Quantization with Knowledge Base Applied to Geometrical Nesting Problem

Grzegorz Chmaj  
University of Nevada, Las Vegas, chmajg@unlv.nevada.edu

Leszek Koszalka  
Wroclaw University of Technology, leszek.koszalka@pwr.wroc.pl

Follow this and additional works at: http://digitalscholarship.unlv.edu/ece_fac_articles

Part of the Algebraic Geometry Commons, Electrical and Computer Engineering Commons, and the Geometry and Topology Commons

Citation Information
http://digitalscholarship.unlv.edu/ece_fac_articles/853

This Article is brought to you for free and open access by the Electrical & Computer Engineering at Digital Scholarship@UNLV. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Publications by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
QUANTIZATION WITH KNOWLEDGE BASE APPLIED TO
GEOMETRICAL NESTING PROBLEM

Grzegorz Chmaj and Leszek Koszalka

Chair of Systems and Computer Networks, Wroclaw University of Technology,
Wroclaw, Poland

Keywords: nesting, quantization, 2D allocation, knowledge base, geometry shapes, algorithm, simulation

Abstract

Nesting algorithms deal with placing two-dimensional shapes on the given canvas. In this paper a binary way of solving the nesting problem is proposed. Geometric shapes are quantized into binary form, which is used to operate on them. After finishing nesting they are converted back into original geometrical form. Investigations showed, that there is a big influence of quantization accuracy for the nesting effect. However, greater accuracy results with longer time of computation. The proposed knowledge base system is able to strongly reduce the computational time.

1 Introduction

Nesting is a geometrical problem of placing two-dimensional shapes on a surface without overlap and with minimizing the surface area used. The goals of nesting algorithms can differ among minimizing wasted surface area and maximizing the amount of shapes placed on or in a specified container. This kind of problems are everyday-questions in the commercial companies, especially factories and cutting manufacturers. Also, nesting problems appear in environmental architecture planning, transport, and many other places. Nesting problems can be divided into [2]: decision problems (having given region and set of shapes the algorithm states whether shapes will fit in the region), knapsack problem (given shapes have to be placed on a given region in a way minimizing used surface), bin packing (there are set of shapes and set of regions, the algorithm minimizes a number of used regions needed to place set of shapes) and strip packing problem (with given set of shapes and a width of rectangular region, the algorithm has to minimize length of region containing all shapes placed).

Some nesting problem implementations allow overlapping shapes in specific situations [2]. Different constraints can be considered. Usually, there is no constraint on shape – it can be rectangle, also can contain roundness. More often constraints are applied to the nested region – due to technological issues, the region is often a fixed width rectangle with unlimited length (e.g. roll of material in clothing industry). According to the problem conditions, appropriate nesting algorithm should be used. There were attempts to solve this problem by using many different ways: e.g. geometry theory, ant algorithms [1], heuristic methods [5], genetic algorithms [4] – but even for small sets of input data – nesting problem is hard to be solved in a reasonable time.

This paper describes a concept of applying quantization with knowledge base to slide the nesting algorithm. It is assumed, that there are no constraints on regions and on shapes. Each shape is converted into binary form, which is further used to pair shapes. As a pairing algorithm the Min-Rectangle (MR) [3] algorithm is used which is able to find a co-placement of two shapes giving smallest bounding rectangle. The proposed QKBMR (Quantization with Knowledge Base in Minimum Rectangle) system gives opportunities for simulations. Research done by the authors is focused on the influence of quantization and knowledge base implementation on the nesting process.

Section 2 states the problem of nesting. Section 3 contains definitions of basic terms and Section 4 presents the proposed QKBMR system. In Section 5 the results of investigations are discussed. Section 6 contains conclusions and perspectives of further research.

2 Problem statement

Nesting is a term that is used to describe several allocation problems of two- or three-dimensional cutting or placing defined set of shapes. Implementations of the problem can vary with different constraints. Constraints can be divided into the following categories:

- shapes constraints – overlapping conditions, known (or not) shapes queue, shapes queue sorting, etc.
- region constraints – shape of region, its infinity in specified dimensions, etc.
- nesting process constraints – minimizing time, minimizing usage area, etc.

In general, the nesting problem statement does not allow shape overlapping, but there are some nesting sub-problems where shapes overlapping can occur.

Figure 1. General nesting scheme

n shapes to place

nesting system

placement

No matter what implementation, the scheme of nesting is always as in Fig. 1.
An example of the nesting problem – with region shape as a finite rectangle and known shape queue is shown in Fig. 2.

![Figure 2. Example of nesting](image)

The considered nesting problem may be stated as follows:
- **given**: the set of shapes with known geometry and the region in which shapes can be placed,
- **to find**: shape placement within the region,
- **such that**: to minimize wasted surface and the time of completing nesting process.

### 3 Nomenclature

To widely describe the nesting problem issues the following nomenclature is introduced:

**Shape** – a geometric closed form defined by geometric characteristic. Any shape can be described by set of points and arcs. Shape combined with $n$ amount of $e$ elements, where each $e$ is a point $P$ (described by position $x,y$) or an arc $A$ (described by two $P$ and radius) is denoted as $$G = \{e_n \in P: (x_n, y_n) \land A: (x_{ni}, y_{ni}, x_{ni+1}, y_{ni+1}, r_{ni})\}.$$  

**Region** – an area of potential placement of shapes. For the considered nesting problem the region is rectangular with infinite length and width. Region containing points $P(x_n, y_n)$ where $x_n, y_n \in R$ is denoted as $$R = \{P(x_n, y_n): -\infty < x_n < +\infty \land -\infty < y_n < +\infty\}.$$  

**Quantum** – a discrete part of geometric area, is characterized by size and logical binary state assigned. Quantum having a size of square side length in geometric interpretation and $(x, y)$ discrete position in mesh $M$ is denoted as $Q(a)(x, y) = [0,1]$.  

**Quantum size** – denoted as $a$, describes the size of quantum, what determines mesh structure and quantization accuracy. 

**Intersection function** – denoted as $\text{INT}(A, B)$ – returns logical true/false result: if area $A$ and area $B$ intersect, logical 1 is returned, otherwise $\text{INT}(A, B)$ returns logical 0.  

**Bounding rectangle $B_G(x, y)$** – minimum size rectangle of width=$x$ and height=$y$ that $G$ can fit inside without intersecting $B_G$ boundaries. $B_G$ assigned to $G$ will satisfy following formula: $$B_G=\{P(x_p, y_p): x_p, y_p \in R \land x_p \leq x_m \land y_p \leq y_m\}:$$  

$$(\forall P(x_p, y_p) \in G, x_m \leq x_p \land y_m \leq y_p) \land x_m \geq x_p \land y_m \geq y_p \}. $$  

**Mesh** – $R$ is divided (Fig. 3) into $Q(a)$ parts, that composes two dimensional mesh on nesting area. According to $R$, mesh having quantum size of $a$ is denoted as $M(a)$ and has infinite length and width: $$M(a)=\{Q(a)(x_n, y_n) \in R: -\infty < x_n < +\infty \land -\infty < y_n < +\infty\}.$$  

**Quantization process $QP$** – process of changing geometry shape into its binary representation. $QP$ is performed using quantization function:

$$QP(G) = \{\forall Q(a)(x_n, y_n) \in B_G, Q(a)(x_n, y_n) = \text{INT}(Q(a)(x_n, y_n), G)\}.$$  

**Binary shape** – denoted as $S(w, h)$ is quantized representation of geometric shape $G$. $S(w, h)$ having width=$w$ and height=$h$ (measured in $Q(a)$ width/height which means multiplying $w$ and $h$ by $a$), consist of $Q(a)(x_n, y_n) \in B_G$ and is the result of $QP(G)$ function.

$$S(w, h) = \{Q(a)(x_n, y_n) \in B_G: Q(a)(x_n, y_n) = \text{INT}(Q(a)(x_n, y_n), G) \land x_n \leq 0, w, y_n \leq 0, h\}; \quad B_G(x_n, y_n): x_n = a \times w, y_n = a \times h.$$

**Binary Shape Set $BSS$** – a finite set of $S$ containing $BSSC$ elements. $BSSC$ can be variable during nesting process, which allows interactive adding of new shapes to $BSS$, where $\text{BSS} = \{S_1, S_2, ..., S_{BSSC}\}$.  

**AND($S_1 S_2$)** – binary AND operation for bit sequences $S_1, S_2$.  

**XOR($S_1 S_2$)** – binary XOR operation for bit sequences $S_1, S_2$.  

**NEG($S$)** – binary negation operation for bit sequence $S$.  

**ROL1($S$)** – binary rotating right of binary sequence. Rotates by one position:

$$\text{ROL1}(S) = S[1] || S[n], S[n] = S[1]; \quad i \in \{<2, n>,$$

where $S_1$ is S before $RO1$, $S_2$ is S after $RO1$, $i$ is a position in a binary sequence, $n$ is amount of bits in sequence.  

**ROR($S$)** – binary rotating right of binary sequence ($S$ shape) by $t$ bits. Performs $RO1$ operation $t$ times.  

**ROL1($S$)** – binary rotating left of binary sequence. Rotates by one position:

$$\text{ROL1}(S) = S[1] || S[n], S[n] = S[1]; \quad i \in \{<2, n>,$$

where $S_1$ is S before $ROL1$, $S_2$ is S after $ROL1$, $i$ is a position in a binary sequence, $n$ is amount of bits in sequence.  

**Knowledge Base Element $KBE$** – a data set containing information about two $S$ and their best combination. $KBE$ containing knowledge about $S_1$, $S_2$, and the best combination $C$ is denoted as $KBE(S_1, S_2) = C$.  

![Figure 3. Region divided into $M(a)$ mesh](image)
Knowledge Base KB – finite set containing KBC (KB Capacity) number of KBE-s. KB expands self during nesting.

\[ KB = \{KBE_1, KBE_2, ..., KBE_{KBC}\} \]

In the paper, the following assumptions are taken into consideration:

- \( R \) has no specified shape,
- resulting area with nested shapes can be of any shape,
- the system tries to afford the best speed of nesting, and next allocating efficiency,
- the shape queue is known to the system and it is able to preprocess it before starting QP and nesting process,
- all shapes are quantized using the same a in \( Q(a) \).
- the shapes can be rotated to obtain better efficiency (UseRotating parameter),
- rotating of \( S \) is performed by angle \( \gamma = \frac{k\pi}{2}, k \in \{0,1,2,3\} \).

4 Nesting system

The core of the proposed nesting system is QKBMR algorithm which reads a list of G, quantizes them and then tries to nest them in \( R \). QKBMR transforms \( R \) in such a way, that (0,0) point can be defined, but it is not related to any \( G \) – no matter what shape of a \( G \) (or \( G \) combination) is – the algorithm places it starting from (0,0) point of \( R \).

During quantization a parameter is used, which has a large influence on precision of quantizing – if \( a \) is smaller, more \( Q(a) \) elements are used for quantizing \( G \), which also changes the final nesting effect. KB is a very important part of nesting system.

![Diagram of the proposed nesting system](image)

It may save a lot of time when nesting a shape that the system already processed. The QKBMR system with cooperating elements is presented in Fig 4.4.

The components of the system are:

- **GQ** – G Queue – input queue of geometrical shapes \( G \).
- **GQPP** – GQ Preprocessor – a module that is able to operate on input GQ before QP. GQPP uses GQPP algorithm set to choose method of processing GQ.
- **QP** – Quantization Process – converting GQ containing \( G \) elements to BSS. Parameter \( a \) determines precision of conversion process.
- **BSS** – Binary Shape Set – set of BSSC number of \( S \) elements. BSS is the result of QP.
- **SP** - Shape pairing – module trying to find the best co-placement of \( S_a \) and \( S_b \). UseRotating parameter determines whether QKBMR is allowed to use rotating of \( S \) elements, Use KB parameters enables to use KB.
- **KB** – Knowledge Base, containing KBC number of KBEL.

Optional external KBEL sets – KB can use knowledge that was generated by another instance of KBE. External sets could also be implemented as global network database, used by many QKBMR systems to share knowledge about shape pairing.

**NM (Nesting Module)** – main part of system that manages \( S \) elements and finally places them in \( R \). \( M(a) \) parameter is used to describe mesh in \( R \).

**DeQP** – DeQuantization Process – converting \( S \) elements into \( G \) elements. DeQP is an optional module, can be omitted.

4.1 Quantization process

Quantization process (QP) outputs with \( S \) having \( G \) object as an input. QP is performed for each \( G \) separately. The size of \( S \) and accuracy of QP depends on \( a \) parameter. The time needed to perform QP on \( G \) can be expressed with a following relation:

\[ T_{QP} = k WHla \]  

where: \( W \) – geometric width of \( G \), \( H \) – geometric height of \( G \), \( k,l \) – coefficients related to the machine used.

The QP algorithm for a given \( G \) works as follows:

1. Find \( B_d(x,y) \).
2. Divide \( B_d(x,y) \) into \( Q_e(x,y) \) elements. The result is a mesh of \( Q_e \) elements with \( w \) amount of \( Q_e \) in each row of mesh and \( h \) amount of \( Q_e \) elements in each column.
3. Assign state to all \( Q_{\alpha}(x,y) \) elements: \( \forall Q_{\alpha}(x,y) \in B_{G}, Q_{\alpha}(x,y) = \text{INT}((Q_{\alpha}(x,y),G), c \in <0,w), d \in <0,h). \)

![Figure 5. Shape G and its quantized representation S](image)

**Example.** The quantized form of the shape \( G \) (Fig. 5) can be denoted as

\[
S(3,2)=1111011110111.
\]

### 4.2 Normalization and denormalization process

The **Normalization Process (NP)** is a process of extending two \( S-es \) according to their properties. Each shape \( (S_1 \) and \( S_2 \) is normalized. \( S_1 \) is always a base – after normalization is centered in its own mesh. \( S_2 \) is placed in left-top corner of its own mesh. \( NP \) adds columns and/or rows containing zeroes to both \( S \) to equalize them by size. \( NP \) allows finding suitable coexistence by using \( SP \) process. The \( NP \) uses the following algorithm:

1. For two received \( S \) objects: \( S_1(x,y)\) and \( S_2(x,y) \),
   - compute: \( mx=x_1+2^x*y_1, my=y_1+2^y*y_2 \).
2. Normalization \( S_1 \):
   - 2.1. Add \( c=mx-x_1 \) columns to \( S_1 \) (contain only zeroes).
   - 2.2. \( x_1=mx \).
   - 2.3. Add \( r=mx-x_1 \) rows to \( S_1 \) (contain only zeroes).
   - 2.4. \( y_1=my \).
   - 2.5. \( t=c/2^r*2^y; \) Perform ROR with \( r \) positions.
3. Normalization \( S_2 \):
   - 3.1. Add \( c=mx-x_2 \) columns to \( S_2 \) (contain only zeroes).
   - 3.2. \( x_2=mx \).
   - 3.3. Add \( r=mx-x_2 \) rows to \( S_2 \) (contain only zeroes).
   - 3.4. \( y_2=my \).

The **QKBMR** algorithm also needs **Denormalization Process (DP)**, which performs removing ordering rows and columns containing only zeroes. The rule states: \( DP(NP(S))=S \). \( DP(S(x,y)) \) is performed using the following algorithm:

1. \( C_\text{r}(2^{x+1}-1)^*2^{y+1} \), if \( \text{AND}(S,C_\text{r})=0 \) - perform remove last \( x+1 \) bits from binary representation.
2. \( C_\text{b}=\text{NEG}(2^{x+1}*(y+1)-2^{x+1}) \), if \( \text{AND}(S,C_\text{b})=0 \) – perform remove last \( x+1 \) bits from binary representation.
3. \( C_\text{c}=C_{\text{r}}+C_\text{b} \), \( C_\text{c}=C_{\text{r}}+C_\text{b}+C_\text{r} \), \( C_\text{c}=2^{x+1} \), if \( \text{AND}(S,C_\text{c})=0 \), perform remove bits from positions \( p \), for whose \( (p \text{ modulo } (x+1)=0) \) formula is satisfied.
4. \( C_\text{d}=\text{NEG}(\text{AND}(C_{\text{d}}, C_{\text{b}}, C_{\text{c}}, C_{\text{r}})), \) where \( C_{\text{d}}=2^{(2^x+1)-2}2^{x+y}(k(x+1)+2^y(1+k(x+1)), \) if \( \text{AND}(S,C_{\text{d}})=0 \), perform remove bits from positions \( p \), for whose \( (p \text{ modulo } (x+1)=0) \) formula is satisfied.

**Example.** \( S \) normalized with \( NP \) is shown in Tab. 1.

<table>
<thead>
<tr>
<th>Before NP</th>
<th>After NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>( S_1 )</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>( S_2 )</td>
</tr>
<tr>
<td>1111</td>
<td>00000000</td>
</tr>
<tr>
<td>0111</td>
<td>00000000</td>
</tr>
<tr>
<td>0011</td>
<td>00111100</td>
</tr>
<tr>
<td>0001</td>
<td>00000000</td>
</tr>
<tr>
<td>0000</td>
<td>00000000</td>
</tr>
<tr>
<td>0000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

### 4.3 Shape pairing

**Shape Pairing (SP)** is an algorithm, that tries to find the best coexistent \( S_2 \) for a given \( S_1 \). \( S_1 \) is always the first \( S \) from \( BSS \). \( SP \) compares \( S_1 \) with every unpaired \( S \) from \( BSS \) and records the actual best pair. The efficiency of a given pair is determined by the \( EFF \) (2) coefficient:

\[
EFF = I/h (S_{i+1}+1)^* (S_{i+1}+1)
\]

The \( EFF \) uses \( w \) and \( h \) from \( S_1 \), because after \( NP \) both \( S_1 \) and \( S_2 \) have the same \( w \) and \( h \). The \( S \) that is the best pair for \( S_1 \) according to the \( EFF \) coefficient is marked as “paired”. \( SP \) works using the following procedure:

1. \( S_1 \) is a base for pairing and “stationary” object (will remain unmoved in normalized mesh).
2. \( S_1 \) is a moving object.
3. For every unpaired \( S \) from \( BSS \) do:
   - 3.1. Normalize \( S(x,y) \) with \( S_1(x,y) \).
   - 3.2. \( S_1=\text{ROR}(S_1) \).
   - 3.3. If \( S_1[0]=0 \lor S_1[1]=1 \), go to 4.
   - 3.4. If \( \text{AND}(S_2,C_1=0) \lor \text{AND}(S_2,C_1=0) \), where \( C_1=C_{2+1}, \) \( C_1=C_{2+1}+C_0 \), \( C_1=2^{x+1}, \) perform \( \text{ROR}(S_2,x+y+1) \), else perform \( \text{ROR}(S_1,1) \).
   - 3.5. \( M=\text{OR}(\text{XOR}(S_1,S_2),S_1,S_2) \).
   - 3.6. If \( M=0 \), go to 3.2.
   - 3.7. Compute \( EFF \) (OR(S_1,S_2))
   - 3.8. If \( EFF<EFF, EFF=EFF \) and \( BF=\text{OR}(S_1,S_2) \).
   - 4. \( BF \) contains best found co-placement of \( S_1 \) and \( S_2 \), \( GLEFF \) contains \( EFF \) for this pair.

After performing the SP algorithm, the **QKBMR** receives information from the SP module about the best fit found for a given \( S_1 \), the number of \( S_2 \) that made the best fit with \( S_1 \), and the value of \( EFF \) coefficient for this fit.

### 4.4 Knowledge base

Every \( S \) is described by some bits and \( x \) and \( y \) coefficients. Depending on the \( a \) factor, the \( QP \) process differs in accuracy. This means, that many shapes can be quantized into the same
bit representation, and a coefficient has large influence on how many shapes from a given set will go into equal binary appearance. SP can use KB to optimize the process of searching for the best fit. Before using the internal pairing algorithm, SP can request KB for a specified pair $S_1$ and $S_2$. If KB has such a record, it will reply to SP with the best fit. This best fit can be placed in KB by the same SP module, or can originate from another module. Many different Nesting systems can share one KB. If KB does not have such a record, according to $S_1$ and $S_2$, it replies to SP with “no result” message. In that case, SP performs an internal pairing algorithm for $S_1$ and $S_2$ and results with the best fit for these two considered shapes. Then, SP sends the $S_1$, $S_2$ and result $S_3$ to KB, which saves it for the future usage. When using KB, the SP works in the following way:

1. $S_1$ is a base for pairing and “stationary” object (will remain unmoved in normalized mesh). $S_2$ is moving object.
2. GLEFF=0
3. For every unpaired $S$ from BSS do:
   3.1. Ask KB for $S_1$ and $S_2$. If KB replied with best fit answer, place answer to BF, count EFF and put in into GLEFF, go to 5.
   3.2. Normalize $S_1(x_1,y_1)$ with $S_2(x_2,y_2)$
   3.3. $S_0=\text{ROR}(S_1)$
   3.4. If $S_0[1]=0 \lor S_0[1]=1$, go to 4.
   3.5. If AND($S_1$,$C_1$)$=0 \lor$ AND($S_0$,$C_2$)$\neq0$, where $C_1=C_0$, $C_2=C_0$\[2^{i+1}+C_0$, $C_0=2^{i+1}$, perform $\text{ROR}(S_0,x_2+1)$, else perform $\text{ROR}(S_0)$.
   3.6. $M=\text{XOR}(\text{XOR}(\text{OR}(S_1,S_2),S_1),S_2)$
   3.7. If $M=0$, go to 3.3.
   3.8. Compute EFF (OR($S_1,S_2$))
   3.9. If GLEFF$<EFF$, GLEFF=EFF and BF=OR($S_1,S_2$)
   4. Send $S_1$, $S_2$, and BF to KB.
   5. BF contains best found co-placement of $S_1$ and $S_2$. GLEFF contains EFF for this pair.

### 4.5 Rotating mechanism

The nesting problem, stated in this paper, allows G objects to be rotated. This means, that also S objects can be rotated. QKBMR algorithm uses RM (Rotating Mechanism) to rotate binary representations of G, to afford better fit of two shapes. Rotating is performed when searching for the best fit, the rotated figure is also used while nesting. The BSS always contains non-rotated objects – also objects that are nested in rotated form, remain in original non-rotated figure. Rotating binary shapes is not easy when rotating angle is other than $\gamma=\pi/2$, $k \in \{0,1,2,3\}$, so QKBMR algorithm uses only these four values while rotating S objects. Rotating can increase the time needed for finding the best fit, but in many cases it is able to find much better fit. When using KB, performing RM is suggested. RM gives better results of pairing, these results will be added to KB, so it is good to add better fits because pairs recorded in KB will not be paired anymore. SP with RM (and also KB) works using the following procedure:

1. $S_1$ is base of pairing and “stationary” object (will remain unmoved in normalized mesh). $S_2$ is moving object.
2. GLEFF=0, $k=1$
3. For every unpaired $S$ from BSS do:
   3.1. Ask KB for $S_1$ and $S_2$. If KB replied with best fit answer, place answer to BF, count EFF and put in into GLEFF, go to 5.
   3.2. $k=k+1$. Rotate S using $\gamma=k\pi/2$ angle
   3.3. Normalize $S_1(x_1,y_1)$ with $S_2(x_2,y_2)$
   3.4. $S_0=\text{ROR}(S_1)$
   3.5. If $S_0[1]=0 \lor S_0[1]=1$, go to 3.11.
   3.6. If AND($S_1$,$C_1$)$=0 \lor$ AND($S_0$,$C_2$)$\neq0$, where $C_1=C_2+1$, $C_2=C_1\ast2^{i+1}+C_0$, $C_0=2^{i+1}$, perform $\text{ROR}(S_0,x_2+1)$, else perform $\text{ROR}(S_0)$. 
   3.7. $M=\text{XOR}(\text{XOR}(\text{OR}(S_1,S_2),S_1),S_2)$
   3.8. If $M=0$, go to 3.4.
   3.9. Compute EFF (OR($S_1,S_2$))
   3.10. If GLEFF$<EFF$, GLEFF=EFF and BF=OR($S_1,S_2$)
   3.11. If $k<3$ go to 3.2.
4. Send $S_1$, $S_2$, and BF to KB.
5. BF contains best found co-placement of $S_1$ and $S_2$. GLEFF contains EFF for this pair.

### 4.6 Nesting module

NM is the main module of QKBMR system. NM sends requests to SP. After performing SP, resulting pair ($S_1$ and $S_2$) is merged and recorded as $S_1$, $S_2$ is marked as “paired” – so it will not be taken into consideration during the next pairing processes. NM has a block, that is able to decide whether to finalize nesting of $S_1$ and $S_2$ (place them on $M_{\text{pair}}$) or to send a request to SP once more, but with special conditions. This can be useful, when the algorithm used in SP does not work well enough – then NM can detect that kind of pair and request to find the pair for $S_1$ again, without using $S_2$, previously rated as the best fit. NM is the only module of QKBMR, that is able to operate on BSS during nesting process – so it is easy to implement some exclusions for pairing algorithms. Also, according to some rules, NM can pre-nest some shapes, before starting to request SP. These additional functions had not been researched and are categorized as future work.

### 4.7 GQPP and DeQP

Two additional mechanisms of operating on shapes, called GQPP and DeQP, are included into QKBMR algorithm.

GQPP is a GQ preprocessor. For some G sets and pairing algorithms, some operations on G set can speed up the nesting process. GQPP performs some sorting or other methods of changing G order in GQ. In the presented research, GQPP was not activated.

DeQP is a module that performs de-quantization: converts binary representation (S) into geometrical figure (G). Due to DeQP, QKBMR is able to restore original form of shapes after the nesting process, so binary conversion is transparent for the user of QKBMR algorithm. Because QKBMR records original geometric form of shape, so there is no loss of information. DeQP also uses recorded relations of quantized form and
original form – so regardless of a parameter, after QP and DeQP, shape will be placed in the same place on the canvas.

5 Investigations

QKBMR algorithm had been implemented in the experimentation system to research the efficiency of the proposed mechanisms. QKBMR algorithm is a complex system, containing many modules presented in Section 4 and requiring the usage of several algorithms.

QP is a process that is required to be performed before the other nesting modules. QP effect depends on a coefficient in an exponential way. The relationship between a and QP, found on the basis of results of simulations is shown in Fig. 6.

A small a coefficient gives (more accurate) mapping of G to S, but requires more time. When performing the nesting process using algorithms that use quantized shapes, it is possible to quantize them once and store in that form, so no quantization would be needed during next nesting processes. For every nesting case, an appropriate a coefficient should be taken. It also affects the nesting accuracy. Many randomly-generated sets of shapes were taken, processed with QP, and then results averaged.

The quantization coefficient a also affects the time of nesting process. When G is transformed into a larger set of bits, SP module has more data to process. This is an exponential relation, shown in Fig. 7, in logarithmic scale. As is visible on the graph, there is an a value, for which time decreases much, and for larger a values, its influence to time is smaller. If results of nesting for this edge value are satisfactory, this value should be used for processing shape sets. The time and a impact can differ according to the algorithm used by SP module.

During the nesting process, in module SP the same S can form a pair with other shapes many times. Because of the fact that the nesting process is very time-consuming it is worth trying to save the time by recording already computed best-fits for shapes. A knowledge base system used in QKBMR algorithm saves all the computed cases for future use, not only by the given system but also by other systems that share the given KB. There is some time needed by KB, but it is very small compared to the pairing process. Investigations showed, that in the standard averaged case taking information about pairs from KB is approx 2% of the time-consumed for computing their best fit (which strongly depends on the a coefficient).

The implementation of RM may result with a better EFF coefficient, but requires more computations, because of the fact that every S is processed 4 times (0, 90, 180 and 270 rotating angle). Investigations showed, that depending on type of S (its geometrical shape), RM can highly increase the EFF coefficient. Table 2 shows the results - the averaged values of EFF computed with (2) for RM (turned on) and RM (turned off) for the same data sets.

6 Conclusions

The proposed QKBMR algorithm seems to be promising for sets of shapes with many objects of the same categories, as in the most industrial nesting processes. Thus, the proposed approach can strongly decrease the time of nesting.

The further work in the area of nesting systems will be concentrated on finding more effective shape pairing algorithms as it is the most time consuming module.

References