4-22-2007

Nesting System with Quantization and Knowledge Base Applied

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

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

Nesting System with Quantization and Knowledge Base Applied

Leszek Koszalka and Grzegorz Chmaj
Chair of Systems and Computer Networks, Wroclaw University of Technology, Wroclaw, Poland
E-mail: leszek.koszalka@pwr.wroc.pl

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 the defined set of shapes. Implementations of the problem can vary with different

Second International Conference on Systems (ICONS'07)
0-7695-2807-4/07 $20.00 © 2007 IEEE

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 – time, usage area, etc.

No matter what implementation, the scheme of nesting is always as in Fig. 1.

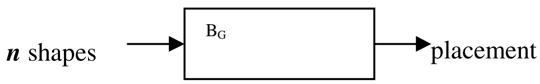

Figure 1. General nesting scheme.

An example of the nesting– with finite rectangle as a region and known shape queue - is shown in Fig. 2.

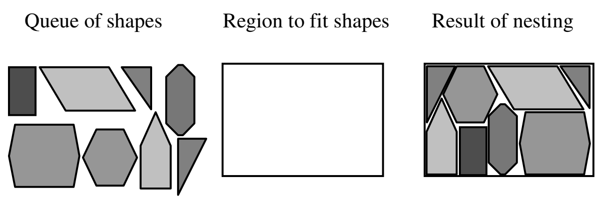

Figure 2. Example of nesting

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.

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

$$G = \{e_n \in P{:}(x_n,y_n) \vee A{:}(x_{n1},y_{n1},\ x_{n2},y_{n2},\ r_n)\}.$$

Region – an area of potential placement of shapes, here, a rectangular with infinite length and width. Region containing points $P(x_n,y_n)$ where $x_n,y_n \in R$ is denoted as

$$R = \{P(x_n,y_n){:}\ -\infty > x_n > +\infty \wedge -\infty > y_n > +\infty\}.$$

Quantum – a discrete part of geometric area, it is characterized by size and logical binary state assigned.

Quantum size – denoted as a, describes the size of quantum, Quantum having a size of square side length in geometric interpretation and (x,y) discrete position in mesh M is denoted as $Q_{(a)}(x,y)=\{0,1\}$.

Intersection function – denoted as $INT(A,B)$ – returns logical true/false result: if area A and area B intersect, logical 1 is returned, otherwise it 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 the following formula:

$$B_G=\{P(x_b,y_b){:}\ x_b,y_b \in R \wedge x_b \in <x_m,x_n> \wedge y_b \in <y_m,y_n>{:}\ (\forall P(x_p,y_p) \in G,\ x_m \le x_p \wedge x_n \ge x_p \wedge y_m \le y_p \wedge y_n \ge y_p)\}.$$

Mesh – R is divided (Fig. 3) into $Q_{(a)}$ parts, that composes two dimensional mesh on nesting area. Mesh having quantum size of a is denoted as $M_{(a)}$ and has infinite length and width:

$$M_{(a)}=\{Q_{(a)}(x_n,y_n) \in R{:}\ -\infty > x_n > +\infty \wedge -\infty > y_n > +\infty\}$$

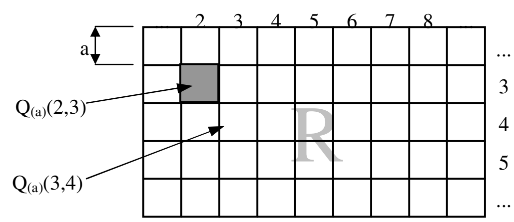

Figure 3. Region divided into $M_{(a)}$ mesh.

Quantization process (QP) – process of changing geometry of a shape into its binary representation. QP is performed using the quantization function:

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

Binary shape (S)– 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.

$$S(w,h) = \{Q_{(a)}(x_n,y_n) \in B_G : Q_{(a)}(x_n,y_n) = INT(Q_{(a)}(x_n,y_n),G) \wedge x_n \in <0,w>,\ y_n \in <0,h>;\ B_G(x_b,y_b){:}\ x_b=a*w,\ y_b=a*h\}$$

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

AND(S_1,S_2), OR(S_1,S_2), XOR(S_1,S_2), NEG(S),– binary logical operation on proper bit sequences.

ROR1(S) – binary rotating right of binary sequence; it rotates by one position:

$$ROR1(S_1) = S_2\text{: } S[i]=S[i+1],\ S[n]=S[1];\ i\in<2,n>,$$

where S_1 is S before *ROR1*, S_2 is S after *ROR1*, i is a position in a sequence, n is amount of bits in sequence.
ROR(S,t) – binary rotating right of binary sequence (S shape) by t bits; it performs *ROR1* operation t times.
ROL1(S) – binary rotating left of binary sequence; it rotates by one position:

$$ROL1(S_1)=S_2\text{: } S[i]=S[i-1],\ S[1]=S[n];\ i\in<2,n>$$

where S_1 is S before *ROL1*, S_2 is S after *ROL1*, i is a position in a sequence, n is amount of bits in sequence.
ROL(S,t) – binary rotating left of binary sequence (S shape) by t bits. Performs *ROL1* operation t times.
Knowledge Base Element (**KBE**) – a data set with information about two S and their best combination C.

In the paper, the following assumptions are taken:

- R has no specified shape,
- the shape queue is known and the system is able to preprocess it before starting *QP* and nesting,
- all shapes are quantized using the same a in $Q_{(a)}$.
- the shapes can be rotated by angle

$$\gamma = \frac{k\pi}{2},\quad k \in \{0,1,2,3\}.$$

3. Nesting system

The QKBMR nesting system is composed of the following elements:

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 the way of processing *GQ*.
QP – *Quantization Process* – converting *GQ* to *BSS*. Parameter a determines *QP* precision (Fig. 4)
BSS – *Binary Shape Set* – set of *BSSC* number of *S* elements. *BSS* is the result of *QP*.
SP - *Shape pairing* – module responsible for finding the best co-placement of S_a and S_b. There are available options: (i) with rotating, (ii) without rotating.
KB – *Knowledge Base* with a set of the *KBC* number of *KBE-s*, i.e. $KB = \{KBE_1, KBE_2, ..., KBE_{KBC}\}$. *KB* expands self during nesting. Moreover, KB can use knowledge acquired from external sources..
NM - *Nesting Module* – the main part of the system that manages *S* elements and finally places them in *R*.
$M_{(a)}$ parameter is used to describe mesh in *R*.
DeQP – *De-Quantization Process* – converting *S* elements into *G* elements. *DeQP* is an optional module.

4. QKBMR algorithm

The core of the proposed nesting system is *QKBMR* regarded as a set of algorithms. In this section the detailed description of ideas of QKBMR are given.
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 by (1):

$$T_{QP} = kWHla \tag{1}$$

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 $B_G(x_m,y_m)$.
2. Divide $B_G(x_m,y_m)$ into $Q_a(x_a,y_b)$ elements. The result is a mesh of Q_a elements with w amount of Q_a in each row of mesh and h amount of Q_a elements in each column.
3. Assign state to all $Q_a(x_a,y_b)$ elements: $\forall\ Q_{(a)}(x_c,y_d) \in B_G$, $Q_{(a)}(x_c,y_d) = INT(Q_{(a)}(x_c,y_d),G)$, $c \in <0,w)$, $d \in <0,h)$.

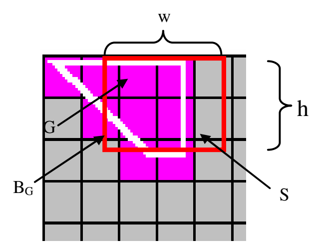

Figure 4. Shape *G* and its quantized representation S(3,2)=111101110011

Normalization Process. *NP* is a process of extending two *S-es* according to their properties. 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:

1. For two received objects: $S_1(x_1,y_1)$ and $S_2(x_2,y_2)$ compute: $mx=x_1+2*x_2$, $my=y_1+2*y_2$,
2. Normalize S_1:
   2.1. Add $c=mx-x_1$ columns to S_1,
   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. $t=c/2+r/2*y_1$; Perform *ROR* with t positions.

3. Normalize S_2:

3.1. Add $c=mx-x_2$ columns to S_2,

3.2. $x_2=mx$,

3.3. Add $r=mx-x_2$ rows to S_2,

3.4. $y_2=my$.

Table 1 shows an example of using *NP*.

Denormalization Process. *DP* performs removing bordering rows and columns containing only zeroes. The rule states: $DP(NP(S))=S$. $DP(S(x,y))$. It is performed in the following way:

1. $C_T=(2^{x+1}-1)*2^{(x+1)*y}$, If $AND(S,C_T)=0$ - perform removing first $x+1$ bits from binary representation.
2. $C_B=NEG(2^{(x+1)(y+1)}-2^{x+1})$, If $AND(S,C_B)=0$ – perform removing last $x+1$ bits from binary representation.
3. $C_L=C_{x+1}$, $C_k=C_{k-1}*2^{x+1}+C_0$, $C_0=2^x$; If $AND(S,C_L)=0$, perform removing bits from positions *p*, for whose *(p modulo (x+1)=1)* formula is satisfied..
4. $C_R= NEG(AND(C_1, C_2, C_3 ,..., C_{(y+1)}))$, where $C_k = (2^{k(x+1)}-2)*2^{(y+1-k)(x+1)}+2^{(y+1-k)(x+1)}-1$. If $AND(S,C_R)=0$, perform removing bits from positions p, for whose *p modulo (x+1)=0* formula is satisfied.

Table 1. Example of nomalization

Before *NP*		After *NP*	
S_1	S_2	S_1	S_2
1111	11	00000000	11000000
0111	01	00000000	01000000
0011		00111100	00000000
		00011100	00000000
		00001100	00000000
		00000000	00000000
		00000000	00000000

Shape Pairing. *SP* tries to find the best coexistent S_2 for a given S_1, where S_1 is always the first shape 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:

$$EFF =1/((S_{1w}+1)*(S_{1h}+1)) \quad (2)$$

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). S_2 is a moving object.
2. $GLEFF=0$
3. For every unpaired *S* from *BSS* do:
   - 3.1. Normalize $S_1(x_1,y_1)$ with $S_2(x_2,y_2)$
   - 3.2. $S_R=ROR1(S_2)$
   - 3.3. If $S_2[1]=0 \vee S_R[1]=1$, go to 4.
   - 3.4. If $AND(S_2, C)=0 \vee AND(S_R,C)\neq 0$, where $C=C_{x2+1}$, $C_k=C_{k-1}*2^{x1+1}+C_0$, $C_0=2^{x1}$, perform $ROR(S_2,x_2+1)$, else perform $ROR(S_2,1)$.
   - 3.5. $M=XOR(XOR(OR(S_1,S_2),S_1),S_2)$
   - 3.6. If $M\neq 0$, go to 3.2
   - 3.7. Compute $EFF (OR(S_1,S_2))$
   - 3.8. If $GLEFF<EFF$, then $GLEFF=EFF$ and $BF=OR(S_1,S_2)$
4. *BF* contains best found co-placement of S_1 and S_2, but *GLEFF* contains *EFF* for this pair.

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

Acquiring and Utilizing Knowledge. Every *S* is described by some bits and *(x, 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* 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 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. $S_R=ROR1(S_2)$

Second International Conference on Systems (ICONS'07)
0-7695-2807-4/07 $20.00 © 2007 IEEE

3.4. If $S_2[1]=0 \vee S_R[1]=1$, go to 4.

3.5. If $AND(S_2,C)=0 \vee AND(S_R,C)\neq 0$, where $C=C_{x2+1}$, $C_k=C_{k-1}*2^{x1+1}+C_0$, $C_0=2^{x1}$, perform $ROR(S_2,x_2+1)$, else perform $ROR(S_2,1)$.

3.6. $M=XOR(XOR(OR(S_1,S_2),S_1),S_2)$

3.7. If $M\neq 0$, 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.

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 $\gamma=k\pi/2,\ k \in \{0,1,2,3\}$, 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_2 using $\gamma=k\pi/2$ angle
   3.3. Normalize $S_1(x_1,y_1)$ with $S_2(x_2,y_2)$
   3.4. $S_R=ROR1(S_2)$
   3.5. If $S_2[1]=0 \vee S_R[1]=1$, go to 3.11.
   3.6. If $AND(S_2,C)=0 \vee AND(S_R,C)\neq 0$, where $C=C_{x2+1}$, $C_k=C_{k-1}*2^{x1+1}+C_0$, $C_0=2^{x1}$, perform $ROR(S_2,x_2+1)$, else perform $ROR(S_2,1)$.
   3.7. $M=XOR(XOR(OR(S_1,S_2),S_1),S_2)$
   3.8. If $M\neq 0$, 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.

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 $M_{(a)}$) 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 will be considered in the future work.

Two additional mechanisms. GQPP and DeQP are operating on shapes. For some *G* sets and *SP* pairing algorithms, nesting process may be speed up. *GQPP* performs sorting or other methods for changing order of G-es within *GQ*.. *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. 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*, 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. The relationship between *a* and *QP,* found on the basis of results of simulations is shown in Fig. 5.

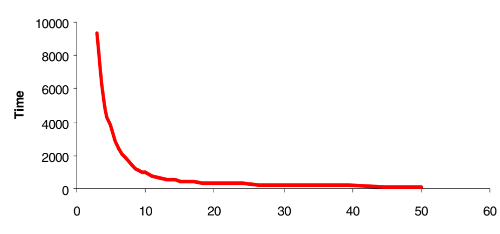

Figure 5. Influence of *a* coefficient on time of *QP.*

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. When *G* is transformed into a larger set of bits, *SP* module has more data to process. This is an exponential relation (Fig. 6).

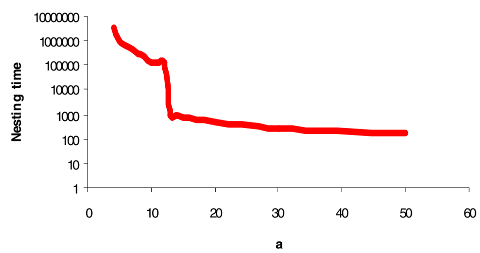

Figure 6. Influence of *a* coefficient to the completion time of nesting .

As is visible on the graph, there exists 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.. 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).

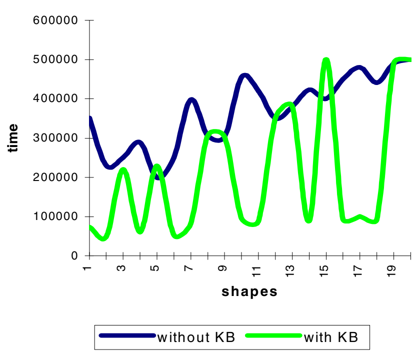

Figure. 7. Knowledge base and time of nesting.

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

6. Conclusions

Nesting process is very time-consuming, thus it is worth to make attempt for shortening the time by recording already computed best-fits for shapes.

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 (i) finding more effective shape pairing algorithms as it is the most time consuming module (ii) developing system by implementing module with data base designed along with ideas given in [8] and [9], and (iii) investigating system efficiency in relation to shape and size of nesting region as well as the distinct categories of shapes to be located.

7. References

[1] E. Burke and G. Kendall, "Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem", *Scientific Report, The University of Nottingham* , 2001.

[2] B. Nielsen B. and A. Odgaard, "Fast Neighborhood Search for the Nesting Problem", *Scientific Report, University of Copenhagen*, 2003.

[3] A. Gomes and J. Oliveira, "A GRASP Approach to the Nesting Algorithm", *Scientific. Report, Porto*, Portugal, 2001.

[4] T. Rintala, "A Genetic Approach to Nesting Problem", *2nd Nordic Workshop on Genetic Algorithms and their Applications*, 1996.

[5] A. Albano and G. Sappupo, "Optimal Allocation of Two Dimensional Shapes Using Heuristic Search Methods", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 10, 2002.

[6] J. Oliveira and J. Ferreira, "Algorithms for Nesting Problems", *Applied Simulated Annealing*, 1993.

[7] M. Adamowicz .and A. Albano, "Nesting Two-Dimensional Shapes in Rectangular Modules", *Computer Aided Design,* vol. 8, 1976.

[8] Pozniak-Koszalka I., Relational data base in Sybase environment. Modeling, Designing, Applications, Wyd. Pol. Wroc., Wroclaw,2004 (in Polish).

[9] M. Wozniak, Lecture Notes in Computer Science, 2005.

IEEE COMPUTER SOCIETY