried out with the neuro-fuzzy method [4]. A black box model for a temperature control pilot plant called RT542, which is equipment for engineering education [5], is another example. The structure can be used to model high nonlinearities such as the dynamics of centrifugal compressors [6]. There can also be found examples of its use for solving the identification and control problem of a combustion engine's exhaust [7]. Besides the modeling, this method can also be used for predictive control [8].

The literature does not detail the process of model selection or a comparison of the different local linear neuro-fuzzy (LLNF) based models which are connected to drive chains. This paper presents a nonlinear black box model for a drive chain of the mobile Gamma-log equipment. The aim is to develop a LLNF model using LOLIMOT algorithm that captures the dynamic properties of the system over a wide operating range.

The paper covers the identification process, starting with a detailed description of the investigated equipment. After that Section 3 deals with measurements on the real system. Section 4 is about preprocessing the measured data. Section 5 shows the basics of the Neuro-Fuzzy LOLIMOT and its dynamical extension, followed by a comparison of the different utilized external dynamics and results.

2. Introduction of the investigated system

The section introduces the investigated Mobile Gamma-log equipment from many aspects. First the aim of the Mobile Gamma-log will be clarified, then the main

construction of the device will be detailed. Following this, the investigated drive chain will be discussed.

2.1. Mobile Gamma-log equipment

During onshore exploratory oil drilling, in order to determine the exact depth of the core which contains oil, the natural gamma-ray spectral of the core is logged. To refine the local measurements, experiments are conducted on the raised bore cores in a laboratory. The result of the site experiments can be refined with the correlation of the two measurements. One of the most important details in the measurements is that the measured gamma spectrum of the bore core section should not slip from the exact depth value. Therefore accurate moving of the gamma-ray detector is needed during the experiments.

The main components of the investigated mobile equipment can be seen in Fig. 1. The bore core lies on a lead case in the centre line of the modular, one meter long, railway part. The detector-carrier-track carries the gamma-ray detector. The necessary power, the control signal of the controller PC and the resulting measurement signals go through the energy chain.

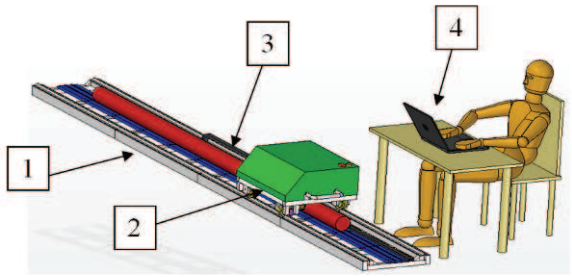


Figure 1. Main components of MGL-01F gamma-log equipment
1: modular railway; 2: detector-carrier-track; 3: energy chain; 4: controller PC

2.2. Drive chain of the track

The main drive in the actuator chain of Gamma-log is a 200W AC servo motor manufactured by Omron. The shaft of the motor is connected to a worm gearbox, which is a product of Bonfiglioli (Fig. 2). The reducing gear ratio of the worm gear is 70. The rear left wheel of the track is the drive wheel which is connected to the worm gear. An incremental encoder is assembled to the front left wheel of the track. The encoder senses the movement of the track in the railway. The resolution

of the encoder is 1000 pulses per rotation. Every wheel of the track has bearings on both sides [9].

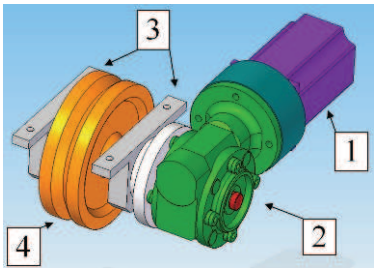


Figure 2. Drive chain of Gamma-log
1: Omron AC servo motor; 2: Bonfiglioli worm gear; 3: Bearings; 4: Wheel with V-profile

3. Measurements

Measurements were performed to model the kinematic and dynamic behaviour of the Gamma-log track. The measurement set-up can be seen in Fig. 3. The main difficulty was that the track was moving while the data acquisition was in progress. During the measurements the track was controlled with different accelerations and different velocities. During the different measurements the moving distance of the track was set to a constant length of 100 millimetres. For the measurement NI CompactDAQ modular equipment was used. The sampling frequency was set to 5 kHz per channel.

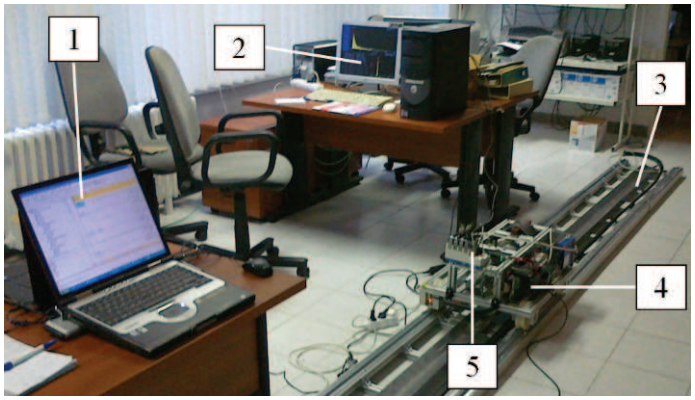


Figure 3. Measurement rig of the drive chain of Gamma-log
1: Control PC during the measurements; 2: PC for data acquisition;
3: Railway of the track; 4: Detector-carrier-track; 5: NI Compact DAQ modular unit

Not all of the parameters of the system could be measured because of the construction of the drive chain. The electrical parameters such as exciting currents (I_1, I_2, I_3) on all 3 phases and line voltages (V_{12}, V_{23}, V_{31}) were measured in addition to the rotation of the wheel with an incremental encoder.

The velocity profile of the track can be calculated from the signal of the encoder (v). The intervals of the measured parameters can be found in Table 1.

Table 1. Intervals of measured parameters

Parameter	Unit	Lower limit	Upper limit
Current (I_1, I_2, I_3)	A	-0.5	0.5
Phase-to-Phase Voltage (V_{12}, V_{23}, V_{31})	V	-25	25
Velocity of the Track (v)	mm/s	0	30

4. Data pre-processing for modeling

The measured data required pre-processing before modeling. The velocity profile of the track was calculated from the signal of the encoder. It was used later as output of the neural network.

The signals of voltages and currents were filtered with a low pass filter. The well-known Park or coordinate-frame transformation for three-phase machinery can provide a useful framework for the investigation. The rotating transformations are commonly used for machine design and control, but the simplifications that result from applying the transformation can also be useful for modeling [10,11].

The three-phase values were calculated in the reference frame of stator using the following formula (1) [12]:

$$\begin{bmatrix} T_\alpha \\ T_\beta \\ T_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \quad (1)$$

Here $T_1 \ldots T_3$ are the three-phase parameters, currents or voltages. T_α and T_β are the same components in the reference frame of the stator.

The measured signals show mainly sinusoidal characteristics with some noise added. Using this transformation, the shapes of the new signals are also sinusoidal but the frequency decreases.

Other transformations were also applied to the signals to get their amplitudes. The angle of the rotating vector can be calculated from the T_α and T_β components. The

changes of the values in the time domain were applied as inputs during the modeling. Between the transformations the signals are filtered and resampled to speed up the modeling process.

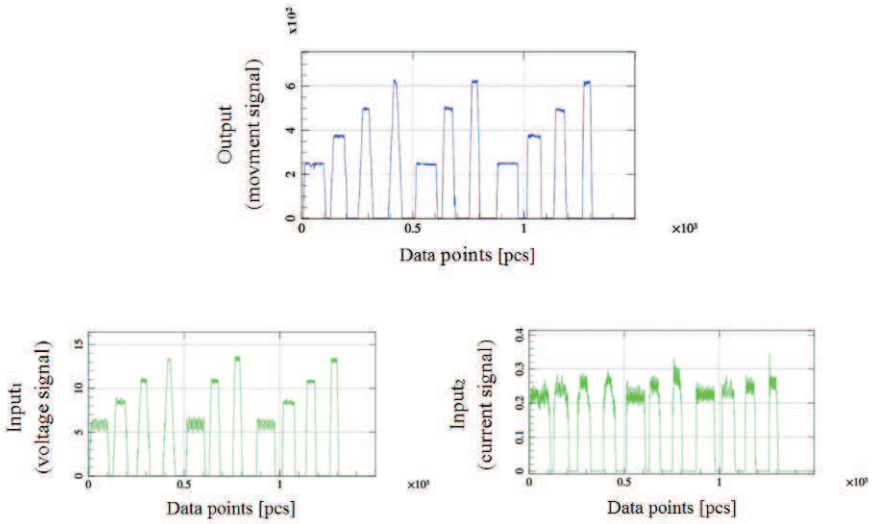


Figure 4. Resampled and transformed training datasets

From the measurements two datasets were created. One of them was the training dataset which was used during the training process (Fig. 4). The other dataset was the validating dataset which was applied during testing the network. The validating set was different from the training set in order to test the extrapolating performance of the model.

5. LOLIMOT model and algorithm

To model the system the Local Linear Model Network, LLMN, which is an extension of the radial basis function network (RBFN) by Nelles is used. This structure also deals with local linear neuro-fuzzy models referred to as the Takagi-Sugeno fuzzy model [13, 14].

In this structure the weights of the output layer are replaced with a linear function of the network's input. Furthermore, the RBFN is normalized [13]. The structure of the network can be seen in Fig. 5.

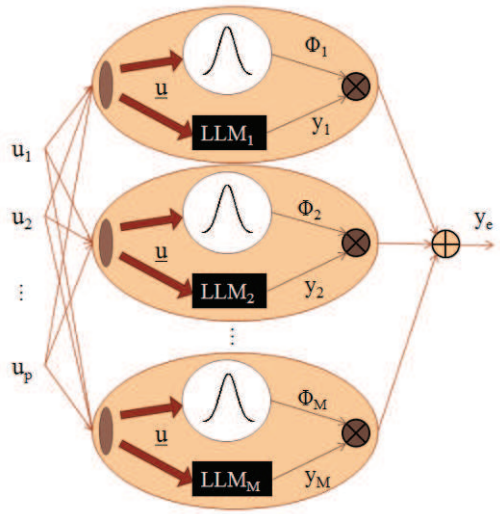


Figure 5. Structure of LOLIMOT neural network

The output of the network (y_e) can be calculated with the following formula (2) [13]:

$$y_e = \sum_{i=1}^{M} \phi_i(\underline{u}, c_i, \sigma_i)\left(w_{i0} + w_{i1}u_i + \ldots + w_{ip}u_p\right), \tag{2}$$

where: M is the number of sub-models; $\underline{u}$ is the input vector; p is the number of inputs; w_{xy} is the yth parameter in the xth neuron. ϕ_i is the normalized Gaussian validity function which determines the regions of the input space where each neuron (Local Linear Model) is active. Furthermore, the nonlinear parameters are c_i (center) and σ_i (standard deviation).

The local linear model can be taught using the local linear model tree algorithm. This training method is stable, very fast and robust and also has a good convergence feature. The training process has two stages: a) in the first part of the training the input space is decomposed by determining the parameters of the validity function, b) in the second step the LLM is optimised to the region by the least square method.

The training process begins with a globally identified linear model. In the first iteration the global region is divided into two local linear models. The generated

local models are valid in their own regions. The models are identified separately and the global model comes from the summation of the local models. In the next iteration the worst model is selected and further divided into two new models. Some steps of the iteration can be seen in Fig. 6.

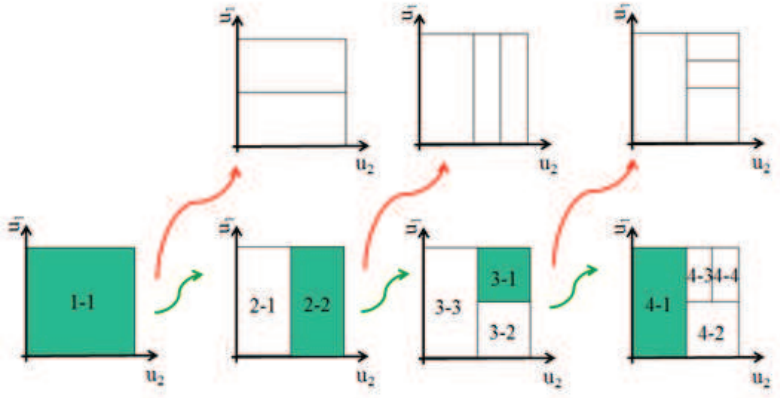


Figure 6. The first four iterations of LOLIMOT algorithm

6. Modeling the system with LOLIMOT

A drive chain is a dynamic system by nature. During the modeling the relationship should be discovered between the inputs and the output. The investigated system can be described by a Multi Input Single Output (MISO) model. The transformed excitation current (u_1) and voltages (u_2) were the inputs and the movement of the track was the output (y) of the mathematical model.

The previously described network structure is able to model static systems but some external dynamics has to be added to the inputs to characterise the dynamic feature of the system.

External dynamics means that virtual inputs are generated by adding new inputs to the network. The new inputs can be transformed from the real inputs using one time delay transformation.

Two different model structures were investigated during the simulations.

The first model uses only the input parameters of the system with some virtual inputs added. This model structure is called FIR model and it is basically a feedforward model (Fig. 7a). This model approximates the function f in the following form:

$$y_e = f\left(u_1(t-1), u_1(t-2), u_2(t-1), u_2(t-2)\right) \tag{3}$$

The other model structure was the ARX model, where beside the inputs (u_1 and u_2) the time delay real output (y) was also applied as input to the network (Fig. 7b). The approximation of the model structure can be described by Eq. 4:

$$y_e = f\left(u_1(t-1), u_1(t-2), u_2(t-1), u_2(t-2), y(t-1), y(t-2)\right). \tag{4}$$

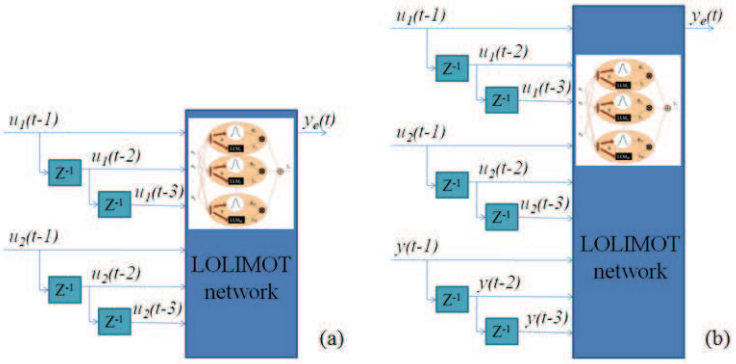


Figure 7. MISO system with FIR input configuration (a) and with ARX external dynamics (b)

To achieve the best result, the structure of the networks was changed by changing the number of virtual inputs. The following six different structures were analysed:

- FIR input configuration with one time delay per input. It is called 2i0o.
- FIR input configuration with two time delays per input. It is called 4i0o (Eq. 3).
- FIR input configuration with three time delays per input. It is called 6i0o.
- ARX input configuration with one time delay per input, the one time delayed transform required output use as input. It is called 2i1o.
- ARX input configuration with two time delays per input, the two time delayed transforms required output use as input. It is called 4i2o. (Eq. 4)
- ARX input configuration with three time delays per input, the three time delayed transforms required output use as input. It is called 6i3o.

On every structure 35 iterations were applied using the training algorithm. In every iteration step the correlation coefficients [15] of both the validating and the training datasets were calculated from the estimated output of the system and the required datasets. The best mode was selected using a performance index which was calculated from correlation coefficients:

$$I_{perf} = \sqrt{CORR_{train}^2 + CORR_{valid}^2} \ . \tag{5}$$

7. Results

Using a performance index the best model can be found. The calculated performance indexes for the 210 models can be found in Fig. 8.

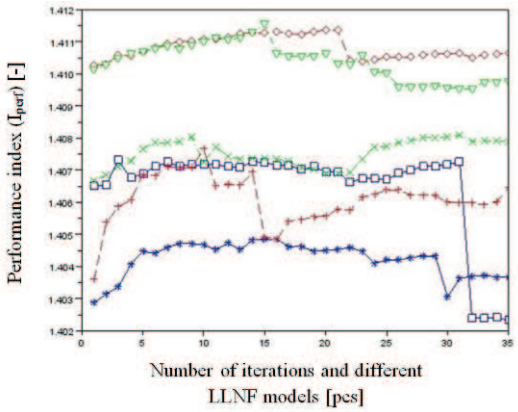


Figure 8. Performance indexes of investigated models
star: 2i0o; square: 2i1o; plus: 4i0o; diamond: 4i2o; cross: 6i0o; triangle: 6i3o

Generally the higher number submodels make better solutions but increase computation time and complexity of the global model. In some cases a new submodel can decrease the performance of the global model because the LOLIMOT training does not analyse whether a separation of a region is good for the global model or not. This is the explanation of the big performance losses of the models with a higher number of submodels comparing to less complicated models.

To find the best model with the best correlation feature, different structure types of the models were compared. The selection criterion of the best model was the previously defined performance index (5). To find the globally best model, the locally selected best model was chosen. The models were compared using performance index and MSE of the datasets. The results are summarised in Table 2. The higher performance index is better.

Table 2. Best model selection using performance index

Structure type	Performance index	Index of best model	MSE of training dataset	MSE of validation dataset
1. FIR structure (2i0o)	1.4048	16	54.948	67.317
1. ARX structure (2i1o)	1.4073	3	48.875	47.509
2. FIR structure (4i0o)	1.4076	10	43.171	48.798
2. ARX structure (4i2o)	1.4113	21	17.280	19.497
3. FIR structure (6i0o)	1.4081	31	23.029	55.013
3. ARX structure (6i3o)	1.4115	15	14.706	19.007

Increasing the number of the inputs makes the models better, but it also increases the computation time and the necessary resources. Both model structures provided good solutions but the best model came from the ARX structure called 6i3o.

Conclusion

An electrical drive chain of real mobile equipment was modeled with an artificial intelligent method. The datasets were collected from the real system during the measurements. Two feedforward structure types were used and compared. The investigation shows that the LOLIMOT algorithm was able to learn the behaviour of the system. Both of the network structures were suitable for modeling the process but the ARX structure gave better results. To select the best model a performance index was established based on correlations of the datasets and estimations. The results can be used for fault detection and fault diagnosis of the system.

Acknowledgements

The work described was carried out as part of the TÁMOP 4.2.1.B-10/2/KONV-0001-2010 project in the framework of the Hungarian Development Plan. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] LIU, J.; WANG, T.; XU, D.; CONG, L.: *A New Algorithm Research and Simulation for Permanent Magnet Synchronous Motor AC Servo,* Proc. of the 2008 IEEE Int. Conference on System, Robotics Automation and Mechatronics (RAM), 2008, pp. 965-969.

[2] ERDOGAN, N.; HENAO, H.; GRISEL, R.: *A Proposed Technique for Simulating the Complete Electric Drive Systems with a Complex Kinematics Chain,* IEEE Electric Machines & Drives Conference, 2007., (IEMDC '07), pp. 1240-1245.

[3] TAKAGI, T.; M. SUGENO, M.: *Fuzzy identification of systems and its application to modeling and control*, IEEE Transaction on Systems Vol. 15, 1985.

[4] HALFMANN, C.; NELLES, O.; HOLZMANN, H.: *Modeling and Identification of the Vehicle Suspension Characteristics using Local Linear Model Trees*, Proc. of the 1999 IEEE Int. Conference on Control Applications, Vol. 2., pp. 1484-1489.

[5] NOURZADEH, H.; FATEHI, A.; LABIBI, B.; ARAABI, B.N.: *An Experimental Nonlinear System Identification Based on Local Linear Neuro-Fuzzy Models*, Industrial Technology, 2006. (ICIT 2006.), pp. 2274-2279, 2006.

[6] DAROOGHEH, N.: *Centrifugal Compressor Identification Using LOLIMOT*, Control and Decision Conference, 2008. (CCDC 2008.), Chinese, pp. 771-774.

[7] HAFNER, M.; SCHULER, M.; NELLES, O.: *Dynamical Identification and Control of Combustion Engine Exhaust*, American Control Conference, 2007., Vol. 1, pp 222-226.

[8] MOHAMMADZAMAN, I.; JAMAB, A. S.: *Adaptive Predictive Control of an Electromagnetic Suspension System with LOLIMOT Identifier*, 14th IEEE Mediterranean Conference on Control and Automation (MED '06), pp 1-6, 2006.

[9] UNIVERSITY OF MISKOLC - RIAES RII, MGL-01F: User's Manual, Miskolc, 2011, pp. 2-14.

[10] CHIASSON, J.: *Modeling and High-Performance Control of Electric Machines*, John Wiley & Sons, Inc. New York, 2005.

[11] KRAUSE, P.C., WASYNCZUK, O., SUDHOFF, S. D.: *Analysis of Electric Machinery and Drive Systems*, IEEE Series on Power Engineering, John Wiley and Sons, 2002, ISBN 047114326X.

[12] LIDOZZI, A.; SOLERO, L.; CRESCIMBINI, F.; BURGOS, R.: *Vector Control of Trapezoidal Back-EMF PM Machines Using Pseudo-Park Transformation*, IEEE Power Electronics Specialists Conference, 2007. (PESC 2007.), pp. 2167-2171.

[13] ISERMANN, R.: *Mechatronic System, Fundamentals*. Springer, London, ISBN 1-85233-930-6, 2005, pp 314-323.

[14] NEKOUI, M.A.; SAJADIFAR, S.M.: *Nonlinear System Identification using Locally Linear Model Tree and Particle Swarm Optimization*, IEEE International Conference on Industrial Technology, 2006., (ICIT 2006), pp. 1563-1568.

[15] REMESAN, R.; SHAMIM, M. A.; HAN, D.; MATHEW, J.: *ANFIS and NNARX based Rainfall-Runoff Modeling*, IEEE Systems, Man and Cybernetics, 2008. (SMC '08), pp. 1454-1459.

Production Systems and Information Engineering
Volume 6 (2013), pp. 15-26

EFFICIENCY ANALYSIS OF QUALITY THRESHOLD CLUSTERING ALGORITHMS

LÁSZLÓ BEDNARIK
University of Miskolc, Hungary
Department of Information Technology
laszlobednarik@gmail.com

LÁSZLÓ KOVÁCS
University of Miskolc, Hungary
Department of Information Technology
kovacs@iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. An important aspect of clustering is to provide a good intra-cluster similarity. Most of the traditional methods do not consider this aspect and they generate weak clusters from this viewpoint. The paper presents a survey of the two dominant candidates for quality threshold clusterings, the QT and BIRCH methods. Besides the analysis, a new variant of BIRCH method which can provide a better performance is proposed.

Keywords: clustering, quality threshold clustering, BIRCH, genetic algorithm

1. Introduction

Data clustering is an important and widely used technique in data analysis and data mining. The goal of clustering is to split the set of elements into subsets where the elements of the same group are more similar to each other than the elements from different groups. The process of clustering usually includes the following steps: select an appropriate representation form of objects; define an appropriate similarity function; determine the appropriate clustering algorithm and parameters; execute the algorithm and interpretat the results. As the clustering problem uses an unsupervised learning algorithm, there are many subjective parameters in the process. One of the key parameters is the aspect of similarity: at which value of similarity can the objects be assigned to the same cluster. There are many standard methods in the literature for clustering. The most widely used standard methods are hierarchical clustering [1], partitioning clustering [2], hybrid method [3], incremental or batch methods, monothetic vs. polythetic methods [4], crisp and fuzzy clustering [8].

Although the literature on clustering is very rich, there is an aspect that has not attracted much interest in recent years. The aspect in question is the similarity threshold within a cluster. Based on the informal definition of clustering, the elements within a cluster should be more similar to each other than the elements of different clusters. Considering the standard methods it can be seen that these methods do not fulfill this requirement at an optimal level. The HAC method [9], for example, can generate arbitrary large clusters where the distance between two elements of the same cluster may be much higher than the distance related to an element of another cluster (see Fig.1).

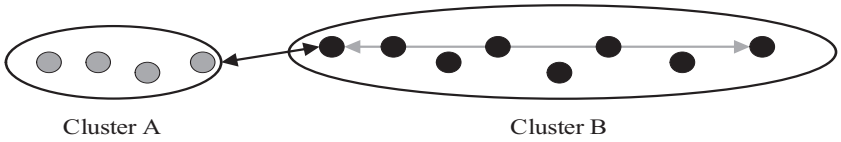


Figure 1. Inter-cluster similarity is higher than intra-cluster similarity

The majority of standard methods use a greedy approach to build up a cluster: while the distance between the candidate and the current cluster is below a threshold value, the cluster is extended with new elements. The threshold criteria are usually independent of the actual size of the current cluster. As a result, the similarity between two points of the same cluster may be arbitrarily low.

The goal of the investigation is to find clustering methods preserving the distance. In order to meet this requirement, the following two constraints are defined on the clustering model:

- for every object pair with a smaller distance than a threshold, there exists a cluster containing both elements of the pair.
- for every object pair with a larger distance than a threshold, there is no cluster containing both elements of the pair.

The first criterion ensures that in examining the target clusters, all object pairs similar to each other can be found. The second constraint says that a cluster does not contain dissimilar objects, it contains only similar objects.

In the literature, there are two dominant variants providing an intra-cluster similarity: the Quality Threshold method and the BIRCH method. The first variant can provide a higher level of quality but it requires a higher execution cost. The second one is a good and robust solution in the case of huge data sets. The paper provides a comparison of these methods and suggests some modifications on the BIRCH algorithm to create an improved intra-cluster quality for large databases.

2. Overview of clustering algorithm with quality threshold

2.1. QT algorithm

The QT (Quality Threshold) clustering method [11] ensures that the distance between any two elements within a cluster should be below a given threshold. The algorithm uses two input parameters: the first parameter is the maximum distance diameter and the second is the minimum cluster size. The diameter is defined in the following way:

$$d = \max_{i,j} \left\{ \sqrt{(x_i - x_j)^2} \right\}. \tag{2.1}$$

The size of the cluster denotes here the number of elements within the cluster. The main steps of QT clustering are the following:

1. Generating candidate clusters for each element where the candidate cluster is built up with a greedy algorithm. Taking an element y, the cluster contains all elements closer to y than the maximum radius.

2. The candidate cluster with the maximum size is selected as a true cluster. The elements of this cluster are removed from the pool, the membership of the remaining candidate clusters is updated.

3. If the largest remaining cluster has a greater size than the minimum limit, go to step 2, otherwise terminate the algorithm.

The QT algorithm generates non-overlapping clusters where some elements remain outside of clusters as outliers. The output of clustering is a set of clusters of limited diameter, thus the similarity values between the elements are above a given threshold.

Considering the execution cost, the algorithm contains three embedded loops. The outer loop runs on the selection of real clusters, the loop in the middle runs on the candidate clusters and the inner loop runs over the elements to generate the candidate clusters. The cost value can be given by

$$O\left(\frac{N^2}{L} D\right),$$

where

N : number of elements in the data set,

L: average size of a cluster,

D: the cost of distance calculation.

Besides the base variant in the literature, there can be found some improved versions using some special techniques such as parallelism [12].

2.2. BIRCH algorithm

The BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [5] is of great importance as a fast and efficient pre-clustering method. Each resulting cluster is represented by one single point for the task-specific further clustering algorithms. The BIRCH algorithm guarantees that the distance of points in sets created is smaller than a threshold given in advance. The threshold is the most important input parameter of the BIRCH algorithm. By decreasing the threshold it can be guaranteed that points in sets are close enough to the centre of the set. Therefore the quality of clustering can be increased.

The structural information of the BIRCH algorithm is stored in Clustering Feature (*CF*) data triplet. The strength of *CF* is to the calculation of core clustering features using only simple arithmetic operations without analysing the individual elements. The structure *CF* is given by the following formula:

$$CF = \left\{ N, \overrightarrow{LS}, SS \right\}.$$

where

N: number of points in the set represented by the given *CF*,

$\overrightarrow{LS}$: d-dimensional linear sum of points in the set,

SS: d-dimensional square sum of points in the set.

The BIRCH algorithm organizes the points to be clustered in a balanced B+ tree. Each leaf-node stores points (or sets of points). Moreover, each leaf-node contains a pointer to the following node and a pointer to the preceding node in the tree to reach them fast. During the insertion operation, if the quality threshold condition does not hold, then, points in the given leaf-node are reassigned into two subsets. In the split operation the two furthest points of the set are used as nuclei of the new clusters. Then all the other points in the original set are moved into the leaf-node whose nucleus is closer to it.

Another core parameter of the algorithm is the branching factor. If the number of child nodes of a given non-leaf node exceeds the maximal branching factor, then the set is split into two parts. The two child nodes which are the furthest from each other are selected as the nuclei of the new sets. Afterwards all the other child nodes of the given non-leaf node move into the non-leaf node whose nucleus is closer.

If B denotes the maximal number of child-nodes of non-leaf nodes, M denotes the maximal size of the tree, then the cost function of the method can be given as [10]:

$$O(d \cdot N \cdot B \cdot (1 + \log_B M)) \quad .$$

Considering all components of the algorithm (e.g. reconstruction), the total time cost of the BIRCH algorithm can be calculated by the following formula [10]:

$$O(d \cdot N \cdot B \cdot (1 + \log_B M) + \log_2 \frac{N}{N_T} \cdot d \cdot (ES - 1) \cdot B \cdot (1 + \log_B M)) \quad . \qquad (2.2)$$

3. Optimal selection of parameters for BIRCH algorithm

Test results of clustering on the base BIRCH algorithm reveal that the time requirement of the procedure considerably depends on parameters of the threshold value and of the maximal branching factor. The test results show that the dominating factor is the threshold value, while the cost is considerably influenced also by the maximal branching factor *CF* of the tree. Tests performed also revealed that if the threshold value parameter is lower than the optimal value then the number of sets resulting by BIRCH algorithm is increased exponentially. The exponential increase in the number of sets results in an exponential increase in the cost of the post-processing algorithms. If the threshold value parameter is larger than the optimal value, then the number of points put into a cluster is increased, which requires a continuously increasing extra cost during insertion and reconstruction operations. If the CF value is too low, the depth of the tree increases exponentially. Administrating the increased number of nodes (creation, decomposition, memory usage, etc.) causes an increased total cost.
The goal function *(C)* is a function of two parameters: the threshold value *(T)* and the maximal branching factor *(B)*: *C=f(T,B)*. The best known representatives of local searching algorithms are: hill-climbing search [6], simulated annealing [7], local beam search, and genetic algorithm. In order to perform parameter optimization we have developed a combination of genetic algorithm and hill-climbing search algorithm.

The formal description of the applied hill-climbing search algorithm is as follows:

1. The actual point is a point given by randomly chosen coordinates *(T,B)*.
2. Determining the cost belonging to the actual point by carrying out the clustering procedure with parameters represented by the actual point.

3. Determining the neighbours of the actual point by increasing and decreasing the coordinates of the actual point by step ε.
4. Determining the costs belonging to the neighbours of the actual point by carrying out the clustering procedure for each neighbour of the actual point with parameters represented by the given neighbour.
5. If the cost of the neighbour having the smallest cost around the actual point is not smaller than the cost of the actual point, then terminate the algorithm.
6. Let the actual point be the point represented by its neighbour with the smallest cost. Take step 3.

The formal description of the algorithm reveals that in order to determine the optimal parameters the same clustering task needs to be performed repeatedly with different parameters. In a minimal case this requires performing the clustering task four times. Hence it is obvious that a clearance of the cost of clustering can only be expected when a relatively great number of samples with the same features are clustered. By denoting the cost of clustering without optimization by S, and the cost of clustering with optimization by O and the cost of finding the optimum by M, then after the i^{th} clustering the cost of clustering without optimizing is $i \cdot S$, and the cost of optimizing carried out with parameters T, B obtained by optimization is: $i \cdot O + M$. The gain of optimization can be given as

$$j \cdot S - (j \cdot O + M) \,. \tag{3.1}$$

To show the results of the simple hill-climbing method, a test was performed on a word-set containing 10,000 words from the Computer science book written by Gábor Kovács (Hungarian Electronic Library). The measured cost values are shown in Figure 2.

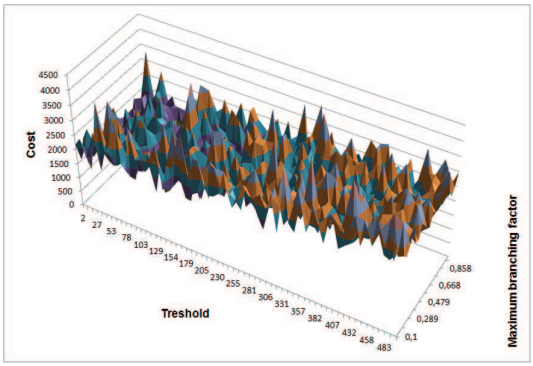


Figure 2. The value of cost calculated at each point of the domain in the example considered

The quality threshold values ran in the interval [0.1, ..., 1] with distance 0.05. For the *CF* parameter, 60 values at an equal distance from each other in interval [2, ..., 500] were chosen. As Figure 2 shows, there are many similar local optimum positions in the problem domain. In order to find a good approximation of the global optimum, the simple hill-climbing method was extended with a stochastic element, namely with a genetic algorithm module.

A group of discrete candidate points existing simultaneously creates a population. Each discrete point of the starting population is generated by coordinates assigned randomly. Each generated discrete point is the starting point of a hill-climbing search algorithm. After the local optimization process, the coordinates of each discrete point of the population are modified to the coordinates of the local minimum obtained by the search. Therefore each discrete point of the population gets into a local minimum. According to the cost of the discrete points of the population obtained in this way, the average cost that describes the population can be determined. If the average cost of the n^{th} population already exceeds the average cost of the population *(n-1),* then the algorithm terminates and the best discrete point in the population *(n-1)* is considered to be the best solution obtained by the optimization. If the n^{th} population is the first population or its average cost is smaller than or equal to the average cost of the population *(n-1),* then according to the discrete points of the n^{th} population a new population *(n+1)* is generated.

The algorithm uses the usual genetic operators such as selection, crossover and mutation, to build up the next generation. During the selection operation, a number of the best discrete points in the n^{th} population are placed into the new population without any change. For the generation of the rest of the points, the following possibilities are carried out:

- Crossover of points: denoting the coordinates of the selected discrete points by (t_i,b_i), (t_j,b_j), then the coordinates of the new point are (t_i,b_j).
- Mutation: denoting the coordinates of the chosen point by (t_i,b_i), the coordinates of the new point are $(t_i+\varepsilon_1,b_i+\varepsilon_2)$.
- A new point is generated randomly, independently of the coordinates of the selected elements.

The optimal values of parameters *T* and *B* are given by the coordinates of the point representing the lowest cost of the previous population.

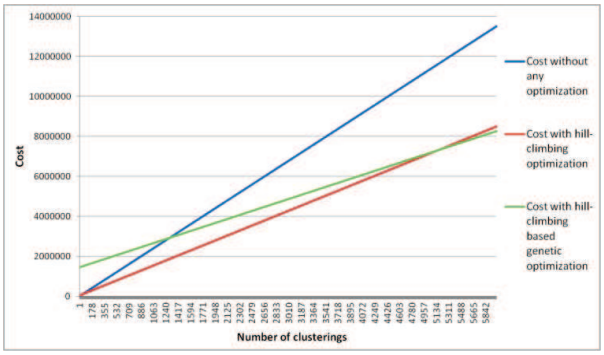


Figure 3. Dependence of costs obtained with and without optimization on the number of clustering

4. Analysing test results of clustering after BIRCH

During the test of the k-means post-clustering method after the BIRCH pre-clustering, we analysed the relationship between the following six parameters:

- the maximal distance (threshold) between points in the leaf-nodes of the BIRCH tree,
- the maximal branching factor of non-leaf nodes of the BIRCH tree,
- the expected number of clusters in a k-means algorithm based on the BIRCH algorithm,
- the number of clustering points,
- the number of alignments of points to be clustered (focus points),
- the dispersion of normal distribution of points to be clustered around focus points.

The first two parameters have a direct effect on the behaviour of the BIRCH algorithm. With the third one the number of clusters expected from the k-means can be set. The last three parameters relate to the problem domain. During tests always only one of the parameters was changed in order to be able to show its effect on the features of the algorithm. The following features were investigated:

- the time requirement of clustering (together with the k-means algorithm),
- the depth of the BIRCH tree,
- the number of leaf-nodes of the BIRCH tree (the number of sets created by the BIRCH algorithm),
- the number of steps in re-arranging the k-means clustering based on the BIRCH algorithm.

Results of the tests according to the above can be seen in the diagrams below.

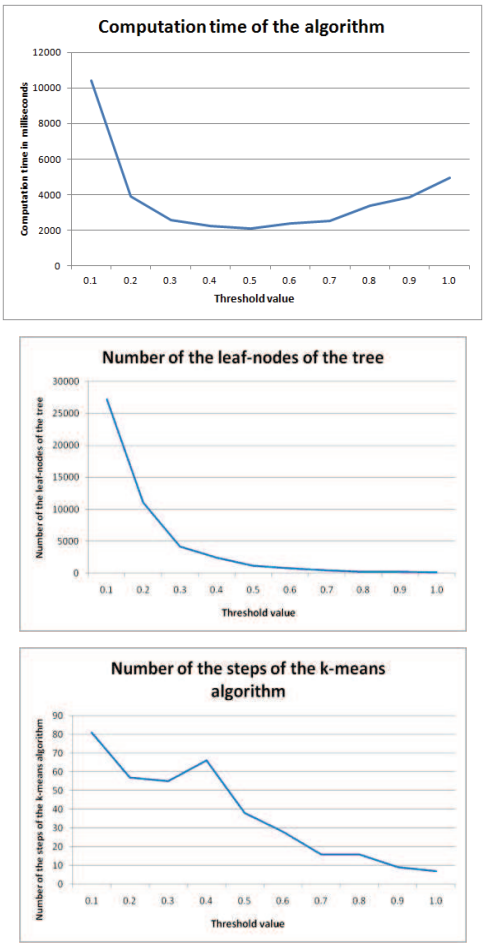


Figure 4. Analysing the effect of the threshold value

According to the test results, we can see that the number of leaf-nodes decreases exponentially with the increase of the threshold value. Therefore the k-means algorithm needs to cluster considerably fewer sets, hence the time requirement of clustering also decreases exponentially. By increasing the value of the threshold above a certain level, too many points are gathered in leaf-nodes and during the separation of sets a considerable number of points needs to be analysed. This effect can be shown in the increase of time consumption.

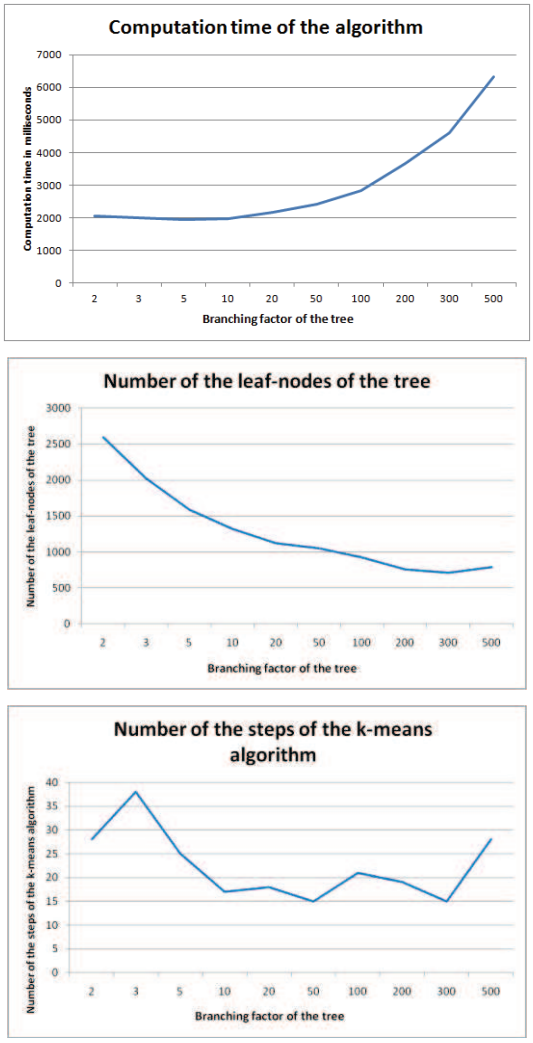


Figure 5. Analysing the effect of the maximal branching factor

By increasing the branching factor of the tree, the depth of the tree decreases exponentially. As in the case of a larger number of branchings, a considerably larger number of child nodes must be processed. This procedure results in an obvious increase in the rate of speed. Also, it is obvious that changing the branching factor does not have any effect on the function of the k-means algorithm.

5. Conclusions

An important aspect of clustering is to provide a good intra-cluster similarity. The BIRCH algorithm is a standard tool used to perform pre-clustering on large data sets using a quality-threshold constraint on the clusters. As the efficiency of the method depends on such parameters as cluster threshold and branching factor, the optimization of the parameters is a crucial step to reduce the costs of clustering operations. The proposed method, hill-climbing with genetic algorithm can be used efficiently to optimize the cluster threshold and branching factor parameters of the BIRCH algorithm. The test results show that a significant cost reduction can be achieved using the proposed optimization method.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] CRACRAFT, J., DONOGHUE, M.: *Assembling the tree of life: Research needs in phylogenetics and phyloinformatics*. Report from NSF Workshop, Yale University, July 2000.

[2] DAY, W.: Complexity Theory: *An Introduction for Practitioners of Classification, Clustering and Classification*, World Scientific Publ., 1992.

[3] MURTY, M., KRISHNA, G.: *A computationally efficient technique for data clustering*, Pattern Recognition, Vol. 12., 1980, pp. 153-158.

[4] SALTON, G.: *Developments in automatic text retrieval*, Science, Vol. 253., 1991, pp. 974-980.

[5] TIAN, Z., RAGHU R., MIRON L.: *BIRCH: A New Data Clustering Algorithm and Its Applications*, 1997.

[6] BART, S., CARLA P. G.: *Hill-climbing Search*, Cornell University, Ithaca, New York, USA, (Jan 15, 2006), pp. 333-336.

[7] SCOTT, K., CHARLES, D. G., MARIO, P. V.: *Optimization by Simulated Annealing*, *Science*, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[8] RUSPINI, E.: *A new approach to clustering*, Information Control, Vol. 15., 1969, pp. 22-32.

[9] MANNING, C., RAGHAVAN, P., SCHÜTZE, H.: *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[10] TIAN, Z., RAGHU R., MIRON L.: *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* Proc. of the 1996 ACMSIGMOD International Conference on Management of Data, Montreal, Canada, pp. 103–114, June 1996.

[11] HEYER, L., RAMAKRISHANAN, R., LIVNY, M.: *BIRCH: An Efficient Data Clustering Method for Very Large Databases*, Genome Research, Vol 9., 1999, pp. 1106-1115.

[12] MOCIAN, H.: *Survey of Distributed Clustering Techniques*, Internal Research Project, http://www.horatiumocian.com/papers/Distributed_Clustering_Survey.pdf, 2009.

Production Systems and Information Engineering
Volume 6 (2013), pp. 27-40

USING GRAPHICAL PROCESSING UNITS FOR DETERMINISTIC SINGLE MACHINE SCHEDULING PROBLEMS

KRISZTIÁN MIHÁLY
SAP Hungary Ltd., Hungary
krisztian.mihaly@sap.com

OLIVÉR HORNYÁK
University of Miskolc, Hungary
Department of Information Engineering
hornyak@ait.iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. This paper gives an introduction to how graphical processing units can be used in non-graphical related problems or tasks. First a history of GPU is provided. The next part focuses on GPU programming. A brief description is given about the available hardware facilities and the available programming languages. As an initial result of the project an easy and well-known scheduling algorithm was implemented for deterministic, single machine models. To check the performance achievement both the CPU and GPU code were implemented. Finally, some of the performance measurements are presented.

Keywords: GPGpu, OpenCL, CUDA, scheduling problems

1. Aims and scope of the paper

This paper aims to give an overview of the history of graphical processing units (GPUs) and how they can be used in non-graphical related tasks. After a short historical introduction it presents a short discussion on two programming languages and shows an easy example, where next to the CPU the GPU is used for solving deterministic, single machine scheduling problems.

2. Historical overview of GPUs

As Sanders put it "the state of graphics processing underwent a dramatic revolution. In the late 1980s and 1990s, the growth in popularity of graphically driven operating systems such as Microsoft Windows helped create a market for a new type of processors. In the early 1990s, users began purchasing 2D display accelerators for their personal computer. These display accelerators offered

hardware-assisted bitmap operations to assist in the display and usability of graphical operating systems" [1]. This accelerator approach was the first step to separate the graphic related tasks from the CPU. In 1992, Silicon Graphics opened the programming interface to its hardware by releasing the OpenGL library. It was a standardized, platform-independent method for developing 3D applications. At that time the computing of rendering was running in the CPU. The demand of the market was strong to have some kind of hardware support for 3D calculations to improve the application speed. The major companies were NVIDIA Corp., ATI Technologies and 3dfx Interactive.

"August 31, 1999 marked the introduction of the graphics processing unit (GPU) for the PC industry, the NVIDIA GeForce 256." [2] First the graphical hardware took care of the calculation of transformation and lighting computations, so these operations could run on the hard-wired graphical processors and the CPU could be used for other tasks. The next breakthrough in parallel-computing was the first graphic card, which supported the Microsoft's DirectX 8.0 standard. This standard introduced the programmable vertex and pixel shaders. This was the first point when the developers were able to influence the control of GPU programs.

Usually the developers used the GPU for graphics related tasks, but there were some developers who wanted to use the calculation power of the GPU cards. Since 2010 there has been a greater focus on the GPU's massive parallel computing capacity. Figure 1 shows how the calculation power of GPUs is increasing.

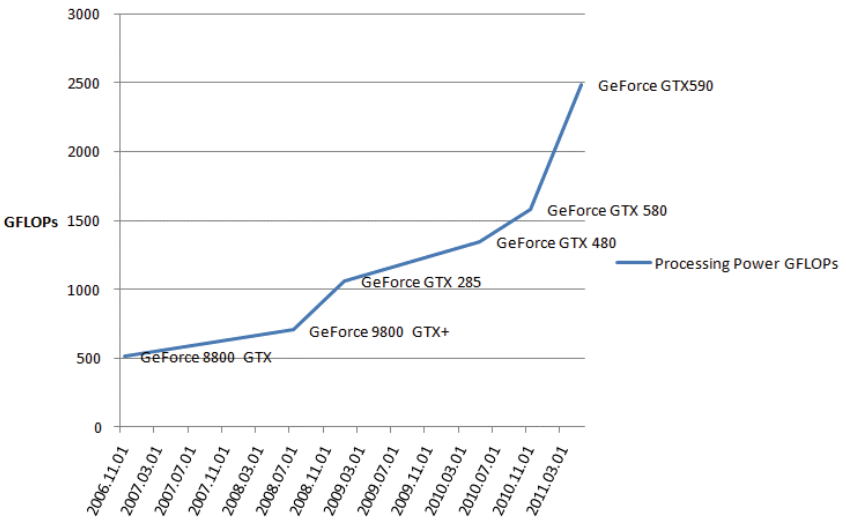


Figure 1. NVidia GPU card GFLOPs evolution

2.1 Tasks of the GPU and the programming environments

Let us have a look at what a GPU card should do. The application runs on the CPU and computes the 3D geometry. The geometry is loaded to the GPU and there is a transformation from 3D to 2D. After the transformation the card creates fragments and composes the image.

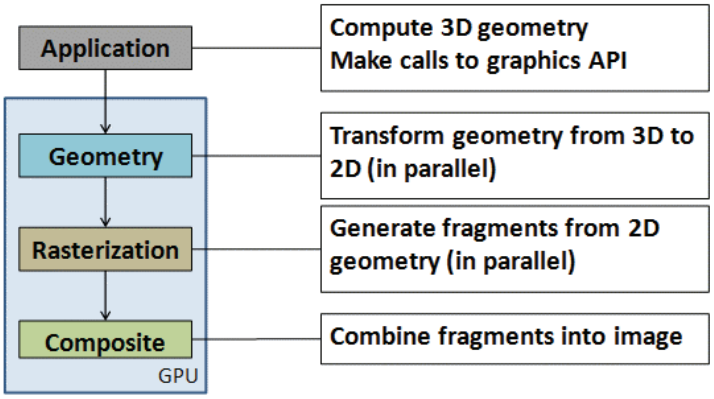


Figure 2. The rendering pipeline of the GPU

As described in the previous section, from the Microsoft's DirectX 8.0 standard on, the developer was able to influence the behaviour of the GPU through the so-called shaders. The geometry transformation can be changed through the vertex shader, the rasterization through the pixel shader.

There were disadvantages of programming GPUs through these shaders. The provided APIs were designed to support graphical APIs and the programmers needed a very deep knowledge of the graphical platform. There were resource constraints; data could be loaded as picture and texture and retrieved data was another picture. So if the algorithm required accessing a memory area to write, it could not run on GPU. It was nearly impossible to predict how your particular GPU can deal with floating-point data. There did not exist any good methods to debug implementations in GPU.

3. New GPU programming languages

Two main programming languages will be described briefly: they are used to develop applications running on GPU, without having any knowledge of the graphical API.

Open Computing Language (OpenCL)

OpenCL is a multivendor open standard for general-purpose parallel programming of heterogeneous systems that include CPUs, GPUs and other processors. OpenCL provides a uniform programming environment for software developers to write efficient, portable code for high-performance computer servers, desktop computer systems and handheld devices. It is managed by the Khronos Group [4]. This standard is applied by industrial companies and academics as well.

OpenCL has four main models

- Platform model

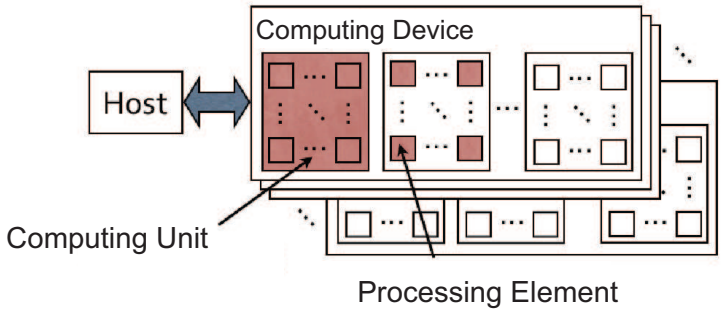


Figure 3. OpenCL platform model [12]

The host coordinates execution and data loading from computing devices. Each computing device has one or more computing units, where one or more processing elements take place.

- Execution model

The host role is context management and controlling of processes. The kernel takes care of controlling the computing units. The kernel program runs on the processing elements, achieves an index range and is grouped into work groups.

- Memory model

There are four main types of memory. The *global memory* is readable/writable from every processing element. Every processing element can access it. The *constant memory* is readable from the processing elements, the host allocates it and fills it with values. The *local memory* is accessed from computing units. Every process element can read/write it within the computing unit. The host cannot access it. The *private memory* is assigned to the processing element and only the assigned process element can read/write it.

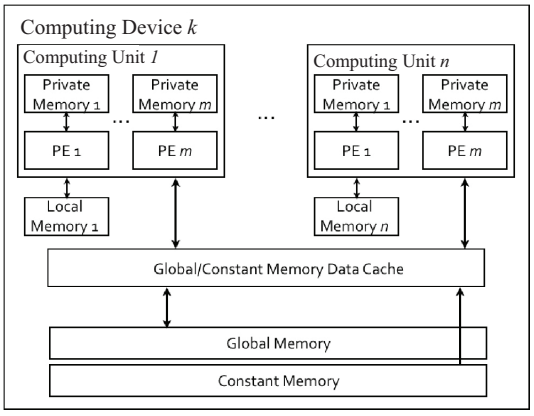


Figure 4. OpenCL memory model [12]

- Program model

  The division of a kernel into work-items and work-groups supports data-parallelism, but OpenCL supports another kind of parallelism as well, called task-parallelism.

OpenCL includes a language for writing so-called kernels. OpenCL provides parallel computing using task-based and data-based parallelism.

CUDA

CUDA was developed by NVidia Inc. and its architecture has two main parts: one is the hardware which supports the CUDA programming and can be called the device and the programming language to be able to create programs using the device capacity. The programming language is based on industry standard C and adds a relatively small number of keywords in order to harness some of the special features of CUDA architecture. There is a public compiler for this language, CUDA C. For further information you can refer to the CUDA Programming Guide by NVidia [5].
As mentioned before, CUDA is an extension of C language and includes GPU relevant APIs and interfaces. There are three main tasks, which are the tasks of the developers, such as thread hierarchy, memory access and synchronisation [6].

Thread hierarchy

To perform computation with CUDA, programmers have to define a special C function, the so-called kernel. This function can be run in a thread using a specified number of lightweight threads in GPU. The kernel is loaded to the device from the

host, where the normal code is running. Threads are grouped into blocks, and blocks form a grid. Threads can communicate through the memory area assigned to the block. The blocks run independently and their behaviour cannot be affected by the programmer.

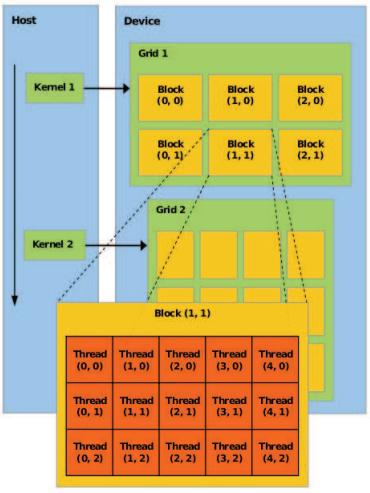


Figure 5. CUDA Thread Hierarchy [7]

Memory hierarchy

There are a great number of types of memory areas, such as global memory, shared memory, constant memory, registers and local memory. The differences between the memory types are the size, the accessing time and the caching property. For more details refer to [5] or to [6].

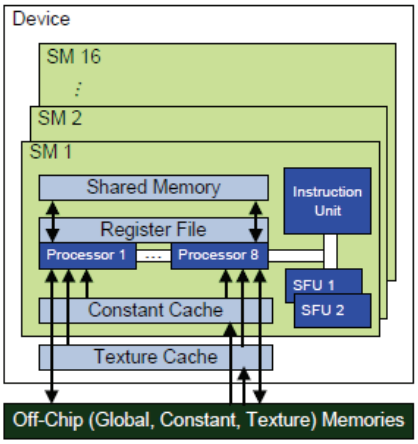


Figure 6. Basic organization of the GeForce 8800 [8]

Synchronisation

Sometimes algorithm behaviour requires a consistent state of the threads within the same block. For example when the shared memory is used by the threads, it could be necessary to have a consistent state in case after writing a thread. The CUDA language provides a __synctrheads( ) method, which defines a waiting point in the kernel code. The processing of a thread will be continued only if each thread reaches this point.

4. Single machine scheduling problems with objective function total weighted completion time

Single machine models are very simple and well-known models in the literature [9] [10]. Only the definition of this model is given here.

There are two main types of objects, the machine and the job. The machine (M) is the processing unit, which is capable of completing the jobs (J). The machine can process only one job at one time and job execution cannot be interrupted. Each job has a processing time (p) on the machine. Each job can have weight (w), which is a priority factor, denoting the importance of the job related to other jobs. Each job can have a completion time (C), which describes the point when the job has been finished. Processing time, weight and completion time will be indexed with j (p_j, w_j, C_j).

Each model can have one or more objective functions, which are used to compare feasible schedulings. This model uses the total weighted completion time as an objective function. This is the summary of the completion time weighted by the priority of a job.

$$TWCT = \sum_{j=1}^{n} w_j c_j \tag{4.1}$$

If there are two feasible schedules, the schedule with the lower TWCT has to be used.

5. First example of implementation

There is a proven theorem for this model [11]: The Weighted Shortest Processing Time First (WSPT) rule is optimal for single machine models, where the objective function is the total weighted completion time. This paper aims at comparing the following implementations for this problem:

- CPU
- GPU with CUDA

Each implementation is based on the following object model. This model is used and generalized later for other problem types.

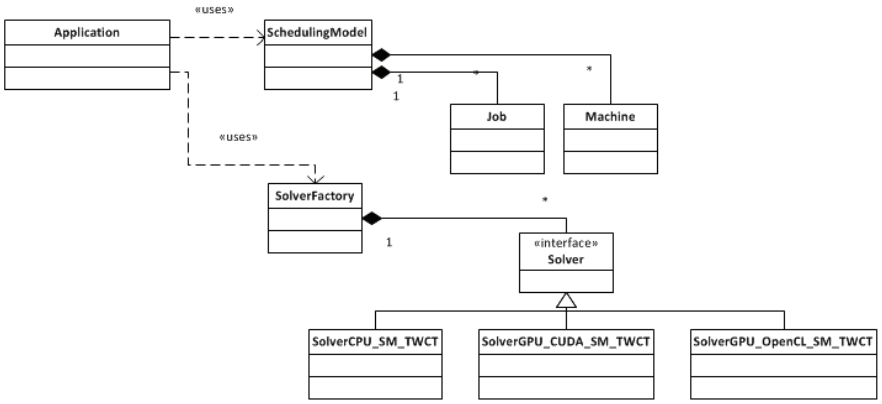


Figure 7. Application object model

Application builds up the scheduling model from a txt file, where the jobs, the job processing time and the job weight are stored. The main flow is described in Figure 8.

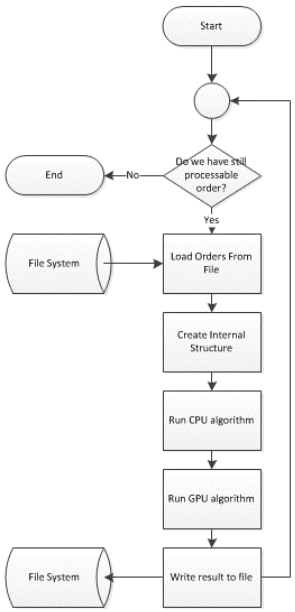


Figure 8. Application main flow

First the file is loaded from the file system and is mapped to the internal structures. In our implementation we always should have enough free memory to store our models. From this general model each algorithm should pre-fill its own data model. During performance measurements this is measured as well. After a running of the algorithms has been finished, the result and the performance result are stored in the file system.

The CPU algorithm

The CPU algorithm uses a vector where the element structure is:

- job identifier
- job processing time
- job weight
- job weight / job processing time calculation result.

The values are calculated in a loop. During one loop phase only one item in the vector can be processed.

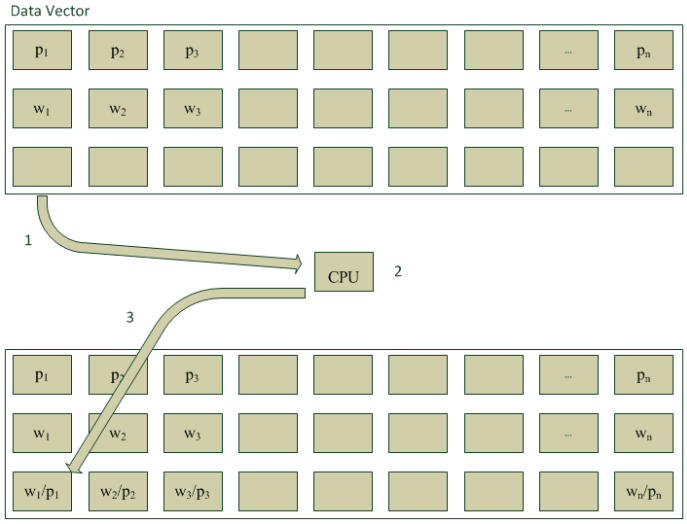


Figure 9. Value calculation flow with CPU

The implementation algorithm is very simple. The data vector is pre-filled with p_j and w_j values of the jobs. There is a loop where an index is used.

Step 1: Values from the index are accessed.

Step 2: The result value is calculated.

Step 3: The result is stored in the memory, the accessing index is increased and the next item will be processed (GoTo Step1) if there exists one.

As can be seen in Figure 9, only one w_j/p_j value is calculated at one time.

The GPU algorithm

The GPU is capable of running the same kernel parallel in several GPU cores. The vector is split into the available blocks and the data are loaded from the host to the device. On the device each processing unit uses the same kernel function and accesses the same memory.

In the single machine model the w_j/p_j values are independent of each other. That is why several independent processing units can be used.

Because the algorithm runs on the GPU, first data have to be loaded from the host (CPU) to the device. Next, memory is allocated to the device, then data are copied from the host. Vectors are used and to each vector item a separated core is assigned. It works only if the number of vector elements is smaller than 65 535. If it is not true, the vector is split into several fragments and the GPU kernel is called more times.

If data is loaded to the device, we start so many parallel kernels as possible to get the highest efficiency. Each processing unit uses the same kernel code.

```
__global__ void gpuCalculateWSPT( int *a, int *b, float *c){
      int tid = blockIdx.x;
      c[tid] = a[tid] / b[tid];
}
```

Step 1: During runtime each GPU core accesses one piece of the data. The indexes come from the CUDA platform and they are called block indexes.

Step 2: The result value is calculated.

Step 3: The result is stored in the device memory.

After the calculations the data have to be copied from the device to the host.

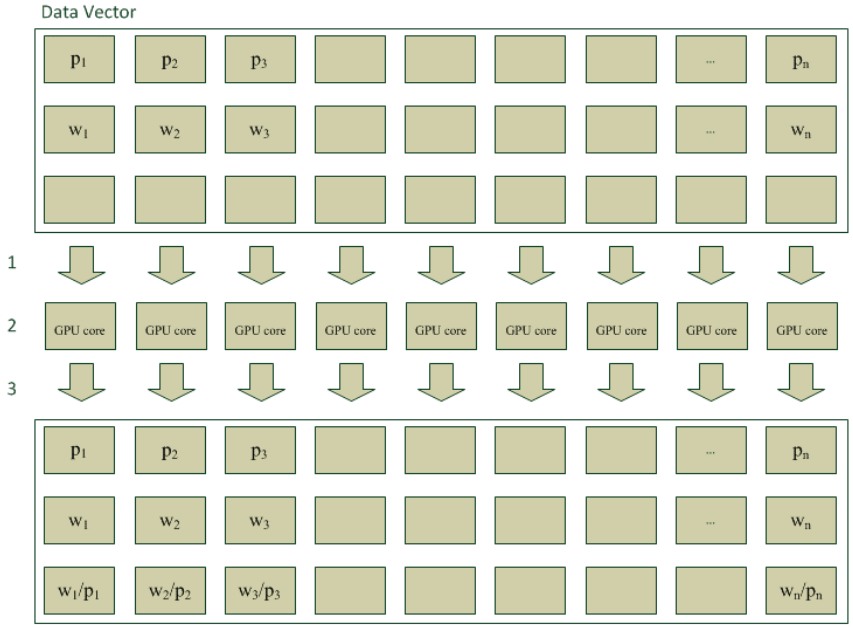


Figure 10. Value calculation flow with CPU

Performance measurements

In order to be able to run performance measurements, a test data set was generated. The main program loads the available test data file from the file system, starts the CPU and GPU solver, measures the processing time and stores the result file back to the file system.

To execute the performance measurements, the following PC configuration was used:

- CPU : Intel Core i3 – 2310M
- GPU: NVIDIA GeForce GT 520M
- Memory: DDRIII 2GB

The test program was developed in C++. The GPU programming model uses the NVida CUDA version 1.1.

The measurements are depicted in the following diagramme.

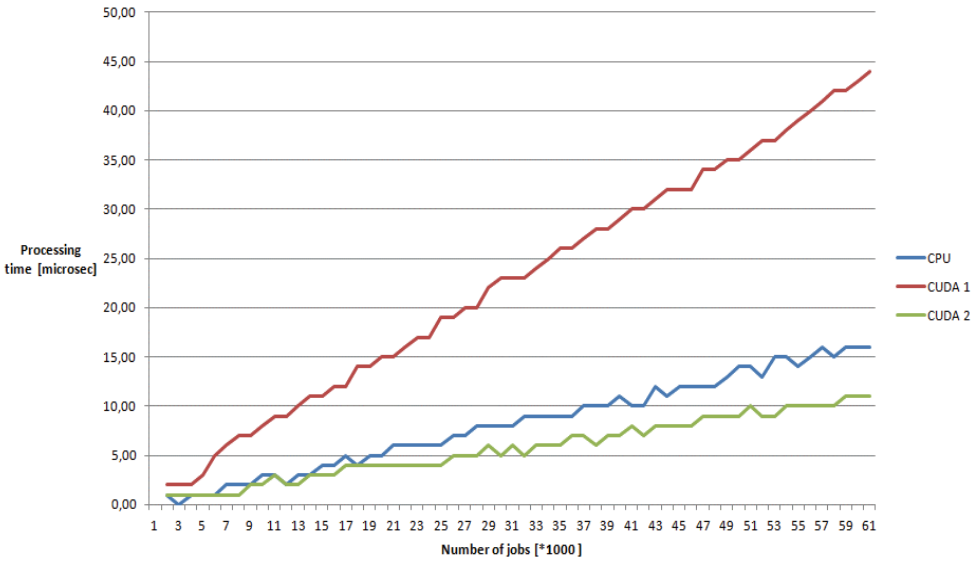


Figure 11. Performance measurements

The axis X shows the number of generated jobs (divided by 1000). The axis Y shows the processing time in microsecunds.

The CPU line shows the processing time of the CPU algorithm.

CUDA 1 line shows the GPU algorithm processing time when the internal data mapping, data copying and the calculation were measured.

CUDA 2 line shows the GPU algorithm processing time when internal data mapping was not necessary. In this case only the processing time of the calculation result and the data copying from the host to the device and vica versa were measured.

Conclusion

As can be seen from the performance results, the solution has O(n) complexity in each case. The CUDA 2 measurement contains the necessary mapping between the structures, the CUDA 1 measurement contains only the calculation of the w_j/p_j rate. As shown, the GPU algorithm has a better performance if the data are structured and stored as appropriate for the GPU.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] SANDERS, J., KANDROT, E.: *CUDA by example,* Addison-Wesley, 2010, p. 3

[2] INTERNET: http://www.nvidia.com/page/geforce256.html, (2012.01.05)

[3] INTERNET: http://www.hardwareinsight.com/nvidia-cuda/, (2012. 01.05)

[4] INTERNET: http://www.khronos.org/opencl/

[5] INTERNET: NVidia Programming Guide 4.0, NVidia Developer Zone http://developer.download.nvidia.com/compute/cuda/4_0_rc2/toolkit/docs/CUDA_C_Programming_Guide.pdf (2011. 08. 25)

[6] CZAPISNKY, M., BARNES, S.: Tabu search with two approaches to parallel flowshop evaluation on CUDA platform, J. Parallel Distrib. Comput. (2011)

[7] INTERNET: http://mohamedfahmed.wordpress.com/2010/05/03/cuda-computer-unified-device-architecture/ (2011. 08. 25)

[8] RYOO, S., RODRIGES, C. I., BAGHSORKHI, S. S., STONE, S. S.: Optimization Principles and Application Performance Evaluation of Multithread GPU Using CUDA

[9] BLAZEVICZ, J., ECKER, K.H., PESCH, E., SCHMIDT, G., WEGLARZ, J.: Scheduling Computer and Manufacturing Processes, Springer

[10] CHEN, B., POTTS, C.N., AND WOEGINGER, G.J.: A review of machine scheduling: Complexity, algorithms and approximability. Handbook of Combinatorial Optimization (Volume 3) (Editors: D.-Z. Du and P. Pardalos), 1998, Kluwer Academic Publishers. 21-169.

[11] PINEDO, M. L.: Scheduling; Theory, Algorithms, and Systems, Springer, 2008, p. 36.

[12] INTERNET: http://developer.amd.com/documentation/articles/pages/opencl-and-the-amd-app-sdk.aspx

Production Systems and Information Engineering
Volume 6 (2013), pp. 41-56

DETAILED PRODUCTION SCHEDULING BASED ON MULTI-OBJECTIVE SEARCH AND SIMULATION

GYULA KULCSÁR
University of Miskolc, Hungary
Department of Information Engineering
kulcsar@ait.iit.uni-miskolc.hu

MÓNIKA KULCSÁRNÉ FORRAI
University of Miskolc, Hungary
Department of Automation and Communication Technology
kulcsfm@mazsola.iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. This paper presents an advanced approach to solving fine scheduling problems of production in practice. It focuses on creating near-optimal feasible schedules considering detailed constraints and capabilities of the resource environment. It is a very important and complicated task to make an efficient schedule for the shop floor to include different types of facilities and operations. The proposed model supports the flexible usage of production goals and requirements simultaneously. The elaborated approach uses a special searching technique with multiple neighbouring operators and problem space transformation based on execution-driven simulation. Successful applications of the methods in real manufacturing environments are also demonstrated.

Keywords: scheduling, simulation, multi-objective optimization, production

1. Introduction

In modern manufacturing/assembling production environments, a great number of scheduling problems may occur. Production scheduling can be defined as the allocation of available production resources over time to perform a collection of tasks [1]. It is a very important decision making process at the operation level. Most of the scheduling problems are highly complicated and hard to solve owing to the complex nature of the applied model. Today, production engineering and management utilize more and more computer integrated application systems to support decision making.

This paper is primarily concerned with industrial scheduling problems, which require advanced scheduling software to assign limited available resources to the operation of jobs and to sequence the assigned operations on each resource over time. It is mainly concerned with discrete manufacturing, in which typically series

of items are produced. The series (batch or lot) can include very different numbers of pieces, from a single product (i.e. special part, complex or unique equipment) to thousands or millions of the same product (simple parts). In discrete manufacturing operations are executed on discrete, separate machines and workplaces. Depending on the arrangement of machines, robots, buffers and material handling devices, manufacturing systems may be characterized by different layouts (i.e. single machine, parallel machines, line, flexible line, group, etc.). In essence the execution of the operations requires the exact prior definition of the feasible routing alternatives [4].

2. Scope of research

The paper focuses on the fine (or detailed) scheduling function of Manufacturing Execution Systems. The main purpose of fine scheduling is to initiate a detailed schedule so as to meet the master plan defined at the Enterprise Resource Planning level. The scheduler is able to get the actual data of dependent production orders, products, resource environment and other technological constraints (tools, operations, buffers, material handling, etc.). The shop floor management configures the actual production goals and their priorities. Obviously, the management may declare various goals time by time. The scheduler has to provide a feasible schedule which meets the management's goals. The result of the scheduling process is a detailed production program which declares the releasing sequence of the jobs and the operations, assigns all the necessary resources to them and proposes the starting time of activities. The execution of the production program has to meet the predefined goals without breaking any of the hard constraints. The computation time of the solving process is also an important issue, especially with a large number of internal orders, jobs, operations, resources, technological variants and constraints.

In the literature, different flexible scheduling functions with various models are found. One of the main groups of these models is the flexible flow shop (FFS) scheme. A detailed survey of the FFS problem is given by Quadt and Kuhn [7], and Wang [10]. The FFS environment consists of stages that represent the fundamental (operation-type) machine groups of the system. At each stage one or more identical machines work in parallel. Each job has to be processed at each stage on any of the parallel concurrent machines.

Considering the production performance, both the allocation of machines and the sequence of jobs are of great importance. A great number of shop scheduling models are known in the literature, but most of them use only one performance measure. Usually the latest finishing time (make-span) of the released jobs appears as a goal function of optimization for Make to Stock (MTS) manufacturing. Frequently, one objective function related to due date plays the main role in scheduling models for supporting Make to Order (MTO) manufacturing. Only a

few of the models deal with multi-objective cases which are very important in flexible and agile manufacturing [2, 6, 8, 9].

The existing models in the operation research field often disregard the machine processing abilities, limited availability time frames of the machines, limited buffer capacities, and shared machine tools, that is why the improvement and extension of flexible shop models is justified. In order to consider the aforementioned important features of real scheduling problems we have focused on the simulation based scheduling approach. This paper presents our approach in which an execution-driven fast simulation is used to transform and reduce the problem spaces preserving important details.

3. Practice-oriented fine scheduling approach

The starting point of the new research was the Extended Flexible Flow Shop model (EFFS) developed by the first author of this paper [5]. The main goal of our new research is the aforementioned model which will be improved in order that the problem class EFFS can also represent the problems for smaller units. From the execution point of view, the units inherit the schedule of the job concerned but are moved along in the shop environment and have their own due date.

In this paper an integrated approach is proposed to solve this complex scheduling problem class as a whole without decomposition. In the approach, all the issues (batching, assigning, sequencing and timing) are answered simultaneously. For solving production scheduling problems in practice, a knowledge intensive searching algorithm has been developed based on execution-driven fast simulation, overloaded relational operators and multiple neighbouring operators. The core of the engine implemented explores iteratively the feasible solution space and creates neighbour candidate solutions by modifying the actual resource allocations, job sequences and other decision variables according to the problem space characteristics. The objective functions concerning candidate schedules are evaluated by producing a simulation which represents the real-world environment with capacity and technological constraints. In this execution-driven simulation, items, parts, units and jobs are passive and they are processed, moved, and stored by active system resources such as machines, material handling devices, manpower and buffers. The numerical tracking of the product units provides the time data of the manufacturing steps. The simulation process extends the pre-defined schedule to a fine schedule by calculating and assigning the time data. Consequently the simulation is able to transform the original searching space into a reduced space by solving the timing sub-problem. This is the part of the approach that encapsulates the dependency of real-world scheduling problems. Successful adaptation of the approach into practice is highly influenced by the efficiency of the simulation algorithm.

4. Application of the approach in practice

4.1. Motivation

The scheduling problem, which will be outlined in this section, is inspired by a real case study concerning only one of the plants of Electrolux-Lehel Ltd. specialized in refrigerator products (Jászberény, Hungary). The firm produces different types of refrigerators with variable series simultaneously. It is typical that the market centres or other distributors require very hard delivery due dates. Another important market trend is that the number of product variants is increasing. The companies are forced more and more to customize products to market demands and to important seasons. These requirements cause a tendency to decrease lot size, to develop better forecast, make more flexible production plans, and to use information technology, operation research and artificial intelligence methods for more efficient shop floor scheduling.

4.2. Problem description

The discrete manufacturing process examined produces various refrigerators as final products. The production planners create the actual production plan for the next time interval (typically one or two weeks) by using ERP system. External orders, forecasts, market trends, seasonal characteristics are considered in decision making. The planning phase is concerned with balancing supply and demand. The production plan generated defines the production program of the final products at the finishing technological step of the manufacturing processes. In other words, the production plan consists of internal production orders and each production order specifies the final product needed, the required quantity, the destination and the due date demanded. The destination means the finishing machine which is assigned to the production order for processing. According to the process plan of the product type, pre-defined technological steps must be executed on the component parts.

The shop contains different machine groups connected to each other in a given configuration (Figure 1). Each machine group contains a pre-defined number of machines. In essence, refrigerator manufacturing includes two fields that can be distinguished: the cabinet and door manufacturing processes. These sub-processes can be modelled and analysed separately. In the system, refrigerators are assembled on two finishing assembly lines (AL 1, AL 2). In this finishing step the cabinet and the door meet with many other component parts to become one final product. Both of them are very important from the viewpoint of overall stability. All the necessary components have to be available at a given time at the destination pre-defined. In order that this requirement should be satisfied, a detailed schedule for all the predecessors are created efficiently to synchronize all the manufacturing steps and sub-processes.

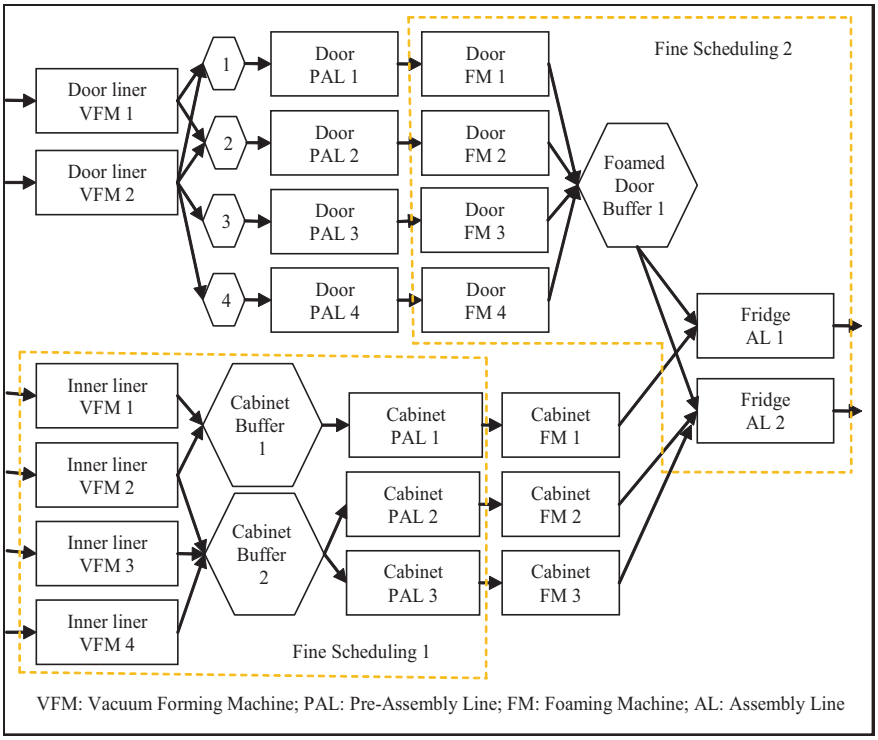


Figure 1. Scheme of refrigerator manufacturing

Based on the results of process analyses, it was clear that the scheduling of the inner liner vacuum forming and the door foaming play key roles. These sub-problems of the shop scheduling problem are denoted by *Fine Scheduling 1* and *Fine Scheduling 2* in Figure 1. The remaining part of this paper describes the applied model and solution method for detailed scheduling of door foaming.

The main purpose of the second R&D project was to develop a fine scheduler software in order to initiate detailed schedule for door foaming machines to realize the production plan and to synchronize the events on the finishing assembly lines of the manufacturing system.

4.3. Modelling the problem

4.3.1. Resource environment

In the problem labelled *Fine Scheduling 2* in Figure 1, z machines M_m $(m = 1, ...,z)$ have to process n jobs J_j $(j = 1, ..., n)$. A job J_j consists of b operations $O_{j1}, ...,O_{jb}$. The execution sequence is defined by the following precedence relations between the operations: $O_{jl} \rightarrow O_{j,l+1}$, for $l=1, ..., b-1$. Operation O_{jl} is not specified by a pre-defined processing requirement p_{jl}, because processing time depends on the

assigned resources. Each operation O_{jl} is associated with a set of machines $\mu_{jl} \subseteq \{M_1, ..., M_z\}$. O_{jl} may be processed on any of the machines in μ_{jl}. Usually, μ_{jl} is not one element set and μ_{jl} is not equal to the set of all machines.

Focusing on the door foaming scheduling, we have two operations: the first one is door foaming (FM) and the second one is final assembly (AL). The machine is dedicated to processing the AL of jobs because the input production plan determines the destination. For processing the door FM, the suitable machines are parallel. The foaming process of different types of doors can be executed on different foaming machines with various production intensities by using suitable tools. The pre-defined assignments and combinations mean hard constraints. In general, the number of available tools and machines is more than one for each door type. So the tools and machines belong to independent resource groups and can be used as shared resources. The foaming machine and the tools which are assigned to a given foaming operation O_{j1}, have to be available simultaneously for O_{j1} during the whole processing period. Scheduling problems of this type are called multi-processor task scheduling problems. A schedule is feasible if no two time intervals overlap on the same machine or the same tool, and if it meets a number of additional problem-specific characteristics.

The machines can work only within the time frames pre-defined. These machine availability intervals (shifts) have constant lengths (i.e. eight hours). Each machine has its own shift distribution. In order to minimize the cost, it is very important for shop floor control management that all the planned and started shifts are utilized as much as possible.

The capacity of the buffer shared among foaming machines and final assembly lines is strongly limited. As in the system different products are manufactured, we have to take into consideration the buffer capacity, which depends on the product types, because the volume of the buffer is constant but the sizes of the products are different. Consequently, in the calculation we use the relative utilization of the buffer.

4.3.2. Job characteristics

In order to create a final product, given components are taken through pre-defined processes. A job includes work-pieces and their operations. A job as a series of work-pieces must be scheduled in the manufacturing system. The actual work-piece depends on the operation to be performed. The operation is an elementary process of the manufacturing and changes one or more significant characteristics of the work-piece. A given sequence of operations must be executed on work-pieces to create final products. A sequence of operations means a technological step which can be performed on a single machine; similarly, a sequence of technological steps fulfilled by an integrated production or assembly line is an execution step.

Pre-emption of operations, technological steps or execution steps on a work-piece is not allowed in a classical sense, but a job execution on a machine can be suspended if the output buffer of the machine is full or the machine is unavailable for processing. Precedence relations between complete jobs are not specified, but restricted precedence chains of jobs (last operations) on finishing machines are given by the input production plan.

In the scheduling problem considered the set I of all jobs is partitioned into disjoint sets $I_1, ..., I_w$, where $I = I_1 \cup I_2 \cup ... \cup I_w$ and $I_f \cap I_g = 0$ for $f, g \in \{1, ..., w\}, f \neq g$. For any two jobs $J_x \in I_f$ and $J_y \in I_g$ to be processed on the same machine M_m, job J_y cannot be started before $sett_{mfg}$ time units after the finishing time of job J_x. Similarly, job J_x cannot be started before $sett_{mgf}$ time units after the finishing time of job J_y. The groups correspond to different types of products and $sett_{mfg}$ may be interpreted as a machine dependent changeover (or set-up) time. During the changeover period, the machine cannot process another job. If $f = g$, then $sett_{mfg} = 0$ for all $f, g \in \{1, ... , w\}$, $m \in \{1, ...,z\}$, and most of cases, if $f \neq g$, then $sett_{mfg} \neq sett_{mgf}$.

Generating a schedule for the following time horizon, it has to be considered that the machines can be loaded with unfinished tasks. So the input data set has to include the actual state variables of the system. It means that the effect of the last confirmed schedule must be considered when creating a new schedule.

4.3.3. Objective functions

In order to express the shop floor management's goals as criteria of a multi-objective optimization problem, we use seven objective functions to be minimalized. These are as follows:
- the number of tardy jobs,
- the sum of tardiness,
- the maximum tardiness,
- the number of set-up activities,
- the sum of set-up times,
- the average waiting rate of machines, and
- the average flow time of jobs.

4.4. Solving the problem

4.4.1. Decision variables

In the approach applied the job plays the role of the basic scheduling item. In order to create the fine schedule of the foaming machines, it is necessary for each job:
- to be assigned it to one of the suitable foaming machines,
- to fix its execution position in the queue for the assigned machine,
- to be assigned it to one of the suitable tools for use on the assigned machine, and

- to pre-set its starting time on the assigned machine.

To make decisions on these issues is very complicated. We developed a new approach in order to answer these questions. The main idea of this approach is a problem space transformation based on simulation. We use the following sets of independent decision variables to represent a candidate schedule as a solution:
- the sequence of job-machine assignments on each foaming machine,
- the specification of the availability time frames (shifts) of each foaming machine.

The availability time intervals of a given machine M_m are expressed by a pre-defined calendar (CAL_m). CAL_m is a list which stores the availability time frames specified by means of start time and end time values. The calendar elements of the foaming machines are decision variables, where the availability time frames of the final assembly machines belong to the set of constraints.

The decision variables of this reduced problem space form a simple schedule which will be extended to a fine schedule (detailed production program) by using simulation. The simulation, which is a fast execution-driven simulation of the simple schedule, answers the remaining issues concerning the tools needed and the starting time data of the execution steps. Consequently, the simple schedule determines the fine schedule. Sizes of production batches are formed dynamically by scheduling the jobs and executing the production units on machines.

To accelerate the simulation we use indexed data structures as attributes of the model objects. This memory model is based on indexes which are non-negative integer values assigned to the entities, to point to the position in the target object. The data model developed supports the association of two or more different type objects (i.e. machines and jobs). Before scheduling a builder method creates the full indexed data model which includes the valid technological and resource constraints and possible alternatives (i.e. job dependent sets μ_{jl} of machines).

4.4.2. Internal due dates

From the execution point of view it is allowed that the job consists of smaller production units (PU). These PUs inherit the schedule of the job concerned but are moved along in the shop environment. The internal due date of a given job on a given foaming machine is equal to the planned starting time of the same job on the pre-defined final assembly line. Considering that the machines can only work in accordance with pre-defined calendars, the internal due date of a given job refers to the first unit of the job in the strict sense. The exact due date of the other units of the job can be calculated by simulating the execution of the job on the final assembly line assuming that all of the units arrive in time. Using this procedure, the precise due date can be adjusted and assigned to each unit. These values are used as indirect input data of the scheduling problem. In the simulation and the evaluation of a candidate schedule, a given job will be delayed if either of its units is delayed.

4.4.3. Execution-driven simulation

An execution-driven simulation can be realized with a rule-based numerical simulation of the production to calculate the time data of the execution steps. Input data determine the production order, the jobs, the production units, the resources and the schedule to be executed. The schedule specifies the assignment of jobs and machines, and in addition defines the execution sequences of jobs on machines and the shift arrangement of machines.

Every job is represented by an individual model object (J_j) in the simulation. A J_j means a set of work-pieces of the same type. All of the work-pieces of a given J_j are processed on the same technological route by the same machines using the same tools. The route and the machines are chosen by the scheduler and defined in the schedule to be executed. A J_j consists of units U_i ($i=1, \ldots, v_j$). Every U_i represents one or more work-pieces (item series) to be moved as an individual atomic unit among machines. An execution step of a given unit on a machine means a processing task.

The main steps of the simulation are as follows:
- build and initialize the model objects,
- choose the next execution step (task) to be performed,
- simulate the execution of the active unit on the active machine.

The most important objects of the simulation model are the production orders, the jobs, the tasks, the machines, the buffer, the tools, the input schedule, the output fine schedule and the object of performance indicators. At the beginning of the calculation these objects are initialized with the actual values of the system state variables.

The execution-driven method calculates and stores the time data of the execution steps. On each machine the execution sequence of the assigned jobs is pre-defined. The main issue is how the limited resources (tools and buffer) affect the execution. To answer this issue, the simulation must perform all of the activities in a suitable sequence. This sequence cannot be pre-defined but it is part of the simulation. In an intermediate situation, the next execution step must be chosen from the set of candidate units. Each machine has a loading list and a pointer that shows the next unit to be processed according to the schedule. The pair of a given machine and its mentioned unit means a candidate execution step for processing if all the starting requirements are satisfied. These are as follows:
- the machine is not blocked by the buffer,
- the machine has finished its previous unit,
- the unit execution has been completed successfully on its previous machine,
- one of the suitable tools is available for processing.

The method chooses the candidate execution step which can be started the earliest. The machine and the unit associated with the chosen execution step become active

entities. The method calculates the time data (start time ST_{mi}, set-up time $SetT_{mi}$, processing time $ProcT_{mti}$, and completion time CT_{mi}) of the active unit on the active machine. The processing time ($ProcT_{mti}$) is determined by the work-piece quantity (q_i) of the unit, the tool and the unit (product type) dependent production rate (pr_{mti}) of the machine. The start time ST_{mi} of a given unit U_i on an assigned machine M_m is determined by the following values:
- the end time of the interval while the machine is blocked by the buffer (BT_{mi}),
- the end time of the interval while the tool is unavailable or engaged (e_t),
- the earliest release time of the unit (r_i),
- the completion time of the unit on the previous machine (ct_{pi}),
- the moving time of the unit from the previous machine (mt_{ipm}),
- the completion time of the previous unit on the machine (ct_{mh}),
- the unit-sequence dependent set-up time on the machine ($sett_{mhi}$),
- the availability time frames of the machine (CAL_m).

Focusing on the simulation of the execution step of unit U_i on the assigned machine M_m, the simplified description of the calculation can be seen in Figure 2 assuming that the set-up activity can be started on the machine before the unit arrives (a_{mi}).

$a_{mi} = ct_{pi} + mt_{ipm};$
$SetT_{mi} = sett_{mhi};$
$ProcT_{mi} = q_i / pr_{mti};$
$ST_{mi} = max(a_{mi} – SetT_{mi}, ct_{mh}, r_i - SetT_{mi}, BT_{mi}, e_t);$
$CT_{mi} = ST_{mi} + SetT_{mi} + ProcT_{mi};$
$Load\_STET\_to\_CAL(ST_{mi}, CT_{mi}, M_m);$

Figure 2. A simplified calculation of the time data of a given execution step

The function *Load_STET_to_CAL* loads the timeframe required by unit U_i on machine M_m. This allocation method inserts the set length time window [ST_{mi}, CT_{mi}] into the first suitable time frame of machine M_m. While the full size of the required time window does not fit in the candidate time interval, the time window is moved right to the next candidate time interval. This version of the load function represents that the execution step of the unit is not pre-empted in time.

Simulation of an execution step covers the handler of the tools and the buffer involved. The work-piece(s) of a given unit must be taken out of the buffer prior to the start of the execution on a final assembly line. If the execution step is finished successfully on a foaming machine, the work-piece(s) of the unit are to be put in the buffer. If the buffer is full, then the work-piece(s) stays on the foaming machine, therefore the next unit cannot be started and the foaming machine will be blocked. If the stock level in the buffer decreases below a given value, the blocked machine or machines will be released.

One of the suitable tools is allocated by the first unit of the job before starting the execution step. If more than one suitable tool is available at the same time, the earliest released tool will be chosen. The exclusive reservation of the chosen tool will be cleared by the last unit of the job.

The most important output data of the execution-driven simulation of the production are coded in a data object *MSTET* which stores the evaluated time data of all units. The simulation extends the pre-defined input schedule to a fine schedule by calculating and assigning *MSTET* in a short time. The performance of the fine schedule can be measured by calculating objective functions based on the data of units, jobs, and machines. In this way the simulation is able to transform the original searching space of the scheduling problem into a reduced space.

4.4.4. Search algorithm

For solving the scheduling problem in an integrated form addressed above, we developed an advanced multi-operator and multi-objective search algorithm based on overloaded relational operators, multiple neighbouring operators and a special taboo list which stores schedules in coded form (MOMOTS). Our approach is based on the taboo searching meta-heuristics that was first suggested by Glover [3] and has been used frequently for different combinatorial optimization problems.

```
MOMOTS
{ s0 ← Generate an initial solution;
  s* ← s0;
  Taboo_List ← NULL;
  while ( Stop criterion is not satisfied )
  { while ( Extension criterion is satisfied )
    {  Nc ← Choose the actual neighbouring operator(priority_list);
       s ← Generate a neighbour solution( s0 , Nc);
       if ( Taboo_List does not include ( s ) )
         {  Insert new taboo into the first position of Taboo_List ( s );
            if ( Number of Taboos > Maximum number )
               Delete the taboo from the last position of Taboo_List;
            if ( This is the first solution of the extension ( s ) ) sk ← s;
            else if ( s < sk ) sk ← s;
         }
    }
    s0 ← sk;
    if ( sk < s* ) s* ← sk;
  }
  return s*;
}
```

Figure 3. Multi-Operator and Multi-Objective Taboo Search (MOMOTS)

Our search algorithm variant (Figure 3) iteratively moves from an actual schedule s_0 to a candidate schedule s in the neighbourhood of s_0 until the stop criterion is satisfied. To reach and examine the unexplored regions of the search space, the method modifies the neighbourhood structure of each schedule as the search progresses. To escape local optimum, the method contains the schedules that have been visited in the recent past (less than a given number of moves ago) in a taboo list. Schedules in the taboo list are excluded from the neighbourhood of the actual schedule. A certain number of neighbours of the current schedule are generated at random successively by using priority controlled neighbouring operators. The operator can only modify the first execution step (decision variables) of jobs in the schedule (solution) because the second one is pre-defined by the input data (constraints). The applied neighbouring operators are as follows:

- operator N_1 moves a randomly chosen job elsewhere,
- operator N_2 moves a randomly chosen tardy job elsewhere,
- operator N_3 chooses a machine randomly and modifies the sequence of jobs on the machine by using a random length permutation cycle,
- operator N_4 exchanges two adjacent jobs on a machine which is chosen randomly,
- operator N_5 chooses a tardy job and moves left one position in the sequence,
- operator N_6 allows a denied calendar element randomly chosen on a machine,
- operator N_7 denies an allowed calendar element randomly chosen on a machine.

The operators listed create new candidate schedules by modifying the values of the decision variables of the initial schedule. The objective functions concerning candidate schedules are evaluated by the execution-driven simulation. The overloaded relational operator $<$ is used to compare the generated schedules according to multiple objective functions described in Section *4.3.3.* These objective functions are given so that:

$$f_k : S \to \Re^+ \cup \{0\}, \forall k \in \{1, 2, ..., K\}. \tag{4.1}$$

Coefficients w_k *(k=1, ..., K)* as input parameters support that the user may calibrate the actual priority of each f_k independently. Each w_k is an integer value within a pre-defined close range [*0, 1, ..., W*] and expresses the importance of f_k.

Let s_x, $s_y \in S$ be two candidate solutions. Function F is defined by (4.2) to express the relative quality of s_y compared to s_x as a real number.

$$F : S^2 \to \Re, F(s_x, s_y) = \sum_{k=1}^{K} (w_k \cdot D(f_k(s_x), f_k(s_y))). . \tag{4.2}$$

Function D defined by (4.3) means the comparison of s_x and s_y according to f_k.

$$D : \Re^2 \to \Re,\ D(a,b) = \begin{cases} 0, \text{if } \max(a,b) = 0 \\ \dfrac{b-a}{\max(a,b)}, \text{otherwise.} \end{cases} \tag{4.3}$$

Using (4.2) the relational operators are overloaded by (4.4):

$$(s_y\ ?\ s_x) := (F(s_x, s_y)\ ?\ 0). \tag{4.4}$$

Any of the relational operators (i.e. in C++ programming language: <, >, <=, >=, ==, !=) can be used between two solutions to compare them as two real numbers. For example: s_y is a better solution than s_x ($s_y < s_x$ is true) if $F(s_x, s_y)$ is less than zero.

After comparing candidate solutions in an actual loop, the best schedule becomes the initial solution of the next loop. When the scheduling process is finished or stopped by the user, the currently best known schedule is returned, so the method can be used in any-time working model.

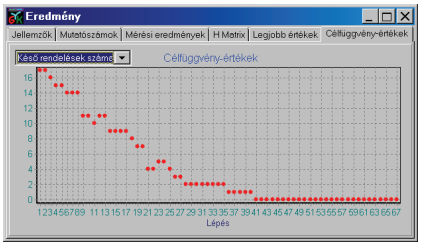
4.5. Some numerical results

For testing the proposed multi-objective predictive scheduling method we have used the data sets developed in Electrolux-Lehel Ltd. (Refrigerator Manufacturing Plant, Jászberény, Hungary), which represent real industrial problems.

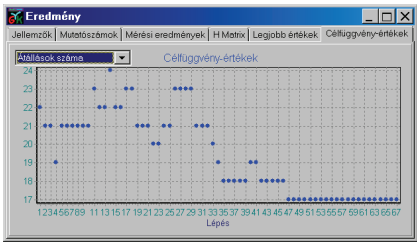
One of the case studies is summarized in Table 1. The main characteristics of the problem are as follows: scheduling time horizon is one week, time unit is one minute, number of jobs is 30, and number of production units is 16201.

Table 1. Sample Results of Multi-Objective Scheduling

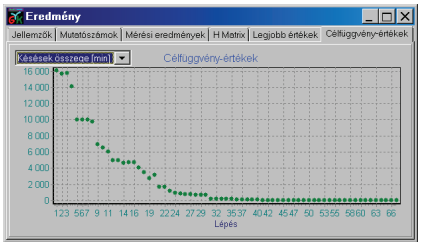
Objective function priority (w_k)	**Objective function (f_k)**	**Initial solution (s_0)**	**Best solution (s^*)**
5	f_1: number of tardy jobs	18	0
5	f_2: sum of tardiness [min]	16044.20	0
10	f_3: maximum tardiness [min]	2289.68	0
5	f_4: number of set-up activities	23	17
5	f_5: sum of set-up times [min]	760	500
5	f_6: average utilization rate of the machines [%]	38.12	91.55
5	f_7: average flow time of jobs [min]	1851.28	1419.48



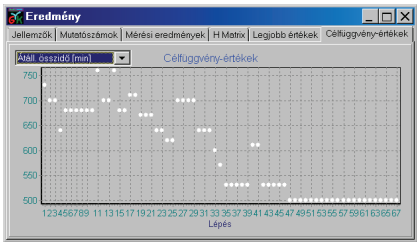
f_1: number of tardy jobs



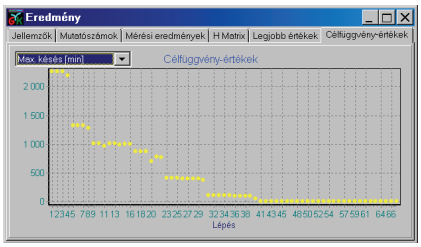
f_4: number of set-up activities



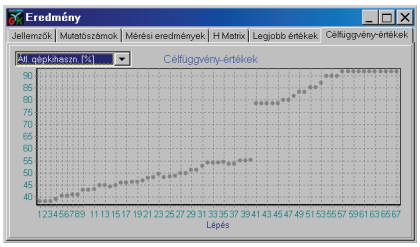
f_2: sum of tardiness [min]



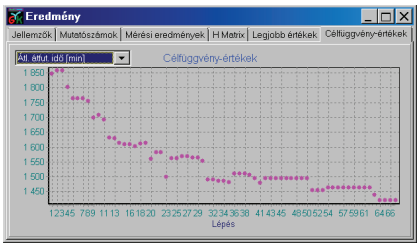
f_5: sum of set-up times [min]



f_3: maximum tardiness [min]



f_6: average machine utilization rate [%]



f_7: average flow time of jobs [min]

Figure 4. Actual values of the objective functions represented in the searching steps

Parameters of the searching algorithm are as follows: the maximum number of elements in taboo list is 150, the number of neighbour solutions in extension is 50, and the priority of each neighbouring operator is 1. In this case the computational time of the solution process is approximately 50 sec. The software was coded in C++ language. Concerning the running test environment we have; Intel^{R} Core^{TM} 2 Duo T9550 2.66GHz CPU, Windows Vista OS, 4GB RAM. The actual values of

the objective functions represented in the searching steps can be seen in Figure 4. All applications of the objective functions occur simultaneously. The diagrams (screen shots) are generated by our software (with a Hungarian user interface).

The proposed method can solve the examined problems effectively in a short time. The software developed is used in daily practice at shop floor control level of the plant.

5. Conclusions

The paper describes the proposition and application of a practice-oriented approach for solving multi-objective scheduling problems. It is based on execution-driven simulation and use of relational operators for comparing qualities of schedules in search algorithms. After developing the software, the concept is successfully tested on extended flexible shop problems considering multiple objectives and constraints originating from an industrial environment. Scheduling based on simulation can consider exactly what the actual manufacturing system should perform in the planned time horizon. In this approach, each schedule created for the shop is a feasible solution, because all of the hard constraints are considered. The results obtained and the independent nature of the approach encourage the application of the method in other multi-objective optimization problems.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] BAKER, K.: *Introduction to Sequencing and Scheduling.* 1st ed. Canada: John Wiley & Sons, 1974.

[2] BAYKASOĞLU, A., ÖZBAKIR, L., DERELI, T.: *Multiple Dispatching Rule Based Heuristic for Multi-Objective Scheduling of Job Shops Using Tabu Search*. In Proceedings of the 5th International Conference on Managing Innovations in Manufacturing, pp. 396-402, Milwaukee, USA, 2002.

[3] GLOVER, F.: *Tabu Search: a Tutorial.* Interfaces, 20, 74-94, 1990.

[4] KULCSÁR, GY., ERDÉLYI, F.: *A New Approach to Solve Multi-Objective Scheduling and Rescheduling Tasks*. International Journal of Computational Intelligence Research, 3, (4), pp. 343-351, 2007.

[5] KULCSÁR, GY., KULCSÁRNÉ, F. M.: *Solving Multi-Objective Production Scheduling Problems Using a New Approach.* Production Systems and Information Engineering, A Publication of the University of Miskolc, 5, 81-94, 2009.

[6] LOUKIL, T., TEGHEM, J., TUYTTENS, D.: *Solving Multi-Objective Production Scheduling Problems Using Metaheuristics*. European Journal of Operational Research, 161, pp. 42-61, 2005.

[7] QUADT, D., KUHN, H.: *A Taxonomy of Flexible Flow Line Scheduling Procedures*, European Journal of Operational Research, 178, pp. 686-698, 2007.

[8] SBALZARINI, L. F., MÜLLER, S., KOUMOUTSKOS, P.: *Multiobjective Optimization Using Evolutionary Algorithms*. In Center of Turbulence Research, Proceedings of the Summer Program 2000, pp. 63-74, 2000.

[9] SMITH, K. L., EVERSON, R. M., FIELDSEND, J. E.: *Dominance Measures for Multi-Objective Simulated Annealing*. In Proceedings of Congress on Evolutionary Computation, pp. 23-30, 2004.

[10] WANG, W.: *Flexible Flow Shop Scheduling: Optimum, Heuristics, and Artificial Intelligence Solutions*, Expert Systems, 22, (2), pp. 78–85, 2005.

Production Systems and Information Engineering
Volume 6 (2013), pp. 57-68

MODERN SOFTWARE RENDERING

PETER MILEFF
University of Miskolc, Hungary
Department of Information Technology
mileff@iit.uni-miskolc.hu

JUDIT DUDRA
Bay Zoltán Nonprofit Ltd., Hungary
Department of Structural Integrity
judit.dudra@bay-zoltan.hu

[Received January 2012 and accepted September 2012]

Abstract. The computer visualization process, under continuous changes has reached a major milestone. Necessary modifications are required in the currently applied physical devices and technologies because the possibilities are nearly exhausted in the field of the programming model. This paper presents an overview of new performance improvement methods that today CPUs can provide utilizing their modern instruction sets and of how these methods can be applied in the specific field of computer graphics, called software rendering. Furthermore the paper focuses on GPGPU based parallel computing as a new technology for computer graphics where a TBR based software rasterization method is investigated in terms of performance and efficiency.

Keywords: software rendering, real-time graphics, SIMD, GPGPU, performance optimization

1. Introduction

Computer graphics is an integral part of our life. Often unnoticed, it is almost everywhere in the world today. The area has evolved over many years in the past few decades, where an important milestone was the appearance of graphic processors. Because the graphical computations have different needs than the CPU requirements, a demand has appeared early for a fast and uniformly programmable graphical processor. This evolution has opened many new opportunities for developers, such as developing real-time, high quality computer simulations and analysis.
The appearance of the first graphics accelerators radically changed everything and created a new basis for computer rendering. Although initially the graphics pipeline and the programmability of the cards followed a very simple model, the previously existing software rasterization quickly lost its importance because CPUs of that

time were not able to compete with the performance of the graphics hardware. From the perspective of manufacturers and industry, primarily speed came to the fore against programming flexibility and robustness.

So in recent years the development of videocard technology focused primarily on improving the programmability of the fixed-function pipeline. As a result, today's GPUs have quite effectively programmable pipelines supporting the use of high-level shader languages (GLSL, HLSL, CG).

Nowadays, technological evolution is proceeding in quite a new direction introducing a new generation of graphics processors, the general-purpose graphics processors (GPGPU). These units are no longer suitable only for speeding up the rendering, but tend the direction of general calculations similarly to the CPU.

However, the problems of GPU-based rasterization should be emphasized. The applied model and the programming logic slowly reach their limits, the intensity of progress is apparently decreasing. Though there are fast hardware supported pipelines in current graphics cards, they do not provide the same level of flexibility for the programmer to manage the rendering process as CPU based rendering. The reason for this is that the pipeline is adapted to hardware limitations and there are many other limitations in utilization of shader languages.

Although the existing pipeline and 3D APIs provide many features for developers, if we would like to deviate from conventional operation, we encounter many difficulties. The general purpose programming of the GPU unit is limited because of the memory model and the fixed-function blocks, which are responsible for performing parallel thread executions [7]. For example, the sequence of pixel processing is driven by rasterization and other dedicated scheduling logic. This is clearly demonstrated by the uniform and predictable look and attitude of today's computer games [1].

Today's GPU architecture is questionable and needs to be reformed, as leading industrial partners strongly suggest [17,18]. What if developers could control every aspect of the rendering pipeline? The answer is practically a return to *software rendering*. A good basis is provided for this by the huge revolution in CPUs occuring in recent years. Processor manufacturers responded with extended instruction sets to market demands making faster and mainly vectorized (SIMD) processing possible also for central units. Almost every manufacturer has developed its own extension, like the MMX and SSE instruction family which are developed by Intel and supported by nearly every CPU. Initially, AMD tried to strengthen with its 3DNow instruction set, but nowadays the direction of development is the Vector Floating Point (VFP) technology and the SSE like NEON instruction set initially introduced at ARM Cortex-A8 architecture.

Due to new technologies, software can reach about 2-10x speedup by exploiting properly the hardware instruction set. All this combined with the GPGPU technology, the question arises: Why could a full software implemented graphical pipeline not be developed where all parts are programmable? Although the

performance probably would not compete completely with a graphical unit, it would offer a more flexible solution than today's only GPU-based solutions.
The main aim of this paper is to examine how it is possible to develop a software renderer built on modern basis, which points forward, is fast enough and takes advantage of technological opportunities inherent in today's central units.

2. Related work

Computer graphics has always been a very crowded area over the years, but with the spread of tablet PCs and mobile devices it has come fore even more. It is common in almost all fields whether physical simulation, modeling, multimedia or the area of computer games. However, although the GPU based display has a detailed literature, the area of software rendering has only a small number of new publications since the release of GPUs.
Software based image synthesis has been there since the first computers and it was focused even more with the appearance of personal computers until about 2003. Thereafter almost all visualization became GPU based. Among the software renderers born during the early years, the most significant results were the Quake I, Quake II renderers (1996), which are the first real three-dimensional engines [5]. These graphics subsystems were developed by the coordination of Michael Abrash, and were typically optimized for the Pentium processor family taking advantage of the great MMX instruction set. Among the later results, the Unreal engine (1998) can be highlighted, whose functionality was very rich at the time (colored lightning, shadowing, volumetric lighting, fog, pixel-accurate culling, etc) [13].
After the continuous headway of GPU rendering, software rasterization was increasingly losing ground. Despite this, some great results have been born, such as the Pixomatic 1, 2, 3 renderers [15] by Rad Game Tools and the Swiftshader by TrasGaming [2]. Both products are highly optimized utilizing the modern threading capabilities of today's Multicore CPUs and have dynamically self-modifying pixel pipelines. In addition, Pixomatic 3 and Swiftshader are 100% DirectX 9 compatible.
Microsoft supported the spread of GPU technologies by the development of DirectX, but besides this, its own software rasterizer (WARP) has been implemented. Its renderer scales very well to multiple threads and it is even able to outperform low-end integrated graphics cards in some cases [3].
In 2008 based on problem and demand investigations, Intel aimed to develop its own software solution based videocard within the *Larrabee* project [7]. The card in a technological sense was a hybrid between the multi-core CPUs and GPUs. The objective was to develop a fully programmable software pipeline using many x86 based cores [4].
Today, based on the GPGPU technology, a whole new direction is possible in software rendering. Loop and Eisenacher [2009] describe a GPU software renderer for parametric patches. Freepipe Software rasterizer [Liu et al. 2010] focuses on

multi-fragment effects, where each thread processes one input triangle, determines its pixel coverage and performs shading and blending sequentially for each pixel. Interestingly, recent work has also been done by NVidia to create a software pipeline which runs entirely on the GPU using the CUDA software platform [8]. The algorithm uses the popular tile-based rendering method for dispatching the rendering tasks to GPU. Like any software solution, this allows additional fexibility at the cost of speed.
Today's leading computer game Battlefield 3 [14] introduced a new SPU (Cell Synergistic Processor Unit) based deferred rendering process, which makes it possible to handle and optimize a large number of light sources.
In [1] the author outlined a modern, multi-thread tile based software rendering technique. The solution utilized only the CPU and had great performance results.
Thus, recent findings clearly underline the fact that in order to increase power and flexibility CPU-based approaches come to the fore again.

3. Software rendering

The imaging process is called software rasterization when the entire image rasterization process is carried out by the CPU instead of a target hardware (e.g. GPU unit). The shape assembling geometric primitives are located in the main memory in the form of arrays, structures and other data. The logic of image synthesis is very simple: the central unit performs the required operations (coloring, texture mapping, color channel contention, rotating, stretching, translating, etc.) on data stored in the main memory, then the result is stored in the *framebuffer* (holding pixel data) and sends the completed image to the video controller. The following figure shows a general pipeline of a software renderer:

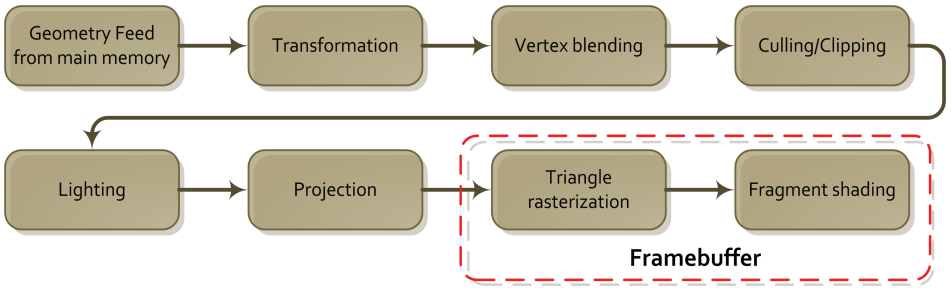


Figure 1. General graphics software pipeline

If we look at the pipeline stages, we can see that two dominant groups are formed during the image rasterization process. The first group includes mainly *vertex transformation operations*, which takes up to framebuffer operations. In these phases, the pixel level rasterization is prepared by several matrix and vector transformations (e.g. coordinate system, vertex, cameras, cutting). The second group includes *per-pixel operations*, such as triangle discretization and pixel

shading. For graphics engines from the perspective of rendering these two groups, but mainly the second, are the computationally intensive task.
However, optimizing stages in both groups can result in significant speedup. In the following several modern performance improvement techniques are outlined.

3.1 Benefits of software rasterization

The software image synthesis has many advantages over the GPU-based technology. As the CPU performs the whole processing, there is less need to worry about compatibility issues because we do not have to adapt to any special hardware, or follow its versions. The image synthetis can be programmed uniformly using the same language as the application, so there is no restriction on the data (e.g. maximum texture size) and the processes compared to GPU language shader solutions. Every part of the entire graphics pipeline can be programmed individually. Preparing the software to several platforms causes fewer problems because displaying always goes through the operating system controller, there is no need for a special video card driver.
In summary, software rendering allows more flexible programmability for image synthesis than GPU technology.

3.2 Disadvantages of software rasterization

The main disadvantage of software visualization is that all data are stored in the main memory. Therefore in case of any changes of data, the CPU needs to contact this memory. These requests are limited mostly by the access time of the specific memory type. Frequent changes on segmented data in memory cause significant loss of speed.
The second major problem, which originates also from the bus (PCIe) bandwidth, is the movement of large amounts of datasets between the main and the video memory. During one second the screen should be redrawn at least 50-60 times, which results in a significant amount of dataflow between the two memories. In case of a 1024x768 screen resolution, 32 bit color depth, one screen buffer holds 3 MB data.

4. General acceleration opportunities and difficulties

Today's modern processor architecture offers many opportunities to increase the performance of the computationally intensive parts in the graphics pipeline. Naturally, to achieve really good results it is necessary to combine these methods, but due to space limitations this article focuses only on the most important techniques.

4.1 SSE based pipeline optimization

In recent years, the most important innovation was built around the SIMD processor instruction sets (Intel – SSE family, AMD – 3DNow, Apple – AltiVec, ARM –

NEON). These allow us to accelerate calculations in a vectorized way and can achieve multiple speed improvement in the pipeline. Besides, another important aspect is the question of programmability: How difficult is it to turn an existing code into an SSE code? Will the code be portable to other operating systems?
Among desktop computers the SSE instruction set is widely accepted today and the instruction set based programming is well-supported by compilers (e.g. GCC, Intel). Modern compilers provide several options to build SSE codes. It is possible to implement the code in assembly language, which requires a deep programming knowledge and the code will not always be portable. Another option is to use the higher level *Intrinsics library* of the compiler. This approach makes the programming level high enough and comfortable (e.g. GCC: __m128 z = mm_setzero_ps(); - fills the vector with zero bytes), and does not limit portability either.

4.1.1 SSE based vector optimization

SSE (Streaming SIMD Extensions) is a SIMD instruction set family (currently SSE 4.2) developed by Intel for x86 architectures. The main innovation is that SSE originally added eight new 128-bit registers, known as XMM0 through XMM7. The extended instruction set provides the opportunity for the processor to execute an operation (e.g. multiplication) on the data of two vectors in parallel.

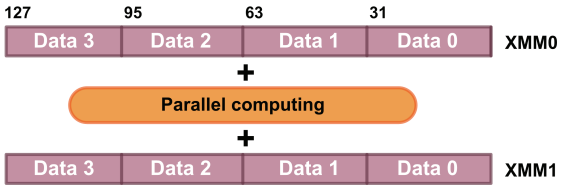


Figure 2. Parallel computing with SSE vectors

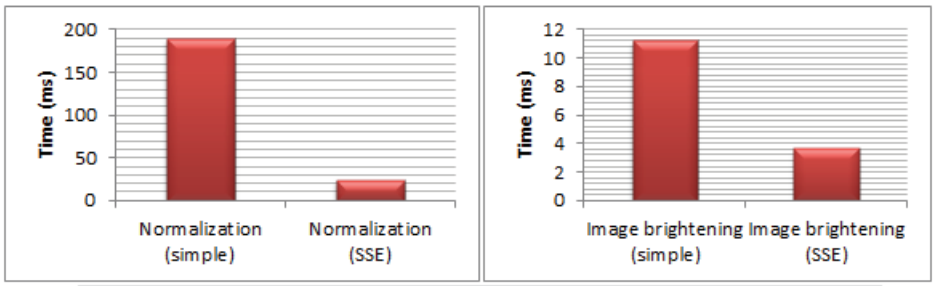
The first operation group of the pipeline typically consists of some kind of vector transformations performed on large datasets. In case of three-dimensional visualization, applying the SSE instruction set makes it possible to store four different 32 bit length floating point numbers (x,y,z,w) in a 128 bit length vector. This means that calculations can be made on these numbers at the same time, which results in significant speed improvements in the execution of the pipeline transformations.

4.1.2 SSE based image processing

The SSE instruction family can be also successfully applied in the rasterization stage of the pipeline or in any other graphical transformation because most of the pixel operations are independent of each other and can be executed in parallel. Today's graphics engines use typically 32-bit (RGBA) color component framebuffers, where each component is 4 bytes long. This mapping fits well with the SSE approach because four pixels colors can be stored in a 128 bit length register and operations can be performed on it in parallel.

4.1.3 SSE test results

In the following, the efficiency of the SSE solution is presented through two test cases. The first test demonstrates a general calculation, vector normalization, which is often used by graphics engines. The formula is compute intensive because it contains square roots and divisions. During the test process 50,000 vector normalizations are repeated 200,000 times.
The second test demonstrates the strength of SSE in an everyday pixel-level image processing task. The test performs 1000 brightening transformations on a 32 bit, 1024x768 resolution image. The test programs were written using SSE2 instruction set, C++ language and GCC 4.4.1 compiler was used and the measurements were performed by an Intel Core i7 870 2.93 GHz CPU. The following table shows the results of each test case.



Normalization (simple)	Normalization (SSE)	Image brightening (simple)	Image brightening (SSE)
188,7903 ms	24,1256 ms	11,2148 ms	3,6719 ms

Figure 3. Comparison of the computing results

Measurement results prove the strength of the SSE-based programming. The performance of the calculations in pipeline bottlenecks can be multiple improved with the appropriate SSE code. While in the first case the speed improvement is 7.8x, the second test shows that SSE was 3.05 times faster compared to the conventional code.

4.1.4 Drawbacks of SSE programming

Applying the SSE-based programming, certain compromises are required. Although this instruction set is well-supported by today's compilers (such as C++), an efficient, fast SSE-based code adaptation requires higher programming skills. SSE is not the Holy Grail, a poorly written code can be slower than the traditional approach.
The only restriction of SSE is that the stored and used data must be 16 byte aligned (evenly divisible by 16) in memory. Without this, the arithmetic instructions cannot be used directly. The disadvantage of the aligned memories is that data storage probably does not require 16-byte alignment, so basically we are wasting memory. In return we can gain high performance.

4.2 Working with Alignment

The graphics pipeline, thus rasterization speed can be even further improved if we store data properly aligned in memory. All data in memory have two properties: value and address. Data alignment means that the address of the data is divisible by one of the numbers (1,2,4,8) representing the byte length of the alignment. In other words, a data object can have 1-byte, 2-byte, 4-byte, 8-byte alignment or any power of 2. The CPU does not read from or write to memory one byte, instead accesses memory in 2, 4, 8, 16, or 32 byte chunks at a time.
Therefore if the related datasets of the graphics pipeline are not properly aligned to 4, 8, or 16 byte order, then these misaligned structures can cause serious performance losses, because CPU has to perform extra work (load 2 chunks, shift and combine) to access the data [12]. A simple case:

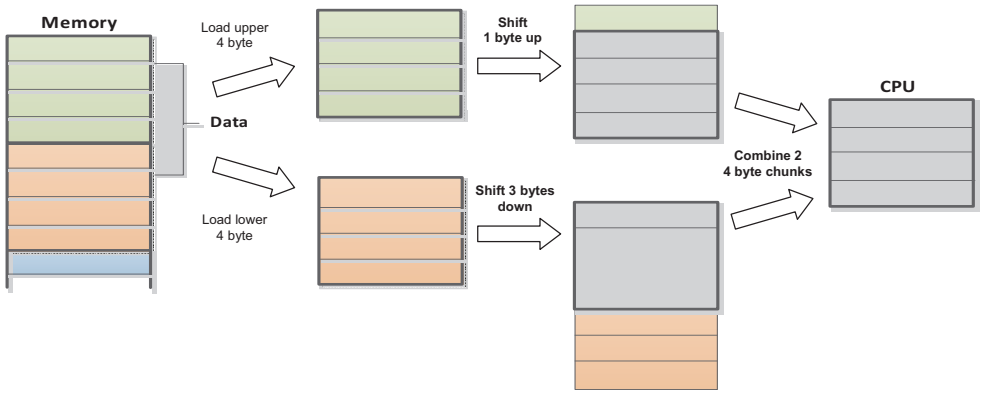


Figure 4. Unaligned data usage by CPU

The alignment problem of the pipeline structures is relatively easy to solve in lower-level languages (e.g. C, C++, D). The principle of member alignment is defined by the current compiler, but in most cases the rules are the same. The alignment should be always based on the most restrictive structure member, which is usually the largest intrinsic type. Therefore, members of a data storage structure should be ordered in descending order according to their size. Thus we get an aligned memory structure. In case of larger structure blocks this not only saves memory but the efficiency of rasterization can also be increased significantly.

4.3 Minimize cache misses

Today's processors have at least one first-level instruction and data caches on chip, and may have second-level cache memory. Memory access speeds are much faster from these storages. If the pipeline wants to access some kind of data during its running and these data are not in one of the caches, then a *cache miss* event is generated and the data will be loaded into the cache. This event is very costly.

While a value of a variable can be loaded during some clock cycles from the cache, loading it from the main memory requires hundreds of clock cycles. Due to this rule an important goal is to reduce cache misses with the following proposals:

- Frequently used data should be stored together and not segmented,
- Avoid pointer indirection, store and access frequently used data in flat, sequential data structures,
- Accessing data sequentially minimizes cache misses, because each cache miss will load *n* number of new data into the cache,
- Group the functions which work on the same data.

5. GPGPU accelerated software pipeline

The GPU-based visualization technology is moving today towards to general purpose processing. This opens up new possibilities for computer visualization and engineering simulations because the graphics processors are no longer limited only to displaying graphics, but can be used for any calculations. The latest GPGPU cards are hiding huge computing power (~1 Tflops/s) because of the inherent growing number of streaming processors (e.g. NVidia GTX 480 has 480 CUDA cores), fast memory (GDDR5) and advanced technology. Since the traditional GPU-based pipeline is not flexible enough, why cannot we use the GPGPU solution of the graphics hardware to implement the pipeline entirely in software?
Logically, the general purpose options of the GPU can be used for any stages of the graphics pipeline with certain restrictions. Since rasterization is a much more computing-intensive task, the computations should be divided between the CPU and GPU in the way that the GPU performs the tasks from the projection stage. The GPU, due to its design and purpose, is very good at parallel task execution. And the process of rasterization typically belongs among well-parallelizable calculations.

5.1 Working together with GPU

The objective of the rasterization stage is to map triangles of the models to screen pixels considering the impact of lights, materials and any other factors. In case of software rasterization the typical rendering process is that the CPU takes triangles sequentially from memory, maps their points to the two-dimensional plane and finally calculates their pixels. The color of pixels is stored in a memory array, adapted to the screen resolution, called the *framebuffer*, and after the rasterization the buffer is copied to the video memory.
GPGPU support of the process can be achieved in several ways. For the ideal solution the characteristics and the programming language (OpenCL, CUDA) options of the GPU should be taken into account [9]. The smallest unit of their programming model is the work-item, which runs the implemented kernel code and is groupped into work-groups. Each work-group has a dedicated processor and runs

separately from the others. The group of work-items also runs inside the same computing unit. Therefore the proper rendering process should be chosen so that we could exploit the hardware features. For this, logically the Tile-Based Rendering [6] is the closest rasterization procedure. The following figure illustrates the TBR rendering solution from the perspective of the GPGPU.

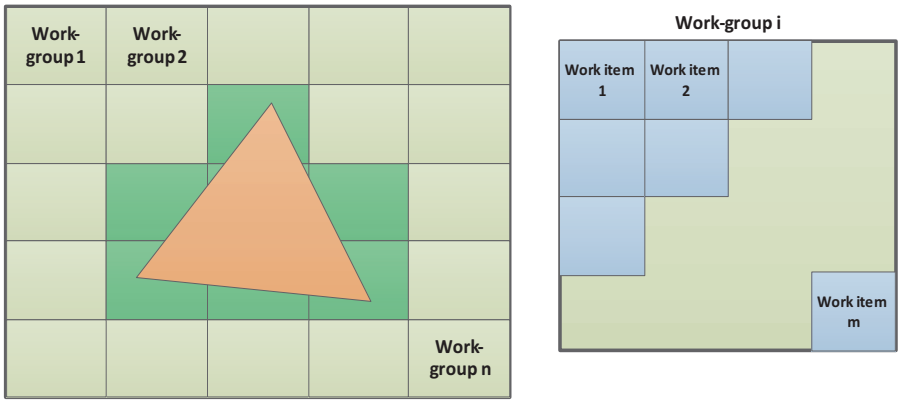


Figure 5. GPGPU model of Tile-Based Rendering

Based on the central idea of TBR, the framebuffer should be divided into equal-size areas, called *bins*. In order to exploit the parallel execution of the GPU, all areas should be assigned to a specific work-group, where work-items perform the computations. The rasterization logic is the following: all triangles of the pipeline are assigned to a tile (binning) based on their 2D mapping, whether the tiles overlap or not. All tiles are processed independently and parallelly on a separate processor, where the pixel level rasterization is performed by work-items.
In frame buffer distribution, typically 16x16 or 32x32 size parts should be chosen. The resolution is then not too high to take advantage of the card parallelism (e.g. Card cache size, local memory size, maximum numbers of work items, etc.), and not too small to result in many unnecessary calculations.
In a group, work-items are responsible for rasterizing the image of a tile. They take the list of triangles belonging to the group, calculate their boundaries and perform the pixel level rasterization. Within a group, work-items run also in parallel and share the same local memory. Each item is associated with a triangle, it is responsible for its rasterization. Because triangles can overlap according to their Z value, synchronization is required between the items, which is supported by the languages (OpenCL, CUDA). The whole image (framebuffer) is ready to display when each group has completed its own task.

5.2 Constraints of GPGPU programming

The GPU and the main memory are away from each other, so moving data between them is strongly limited by the PCIe bus transfer rate (~2.4 GB/s). It is therefore

appropriate to store all triangle data of the pipeline in a card's memory and minimize data movement. As a short test, an empty, 1024x768 size, 32 bit framebuffer was shared with the GPU and performance was measured. No other calculations were performed, only data sharing and framebuffer displaying. The hardware used for the test was an ATI Radeon HD 5670 1 GB RAM. In the first test case an empty framebuffer was displayed on the screen and no GPU share was applied. The average rendering speed was 1100 FPS. In the second test, the software framebuffer was shared with the GPU in each rendering frame using OpenCL and the average performance dropped to 620 FPS without any GPU calculations.

Another problem is that running a kernel (even an empty one) requires a specific preparation time in execution. In the third test it was investigated how this kernel initialization affects the rendering performance. During the test process an empty kernel was created and four float type variables were shared with the kernel. Rendering speed dropped to 580 FPS this time.

Naturally, there are opportunities to improve the loss of speed arising from the communication. For example, if the entire framebuffer is stored as an OpenGL texture in the card's memory and is shared for GPGPU computations. However, applying this method, we lose a part of the characteristics of software pipeline.

We can say that the modern GPU is very efficient in parallel processing, but is not a wonder tool. Tests show that the technology can be applied in real-time applications, but it provides sufficient efficiency only in case of well-prepared and GPU uploaded data.

Conclusion

It can be said that the future is bright for a software rendering revolution. The techniques presented in this article highlight the fact that developing a really fast software renderer requires a lot of effort. It is essential to combine several technologies and to use lower-level languages for programming. The CPU has evolved over the past few years: utilizing its potential properly, the performance of the software rendering pipeline can be improved to a large extent. This paper has presented how the GPGPU technology can be applied as a new approach in the rasterization stage. Its parallel potentials are adaptable to the TBR rendering method but only within certain limits.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] ZACH, B.: *A Modern Approach to Software Rasterization*. University Workshop, Taylor University, 14. dec 2011.

[2] TRANSGAMING INC: *Swiftshader Software GPU Toolkit,* 2012.

[3] MICROSOFT CORPORATION: *Windows advanced rasterization platform (warp) guide*. 2012.

[4] ABRASH, M.: *Rasterization on larrabee*. Dr. Dobbs Portal, 2009.

[5] EBERLY H, D.: *3D Game Engine Design*: *A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann/Academic Press, 2001.

[6] ANTOCHI, I.: *Suitability of Tile-Based Rendering for Low-Power 3D Graphics Accelerators.* Dissertation, Delft University of Technology, 2007.

[7] SEILER, L., CARMEAN, D., SPRANGLE, E., FORSYTH, T., ABRASH, M., DUBEY, P., JUNKINS, S., LAKE, A., SUGERMAN, J., CAVIN, R., ESPASA, R., GROCHOWSKI, E., JUAN, T., HANRAHAN, P.: *Larrabee: a many-core x86 architecture for visual computing.* ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008 Volume 27 Issue 3, August 2008.

[8] LAINE, S., KARRAS T.: *High-Performance Software Rasterization on GPUs.* High Performance Graphics, Vancouver, Canada, aug 5. 2011.

[9] OWENS, J., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRUGER, J., LEFOHN, A., PURCELL, T.: A *Survey of General Purpose Computation on Graphics Hardware*. Computer Graphics Forum. v.26, n. 1, pp. 80-113, 2007.

[10] SUGERMAN, J., FATAHALIAN, K., BOULOS, S., AKELEY, K., AND HANRAHAN, P.: *Gramps: A programming model for graphics pipelines*. ACM Trans. Graph. 28, 4:1–4:11, 2009.

[11] FANG, L., MENGCHENG H., XUEHUI L., ENHUA W.: FREEPIPE: *A Programmable, Parallel Rendering Architecture for Efficient Multi-Fragment Effects*. In Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2010.

[12] AGNER, F.: *Optimizing software in C++ An optimization guide for Windows, Linux and Mac platforms*. Study at Copenhagen University College of Engineering, 2011.06.08.

[13] SWENNEY, T.: *The End of the GPU Roadmap*. Proceedings of the Conference on High Performance Graphics, pp. 45-52, 2009.

[14] COFFIN, C.: *SPU-based Deferred Shading for Battlefield 3 on Playstation 3*. Game Developer Conference Presentation, March 8, 2011.

[15] RAD GAME TOOLS: *Pixomatic advanced software rasterizer,* 2012.

Production Systems and Information Engineering
Volume 6 (2013), pp. 69-80

A FAST METHOD FOR SECURING USER SUPPLIED CODE EXECUTION IN WEB SERVERS

DÁVID VINCZE
University of Miskolc, Hungary
Department of Information Technology
david.vincze@iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. This paper presents a novel method for securing web servers while keeping performance overhead as low as possible. There are already existing methods for separating the execution user and group identities for different websites on the same web server, hence improving security, but all of them have performance and resource usage weaknesses. The mechanism presented here requires modifications also in the kernel of the operating system, not only in the web server. The method works by extracting the calling address of the process invoking a specific newly implemented system call in the Linux kernel. Based on this address, the new system call can decide whether to grant additional privileges to the invoking process or not. Integrating this method into the Apache web server (the most popular web server application for many years as of writing) makes it possible to create a secure environment for the different websites belonging to different users on the same server. Compared to similar solutions, the overhead of the method is very low. The last chapter presents measurement results comparing the performance of the original version and that of the modified version of the web server.

Keywords: operating systems security, Linux kernel, web server security, Apache web server

1. Introduction

Web based applications are getting more and more widespread nowadays. In most cases on multi-user systems for performance and resource benefits, separate web applications are served by the same web server running with the same user identities and permissions. Hence the served applications are equal when it comes to accessing resources, independently from which user they belong to, because they are served under the identity of the web server. In some cases this can be uncomfortable for the users, moreover malicious users could exploit this easily, e.g. they can access each other's source code, database access passwords, in some

cases they could disturb the web server causing it to malfunction, hide backdoors, etc. [5] [14]

In production environments these problems emerge quite soon if the given web server has many users. Most commonly: uploaded files must be separately adjusted by the system administrator or the user to the correct privileges; working with the uploaded files can be difficult because the web server created these files with its own user identity, deleting these files via File Transfer Protocol (FTP) [11] often causes problems (because FTP separates identities); these files are accounted to the resources of the web server and not to the user who had the file uploaded; limitation and logging of resource usage cannot be distinguished among users; setting arbitrary headers in e-mails when sending mail using the *mail()* function in PHP [1] scripts; session manipulation [10] and other seemingly small, but serious problems [13] [5] [14].

As nowadays most of the web servers in production apply this structure and configuration, these problems affect a considerable number of users: internet service providers, educational institutions, various enterprises, etc.

This paper presents a possible solution to eliminate these problems, providing a fast and secure environment for web applications using runtime user and group identity switching.

Currently the Apache HTTPD is one of the most commonly used web servers [20], and most of its installations are deployed on Linux based systems. The Apache web server and the Linux kernel are both open source software, therefore it is possible to modify their source code freely. The presented solution has been developed for the Linux operating system in C and Assembly (x86 32-bit and 64-bit) programming languages.

First, the paper gives an overview of the operation of web servers at a glance, especially the operation of the Apache HTTPD, then of the possibilities of run-time user and group identity switching in the Linux kernel. A novel method is introduced to determine whether a privilege elevation request is legitimate or not. The modifications required for this new mechanism to work are also presented both in the Linux kernel and the Apache web server. Finally, results of performance benchmarks are presented.

2. Overview of the operation of web servers

In the beginning of the web-era, the first web servers were only able to serve regular files. This practically means that only static content was available, e.g. text files, documents, images, etc. These simple web servers only read the contents of the requested files and sent it to the client unmodified. This is called a static web server, which is still in use nowadays, because a great deal of static content can still be found (e.g. images, videos, stylesheets, JavaScript scripts, etc.) on regular web pages.

Nowadays it is common that web pages show dynamic content, which means the pages are generated on-the-fly, taking various input data into consideration (usually exchanging data with databases). In this case the web server, instead of sending the exact content, runs the specified program/script corresponding to the request, and sends back its output (the generated web page) to the requesting client. This is called dynamic serving mode.

The first widespread method for generating dynamic content for web pages was the Common Gateway Interface (CGI) mechanism [12]. The concept of CGI is simple: the requested file (CGI application) is executed on the web server and the output of the CGI application is sent as an answer to the request. Scripts can be also used as CGI application, in this case the script interpreter must be defined. The CGI application is executed in a newly created independent separate process, and after the application finishes, this process is also terminated. CGI is still in use, but not very common nowadays.

After the success of CGI, it was realized that CGI was mainly used for running script interpreters. Interpreters embedded in the web server were developed for various script languages (e.g. Perl, PHP, Python, Ruby, etc.). These were much faster because there was no need for forking new processes and initialization every time a request arrived. Also these embedded interpreters provide an environment which is easier for programmers to use. Nowadays these solutions are in use, superseding CGI.

According to the latest NetCraft survey [20], the most commonly used web server on public web sites is the Apache HTTPD. Most of these are installed on Linux-based servers.

Traditionally the Apache HTTPD uses the *prefork* model [21]. The prefork model has a control process which does not serve incoming Hypertext Transfer Protocol (HTTP) [4] requests, but only starts and monitors child processes, which do the actual work. A child process can serve many independent HTTP requests (configurable, some hundred on an average basis), hence it is not required to start a new process and perform initialization for every request, which yields a considerable performance gain.

The *worker* model [21] is similar to the prefork model, the difference being that it uses threads instead of processes. That is the main reason why it is not used in multi-user environments where users are not trusted.

Using either the prefork or the worker model, the user and group identities stay the same through every served request. This means that all programs (e.g. CGI) are running with the same permissions, which is considered a security weakness for various reasons, see e.g. [13].

For securing CGI executions there is a method called *suexec CGI* [21], which

extends the CGI mechanism with the capability of running each CGI program as a different user/group identity. The main idea is that CGI programs are executed through a wrapper program. This wrapper program has a special permission setting (suid – setuid, see [2] and [3] for details) allowing it to run with system administrator privileges. The wrapper checks which user/group identities must be set for the CGI program, and after changing the identities it loses all of the administrator privileges (hence the CGI program cannot revert to be administrator again), and the original CGI application will be executed. Injecting this wrapper program in the chain of execution on one hand increases security, but the serving requests becomes slower compared to normal CGI execution.

Also there was a model called *perchild* [21], which ran separate dedicated threads for every website configured. These separated threads or thread groups had their own configured user and group identities set, hence the requests were served with the correct permissions. A drawback of this concept was that a certain number (the number of the sites configured) of thread groups were always running even if they never served a request. In case of many configured websites this consumes resources unnecessarily. The main drawback of the perchild model is that the model was never finished, furthermore it was dropped from the newest versions of the Apache web server [21].

Another implementation of this concept was *metuxmpm* [16], but as with perchild, the development was cancelled. Then came the peruser model [9], which tends to be a working implementation, but it uses separate processes instead of threads. This means that in case of many websites the amount of unnecessarily allocated resources can be huge.

A promising model based on prefork is the *itk* model [6]. It can run the programs required to generate pages with their own identities, without using dedicated thread/process groups. All the children processes run with administrator privileges till the necessary information (which identities to set) can be extracted from an incoming HTTP request. After changing the user and group identities and serving the requests belonging to the same website (user id), the identities cannot be set back to administrator level, hence the process has to be terminated. This means that processes cannot be reused, new processes should be created if a new incoming request belongs to another website (user id). Thus compared to the prefork model, this module also has performance and resource usage drawbacks.

A similar, experimental solution called Harache has been proposed in [7]. Harache works very much like the *itk* model: sets the identifiers before processing the incoming HTTP request, and after processing the request, terminates the process.

Another solution from the same authors called Hi-Sap [8] uses a complex model consisting of a dispatcher/content scheduler and Apache worker pools. While the performance is better compared to Harache, the overall resource usage is

considerably higher.

Being aware of these facts, it is clear that system administrators have to make a choice between security and performance. As impacts of performance can usually be experienced directly (faster web pages, more customers can be handled by fewer servers, etc.), performance is chosen over security.

The next chapter introduces a novel method which retains both performance and security in the previously described situations.

3. Enhancing security while retaining performance

The main difficulty is to unambiguously determine whether the currently executed code in the process is a normal web server code or a user-written code (e.g. embedded script interpreter running a user's script), as a normal web server process does not have any properties which could be used to distinguish between these two states. For example the operating system kernel does not know whether the code of the Apache web server itself is executed, or a user code is being interpreted by e.g. the embedded PHP interpreter: *mod_php*.

Therefore, first a method should be developed which can distinguish between these two conditions inside a process. This paper presents a novel approach which is based on the calling memory address of a privilege elevating system call. In the possession of this calling address it becomes possible to determine whether the privilege elevation request is legitimate or not. If the request is allowed, then various privileges can be granted for the requesting process.

In this section a new system call implemented in the Linux kernel will be presented, which gives certain privileges to the calling process based on the return address to the user-space (also referred to as calling address earlier). Then the modifications required in the Apache web server to make use of this newly added system call will be discussed.

First, some concepts required for understanding the new method will be overviewed shortly.

3.1. The setuid() / setgid() system call family

On multi-user POSIX (Portable Operating System Interface) [17] compatible systems, users and groups posses their own identifiers (one user id - uid, and at least one group id - gid) for the purpose of separating the users. Authentication and access control are based on these identities. The following system calls are standardized in POSIX.1 [17]: setuid() / seteuid() / setreuid() / setresuid(). These syscalls allow an arbitrary process in the system to change their user identities. The same syscalls exist regarding changing group identities: setgid() / setegid() / setregid() / setresgid().

The case of normal usage is when a process is running with administrator privileges and after the application completes the procedures requiring administrator privileges, the process switches its user and group identities to unprivileged identities. These are carried out with calling the setuid()/setgid() system calls, hence it drops the administrator privileges and continues to run as a normal user process. See [2] [3] for more details.

The Apache web server works the same way as described. The system administrator privileges are needed only for binding to TCP ports below 1024 – by default port 80 or port 443 (HTTP / HTTPS ports) - and also for opening the log files. Once these tasks are completed, the web server switches to a normal user for the rest of its lifetime.

3.2. The Linux capabilities system

Access control of traditional UNIX systems (including Linux) distinguishes two privilege levels: one level for the system administrator (user identity equals 0) and another one for normal user level (user identity other than 0). Controlling access is very simple by default: at the system administrator level no checking is performed for accessing resources, but at the user level the current user and group identities are compared with the identities required by the desired resource.

The capabilities mechanism is described in chapter 25 of the POSIX.1e [18] standard. This allows finer granularity for permission distribution and access control. The POSIX.1e standard has never been finalized, only a draft has been published, and even that has been withdrawn later. Despite this fact, the capabilities mechanism has been implemented in some operating systems, e.g. Linux. The capabilities mechanism describes elemental privileges which can be assigned to processes, independent from their current user identities. Some of the implemented capabilities in the Linux operating system [15] are:

- CAP_NET_BIND_SERVICE: Allows binding to TCP/UDP ports under 1024.
- CAP_SETUID: Allows setting the user identity to an arbitrary value on the current process (allows the successful calling of the setuid() syscall family).
- CAP_SETGID: Allows setting the groups identity and supplemental groups to arbitrary groups on the current process (allows the successful calling of the setgid() syscall family).

Naturally, there are many more capabilities defined. By default, the system administrator possesses all the capabilities and normal users do not have any of the system capabilities. The proposal in this paper makes use of the CAP_SETUID and the CAP_SETGID capabilities.

3.3. The new system call

As described earlier, a new method has been implemented, which elevates process privileges based on the memory address in the memory area of the process where the system call was invoked from. A new system call was implemented in the latest Linux kernel (version 3.1.10 as of writing). Unfortunately the code is architecture specific, currently the implementation is for the x86 and the x86_64 architectures only. Details of the implementation are presented in this subsection.

Before a system call executes, the memory address of the next instruction to be executed (the return address) is stored on the stack before entering into the kernel. After the system call completes its task in the kernel, the control will be returned to the user-space application, which will continue its execution exactly at the memory address previously stored on the stack. The control returns to the saved memory address in the process of the web server with a *ret* instruction. The return address is popped from the stack, which was pushed to the stack earlier just before the actual system call. This return address has to be traced back on the stack to be used for authentication.

Tracing back the return address is done in the new syscall named *getmyretaddr*(). It does not have any input parameters as its only task is to check whether to allow the caller process elevated privileges or not. The newly implemented system call first traces back the return address on the stack to determine where exactly the system call came from the memory area of the calling process. If this is the first invocation of the new system call within a given process, the traced return address will be stored in the process context. Otherwise the traced memory address will be compared with the previously stored memory address by the first invocation. In case of a match, the privilege elevation will be successful, otherwise the privilege elevation request will be ignored.

In possession of the calling address, the access verification is realized as described in the following. In the case of the first invocation of the *getmyretaddr()* system call, a new attribute in the process context (called *retaddr* – default value is zero) will be initialized with the gathered return address. The initialization will only be successful if the process possesses the CAP_SETUID and the CAP_SETGID capabilities. Any next invocation of the *getmyretaddr()* system call checks the return address against the stored *retaddr*; if these two match, then the calling process will be granted both with CAP_SETUID and CAP_SETGID. Then the system call finishes and the control is returned into user space. Now it is the task of the privilege elevation requesting process to change the user and group identifiers, then drop the two capabilities.

The next chapter describes how this new system call has been integrated into the Apache web server.

4.2. Modifications in the Apache web server

The prefork module [21] was modified to make use of the presented mechanism. The required modifications will be described briefly in the following.

Right after staring the web server, initially forked child processes wait for incoming requests with the default user and group identities. The web server was modified to leave both CAP_SETUID and CAP_SETGID capabilities set on the inital processes. When a request arrives, the child process does its usual initialization and just before serving the request, the *getmyretaddr()* syscall will be invoked. If this is the first request served by the child process, the *getmyretaddr()* does only initialization (stores the valid return address in the process context: *retaddr*). The pre-forked child process should be in possession of the CAP_SETUID and the CAP_SETGID capabilities, otherwise the system call will be unsuccessful. Next, after successfully changing the user and group identities with *setresuid()* and *setresgid()* calls, the two capabilities are not required anymore, so they will be dropped. Finally the actual request will be served but now with the correctly set unique identities. When serving the forthcoming requests, the child process is no longer in possession of the two capabilities when calling *getmyretaddr()*, but if it is called from the permitted memory address (the first call's return address), then the syscall will be successful, and the kernel will grant the two capabilities to the calling process. Hence the *setresuid()* and *setresgid()* syscalls will be also successful when called. In case when the request is for another user's resource, then the capabilities can be dropped again. This way the process executes user supplied code (e.g. embedded interpreter runs a script) with the user and group identities configured for the actual website.

Serving the incoming requests is the task of the child processes of the web server. Every request triggers the *ap_process_request_internal()* function in the Apache web server, hence it is an adequate function where the *getmyretaddr()* syscall and its supplemental code can be injected. Just before invoking *getmyretaddr(),* it should be determined which identities should be used for serving the request. A new configuration directive called *ServeAsUIDGID* was defined, which can be used in the configuration of the web server to specify which identities should be set on a website. If the newly added directive was supplied in the configuration, then it is straightforward which identities should be used (e.g '*ServeAsUIDGID 1303 1011'* – the user id will be set to 1303, and the group id will be set to 1011). Otherwise (very useful when applying solutions to automatize the publishing of users' web pages, e.g. *mod_userdir* [21]) the same identities as the owner of the requested file will be set. If the ownership of the file cannot be determined, a HTTP error 404 (not found) is returned. And in case the owner of the file to be served is the system administrator (root – uid 0), the request is denied with

returning a HTTP error 403 (access forbidden).

It is possible that an incoming request requests a resource belonging to the same identity as the one which was already set for the previous request. For achieving better performance, it is first decided whether it is required to change identities or not, by comparing the identities needed to be set and the currently set identities. If these match, then the *getmyretaddr()* system call will not be invoked.

In the unfortunate case when an error occurs while dropping the now unwanted capabilities, the web server fails with an internal error returning a HTTP error 500 (internal server error) to the client, and the serving of the request will be terminated.

With these modifications the Apache web server became capable of using the presented mechanism. After compiling the modified source code, the method was tested and verified with the usage of the GNU Debugger (not covered in this paper).

The modifications in the source code of both the Linux kernel and the Apache web server in the form of a patch can be found at [22].

4. Performance analysis

According to the modifications, the Apache web server and the Linux kernel contain additional codes to be executed in every iteration when a request is served, which yields additional performance overhead. This overhead was measured empirically with the help of the *siege* HTTP testing utility [19]. The measurement tests were conducted both using the original and modified web server in conjunction with the modified kernel, and also using the previously mentioned ITK module in the Apache web server.

The siege utility simply sends requests to a given web server, and measures the elapsed time till the requests finish. The web server was configured with 16 separate websites with different user and group identities to be set for each site. Siege was configured to send requests to these 16 sites in a round-robin fashion. This results in an identity switch with every request, which means the worst possible case was simulated and benchmarked.

The same measurement was conducted several times with various concurrency level settings (1, 4 and 16 respectively) to minimize measurement incorrectness. An average of these results were evaluated. The results corresponding to the unmodified web server were taken as reference (100%). Accordingly, the results are shown in Figure 1.

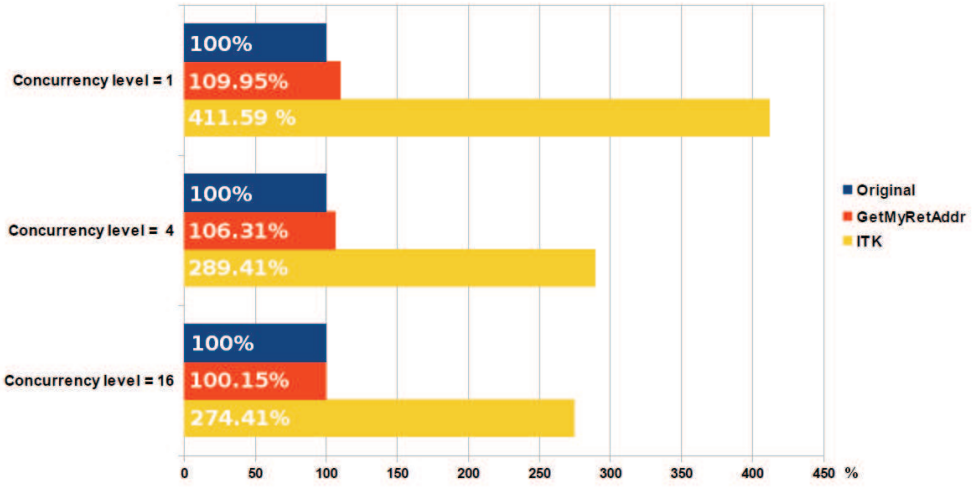


Figure 1. Performance overhead measurement results

It can be noticed that the overhead is very slight (+0.15%) when using concurrency level 16. On production servers many clients access many sites, hence concurrency level could be high. Also, keeping in mind that the tests were conducted simulating the worst possible scenario, and on real world web servers many request batches are for the same website, the average overhead should be lower. Therefore the performance overhead can be so low that the overhead can go unnoticed in real environments.

Conclusion

The mechanism introduced in this paper defines a novel approach for access control. The main idea of the mechanism is to check the calling memory address in the invoking process, and if this value is the same as it was during initialization, then various privilege elevations can be performed. The mechanism was implemented in the latest Linux kernel (version 3.1.10 as of writing) as a new system call. Integrating this new system call into the most commonly used web server nowadays - the Apache HTTPD web server - makes it possible to serve every HTTP request with different user and group identities, with very slight performance overhead. Hence the mechanism introduced in this paper is significantly faster compared to other existing solutions with the same security approach.

The separated identities allow some additional advantages for users and administrators of web servers. E.g. limiting and accounting resources on a per user

basis: OS level authentication for database access, controlled network access, file system access, quota management for uploaded files, etc.

Further development possibilities include porting to other operating systems (e.g. BSD), and also the new security risks possibly opened by the new system call (e.g. rewriting code on the trusted memory address) should be investigated closely.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] ACHOUR, M., BETZ, F., DOVGAL, A., ET. AL.: *PHP Manual*, PHP Documentation Group, 2012

[2] BUNCH, S.: *The Setuid Feature in Unix and Security,* Proceedings of the 10th National Computer Security Conference, pp. 245 – 253, 21-24 Sept. 1987

[3] CHEN, H., WAGNER, D., DEAN, D.: *Setuid Demystified,* Proceedings of the 11th USENIX Security Symposium, pp. 171 – 190, 2002

[4] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., BERNERS-LEE, T.: *Hypertext Transfer Protocol – HTTP/1.1, Request for Comments #2616*, Internet Engineering Task Force, June 1999

[5] GARFINKEL, S.: *Web Security, Privacy & Commerce*, Second Edition, O'Reilly Media Inc., ISBN: 978-0596000455, Jan. 2002

[6] GUNDERSON, S. H.: *The Apache 2 ITK MPM*

HTTP://MPM-ITK.SESSE.NET (ACCESSED JAN. 2012)

[7] HARA, D., OZAKI, R., HYOUDOU, K., NAKAYAMA, Y.: *Design and Implementation of a Web Server For a Hosting Service*, Proceedings of the Ninth IASTED International Conference, Internet and Multimedia Systems and Applications, August 15-17, 2005, Honolulu, Hawaii, USA

[8] HARA, D., NAKAYAMA, Y.: *Secure and High-performance Web Server System for Shared Hosting Service*, Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06), Minneapolis, USA, 2006

[9] HEACOCK, S. G.: *Peruser MPM for Apache 2.x Technical Details*

HTTP://WWW.PERUSER.ORG/ (ACCESSED JAN. 2012)

[10] NIKIFORAKIS, N., JOOSEN, W., JOHNS, M.: *Abusing Locality in Shared Web Hosting,* Proceedings EUROSEC '11 Proceedings of the Fourth European Workshop on System Security, Salzburg, Austria, 2011

[11] POSTEL, J., REYNOLDS, J.: *File Transfer Protocol, Request for Comments #959*, Internet Engineering Task Force, October 1985

[12] ROBINSON, D., COAR, K.: *The Common Gateway Interface (CGI) Version 1.1, Request for Comments #3875.* Internet Engineering Task Force, Oct. 2004

[13] SHIFLETT, C.: *Essential PHP Security*, O'Reilly Media Inc., 2005, ISBN: 0-596-00656-X

[14] SNYDER, C., SOUTHWEL, M.: *Pro PHP Security*, Apress Media LLC, ISBN: 978-1590595084, Sept. 2005

[15] TOBOTRAS, B.: *Linux Capabilities FAQ*, 1999
HTTP://FTP.KERNEL.ORG/PUB/LINUX/LIBS/SECURITY/LINUX-PRIVS/KERNEL-2.4/CAPFAQ-0.2.TXT

[16] WEIGEL, E.: *Muxmpm/metuxmpm for Apache 2*

HTTP://WWW.SANNES.ORG/METUXMPM/ (ACCESSED JAN. 2012)

[17] IEEE STANDARD FOR INFORMATION TECHNOLOGY 1003.1-2004: Portable Operating System Interface (POSIX)

[18] IEEE STANDARD FOR INFORMATION TECHNOLOGY P1003.1E Draft: Portable Operating System Interface (POSIX)

[19] JOE DOG SOFTWARE: Siege HTTP Testing Utility

HTTP://WWW.JOEDOG.ORG/INDEX/SIEGE-HOME (ACCESSED JAN. 2012)

[20] NETCRAFT LTD. Web Server Survey January 2012:

HTTP://NEWS.NETCRAFT.COM/ARCHIVES/2012/01/03/JANUARY-2012-WEB-SERVER-SURVEY.HTML

[21] APACHE HTTP SERVER DOCUMENTATION

HTTP://HTTPD.APACHE.ORG/DOCS/ (ACCESSED JAN. 2012)

[22] THE SOURCE CODE MODIFICATIONS CAN BE ACCESSED AT:

HTTP://USERS.IIT.UNI-MISKOLC.HU/~VINCZED/GETMYRETADDR/

Production Systems and Information Engineering
Volume 6 (2013), pp. 81-90

LOCALIZATION TECHNIQUES IN WIRELESS SENSOR NETWORKS

ATTILA K. VARGA
University of Miskolc, Hungary
Department of Automation and Communication Technology
varga.attila@uni-miskolc.hu

[Received February 2012 and accepted September 2012]

Abstract. The drawback of wired networks is that if we want to communicate on it wired communication has to be established. Actually, wired communication limits our mobility. In wireless networks there is no need for wires, we can connect our devices to the network. Since wireless network applications have been deployed widely, wireless sensor networks have become an important research area.

The development of wireless technology enabled us to use cheap and small sized sensors in short range communications. A sensor network consists of several nodes that are low in cost and have a battery with low capacity. Localization in wireless sensor networks is a vital issue, i.e. determining the position of a given device in the network. Location information of mobile nodes is a demand in many wireless systems.

Localization involves determining the location of the sensor node based on other sensor nodes with known locations. The node can calculate its distance and/or angle between itself and the reference points. In the 2D space, if a node knows its distance from three reference points, its position can also be determined. One more reference point is needed in the 3D space to determine the current position of the target device. The paper deals with various techniques of localization used in wireless sensor networks.

Keywords: wireless sensor network, localization, anchors, mobile nodes

1. Introduction

Distributed sensor networks have already been applied for years, but wireless sensor networks [1] have recently been focused on. The rapid development of wireless sensor networks opened the door to create low-cost, low-power and

multifunctional sensor devices that are integrated with sensing, processing, and communication capabilities.

Estimating the location of a sensor is a critical task in sensor networks. There exist several location estimation techniques used in sensor networks. In the sensor network there are two types of nodes:

- nodes that know their location (they are fixed nodes and are often called anchor nodes),
- and some other nodes (called mobile sensor nodes) having the ability to estimate their location using information about their position (i.e. coordinates of a node, properties of a signal such as signal strength [2], time difference of arrival, etc.) received from the fix nodes.

After performing such measurements for different nodes, the sensor node has to combine all this information for estimating its location. The location estimation algorithm has two main requirements:

- the sensor nodes should avoid complex and time consuming computations, which would deplete their energy supply rapidly,
- the computations should take into consideration the error in the measurements, which can be significant.

Several approaches use computationally demanding methods, such as convex optimization [3], systems of complex equations [4], minimum mean square error (MMSE) methods, and Kalman filters. In these approaches, the measurement model is not adequately analyzed and the error is assumed to be small, which is not the case in most real applications of sensor networks.

Other algorithms estimate the location of a node using the Received Signal Strength Indicator method, which is the most realistic model for sensor network communication [5].

Many localization techniques used in sensor networks can be applied in a variety of wireless networks with which a wireless node can estimate its distance or its relative location to a reference point. In the past few years, several algorithms for solving the localization problem have been proposed.

2. Basis of localization

A sensor node (often called mote) is practically a device in the wireless network that is capable of data processing, information gathering and establishing communication with the other nodes in the network.

Wireless localization techniques [6] are used to give the positions of the mobile nodes considering the known location information. In 2D and 3D space reference points are needed to determine the position of a mobile device.

Using reference points, the angle and distance of a device can be calculated from the reference points. Observing the well-known scenarios it is a widely used technique that a mobile device computes its own location according to the position information of fix devices. These fix devices are also called anchors (see Figure 1).

Many applications of sensor networks require knowledge of physical sensor positions. Location information can be used not only to minimize the communication but also to improve the performance of wireless networks and provide new types of services.

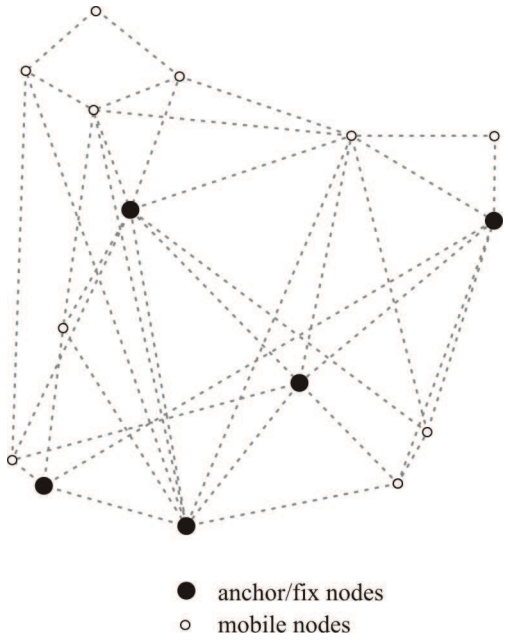


Figure 1. Nodes in a self-organizing wireless sensor network

The positioning algorithm must be distributed and localized in order to scale well for large sensor networks. Often, price, size and the precision of the localization are the main factors when choosing the position determining technique. The wireless solution generates a number of problems in connection with localization:

- a number of measurements have to be done for determining the position of a mobile node,
- choice of the localization technique depends on the given environmental conditions,
- wireless sensors are cheap devices but have limited computation capabilities,
- localization techniques need implementation with minimal hardware investment considering the given measurement possibilities,
- in many cases sensor networks should be designed for using them in multi-hop networks.

Localization algorithms used in large-scale ad-hoc sensor networks should meet some requirements. These algorithms should be

- self-organizing,
- tolerant to node failures,
- energy and computation efficient (little consumption and low-computational steps).

The most important and user-oriented factors are the accuracy and the precision of the given positioning algorithm.

2. Types of localization

Localization can be classified in many aspects (Fig. 2). Localization techniques can be divided into two categories based on the communication between nodes:

- centralized localization techniques,
- and decentralized localization techniques.

Centralized localization techniques involve data transfer to a central node in order to compute the location for each node. Communication with centralized computing is expensive, and sending data serially by time within the network introduces latency, and it consumes energy and network bandwidth as well.

Decentralized or distributed localization techniques depend on each sensor node being able to determine its location with only limited communication with nearby nodes. Distributed localization techniques do not require centralized computation.

Distributed localization techniques involve two kinds of techniques such as [7]:

- range-based,
- and range-free localization techniques.

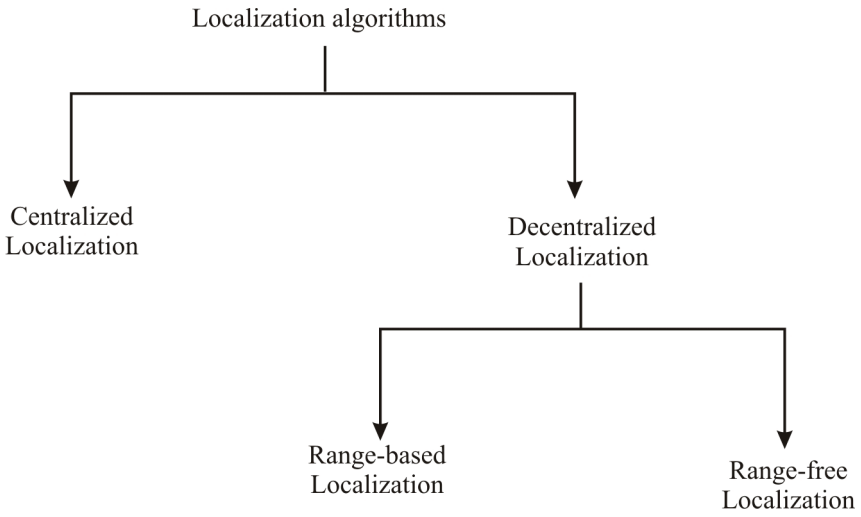


Figure 2. Types of localization

Range-free methods calculate the distance between two nodes in a number of hops and do not take into consideration any coordinate system. In range-free algorithms [7] nodes can estimate their position according to the known localization information of the neighbouring nodes. In this case, it is assumed that not all the nodes have distance, angle or other metric information. Position of the target node can be calculated as the 'centroid' of the surrounding fix nodes or it can be derived from geometric relations.

Range-based methods estimate distance and direction of two nodes from a measurement (i.e. angle of arrival measurements, distance related measurements and received signal strength measurement). Range-based algorithms make distance estimations between the fix and mobile nodes. The positions of a group of nodes in the wireless network are known by means of simple distance measurement. The target nodes try to estimate their position relative to the fix nodes.

3. Localization methods

There are several advanced localization algorithms based on machine learning and data fusion techniques.

3.1. Weighted centroid method

The idea of the centroid method (Fig. 3) is that the position of the target device is calculated by the known positions of the anchor nodes in the transmission range. Although this algorithm is very simple, efficient, easy to implement and needs low computational operations, it produces a lower level of precision. It is widely used in such dense sensor networks – there are fix nodes having known positions – that contains overlapping ranges. The known positions can be weighted, so the unknown position can be calculated as follows:

$$\tilde{x} = \frac{\sum_{i=1}^{N} \omega_i x_i}{\sum_{i=1}^{N} \omega_i} \quad , \tag{3.1}$$

where N is the number of anchors in the transmission zone, x_i is the position and ω_i is the weight of the *i-th* fix node. An adequately selected weighting method can result in more precise information about the position of the given node.

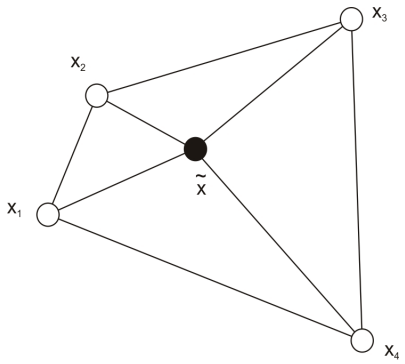


Figure 3. Centroid method

In practice, mobile nodes broadcast messages that are received by the fix nodes in the transmission range. This message sending and receiving mechanism helps the nodes to establish connection between themselves.

According to the connection metric, the weights can be calculated as follows:

$$\omega_i(t) = \frac{n_{received}^{(i)}(t)}{n_{sent}^{(i)}(t)} \quad , \tag{3.2}$$

where t is the duration of transmission of the broadcast messages, $n_{received}(t)$ and $n_{sent}(t)$ are the number of received and sent messages during t. Only those nodes will be considered that reached a given ratio – usually over 90 %.

3.2. Bounding box method

The bounding box method (Fig. 4) is a simple and low computational localization technique. The accuracy of this technique is limited, but it is simple, fast to implement and to run on sensor nodes.

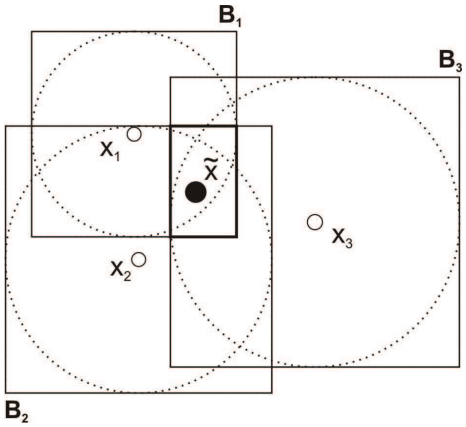


Figure 4. Bounding box method

The main concept is that boxes are created around the transmission range of the nodes, and the target device is located in the intersection of these boxes. The distance between the target device and the *i-th* fix node can be estimated by the side length of the *i-th* bounding box. The result of the position estimation is the intersecting box or its centre:

$$\tilde{x} \in \left\{ \bigcap_i B_i \right\}, \tag{3.3}$$

where B_i is the box created around the x_i-th node, and the side length of the box is twice the transmission range.

3.3. Point in Triangle method

The Point in Triangle method (Fig. 5) is a range-free localization scheme. In this approach the target device sends a beacon message that is received by the fix nodes. After that, the anchor nodes create communication triangles, i.e. they form all the possible subsets of three nodes.

$$\begin{aligned} \tilde{x} &\in T_1 = \{x_1, x_2, x_3\} \\ \tilde{x} &\in T_2 = \{x_1, x_2, x_4\} \\ \tilde{x} &\in T_3 = \{x_1, x_3, x_4\} \\ \tilde{x} &\in T_4 = \{x_2, x_3, x_4\} \end{aligned}, \tag{3.4}$$

where T_i is the *i-th* subset that forms the *i-th* triangle. In each triangle the so-called *PiT* test decides whether the target device is in the current area or not.

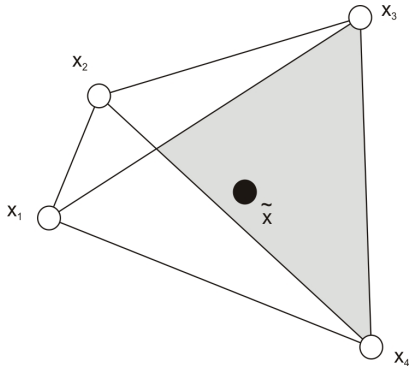


Figure 5. Point in Triangle based localization

Finally, the intersection of these areas will produce the position of the target:

$$T_{target} = \bigcap_{i=1}^{4} T_i \,. \tag{3.5}$$

The main disadvantage of this method is the great number of computational steps, as the PiT test must be done for all the triangles to examine whether the current area includes the target device or not.

Actually the PiT test is based on geometry: for a given triangle the mobile node is outside the triangle if the gradient of the distance estimated to each vertex of the triangle is positive, i.e. the mobile node moves farther away from the points.

Conclusions

In wireless communication the localization of mobile devices is a major problem. Location-based applications are very close to our daily life; it is a process to compute the locations of wireless devices and relies on the geometric relationship of network nodes.

As outlined in the paper, the positioning method should not increase the cost and complexity of a sensor because an application may require a great number of sensors. Communication and collaboration between nodes should be minimized for achieving energy saving. Most wireless sensor networks have a limited computation ability, so the main goal is to implement and run simple distance estimation functions. The estimation must be very close to the real position. If the position estimated is close to the average value, then estimation is said to have great precision. When the position estimated is almost the same as the real position, the degree of accuracy is very high. Absolute precision can never be achieved, although a well-chosen estimator can improve the accuracy.

Precision and costs may contradict each other, but these are important factors in location-based services. Extra hardware may be required to achieve higher localization accuracy using the existing localization algorithms that generate higher cost.

Fusion of techniques can improve localization precision and reduce the variance of the position estimation that may minimize communication and the computation overhead of the sensors.

As regards further work, software development is planned to simulate localization techniques presented in the paper for comparing them in several aspects such as speed, time of running, performance and maintainability. The simulation is expected to provide useful results for providing solution to performance and energy optimization in the localization process. After drawing conclusions from these results an algorithm is planned to be recommended which is optimized for performace, as well as to give correction factors for the localisation methods described in the paper.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] BISCHOFF, U., STROHBACH, M., HAZAS, M., KORTUEM, G.: *Constraint-based distance estimation in ad-hoc wireless sensor networks*. In: Proceedings of the Third European Workshop on Wireless, New York, USA, 2007.

[2] WHITEHOUSE, K., KARLOF, C., CULLER, D.. *A practical evaluation of radio signal strength for ranging-based localization*. In ACM Mobile Computing and Communications Review (MC2R), vol.11, no.1, pp.41-52, 2007.

[3] DOHERTY, L., PISTER, K. S. J., GHAOUI, L. E.. *Convex optimization methods for sensor node position estimation*. In Proc. IEEE Conference on Computer Communications (INFOCOM), pp. 1655–1663, 2001.

[4] NICULESCU, D., BADRINATH, B. R.. *Ad hoc positioning system (APS) using AOA*. In Proc. IEEE Conference on Computer Communications (INFOCOM), 2003.

[5] PATWARI, N., HERO III, A O, PERKINS, M., CORREAL, N. S., O'DEA, R. J.. *Relative Location Estimation in Wireless Sensor Networks*, IEEE Transactions on Signal Processing : A Publication of the IEEE Signal Processing Society., vol. 51, pp. 2137, 2003.

[6] TAREQ ALI ALHMIEDAT & SHUANG-HUA YANG (*Senior Member, IEEE*): *A Survey: Localization and Tracking Mobile Targets through Wireless Sensors Network*, IEEE, PGNet, 2007.

[7] HE, T., HUANG, C., BLUM, B.M., STANKOVIC, J.A., ABDELZAHER, T.: *Range-free localization schemes for large scale sensor networks*. In: MobiCom '03: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, pp. 81–95. ACM, New York, 2003.

Production Systems and Information Engineering
Volume 6 (2013), pp. 91-100

PARAMETER APPROXIMATION FOR TOOL LIFE EQUATIONS

SÁNDOR FEGYVERNEKI
University of Miskolc, Hungary
Department of Applied Mathematics
matfs@uni-miskolc.hu

ATTILA KÖREI
University of Miskolc, Hungary
Department of Applied Mathematics
matka@uni-miskolc.hu

JÁNOS KUNDRÁK
University of Miskolc, Hungary
Department of Production Engineering
janos.kundrak@uni-miskolc.hu

[Received March 2012 and accepted September 2012]

Abstract. The aim of this research is to investigate a model for estimating tool life (Kundrák, 1996) during high speed hard turning based on the relationship between the wear progress (speed) and time. The tool life equation is valid in the whole cutting speed range for a given feed rate and depth of cut, and has two extreme values. Here the coefficients A, B and K of the equation are calculated under different cutting conditions and the results are in good agreement with experimental values. It is also shown that different feed rate and depth of cut values change the location of the two extreme values of the tool life curve. A Matlab model was developed to determine the parameters of the tool life equation. Furthermore the properties of estimated parameters were investigated by Statistica 9 program package.

Keywords: tool life equation, least squares, Levenberg-Marquardt

1. Introduction

The time spent cutting up to the allowed wear of the tool is considered tool life. Among the cutting data it is the cutting speed that has the most significant effect on the wear intensity.

The famous formula for estimating tool life was established by Taylor [11] about one hundred years ago. It gave the tool lives belonging to different cutting speeds by v_c-T tool life curve ('Taylor relation'), which is still in use.

The prediction of tool deterioration is done even today by calculating the tool life based on experimental work, using the empirical tool life equations.

Although Taylor's equation gives a simple relationship between tool life and cutting speed, and it is easy to use, at the same time the information is limited only to one cutting datum. In addition, Taylor equations consume a lot of money and time since they give reliable results only in a relatively narrow range of cutting speed [4, 7, 11].

The results of a wide range of tool life experiments proved that the real tool life change depending on the cutting speed has a relative maximum and a relative minimum value.

This is caused first of all by the layers of different thicknesses and origins (for example metallic and non-metallic build-ups) which are created by the resultant of highly complicated mechanical, physical and chemical procedures.

In the last one hundred years a great number of suggestions have been made to describe the connection between the cutting parameters and tool life using simplified approximation curves (Safonov, Temchin, Wu, Kronenberg, Metchisen, Granovski, König-Depiéreaux etc) [2, 8].

Among the relationships found in technical literature only some are able to describe tool life in a wide range of cutting parameters.

A new tool life equation valid for the whole cutting speed range created by Kundrák [6] is suggested to be used for the description of tool life [5, 7, 8,9].

$$T = \frac{K}{V^3 + AV^2 + BV}, \tag{1.1}$$

(time T, speed V, parameters A, B and K).

This tool life equation is valid in the whole cutting speed range for a given feed rate and depth of cut, and has two extreme values. Here the coefficients A, B and K of the equation are calculated under different cutting conditions and the results are in good agreement with experimental values. It is also shown that different feed rate and depth of cut values change the location of the two extreme values of the tool life curve.

A Matlab simulink model was developed to simulate the tool life based on the flank wear rate [1].

In this paper a good iterative, numerical algorithm is given with starting values. Furthermore the properties of estimated parameters are investigated by Statistica 9 program package.

2. Mathematical description of tool life

Denote speed by x and tool life by $f(x)$. From the former investigations we have the following conditions for the approximation of the tool life curve:

F1 $f:(0,\infty)\to(0,\infty).$

F2 $\lim\limits_{x\to 0+0} f(x)=+\infty.$

F3 $\lim\limits_{x\to+\infty} f(x)=0.$

F4 There exist $0<a<b$ such that $f'(a)=f'(b)=0.$

F5 $\lim\limits_{x\to 0+0} f'(x)=-\infty.$

F6 $\lim\limits_{x\to+\infty} f'(x)=0.$

The relationship suitable to describe the complete tool life curve can be written in the following form using the symbols applied in cutting:

$$f(x)=\frac{K}{x^3+Ax^2+Bx}. \tag{2.1}$$

Remark. The function f was given by investigating reciprocal polynomials. Conditon F4 implies that the degree of polynomials is at least 3 and the constant of polynomials is equal to 0 by condition F2.

$$f(x)=\frac{K}{x^3+Ax^2+Bx+C} \tag{2.2}$$

$$f'(x)=\frac{-K(3x^2+2Ax+B)}{(x^3+Ax^2+Bx+C)^2} \tag{2.3}$$

The expansion of further constraints

1. $K > 0.$
2. $C = 0.$
3. $A^2 < 4B$
4. $A^2 > 3B$
5. $A < 0$ and $B > 0.$

Thus $C = 0.$ The possible place of extrema are

$$x_1 = -\frac{1}{3}A - \frac{1}{3}\sqrt{A^2 - 3B}, \qquad x_2 = -\frac{1}{3}A + \frac{1}{3}\sqrt{A^2 - 3B}\,. \tag{2.4}$$

The second derivative

$$f''(x) == \frac{2K(3x^2 + 2Ax + B)^2}{(x^3 + Ax^2 + Bx)^3} - \frac{K(6x + 2A)}{(x^3 + Ax^2 + Bx)^2} \tag{2.5}$$

implies that there is a local minimum at x_1 and a local maximum at x_2.

By the classical relation of roots and cofficients we have

$$x_1 + x_2 = -\frac{2A}{3} \quad \text{és} \quad x_1 x_2 = \frac{B}{3}. \tag{2.6}$$

The optimization problem can be defined as follows: Given the measurements

$$(x_1 y_1), (x_2 y_2), \ldots, (x_n y_n), \tag{2.7}$$

determine the parameters K, A and B such that

$$\sum_{i=1}^{n} \left(y_i - \frac{K}{x_i^3 + Ax_i^2 + Bx_i} \right)^2 \to \min. \tag{2.8}$$

Let

$$Q := \sum_{i=1}^{n} \left(y_i - \frac{K}{x_i^3 + Ax_i^2 + Bx_i} \right)^2. \tag{2.9}$$

Then

$$\frac{\partial Q}{\partial K} = \sum_{i=1}^{n} -2\left(y_i - \frac{K}{x_i^3 + Ax_i^2 + Bx_i} \right)\left(x_i^3 + Ax_i^2 + Bx_i \right)^{-1} = 0, \qquad (2.10)$$

$$\frac{\partial Q}{\partial A} = \sum_{i=1}^{n} 2\left(y_i - \frac{K}{x_i^3 + Ax_i^2 + Bx_i} \right) Kx_i^2 \left(x_i^3 + Ax_i^2 + Bx_i \right)^{-2} = 0, \qquad (2.11)$$

$$\frac{\partial Q}{\partial B} = \sum_{i=1}^{n} 2\left(y_i - \frac{K}{x_i^3 + Ax_i^2 + Bx_i} \right) Kx_i \left(x_i^3 + Ax_i^2 + Bx_i \right)^{-2} = 0. \qquad (2.12)$$

This is a system of nonlinear equations where the order of magnitude of parameters is determined strictly by data. We can solve this system by an iterative method (Levenberg-Marquardt, trust region) [3, 4]. The starting values are obtained by the modified optimization problem

$$\sum_{i=1}^{n} \left(\frac{1}{y_i} - \frac{x_i^3 + Ax_i^2 + Bx_i}{K} \right)^2. \qquad (2.13)$$

If we solve this modified least squares problem, then the system of equations is linear and its solution is a good starting value for the original least squares problem. By further investigations it was obtained that the solution of the problem

$$\sum_{i=1}^{n} \left(\frac{1}{y_i x_i} - \frac{x_i^2 + Ax_i + B}{K} \right)^2 \qquad (2.14)$$

would be better for starting value.

3. Experimental results

Table 1 contains the data of six experiments and the results of our calculations. Having fixed the values of feedrate and depth of cut, tool life was measured depending on different cutting speeds. The bottom part of the table shows the calculated parameters of the six tool life curves and the coordinates of the extrema on them where E is the error of the approximation.

Table 1. Data and results of calculations

v_c [m/min]	T [min]					
X	Y1	Y2	Y3	Y4	Y5	Y6
11	450	300	260	400	310	290
20	310	220	210	270	240	210
29	260	210	190	260	220	210
35	260	220	200	250	220	210
40	260	230	210	260	230	220
50	270	210	160	270	230	220
59	270	170	110	260	200	180
68	270	110	60	220	150	130
80	230	60	30	150	90	70
92	160	40	30	90	50	40
105	100	20	10	60	30	20
120	60	10	6	30	20	10
150	20	4	2	10	7	5
f[mm/rev]	0.025	0.075	0.125	0.05	0.05	0.05
a_p[mm]	0.1	0.1	0.1	0.05	0.15	0.25
K	26.03×10^6	7.67×10^6	5.05×10^6	16.12×10^6	9.86×10^6	8.06×10^6
A	-146.61	-102.97	-90.44	-125.44	-112.03	-107.66
B	6772.17	3373.07	2647.84	4975.66	3959.55	3623.01
E	145.30	152.51	450.12	255.94	104.11	95.93
minx	37.43	26.99	25.02	32.33	28.70	26.93
miny	258.94	215.06	199.69	253.72	219.15	206.57
maxx	60.31	41.66	35.27	51.30	45.98	44.85
maxy	275.35	225.00	204.03	268.12	232.46	223.01

In order to make the discussion of different tool life curves convenient, a user interface is developed with the help of Matlab GUI (Fig. 1). After loading the list of cutting speeds and the corresponding tool life values, the program calculates the parameters of the tool life equation, presents the graphs of v-T and v-L functions and displays the extrema on these functions.

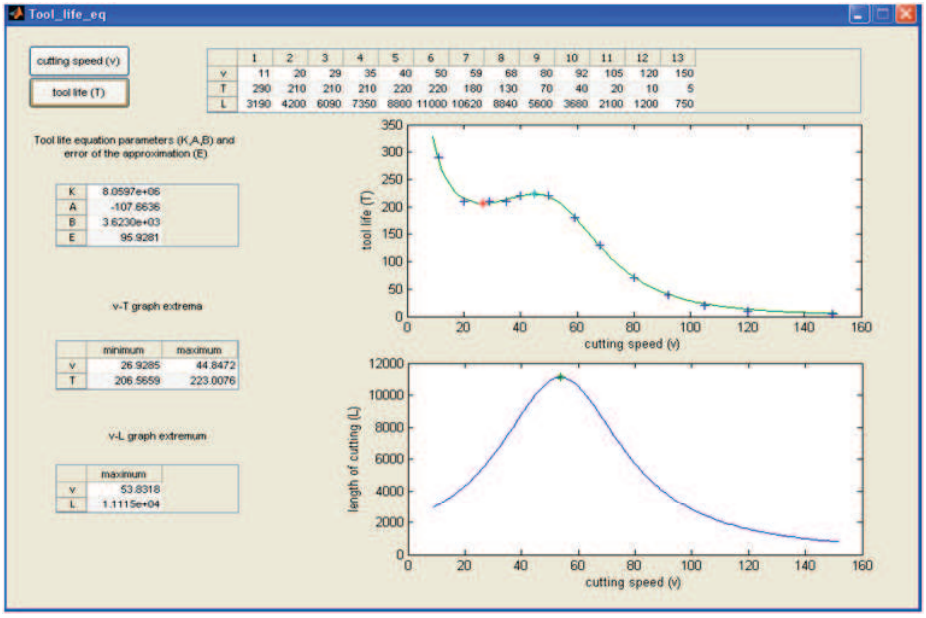


Figure 1. Matlab user interface for the problem

4. Statistical results

Main results are given by *Statistica 9* software where (x11_Y1Y6) is the name of data file.

Model is: Y2=K/(X^3+B*X^2+C*X); Dependent variable: Y2;

Independent variables: 1; Loss function: least squares;

Final value: 152,51007128;

Proportion of variance accounted for: 0,99877281 R =0,99938622.

A common alternative to the least squares *loss function* is to maximize the likelihood or log-likelihood function (or to minimize the negative log-likelihood function; the term maximum likelihood was first used by Fisher, 1929).

Table 2. Iteration history (Statistica 9 software)

	Model is: Y2=K/(X^3+B*X^2+C*X) (x11_Y1Y6) Dep. Var. : Y2			
	Loss Function	K	B	C
1	612,0588	0	0,100	0,100
2	603,7063	19131	0,100	0,100
3	600,1535	22883	-1,904	-0,187
4	592,9964	28799	-4,053	-1,180
5	579,2200	36857	-5,794	-5,159
6	546,9349	46742	-6,928	-17,346
7	525,4859	58241	-7,394	-22,311
8	523,4034	75313	-7,184	-19,546
9	519,6936	108030	-6,796	-14,177
10	513,7193	167372	-6,138	-4,121
11	503,8548	285899	-4,974	17,179
12	489,0267	520740	-3,234	64,050
13	468,1469	971452	-1,915	171,837
14	436,5744	1745041	-6,741	424,369
15	355,1769	2638343	-36,759	972,743
16	138,1249	4731877	-80,710	2119,442
17	35,0382	7526846	-99,462	3173,117
18	12,5452	7703253	-102,898	3370,423
19	12,3495	7676002	-102,973	3373,188
20	12,3495	7675579	-102,971	3373,075

Table 3. Summary: parameter estimates (Statistica 9 software)

	Model is: Y2=K/(X^3+B*X^2+C*X) (x11_Y1Y6) Dep. Var. : Y2 Level of confidence: 95.0% (alpha=0.050)					
	Estimate	Standard error	t-value df = 10	p-value	Lo. Conf Limit	Up. Conf Limit
K	7675579	196731,1	39,016	0,000000	7237235	8113923
B	-103	0,7	-140,242	0,000000	-105	-101
C	3373	53,2	63,393	0,000000	3255	3492

The estimated parameter values are rejected by t-test at arbitrary confidence level.

Table 4. Correlation of parameters (Statistica 9 software)

	Model is: Y2=K/(X^3+B*X^2+C*X) (x11_Y1Y6 Dep. Var. : Y2		
	K	B	C
K	1,000000	-0,736468	0,909304
B	-0,736468	1,000000	-0,937510
C	0,909304	-0,937510	1,000000

The correlations are high in every case.

Table 5. Observed and predicted values (Statistica 9 software)

	Model is: Y2=K/(X^3+B*X^2+C*X) (x11_Y1Y6 Dep. Var. : Y2		
	Observed	Predicted	Residuals
1	300,0000	295,4953	4,50468
2	220,0000	223,9542	-3,95422
3	210,0000	215,5498	-5,54976
4	220,0000	220,6083	-0,60835
5	230,0000	224,6363	5,36375
6	210,0000	211,8833	-1,88328
7	170,0000	167,0517	2,94834
8	110,0000	113,4403	-3,44033
9	60,0000	62,4896	-2,48961
10	40,0000	35,2962	4,70379
11	20,0000	20,3845	-0,38454
12	10,0000	11,8089	-1,80890
13	4,0000	4,9073	-0,90732

The approximation function f of observed values is uniform in the whole interval.

5. Conclusions

With the appearance of superhard tools, the possibility of precision machining applications has significantly widened. The results reported in this paper should help us to promote the economical application of CBN tools. Application of a new tool life equation is suggested and its advantages have been shown, i.e. it is valid for the whole range of the cutting speed, it considers the joint influence of

technological data elements, and it can be applied in practice with our Matlab user interface. This interface is simple, quick, accurate and user friendly.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010/0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] ADESTA, E.Y.T., AL HAZZA, M., RIZA, M., AGUSMAN, D.: Tool Life Estimation Model Based on Simulated Flank Wear during High Speed Hard Turning. *European Journal of Scientific Research,* 39, 2010, pp. 265-278.

[2] BENO, J.: *Theory of metal cutting*, (in Slovakian) Edition Vienala Kosice, 1999.

[3] FORST, W., HOFFMANN, D.: *Optimization - Theory and Practice,* Springer-Verlag, Berlin, 2010.

[4] HORVÁTH, M.: Computer Aided Planning of Part Machining Processes, *DSc thesis (In Hungarian),* MTA SZTAKI Tanulmányok, Budapest, 169/1985.

[5] KARPAT YIGIT; OEZEL TUGRUL: Multi-objective optimization or turning processes using neural network modeling and dynamic-neighborhood particle swarm optimization *Int. J. Advance Manufacturing Technology*, 35, Issue: 3-4, 2007, pp. 234-247.

[6] KUNDRÁK, J.: The scientific principles of increasing the effectiveness of inner surfaces' cutting with CBN Tools, *DSc thesis (In Russian)* Kharkov, 1996.

[7] MAMALIS, A. G., KUNDRÁK, J., HORVÁTH, M.: Wear and Tool Life of CBN Cutting Tools. *Int. J. Advance Manufacturing Technology*, 20, pp. 475–479, 2002.

[8] MAMALIS, A. G. KUNDRÁK, J. HORVÁTH, M.: On a novel tool life relation for precision cutting tools. *J. Manuf. Sci. Eng.*, 127, 2005, pp. 328–332.

[9] ROY R.; MEHNEN J.: *Dynamic multi-objective optimisation for machining gradient materials* CIRP ANNALS-MANUFACTURING TECHNOLOGY, 57, Issue: 1, 2008, pp. 429-432**.**

[10] TAYLOR, F. W.: On the art of metal cutting, *Transaction of the ASME,* V.28, (1901).

[11] TÓTH, T.: Automated technical planning in production engineering, *DSc thesis (In Hungarian),* Miskolc, 1988.

Production Systems and Information Engineering
Volume 6 (2013), pp. 101-112

EFFICIENT 2D SOFTWARE RENDERING

PETER MILEFF
University of Miskolc, Hungary
Department of Information Technology
mileff@iit.uni-miskolc.hu

JUDIT DUDRA
Bay Zoltán Nonprofit Ltd., Hungary
Department of Structural Integrity
judit.dudra@bay-zoltan.hu

[Received March 2012 and accepted September 2012]

Abstract. The market of computer graphics is dominated by GPU based technologies. However today's fast central processing units (CPU) based on modern architectural design offer new opportunities in the field of classical software rendering. Because the technological development of the GPU architecture almost reached the limits in the field of the programming model, the CPU-based solutions will become more popular in the near future. This paper reviews the problems and opportunities of two-dimensional rendering from a practical point of view. An efficient, software based rasterization method is presented for textures having a transparent color component. The applicability of the solution is proved through measurement results compared to other methods and the GPU based implementation.

Keywords: real-time rendering, software rasterization, optimization

1. Introduction

Computer graphics has gone through dramatic improvements over the past few decades, where an important milestone was the appearance of the graphic processors. The main objective of the transformation was to improve graphical computations and visual quality. Initially, the development process of the central unit was far from being a fast paced evolution like today. So based on industry demands, there was a need for dedicated hardware which takes over the rasterization task from the CPU.
Graphical computations have different requirements from the other parts of the software. This allowed for the graphics hardware to evolve independently from the central unit opening new opportunities before developers, engineers and computer games. From the perspective of manufacturers and industry, primarily speed came to the fore against programming flexibility and robustness. So in recent years the

development of videocard technology focused primarily on improve the programmability of its fixed-function pipeline. As a result, today's GPUs have quite effectively programmable pipeline supporting the use of high-level shader languages (GLSL, HLSL, CG).
Nowadays, the technological evolution proceeds to a quite new direction introducing a new generation of graphics processors, the general-purpose graphics processors (GPGPU). These units are no longer suitable only to speed up the rendering, but tend the direction of general calculations similarly to the CPU.
However, the problems of the GPU-based rasterization should be emphasized. The applied model and the programming logic slowly reach its limits, the intensity of progress is apparently decreasing. Even there are fast hardware supported pipeline in current graphics cards, they do not provide the same level of flexibility to the programmer to manage the rendering process as CPU based rendering. Although the existing pipeline and 3D APIs provide many features for developers, if we would like to deviate from the conventional operation, we encounter many difficulties.
Today's GPU architecture is questionable and needs to be reformed, as leading industrial partners strongly suggest [12,13]. The video card industry is currently under development. For example AMD's new Fusion technology (APU - Accelerated Processing Units) represents a whole new generation by the integration of the graphical processor and the central unit.
For the problem posed also by game industry leaders [12], the solution is to return to the software rendering technique, where the display content should be programmed logically and technically using the same language as the application.
This would permit creating a more flexible development environment driving computer graphics to a new direction.
A good basis for this is provided by the huge revolution of the CPUs occuring in recent years. Processor manufacturers have responded with extended instruction sets to market demands making possible faster and mainly vectorized (SIMD) processing also for central units. Almost every manufacturer has developed its own extension e.g. the MMX and SSE instruction family which are developed by Intel and supported by nearly every CPU. Due to new technologies, a software can reach about 2-10x speedup by exploiting properly the hardware instruction set.
This paper investigates practical realization questions of two-dimensional software rasterization. A special optimization solution is presented, which helps to improve the non GPU based rendering for transparent textures.

2. Related work

The software based image synthesis has existed since the first computers and it has been focused even more with the appearance of personal computers until about 2003. After this time almost all the rendering techniques become GPU based. However there were born many interesting software renderers during the early years. The most significant results were the Quake I, Quake II renderers in 1996

and 1998, which are the first real three-dimensional engine [8]. The rendering system of the engines was brilliant compared to the current computer technology and was developed by the coordination of Michael Abrash. The engine was typically optimized for the Pentium processor family taking advantage of the great MMX instruction set. The next milestone of computer visualisation was the Unreal Engine in 1998 with its very rich functionality the time (colored lightning, shadowing, volumetric lighting, fog, pixel-accurate culling, etc) [12]. Today Unreal technology is a leader in the area of computer graphics.

After the continuous headway of GPU rendering software rasterization was increasingly losing ground. Fortunately there are some notable great results also today, such as the Swiftshader by TrasGaming [2] and the Pixomatic 1, 2, 3 renderes [14] by Rad Game Tools. Both products are very complex and highly optimized utilizing the modern threading capabilities of today's Multicore CPUs. The products have dynamically self-modifying pixel pipelines, which maximises rendering performance by modifing its own code during runtime. In addition, Pixomatic 3 and Swiftshader are 100% DirectX 9 compatible. Unfortunately, since these products are all proprietary, the details of their architectures are not released to the general public.

Microsoft supported the spread of GPU technologies by the development of DirectX, but beside of this its own software rasterizer (WARP) has been implemented. Its renderer scales very well to multiple threads and it is even able to outperform low-end integrated graphics cards in some cases [3].

In 2008 based on problem and demand investigations, Intel aimed to develop an own software solution based videcard within the *Larrabee* project [5]. The card in technological sense was a hybrid between the multi-core CPUs and GPUs. The objective was to develop an x86 core (many) based fully programmable pipeline with 16 byte wide SIMD vector units. The new architecture made possible to graphic calculations to be programmed in a more flexible way than GPUs with x86 instruction set [4].

Today, based on the GPGPU technology, a whole new direction is possible at a software rendering. Loop and Eisenacher [2009] describe a GPU software renderer for parametric patches. Freepipe Software rasterizer [Liu et al. 2010] focuses on multi-fragment effects, where each thread processes one input triangle, determines its pixel coverage and performs shading and blending sequentially for each pixel. Interestingly, recent work has also been done by NVidia to create a software pipeline which runs entirely on the GPU using the CUDA software platform [7]. The algorithm uses the popular tile-based rendering method for dispatching the rendering tasks to GPU. Like any software solution, this allows additional fexibility at the cost of speed.

A new SPU (Cell Synergistic Processor Unit) based deferred rendering process has been introduced in today's leading computer game, Battlefield 3[13]. Its graphical engine, Frostbite 2 engine makes possible to handle large number of light sources effectively and optimized.

In publication [1] a modern, multi-thread tile based software rendering technique was outlined where only the CPU has been used for calculations and had great performance results.
Thus, recent findings clearly support the fact that CPU-based approaches are ready to come back in order to improve performance and flexibility. So, the aim of this publication is to investigate performance in the area of the 2D software rendering utilizing today CPUs.

3. Software rendering in practice

Software rasterization is the process, where the entire image rendering is carried out by the CPU instead of a target hardware (e.g. GPU unit). The main memory stores the shape assembling geometric primitives in in the form of arrays, structures and other data. The logic of image synthesis is very simple: the central unit performs the required operations (coloring, texture mapping, color channel contention, rotating, stretching, translating, etc.) on data stored in main memory, then the result is mapped into the *framebuffer* and the completed image are sent to the video controller. Framebuffer is an area in memory which is being streamed by display hardware directly to the output device. Usually it is on video card, but can be mapped into address space of the application and accessed directly like normal RAM.
Software image synthesis has many advantages over the GPU-based technology. As the CPU performs the whole processing, there is less need to worry about compatibility issues because we do not have to adapt to any special hardware, or follow its versions. The image synthes can be programmed uniformly using the same language like the application, so there is no restriction on the data (e.g. maximum texture size) and the processes compared to GPU language shader solutions. Every part of the entire graphics pipeline can be programmed individually. Preparing the software to several platform causes less problems because displaying always goes through the operating system controller, there is no need for special video card driver.
Developing a fast software renderering engine requires lower level programming languages (e.g. C, C++, D) and higher programming skills. Because of the utilized techniques it is necessary to use operating system-specific knowledge and codings. A typical example is when the framebuffer should be copied into the video card's memory for displaying. To support this data transfer, several solutions are developed in practice.
Firstly, we can use the operating system routines for framebuffer transfer, but it is strongly platform-dependent. This method requires writing the bottom layer of the software separately for all the operating systems. A more elegant solution is to use the OpenGL's platform-independent (e.g. glDrawPixels, texture) [6] or the DirectX (e.g. DirectDraw surface, texture) solutions.

4. 2D software rendering solutions

Two-dimensional visualization plays an important role beside today's modern three-dimensional rasterization. We can say that the world moved to the direction of the field of 3D vizualization, but the two-dimensional solutions have always been and will be present as a complementary technique. Several layers of the computer applications belong here: any software that has some kind of graphical menu or windowing system, and mainly the two-dimensional computer games. There are always attempts to make menu systems more illustrative using three dimensional graphics, but these solutions usually return to 2D mapping. The rendering speed at these systems is not critical compared to today's system performance. Mostly a small number of static images vary, other transformations (e.g. rotate, stretch, etc.) are not very typical.
In computer games the rasterization performance requirements are the opposite of this. Generally a large number of continuously changing and moving sets of objects have to be rendered, which consumes significant system resources. Typical features of today's systems are high screen resolution, large texture sets, animations and transformations to reach a higher user experience.
In the following several 2D rasterization techniques are presented, which make it possible to develop a high performance and robost software-based rendering system.

4.1 Classical 2D rendering

The classical two-dimensional software rasterization has to solve a great number of difficulties. The main disadvantage of the GPU-based solutions is that there is no special hardware for rendering, any calculation should be done on the CPU.
The basic building blocks of 2D rendering are two-dimensional images (textures) and objects (animations). The graphics engine is responsible for image generation, it takes objects one by one and draws them into the framebuffer. So the final image is created as a combination of these. In all cases, textures are stored in the main memory represented as a block of arrays. Arrays contain color information about the objects, their size depends on the quality of the texture. Today software works with 32-bit (4 bytes - RGBA) type color images, where image resolution can be up to 1024x768 pixels depending on the requirements of the items to be displayed.
Textures can be classified into two main groups based on the color channels included: there are visual elements with and without alpha channel (A). The distinction is important because the displaying process, the potential acceleration techniques differ in these two groups.

4.1.1 Rendering textures without alpha channel

Textures without alpha channel lack transparency and only have RGB color components. This means that any two objects can draw on each other without

merging any colored pixels of the overlapping objects. The rendering process will be faster and simpler.

The classical mechanism for drawing any type of texture is the per-pixel rasterization (just like in 3D area), which is slow in terms of today's high quality requirements. The graphical engine asses through the graphical objects pixel by pixel and generates the final image. The disadvantage of this is that many elements, which can also consist of many points, have to be drawn on the screen. The per-pixel drawing requires thousands of redundant computations and function calls. In case of each pixel the color information should be read from memory, then depending on the environmental data its position should be determined and finally the color should be written into the framebuffer (e.g. pFrameBuffer[y * screenWidth + x] = color). So the pixel-by-pixel realization is not enough to provide a fast solution because too many small operations are performed, which consumes CPU resources. In a real-time computer game up to 100 different moving objects should be drawn simultaneously to the screen.

In order to achieve the appropriate speed, additional solutions and optimizations are needed. In case of textures without alpha channel, the solution is relatively simple. Move the picture array at once in one or more blocks into the frame buffer and not pixel by pixel. So the main objective of a rendering optimization (for 3D as well) is to try to perform all the operations in blocks as wide as possible. This minimizes unnecessary movement of data and calculations. The following figure shows the process:

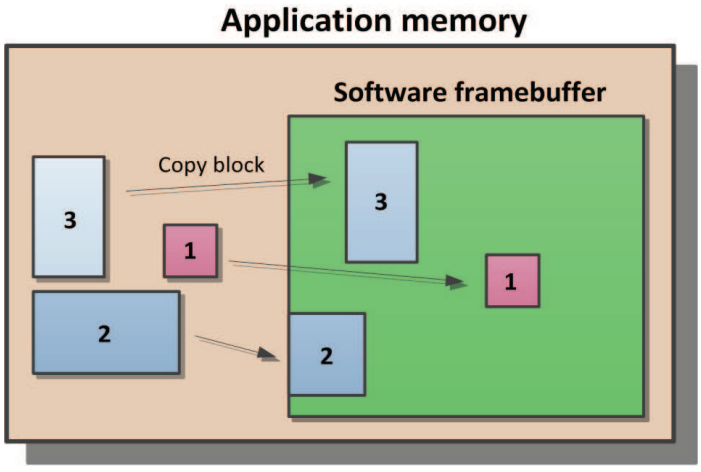


Figure 1. Block oriented texture copy

In this case drawing means that the central blocks of the main memory are copied to a specified area of the framebuffer using system level memory copy operations (e.g. C - *memcpy()*). The solution can achieve significant performance speedup.

However, the method is not complete. The reason is that at pixel level rasterization screen bounding check calculations can be performed easily, but in case of block

oriented rendering the data blocks should be segmented based on the object's position. If any object locates out of the screen bounding rectangle in any direction, a viewport culling should be performed. Although it requires further calculations, the solution remains still fast enough.

4.1.2 Rendering textures with alpha channel

Textures having also a fourth, alpha (A) color channel belong to another group of images. The role of this type of images has increased today, they are used in many areas in order to improve visualization experience (e.g. window shadows, animations with blurred edges, semi-transparent components, etc.). Handling this extra information is not more complicated, but more computing-intensive. The reason is that transparent and non-transparent areas can arbitrarily vary within a texture image (e.g. character animation, particle effects, etc.). Due to this the rendering process is made at per-pixel level because transparent or semi-transparent parts of the objects should be merged with the overlapped pixels. As mentioned earlier, basically the procedure is not very computation intensive, but in systems working with large amounts of objects and larger textures, the solution performance will be insufficient. The following diagram illustrates the problem:

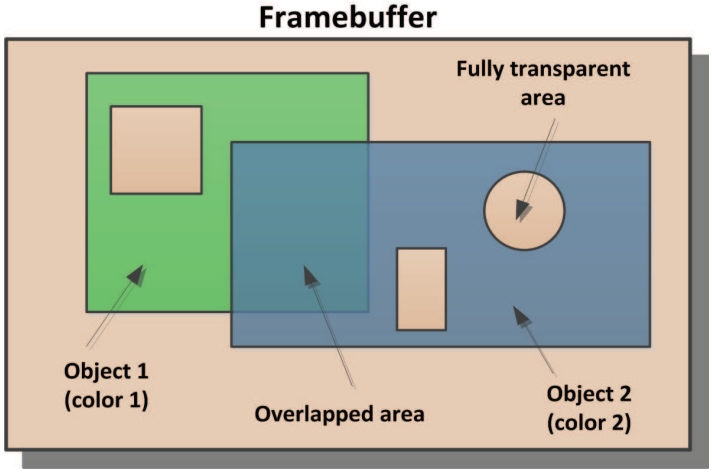


Figure 2. Overlapping RGBA textures

4.2 RLE like object rendering

The above implies that the main problem of two-dimensional software rasterization is to handle semi-transparent textures or textures with 'hole' areas in a performance friendly way. In the following we present a technique which offers an efficient solution to this problem at the cost of some compromises.

It is foreseeable from the above that a process needs to be developed which somehow tries to take advantage of block-based visualization capabilities handling

pixels in blocks. To resolve this issue, let us start from the investigation of a typical texture. If we analyze pixels, we can see immediately that the majority of images have parts where the colors of adjacent pixels are equal to each other or there are fully transparent areas. This provides a good basis to develop an algorithm with a custom data structure, which works like RLE (Run-length Encoding) encoding and can group the same color and adjacent pixels into blocks.

4.2.1 Texture pre-processing

The basic idea of the solution is to collect the same colored pixels into blocks. This requires pre-processing operations for all the loaded textures. During the process the appropriate describing data structure will be established, which helps performing the block oriented rendering in the rasterization stage. The following figure shows the steps and logic of the pre-processing task.

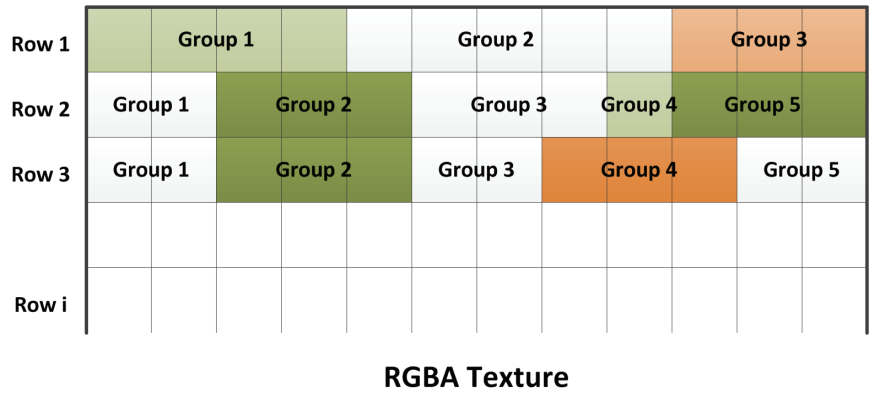


Figure 3. Preparing textures for block oriented rendering

The figure illustrates that an even more complex structure is needed to store the color blocks. Pre-processing is performed row by row, where a separate block is created for all the coherent sets of pixels. For this the following should be stored: color and length of the group, whether the group is transparent or not, and finally a pointer pointing to the first pixel in the original texture address space. Moreover a global data structure should be prepared, which stores the precalculated color groups in rows, their counts and a pointer to the original texture memory space.
The proper implementation of the storage is especially important because in case of systems containing a large amount of objects, the number of loops, function calls, and operations are significant and slow down the rendering process.

4.2.2 The rendering process

During the rasterization stage object mapping is performed by the prepared data structure. This data makes it possible that while the image has ‘holes’ (fully transparent parts), a color group oriented block based blitting can be achieved on the

framebuffer. The result is that the rendering performance of images with alpha-channel can be considerably increased. The rasterization will consist of so many blocks as many were created during the texture pre-processing stage. In addition, the rasterization is performed row by row. One reason is that the framebuffer has been implemented in a row oriented form like in most systems, so a row is a logical unit.
Another reason is that each object can overrun the screen. Although the colors are grouped, parts that are out of screen should be culled during the rasterization.
The row based approach results again in a speed improvement here because if the beginning or the end of the row is out of the screen, other parts (groups) of the row should not be checked. This prevents performing additional computions. The following code summarizes the drawing process as a sample.

```
CFrameBuffer* framebuffer = g_Graphics->GetFrameBuffer();
  for(unsigned int i=0; i < row_group.size(); i++){
    vector<CRLEColor*> image_row = row _group[i];
    for(unsigned int j=0; j < image_row.size(); ++j){
      CRLEColor * c = image_row[j];
      if (c->invisible == false){
       framebuffer->BlitArray(c->offset,c->length,pos.x+c->x,pos.y+c->y);
      }}}
```

4.3 Test results

The following section presents the performance differences of the rasterization techniques with the help of three test cases. The test programs were written using the C++ language applying the GCC 4.4.1 compiler and the measurements were performed by an Intel Core i7 870 2.93 GHz CPU. The chosen screen resolution and color depth were 800x600x32 in windowed mode.
Because of the results' validity we considered it important to implement all the test cases with the GPU based technology as well. With this reference value, the relative performance ratio of the methods will be visible and clear. The hardware used for the test was an ATI Radeon HD 5670 with 1 GB RAM. To display the framebuffer, the OpenGL glDrawPixels solution was applied in an optimized form.
The GPU based reference implementation was also developed with the OpenGL API, where all visual elements were stored in the high-performance video memory and the VBO (Vertex Buffer Object) extension was applied for the rendering. Currently, VBO is the fastest texture rendering method in the GPU area.
The alpha-channel images used in the tests contained an average number of transparent pixels.

Test case 1: during the test we were looking for an answer to the question of how the presented methods can handle a relatively big texture. Although the RLE-based solution logically does not fit this example because the picture does not contain transparent areas, it is advisable to perform this measurement.

Test case 2: the aim of this test is to measure the speed of the RLE based rendering implementation against the classical method in case of an average size (256x256) image with alpha-channel. The fully block oriented approach cannot be used for this type of images.

Test case 3: in test three a heavily loaded rendering system was simulated applying 200 64x64 size RGBA type textures.

During the tests the average *Frame Rate* (Frames Per Second) was recorded at least one minute run-time. The following Table contains the results for all test cases.

Table 1. Benchmark results

	Count	Speed of rasterization methods (FPS)			
		Pixel level	**Block oriented**	**RLE based**	**GPU based reference implementation**
800x600 texture	1	119	1290	1224	3522
256x256 texture (with alpha)	1	910	-	1798	3689
64x64 texture (with alpha)	200	143	-	794	1690

The findings demonstrate that pixel-level rendering was proved to be the slowest because of the large number of operations. However the RLE based approach has good results in all test cases. The frame rate was only lower in the first test, because RLE based rendering requires extra data structures and loops to rasterize the image. This supports the fact well that moving pixels in larger blocks results in significant performance improvements.
Naturally the GPU based implementation produces always the fastest frame rate. But we must not forget that in this case the calculations are carried out by the dedicated hardware. All the data are stored in video ram, so there is no need to move data between the main memory and GPU memory.

4.4 Other optimization issues and features

Developing a really fast software renderer is not an easy task. During the implementation the programmer should take care of several seemingly small coding tricks, which have a strong influence on the performance. For example current 3D hardware is highly optimized for texture and vertex uploads, but framebuffer transfers have been neglected. Therefore the first optimization technique is to implement the framebuffer as an *uint32_t* type array and not as a *floating-point* or *unsigned char* type buffer. This storage type makes it possible to handle all the color components of a single pixel in one unsigned integer type variable, in one single block (e.g. color = A << 24 | R << 16 | G << 8 | B). With this modification at least 20% speed improvement can be achieved.

The built-in data structures (e.g. vector) provided by the C++ STL library are slow, it is not practical to use them [11]. The number of array iterations and unnecessary assignments should be minimized. To detect bottlenecks in the code, use a Profiler application and the component of the compiler which can display the assembly code of a specific code section. These can help to localize the problematic code segments.

Conclusion and further work

Although today's computer graphics is dominated by the GPU market, we cannot forget the opportunities offered by software based image synthesis. The central units have undergone a huge revolution during the recent years, which makes it possible to turn back to CPU based image rasterization in order to gain more flexibility. The techniques presented in this article highlight that developing a really fast software renderer requires a great deal of effort. It is essential to combine several technologies and to use lower-level languages (e.g. C, C++, D) for programming.
This paper discussed software rendering solutions in two-dimensional space. Our further objective is to perform a comprehensive analysis of the triangle rasterization problems and optimization techniques in the area of 3D computer graphics.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] ZACH, B.: *A Modern Approach to Software Rasterization*. University Workshop, Taylor University, 14. dec 2011.

[2] TRANSGAMING INC: *Swiftshader Software GPU Toolkit,* 2012.

[3] MICROSOFT CORPORATION: *Windows advanced rasterization platform (warp) guide*. 2012.

[4] ABRASH, M.: *Rasterization on larrabee*. Dr. Dobbs Portal, 2009.

[5] SEILER, L., CARMEAN, D., SPRANGLE, E., FORSYTH, T., ABRASH, M., DUBEY, P., JUNKINS, S., LAKE, A., SUGERMAN, J., CAVIN, R., ESPASA, R., GROCHOWSKI, E., JUAN, T., HANRAHAN, P.: *Larrabee: a many-core x86 architecture for visual computing.* ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008 Volume 27 Issue 3, August 2008.

[6] ROST, R.: *The OpenGL Shading Language*. ADDISON WESLEY, 2004.

[7] LAINE, S., KARRAS, T.: *High-Performance Software Rasterization on GPUs.* High Performance Graphics, Vancouver, Canada, aug 5. 2011.

[8] AKENINE-MÖLLER, T., HAINES, E.: *Real-Time Rendering*, A. K. Peters. 3nd Edition, 2008.

[9] SUGERMAN, J., FATAHALIAN, K., BOULOS, S., AKELEY, K., AND HANRAHAN, P.: *Gramps: A programming model for graphics pipelines*. ACM Trans. Graph. 28, 4:1–4:11, 2009.

[10] FANG, L., MENGCHENG H., XUEHUI L., ENHUA W.: FREEPIPE: *A Programmable, Parallel Rendering Architecture for Efficient Multi-Fragment Effects*. In Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2010.

[11] AGNER, F.: *Optimizing software in C++ An optimization guide for Windows, Linux and Mac platforms*. Study at Copenhagen University College of Engineering, 2011.06.08.

[12] SWENNEY, T.: *The End of the GPU Roadmap*. Proceedings of the Conference on High Performance Graphics, pp. 45-52, 2009.

[13] COFFIN, C.: *SPU-based Deferred Shading for Battlefield 3 on Playstation 3*. Game Developer Conference Presentation, March 8, 2011.

[14] RAD GAME TOOLS: *Pixomatic advanced software rasterizer,* 2012.

Production Systems and Information Engineering
Volume 6 (2013), pp. 113-120

EVALUATION OF DATA OF FLATNESS MEASUREMENT

SÁNDOR FEGYVERNEKI
University of Miskolc, Hungary
Department of Applied Mathematics
matfs@uni-miskolc.hu

FERENC NAGY
University of Miskolc, Hungary
Department of Applied Mathematics
matnf@uni-miskolc.hu

PÉTER RAISZ
University of Miskolc, Hungary
Department of Applied Mathematics
matrp@uni-miskolc.hu

[Received March 2012 and accepted September 2012]

Abstract. The main point of the milling technology in this paper we are dealing with is that a horizontal rod - of the same breadth as the ingot – 'sits up' onto the highest point of the ingot, and at a given distance (depth) d the surface is milled down. The task to be solved is the determination of this distance d in such a way that the depth of the milling should be at least 8 mm even at the lowest point of the surface.

Keywords: aluminium alloys, depth of the milling

1. Introduction

From aluminium alloys so-called rolling ingots are made to be rolled. The profiles and lengths of the ingots are varied. One of the worst defects arising in the course of rolling is the surface blistering caused by the accumulation of unwanted elements. According to the standpoint of the majority of experts, this accumulation may occour to a higher extent on the surface of the ingots, thus the outer surface is to be removed. If this removal is not of the sufficient extent, then the chance of arising surface blistering is higher, while if the extent of the removal is exorbitant, then it cuts the output down substantially. The two ends and the sides are sawed off (by circular or endless saw), while on the sides of the largest surface the unwanted zone is removed by milling [3].

The main point of the milling technology is that a horizontal rod - of the same breadth as the ingot – 'sits up' onto the highest point of the ingot, and at a given distance (depth) *d* the surface is milled down.

The task to be solved is the determination of this distance *d* in such a way that the depth of the milling should be at least 8 mm even at the lowest point of the surface. The task is easier to understand when looking at the picture of the ingots waiting for measurement.

2. Measurement, Processing Data

Figure 1. Flatness measurement

The surface to be treated can be observed much better in Figures 2 and 3.

Figure 2. The face of an ingot

A zoomed part of this ingot is in Figure 3.

Measurements were performed on the surface of the ingots by a laser device. The accuracy of the measurements was 0.01 mm. There were two-two sequences of measurements for each of the 10 faces of 5 aluminum ingots, in 5 x 10 point per faces. The first sequence of measurements gave in five-five points transversely the vertical deviations with respect to the left-wing edge, while the second sequence lengthwise in ten-ten points the vertical deviations with respect to the first edge.

Figure 3. A zoomed part of the ingot **from Figure 2**

Table 1. Here are the data for one face of an ingot as an illustration

	Ingot **A** face *A*				
405 cm	-3.43	-0.64	-3.61	-1.86	-1.33
360 cm	-1.85	-1.3	-5.11	-1.83	-1.32
315 cm	-1.65	-1.21	-4.27	-3.02	-1.73
270 cm	-1.48	-1.24	-4.05	-2.84	-1.4
225 cm	-0.72	-0.34	-2.77	-1.73	-0.55
180 cm	0.07	0.25	-1.35	-0.82	0.22
135 cm	0.89	0.99	-0.01	0.34	1.03
90 cm	-3.3	1.36	0.71	1.01	1.62
45 cm	1.45	1.15	0.86	0.88	1.33
0 cm	0	0	0	0	0
	x0	**x1**	**x2**	**x3**	**x4**

The problem was to decide with a satisfactory safety, whether the present technology – the sensor of the milling machine is a rod laid on the peaks of the face – is able to ensure the required minimal milling depth (at least 8 mm).

Processing the data the following steps were performed:

1. The vertical deviations with respect to the first left-hand point were calculated.
2. The maximum of each row was determined.
3. The minimum of each face was calculated.
4. For each row the difference of the row-maximum and the face-minimum were calculated.

Thus the results for the previous face **A** of ingot *A* are the following:

Table 2. The calculated results for the previous face **A** of ingot *A*

Ingot A face *A*

					Row-max	absmin	difference
-3.43	-2.38	-4.4	-4.26	-2.66	-2.38	-7.78	5.4
-1.85	-3.94	-7.78	-4.69	-1.26	-1.26	-7.78	6.52
-1.65	-3.76	-6.95	-6.28	-2.23	-1.65	-7.78	6.13
-1.48	-4.18	-7.04	-6.57	-2.44	-1.48	-7.78	6.3
-0.72	-2.91	-5.71	-5.21	-1.39	-0.72	-7.78	7.06
0.07	-2.44	-4.19	-4.57	-0.82	0.07	-7.78	7.85
0.89	-1.28	-2.08	-2.89	0.91	0.91	-7.78	8.69
-3.3	-1.02	-1.94	-2.37	1.51	1.51	-7.78	9.29
1.45	-1.58	-1.8	-2.39	0.94	1.45	-7.78	9.23
0	-2.58	-2.67	-2.87	-0.79	0	-7.78	7.78

It is to be mentioned that in the above model the row-max was chosen from the 5 measured data, while the face minimum was chosen from the 50 measured data. Thus the actual deviation is somewhat greater, but a more precise estimation could be reached only by a much more time-consuming and more expensive sequence of measurements. Both the experience and the photos of the ingots support the supposition of the independence of these data. Therefore we considered them as the elements of a sample (VAR1) [2].

VAR1

5.400	6.520	6.130	6.300	7.060	7.850	8.690	9.290	9.230
7.780	7.090	6.420	7.970	6.440	6.170	5.860	5.680	5.150
6.010	7.460	5.160	4.640	4.600	4.430	4.240	4.030	3.800
3.870	4.290	5.210	4.000	4.710	4.910	5.350	5.580	6.080
6.330	6.320	6.080	4.960	4.400	5.210	4.690	4.820	4.560
4.000	4.530	4.140	4.580	5.450	4.510	4.470	4.800	5.260
5.730	6.320	6.870	7.080	6.950	6.240	3.740	4.940	4.970
5.050	5.310	4.760	3.690	3.780	4.480	5.270	5.100	4.950
5.200	5.400	5.790	6.360	6.840	7.040	6.890	6.210	3.250
3.640	4.090	4.250	4.550	5.090	5.260	5.310	5.830	5.380
3.260	2.620	2.850	3.230	4.190	3.370	3.730	4.240	4.410
4.990								

3. Statistical Results

Since at the beginning of the machining the measuring rod sits onto the maximum point of the first edge and even from the lowest point 8 mm milling is required, thus VAR1 can be treated as 100 ingot-faces.

Some of the descriptive statistics of sample VAR1 by Statistica for windows 9 [2]:

Sample size	100	
Mean	5.310100	
Confidence interval (95%)	5.046214	5.573986
Median	5.155000	
Minimum	2.620000	
Maximum	9.290000	
Range	6.670000	
Lower quartile	4.420000	
Upper quartile	6.190000	
Quartile range	1.770000	
Variance	1.768700	
Standard deviation	1.329925	
Skewness	0.681855	
Kurtosis	0.575884	

On the basis of these statistics and considering the histogram, we performed tests for distribution fitting for normal (Gaussian), lognormal, gamma, extreme value and Weibull distributions [1].

Using χ^2-test for the distribution fitting for lognormal distribution, the value of the χ^2-statistic is 2.246493, with degree of freedom 8. For the other plausible distributions the χ^2-statistics are greater.

Considering the density histogram, the above statistics and the χ^2-test, we can consider the sample VAR1 as of lognormal distribution [2].

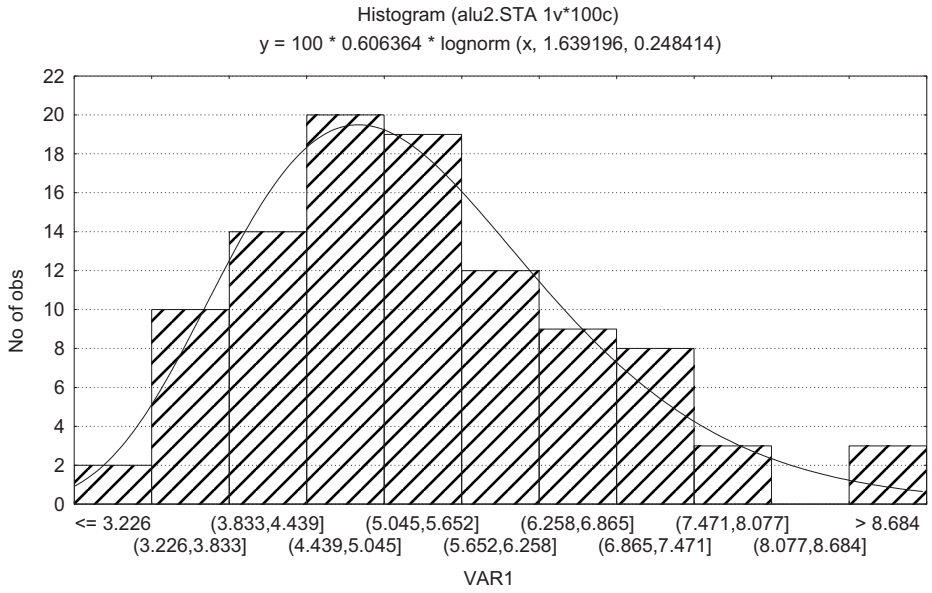


Figure 4. The frequency histogram was made using Statistica for Windows

The practical consequence of this statement can be demonstrated by some examples [2]:

If the distance of the measuring rod and the face-minimum is supposed to be 9 mm, then it will be right in 98.76 % of the cases.

If the distance of the measuring rod and the face-minimum is supposed to be 8 mm, then it will be right in 96.18 % of the cases.

If the distance of the measuring rod and the face-minimum is supposed to be 7 mm, then it will be right in 89.15 % of the cases.

Approaching the situation from the point of view of reliability, we get the following values.

To get a 99% reliability, the distance of the measuring rod and the face -minimum should be supposed to be 9.180 mm,

To get a 98% reliability, the distance of the measuring rod and the face -minimum should be supposed to be 8.580 mm,

To get a 95% reliability, the distance of the measuring rod and the face -minimum should be supposed to be 7.751 mm,

To get a 90% reliability, the distance of the measuring rod and the face -minimum should be supposed to be 7.082 mm,

Subsequently seven sequences of measurements were performed – under various conditions – and the measured data fully supported the abovementioned inferences [1].

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010/0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] FEGYVERNEKI, S.: *Robust estimators and probability integral transformation*, Math. Comput. Modelling, 38, (2003), pp. 803-814.

[2] JOHNSON, J. L.: *Probability and Statistics for Computer Science*, Wiley New York, 2008.

[3] FEGYVERNEKI, S., RAISZ, P.: *Sampling from delivered scrap, Report* (in English) - ALCOA-KÖFÉM Kft. Öntöde Üzletág, (2008), pp.1-29.