

碩士學位論文

비트연산을 이용한  
최적배치 연구



濟州大學校 大學院

機 械 工 學 科

金 永 根

2006年 6月

# 비트연산을 이용한 최적배치 연구

指導教授 趙慶鎬

金永根

이 論文을 工學 碩士學位 論文으로 提出함



金永根의 工學 碩士學位 論文을 認准함

審査委員長

한민기

委員

조경호

委員

정동원

濟州大學校 大學院

2006年 6月

A study on optimal nesting  
using Bit operations

Young-Gun Kim

(Supervised by professor Kyung-Ho Cho)



A thesis submitted in partial fulfillment of the requirement  
for the degree of Master of Engineering

Department of Mechanical Engineering  
GRADUATE SCHOOL  
CHEJU NATIONAL UNIVERSITY

2006. 6

# 목 차

I. 서론 .....	1
1. 연구 배경 .....	1
2. 연구 동향 .....	2
3. 연구 목적 .....	5
II. 픽셀을 기반으로 한 네스팅 .....	6
1. 개요 .....	6
2. 픽셀모델을 이용한 형상표현 .....	7
3. 픽셀모델을 이용한 네스팅 .....	9
4. 픽셀을 기반으로 한 네스팅의 장단점 .....	11
III. 비트연산을 이용한 네스팅 .....	12
1. 개요 .....	12
2. 워드모델을 이용한 형상표현 .....	13
2.1 1D 워드모델 .....	13
2.2 2D 워드모델 .....	16
2.3 Shifted 워드모델 .....	18

3. 비트네스팅 연산 .....	22
3.1 부재 배치 비트연산 .....	23
3.2 부재 삭제 비트연산 .....	27
3.3 중첩 검사 비트연산 .....	31
4. 쿼드트리를 이용한 간극탐색 .....	37
4.1 쿼드트리 개요 .....	39
4.2 쿼드트리 형성과정 .....	41
4.3 간극탐색 .....	44
IV. 적용사례 .....	47
V. 결론 .....	55
VI. 참고 문헌 .....	56



# List of Figures

Fig. 1 No Fit Polygon .....	2
Fig. 2 Allocation region and space available .....	3
Fig. 3 Representation of pixel model .....	3
Fig. 4 Pixel models of raw sheet and part .....	7
Fig. 5 Reference shape representation by pixel model .....	8
Fig. 6 Pixel based part allocation .....	9
Fig. 7 Pixel based overlap test .....	10
Fig. 8 Pixel based part de-allocation .....	10
Fig. 9 Example of bit operations .....	12
Fig. 10 Pixel and 1D word models of part .....	15
Fig. 11 1D word models of waste space .....	15
Fig. 12 Pixel and 2D word models of part .....	17
Fig. 13 Comparison of the grid size .....	17
Fig. 14 Shifting bit / 1D word patterns .....	19
Fig. 15 Shifted pixel and 1D word model .....	19
Fig. 16 Shifting bit / 2D word pattern .....	20
Fig. 17 Shifted pixel and 2D word model .....	20
Fig. 18 Shifted 1D word models and 2D word models of part .....	21

Fig. 19 1D word based part allocation .....	25
Fig. 20 2D word based part allocation .....	26
Fig. 21 1D word based part de-allocation .....	29
Fig. 22 2D word based part de-allocation .....	30
Fig. 23 1D word based overlap test(non-overlapped) .....	33
Fig. 24 1D word based overlap test(overlapped) .....	34
Fig. 25 2D word based overlap test(non-overlapped) .....	35
Fig. 26 2D word based overlap test(overlapped) .....	36
Fig. 27 Gap searching .....	38
Fig. 28 Quad-tree data structure .....	40
Fig. 29 Quad-tree level .....	43
Fig. 30 Gap searching and filling .....	46
Fig. 31 Schematic flow diagram of the nesting .....	47
Fig. 32 Comparison of the grid size between the pixel model and the word model .....	48
Fig. 33 Gap searching using quad-tree and part allocation .....	49
Fig. 34 Nesting samples of aircraft parts .....	52
Fig. 35 Nesting samples of heavy machinery parts .....	53
Fig. 36 Nesting samples of shipbuilding parts .....	54

# List of Table

Table 1 Bit Operators .....	22
Table 2 Nesting results of aircraft parts .....	52
Table 3 Nesting results of heavy machinery parts .....	53
Table 4 Nesting results of shipbuilding parts .....	54





# Summary

Recently, as the amount of production is vastly increased, the demand of the optimal nesting algorithm is increased. The optimal nesting helps reducing the production cost and saving the processing time. The several optimal nesting methods are developed.

Among them, vector-based nesting in which the patterns with arbitrary shape are approximated as straight lines is mainly used at present. However, the process to find the intersection points among the line segments is required to check the overlaps of the patterns in the vector-based nesting method. Therefore, pixel-based nesting is proposed to avoid the process to find the intersection points. In pixel-based nesting, all the target patterns are converted as pixels. Then, the allocation, the de-allocation, and the overlap test of the target patterns are performed in pixel basis. However, the processing time and the operation numbers are highly proportional to the pixel resolutions.

In this thesis, a new nesting method based upon the 'word representation' of parts and raw sheets is proposed to improve the performance of pixel-based nesting. In this new representation, the pixel patterns of parts and sheets are converted to proper word-sets. And all the usual nesting processes such as part allocations, de-allocations, overlap testings and etc. can be performed through the 'bit operations' between words of the parts

and sheets. The proposed method shows some possibilities to overcome the shortcomings of the traditional pixel-based nesting. In addition, a new efficient gap searching method based upon the quad-tree is presented to locate proper allocation positions of the parts.



# I. 서론

## 1. 연구 배경

최적배치(optimal nesting)는 다양한 부재(part)를 정해진 크기의 원자재(sheet)에 배열하는 방법으로 주로 2차원 형상의 절단응용 분야인 조선, 의류, 블랭킹 금형, 종이, 유리, 목재 등의 제조업 분야에서 널리 쓰인다. 일반적으로 배치 작업은 원자재의 버림률(waste or scrap ratio)을 최소화하는 작업으로 인식되는 경우가 많지만 설계, 생산 계획, 자재 관리, 절단 공정 등과 같이 생산 단계 전 분야와 관련이 되며 이를 함께 고려하여야 생산성을 향상시킬 수 있다. 과거 이러한 최적배치는 숙련된 작업자들의 수작업으로 이루어졌다. 하지만 작업자의 숙련도에 따른 배치효율문제나 인건비 부담뿐만 아니라, 작업시간이 길어짐에 따른 공정의 병목현상 등 많은 문제점이 발생하였다.

이와 같은 문제점을 해결하기 위해 최적배치에 관해 많이 연구되고 있지만, 배치문제의 성격상 NP-complete에 속하여 다루기가 쉽지 않고 적용되는 분야에 따라 요구사항과 구속조건이 변하여 이로 인해 배치전략이 달라지기 때문에 효과적인 배치방법을 찾기가 매우 힘들어 진다. 또한 현대사회가 급속도로 발전함에 따라 생산규모가 커지고 자동화 공정이 확대되어 효과적인 최적배치 알고리즘의 필요성이 크게 대두 되고 있는 실정이다. 특히 사용 원자재가 고가이거나, 원자재 사용량이 증가될수록 원가를 절감하기 위해서 부재의 최적배치를 위한 프로그램 필요성이 증대되고 있다. 따라서 효율적인 배치 알고리즘의 개발 및 시스템 구현 요구가 계속 되어지고 있다.

## 2. 연구 동향

최적배치 알고리즘에 관한 연구는 다양한 방법으로 진행되어 왔으며 형상을 효율적으로 배치하면서 처리시간을 줄이기 위해 각각 고유한 형상의 데이터를 여러 가지 방법으로 정의하여 최적배치 알고리즘에 적용하였다.

1970년대 Adamowicz와 Albano[1]는 형상을 Fig. 1(a)와 같이 기준점(reference point)를 갖는 다각형으로만 정의하여 Fig. 1(b)와 같이 고정형상 A에 대해 겹치지 않고 최대한 근접하면서 이동형상 B의 기준점이 지나는 자취인 NFP(No Fit Polygon)을 정의하여 이 NFP 위의 점을 이동부재 B의 기준점이 놓여질 위치로 선택하여 형상을 배치하였다. 하지만 이 방법은 형상 B의 모양에 따라 NFP를 구하는 것이 매우 어렵고, 형상 B의 고정된 방향에서만 NFP를 구할 수 있으므로 이동부재의 방향(orientation)을 미리 예측해야 하는 어려움이 있다.

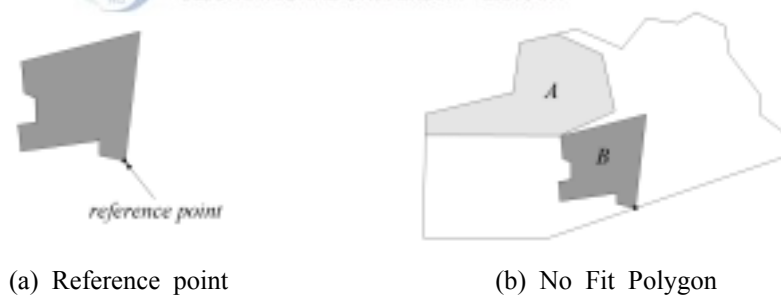
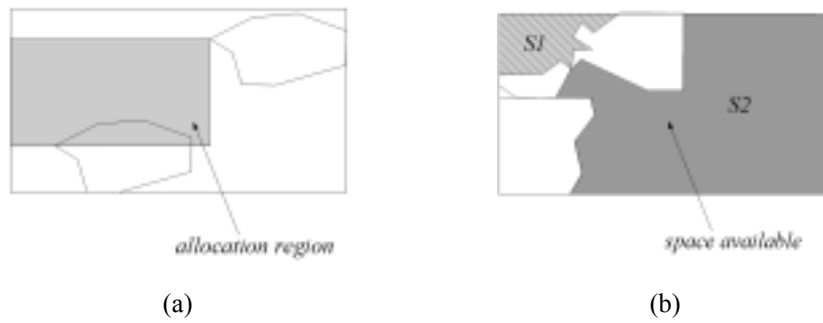


Fig. 1 No Fit Polygon

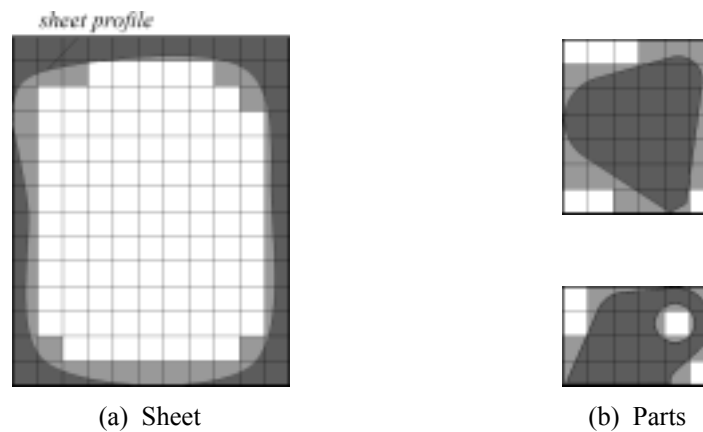
Albano와 Sapuppo[2]는 한 형상이 이미 고정된 형상들과 겹치지 않으면서 고정될 수 있는 모든 가능한 위치로 움직여 나갈 때 형상의 기준점이 지나는 영역을 배치영역(Fig. 2(a) 참조)으로 정의하여 이 배치영역 내에서 버림이 가장 작은 경우를 택하여 배치하는 알고리즘을 개발하였다. 이 알고리즘에서는 배치영역이

항상 형상(profile)의 오른쪽에만 존재할 수 있도록 하여 Fig. 2(b)와 같이  $S1$  영역에는 어떤 형상도 배치되지 못하는 단점이 있다.



**Fig. 2** Allocation region and space available

1983년에는 모의 어닐링(simulated annealing)알고리즘을 이용한 최적화 기법이 Kirkpatrick 등에 의해 소개된 이래 여러 응용 분야에서 적용되었다.[3, 4] 특히 1990년 방기범[5]과 1993년 조경호[6, 7]는 임의 형상의 배치에서 픽셀 표현법(pixel representation)으로 접근하였다.



**Fig. 3** Representation of pixel model

Fig. 3과 같이 픽셀 표현법을 근간으로 하는 배치는 원장 경계가 불규칙하고 원장 내부의 결함을 표현하는 것이 용이하여 다수 원장에의 동시배치, 간극탐색, 중첩검사시 수학적 연산 불필요성 등의 장점이 있지만 원장과 부재의 정확한 경계를 모사하기 위해서는 픽셀수가 기하급수적으로 증가하여 메모리 및 처리시간이 증가하는 문제점이 있다.

그리고 1992년 Ismail과 Hon[8]은 유전자 알고리즘을 이용하여 2개의 부재를 배치하는 방안을 제시하였고, 1993년에는 Fujita[9]등에 의해 유전자 알고리즘을 형상배치에 적용하는 방법이 제시되었다. 유전자 알고리즘에서는 효율적인 배치를 위해 형상을 생명체의 염색체(chromosome)와 나열 구조가 유사한 문자(string) 구조로 표현하여 문자열이 잘 맞물리는 쌍을 찾아내는 클러스터링(clustering)단계와 클러스터의 배치 단계를 거치는 방법을 이용한다.



### 3. 연구 목적

본 연구에서는 픽셀을 기반으로 한 네스팅(pixel-based nesting)의 단점을 일부 보완할 수 있는 ‘비트연산을 이용한 최적배치’ 라는 새로운 방법을 개발하였다.

픽셀을 기반으로 한 네스팅은 배치가 가능한 공간과 불가능한 공간을 0과 1만의 정수로 표현하여 경계가 불규칙한 원자재나 원자재 내부의 결함을 표현하는데 있어 용이하고 큰 부재의 내부 홀에 작은 부재를 배치하는 경우 등에 있어서 별도의 추가 알고리즘 필요 없이 구현이 가능하다는 장점이 있다. 또한 배치 중간 과정에 수없이 수행되는 간극탐색과 중첩검사 작업을 추가적인 수학적 연산을 도입하지 않고도 단순히 원자재의 픽셀값만을 평가하는 작업으로 대신할 수 있다는 장점이 있다. 하지만 원자재와 부재의 정확한 경계를 모사하기 위해서는 픽셀의 분해능을 증가시켜야 하고 이 경우 처리될 픽셀의 수가 기하급수적으로 증가하여 처리시간 및 연산횟수가 증가하는 문제점이 있다.

이를 위해 본 연구에서는 픽셀을 기반으로 한 네스팅에 사용되는 픽셀모델(pixel model)을 이진값으로 변환한 비트(bit)들의 묶음으로 구성된 워드모델(word model)로 표현하여 단순화 시켰으며, 워드모델로 표현된 부재들은 서로 대응되는 워드간의 비트연산에 의해 배치, 삭제, 중첩검사 등을 수행하도록 하였다.

또한 배치 중간 과정에 수행되는 간극탐색을 비교적 효율적이고 간편한 계층적 자료구조의 하나인 쿼드트리를 이용한 간극 탐색법을 개발하였다.

## II. 픽셀을 기반으로 한 네스팅

### 1. 개요

대부분의 최적배치 프로그램에서는 임의의 형상을 직선선분들로 근사화시킨 다각형 근사법으로 형상을 표현한 후 배치하는 방법이 주를 이루고 있다. 다각형 근사법은 이미 배치된 형상들과 중첩을 피하기 위해서 형상을 이루는 직선선분들 간의 교차점을 구하는 과정이 필요하다. 이를 위해 NFP 개념을 이용하지만 아래와 같은 단점을 가지고 있다.

- 형상의 모양에 따라 NFP를 구하는 과정이 복잡함
- 형상의 고정된 방향에서만 NFP를 구할수 있으므로 적당한 방향을 미리 예측해야 함
- 내부 홀이 있는 형상에 대해서는 특별히 고안된 알고리즘이 필요함
- 형상간 중첩을 피하기 위해서 형상을 이루는 꼭선끼리의 교차점을 구하는 과정이 필요함

특히 형상이 복잡하고 많은 형상을 배치시킬 때에는 교차점 계산과정이 복잡하고 연산횟수가 증가함에 따라 처리시간이 증가된다. 이러한 교차점을 구하지 않고 배치를 수행하기 위해서 픽셀을 기반으로 한 네스팅이 제안되었다. 이 방법은 중첩검사시 교차점을 구하지 않고 모든 형상들을 픽셀화 시켜 픽셀의 정수값을 비교하여 중첩여부를 판단하므로 기존의 알고리즘에서의 복잡성을 피하고 있다.



## 2. 픽셀 모델을 이용한 형상표현

픽셀을 기반으로 한 네스팅에서 원자재와 부재의 형상은 지정된 분해능 (resolution)으로 나누어진 픽셀들로 표현되고 이런 픽셀들로 근사화된 형상을 픽셀모델이라 한다. 즉 픽셀모델은 원자재의 경우 Fig. 4(a)와 같이 배치 불가능한 영역의 픽셀값을 1로, 배치 가능한 영역의 픽셀값을 0으로 초기화 시키고, 부재의 경우는 Fig. 4(b)와 같이 형상 내부의 픽셀값을 1로, 형상 외부(내부홀 포함)의 픽셀값을 0으로 초기화 시켜 각각 형상을 표현한다.

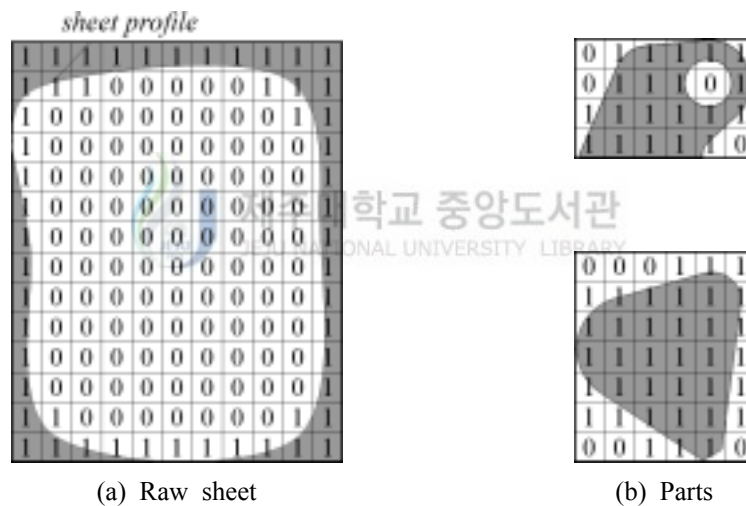
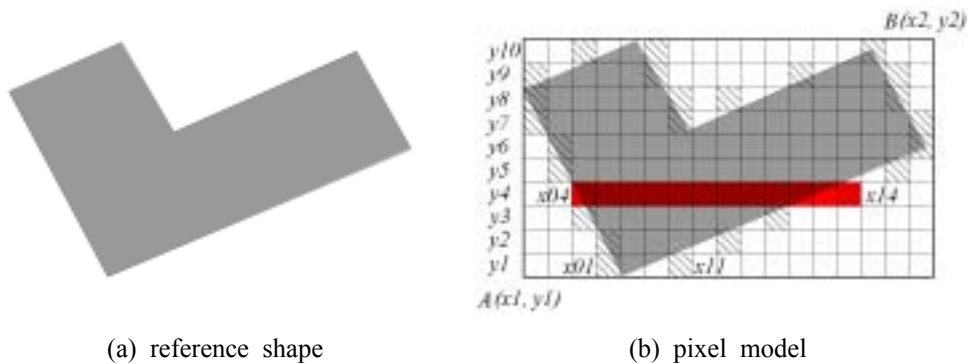


Fig. 4 Pixel models of raw sheet and part

하지만 위와 같이 픽셀 모델을 생성하기 위해서는 일반적으로 많은 교차계산을 수반하게 되는데, 1993년 조경호[6]는 그래픽 유틸리티에서 제공하는 픽셀 정보 참조기능을 이용하여 위 방법에 비해 간결하면서도 신뢰도 높은 픽셀 모델 생성법을 제시하였다. 아래는 이에 대한 방법을 간략하게 기술한 것이다.

그래픽 화면의 고유 분해능을 고려하여 적절한 축적으로 Fig. 5(a)와 같이 기준형상을 표현한 후, 모든 형상 경계를 포함하는 최소의 직사각형의 양단 끝점을  $A(x1, y1)$ ,  $B(x2, y2)$ 이라 지정하면 `rectread(x1, y1, x2, y2, parray)`라는 그래픽 함수를 이용하여 지정된 직사각형 내의 모든 화소의 16 bit color index 정보를 `parray`라는 배열을 통해 참조할 수 있다.

픽셀모델의 픽셀정보의 저장은 Fig. 5(b)의 □□□□□ 부분만 저장하도록 하여 기억공간을 효율적으로 사용하도록 하였다. 즉 그림의 ■로 표시된 영역(grid strip)의 픽셀정보 저장은 시작점( $x04$ ), 끝점( $x14$ ) 그리고  $y$ 좌표( $y4$ )만 알면 된다. 이 작업은 지정된 직사각형의 최하단부터 상단까지 일정  $y$ 좌표값에서  $x1$ 부터  $x2$ 까지 `parray`값을 참조하는 간단한 알고리즘으로 각 strip의 양 끝점을 추출한다. 이러한 픽셀정보의 저장방법은 모든 경계픽셀정보를 저장하는 일반적인 방법과 비교하여 볼 때 훨씬 적은 기억 공간을 사용하게 되며 본 연구에서는 이 방법을 도입하여 워드모델을 생성하였다.



**Fig. 5** Reference shape representation by pixel model

### 3. 픽셀 모델을 이용한 네스팅

픽셀을 기반으로 한 네스팅에서는 배치되어야 할 모든 형상에 대해서 앞절에서와 같이 기준픽셀모델을 완성한 후 이미 픽셀모델로 변환된 원자재에 형상을 배치, 중첩검사, 삭제를 수행한다.

이때 부재 형상의 배치는 Fig. 6과 같이 대응되는 원자재 픽셀모델의 각 픽셀값을 1씩 증가시키는 일에 해당한다. 즉 원자재 임의의 픽셀에 대한 픽셀값은 그곳에 배치된 형상의 개수를 의미하며 이를 이용하면 중첩여부에 대해서도 판단할 수 있다. 즉 Fig. 7과 같이 픽셀값이 1이하이면 중첩이 발생하지 않는 경우로 판단하고, 픽셀값이 2이상이면 형상간의 중첩이 발생한 경우로 판단하여 최종배치에서는 중첩이 나타나지 않도록 한다.

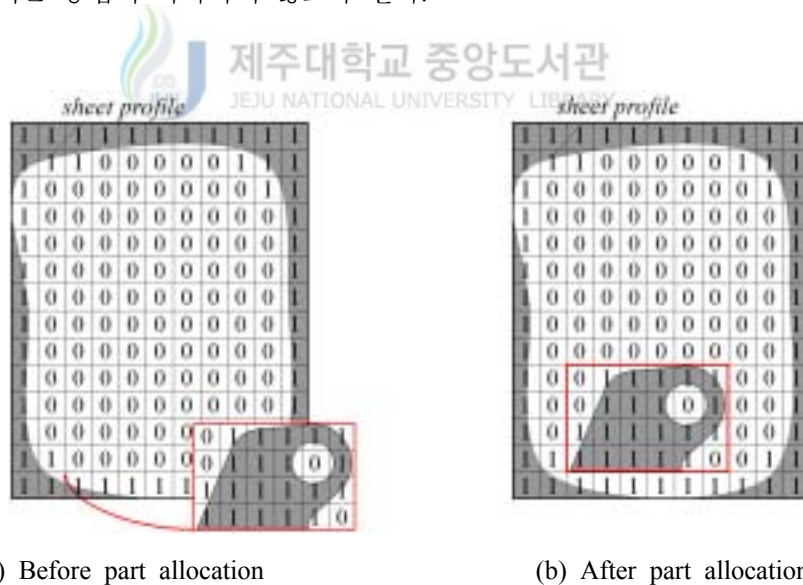


Fig. 6 Pixel based part allocation

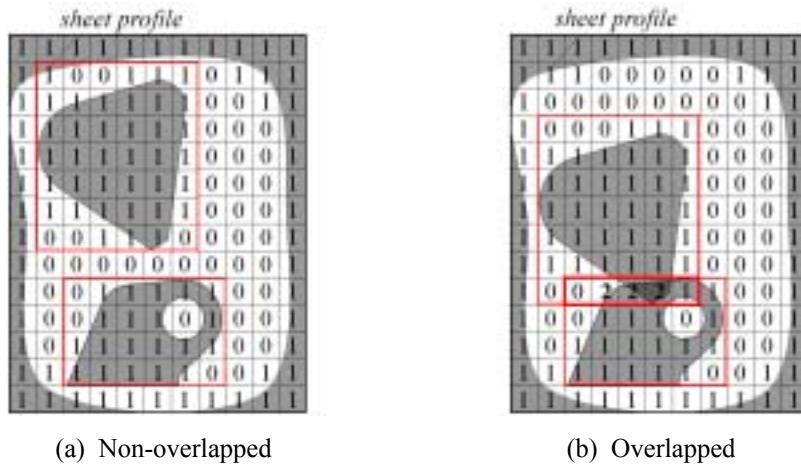


Fig. 7 Pixel based overlap test

부재 형상의 삭제는 Fig. 8과 같이 대응되는 원자재 픽셀모델의 각 픽셀의 값을 1씩 감소시키는 작업이다.

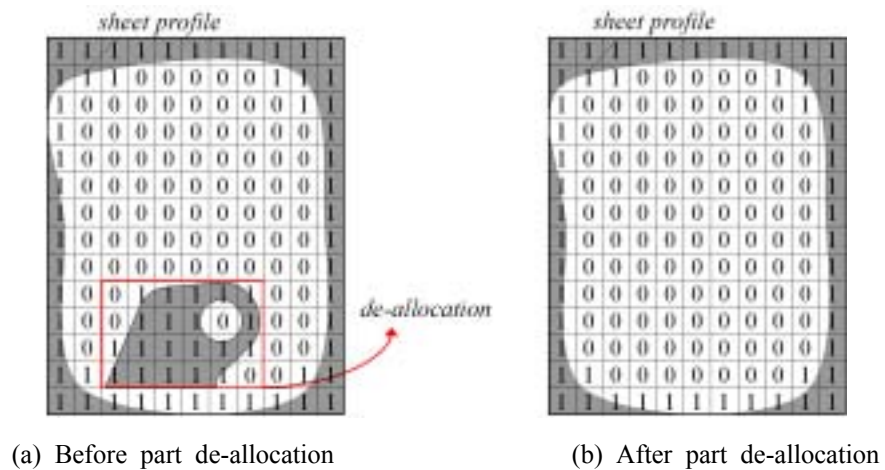


Fig. 8 Pixel based part de-allocation

#### 4. 픽셀을 기반으로 한 네스팅의 장단점

픽셀을 기반으로 한 네스팅은 앞절에서 언급된 바와 같이 원자재와 부재를 배치 가능한 공간과 불가능한 공간을 0과 1이상의 정수만으로 표현 가능하기 때문에 아래와 같은 장점을 가진다.

- 경계가 불규칙하거나 내부 결함이 있는 원자재의 표현이 용이함
- 내부 홀을 가진 부재에 별도의 추가 알고리즘 필요 없이 작은 부재를 내부 홀에 배치 가능함
- 원자재내 동시 배치가 가능함
- 간극 탐색 및 중첩 검사시 수학적 연산이 불필요함

하지만 원자재와 부재의 정확한 모사를 위해서 픽셀의 분해능을 증가시킬 경우 처리해야 될 픽셀의 수가 기하급수적으로 증가하기 때문에 아래와 같은 단점을 가진다.

- 분해능 증가에 따른 소요메모리 증가
- 간극 탐색 및 중첩 검사시 연산횟수 증가
- 연산횟수 증가에 따른 처리 시간 증가

### III. 비트연산을 이용한 네스팅

#### 1. 개요

앞절에서 언급된 바와 같이 픽셀을 기반으로 한 네스팅의 단점을 보완하기 위해서 본 연구에서는 픽셀을 기반으로 한 네스팅에 비트연산을 도입한 새로운 네스팅 방법을 개발하였다. 이때 비트연산(bit operations)은 비트단위 연산자(bitwise operators)를 이용하여 개개의 비트(bit)를 조작하는 행위를 의미하며, Fig. 9와 같이 대응되는 여러 비트들의 이진값(binary values, 0, 1)들을 동시에 처리하는 기능을 포함하고 있다. 하지만 비트연산을 수행하는 비트단위 연산자는 기억장치의 최소단위인 비트들을 연산대상으로 하기 때문에 픽셀모델을 비트연산이 가능하도록 적절히 변형하여야 한다. 이를 위해 본 연구에서는 형상의 픽셀을 0 또는 1만의 값을 가지는 비트로 변환한 후 일정영역의 비트를 워드에 대응시킨 워드모델을 도입하였다. 또한 부재배치 과정에서 수행되는 간극탐색은 픽셀을 기반으로 한 네스팅을 근간으로 하기 때문에 이를 제시된 방법에서 그대로 사용하는 것은 부적절하여 본 연구에서는 비교적 효율적이고 간편한 계층적 자료구조의 하나인 쿼드트리를 이용한 간극탐색법을 도입하였다.

$$\begin{array}{l} 1010 \ \& \ 1001 \ = \ 1000 \\ 1010 \ \mid \ 1001 \ = \ 1011 \\ 1010 \ \wedge \ 1001 \ = \ 0011 \\ 1010 \ \ll \ 2 \ = \ 1000 \end{array}$$

Fig. 9 Example of bit operations

## 2. 워드모델을 이용한 형상표현

일반적으로 최적배치 프로그램에서는 형상을 효율적으로 저장하고 처리시간을 감소시키기 위해 실제형상을 근사화시킨 근사표현방법을 사용하며 앞절에서 언급한 다각형 근사법과 픽셀 표현법이 대표적인 근사표현방법이라 할 수 있다. 다각형 근사법은 임의의 형상을 직선선분들로 근사화시켜 처리시간을 단축시키기 위함이고, 픽셀 표현법은 형상간 겹침 판별을 효율적으로 하기 위함이다.

본 연구에서는 픽셀모델의 단점을 보완한 워드모델에 대해서 정의하고 다루기로 한다. 워드모델(word model)은 픽셀모델에서 일정수의 픽셀(비트)을 연결하여 하나의 워드에 대응시켜 단순화한 것을 의미하는데, 이때 워드(word)는 컴퓨터에서 연산의 기본단위가 되는 정보의 양을 의미하며, 일정수의 비트로 구성된다. 이러한 워드모델로 표현된 형상은 픽셀모델로 표현된 형상을 보다 단순화 시킬수 있어 부재의 배치, 삭제, 중첩검사시 연산횟수를 감소시킬수 있으며 워드간 비트 연산이 가능하므로 처리시간을 단축시킬수 있다.

### 2.1 1D 워드모델

1D 워드모델은 초기 원자재와 부재의 픽셀모델을 가로방향으로 연속한 일정 길이를 비트의 집합인 워드에 대응시킨 자료구조이며 아래와 같은 과정으로 생성된다.

과정 1)  $n \times m$ 개의 픽셀로 근사화된 원자재와 부재의 각 픽셀을 0과 1로 구성된 비트로 변환한다.(Fig. 10(b) 참조)

과정 2) 워드에 대응시킬 픽셀의 범위 nbit(32, 16, 8 bit)를 결정한다.

본 1D 워드모델에서는 서술의 편의상 가로 16개의 픽셀을 1개 워드에 대응시킨 16 bits/word를 사용한다.

과정 3) 픽셀모델의 좌측하단(0, 0)을 시작점으로 픽셀의 범위만큼 연속한 가로방향의 픽셀을 워드에 대응시킨다. 이때 (n/nbit)의 나머지가 0이 아닐 경우 (nbit-나머지)개수만큼 픽셀을 생성하고 0으로 초기화 시켜 워드에 대응시킨다.(Fig. 10(c) 참조)

과정 4) 과정 3을 픽셀모델의 우측상단점인 (n, m)까지 y방향으로 m번 반복한다.

과정 5) 각각의 워드에 대응시킨 픽셀값(이진수)을 하나의 새로운 정수값으로 변환한다.(Fig. 10(d) 참조) 정수값 변환시 데이터 유형(int, short, long)에 따라 1번 비트가 부호 비트로 사용 되는 경우는 이를 적절히 변형해 주어야 한다. 또한 비트연산자는 정수값을 구성하는 비트들에 적용되므로 double형 데이터나 float형 데이터는 지원하지 않으므로 주의해야 한다.

Fig. 10은 부재가 픽셀모델에서 1D 워드모델로 단순화 되는 과정을 보여주며 원자재의 워드모델도 이와 동일한 과정으로 생성된다. 또한 Fig. 10(b)의 분해능으로 (a)의 부재를 표현하는데 최소 6×4개의 픽셀이 필요한 것에 반해 (d)의 워드모델은 단지 4개의 워드만으로 동일한 부재를 표현하고 있음을 보인다.

하지만 Fig. 11과 같이 1D 워드모델은 (n/nbit)의 나머지가 0이 아닐 경우 (nbit-나머지)개수만큼 *waste space*가 발생한다.



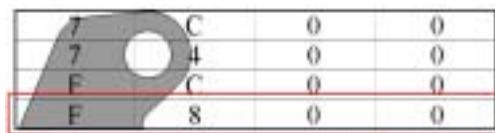


(a) Part

(b) Pixel model



(c) Bit pattern



1D word



(d) 1D word model

Fig. 10 Pixel and 1D word models of part

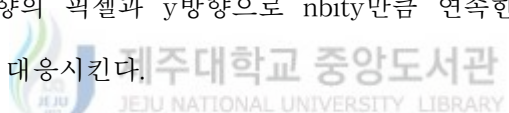


Fig. 11 1D word models of waste space

## 2.2 2D 워드모델

2D 워드모델은 가로방향으로 연속한 일정길이의 픽셀을 연결한 1D 워드모델을 세로방향으로 확장하여 일정영역의 픽셀을 워드에 대응시킨 자료구조이며 1D 워드모델 생성과정에서 다음 과정이 수정된다.

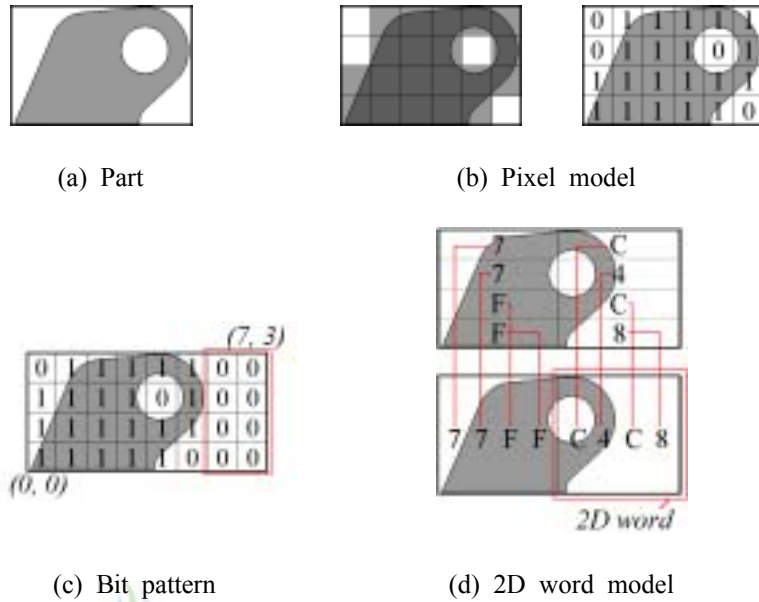
과정 2) 워드에 대응시킬 픽셀의 범위  $nbit(32, 16, 8 \text{ bit})$ 만큼  $x$ 방향으로  $nbitx$ ,  $y$ 방향으로  $nbity$ 로 적절히 결정한다. 이때  $nbit = nbitx \times nbity$ 가 된다. 본 2D 워드모델에서는 서술의 편의상  $4 \times 4$  픽셀을 1개 워드에 대응시킨  $16 \text{ bits/word}$ 를 사용한다.

과정 3) 픽셀모델의 좌측하단  $(0, 0)$ 을 시작점으로  $x$ 방향으로  $nbitx$ 만큼 연속한 가로방향의 픽셀과  $y$ 방향으로  $nbity$ 만큼 연속한 세로방향의 픽셀을 워드에 대응시킨다.  이때  $(n/nbitx)$  혹은  $(m/nbity)$ 의 나머지를  $n_0, m_0$ 라 하고  $n_0, m_0$ 가 0이 아닐 경우 각각  $(nbitx - n_0), (nbity - m_0)$  개수만큼 픽셀을 생성하고 0으로 초기화시켜 워드에 대응시킨다.

과정 4) 과정 3을 픽셀모델의 우측상단점인  $(n, m)$ 까지 반복한다.

Fig. 12는 부재가 픽셀모델에서 2D 워드모델로 단순화되는 과정을 보여주며 원자재의 워드모델도 이와 동일한 과정으로 생성된다.

Fig. 13은 픽셀모델로 표현된 부재 형상 (a)를  $16 \text{ bits/word}$ 로 구성된 1D 워드모델 (b)와 2D 워드모델 (d)를 격자로 표시한 것으로 부재를 표현하는데 픽셀모델은  $6 \times 4$ 개의 픽셀이 필요하고, 1D 워드모델은 4개의 워드가 필요하다. 이에 비해 2D 워드모델은 2개의 워드로 부재를 단순화하여 표현할 수 있음 나타낸다.



제주대학교 중앙도서관  
 Fig. 12 Pixel and 2D word models of part

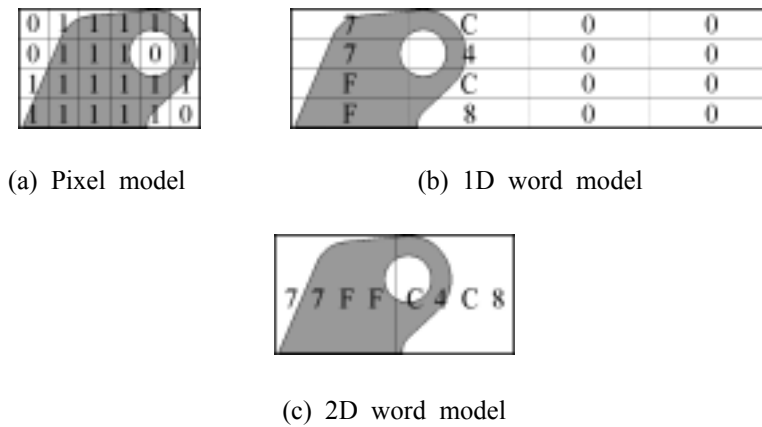


Fig. 13 Comparison of the grid size

이와 같이 워드모델을 픽셀모델과 비교하면 다음과 같은 장점을 가진다.

- 픽셀을 워드로 단순화함에 따라 기존 픽셀모델의 해상도를 증가시켜 원자재와 부재의 형상을 정밀히 묘사하고 최적배치 알고리즘에 적용함에 따라 배치효율을 높일 수 있다.
- 해상도 증가에 따라 기하급수적으로 증가한 픽셀을 워드로 대체하여 단순화함으로써 처리시간을 단축할 수 있다.

### 2.3 Shifted 워드모델

워드모델로 표현된 원자재와 부재에서 네스팅을 수행하는 과정 중에 부재가 임의의 위치와 방향으로 배치되는데 워드모델로 표현된 원자재와 부재는 워드별 비트연산을 수행하기 때문에 이 과정에서 영향을 받게 되는 원자재의 해당 워드들과 부재 워드를 정확히 대응시켜야 한다. 이때 원자재 워드와 부재 워드를 정확히 대응시키기 위해 부재의 기본 워드모델을 x, y 방향으로 픽셀 단위씩 적당히 이동시킨 워드모델이 필요하게 되는데 이를 shifted 워드모델이라 한다.

Shifted 워드모델 생성시 비트단위의 자리이동 연산자인 <<(left shift)와 >>(right shift)가 사용되며 각 연산자는 우측 피연산자에 의해 주어진 자리수만큼 좌측 피연산자의 비트들을 왼쪽 혹은 오른쪽으로 이동시키며 이때 공석이 된 위치에는 0이 채워지고 범위를 넘어간 비트들은 무시된다. 즉 아래 식(1), (2)와 같이 연산이 이루어진다.

$$(10010010) \ll 2 \rightarrow (01001000) \quad (1)$$

$$(10010010) \gg 2 \rightarrow (00100100) \quad (2)$$

Fig. 14는 (a)와 같이 연속된 16개의 비트로 구성된 워드를 1비트 오른쪽으로 이동하였을 때 이와 대응되는 비트패턴 (b)를 나타낸 것이고, (c), (d)는 (a), (b)의 비트들에 대응하는 워드들의 변화를 보인 것이다. 같은 원리로 Fig. 15는 픽셀모델과 1D 워드모델을 2 비트만큼 오른쪽으로 이동시킨 모델을 나타낸 것이다.

Fig. 16은 연속된 4×4개의 비트로 구성된 워드 (a)를 오른쪽으로 2비트 이동시키고 다시 위쪽으로 3비트 이동시켜 이에 대응되는 비트패턴 (b)를 나타낸 것이고, (c), (d)는 (a), (b)의 비트들에 대응하는 워드들의 변화를 보인 것이다. 같은 원리로 Fig. 17은 픽셀모델과 2D 워드모델을 오른쪽으로 2비트 이동시킨 후 위쪽으로 1비트 이동시킨 모델을 나타낸 것이다.

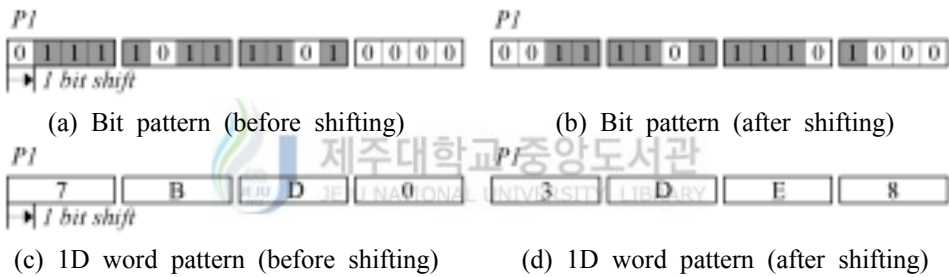


Fig. 14 Shifting bit / 1D word patterns

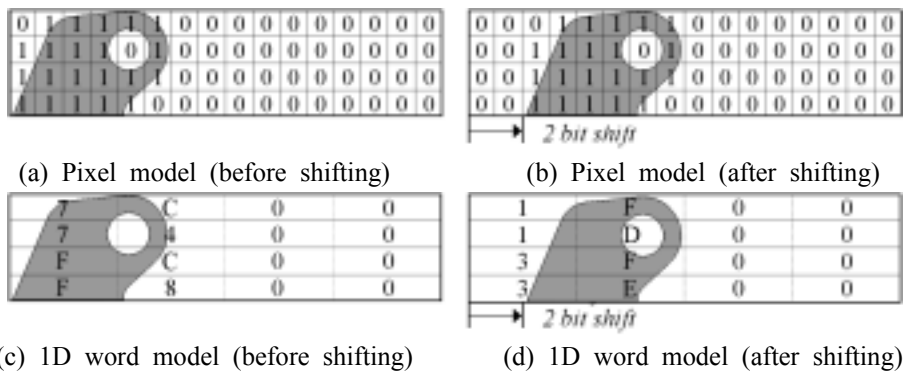


Fig. 15 Shifted pixel and 1D word model

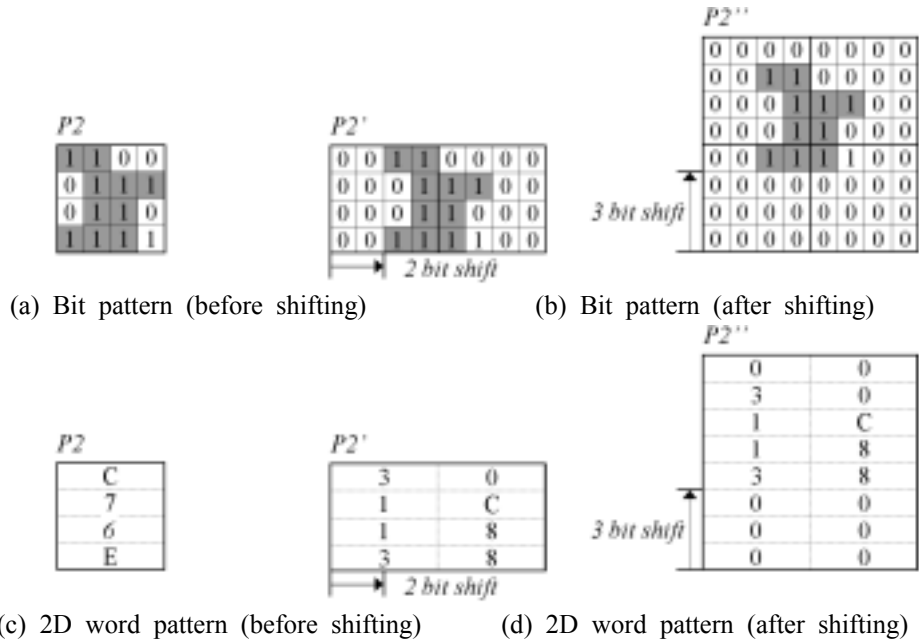


Fig. 16 Shifting bit / 2D word pattern

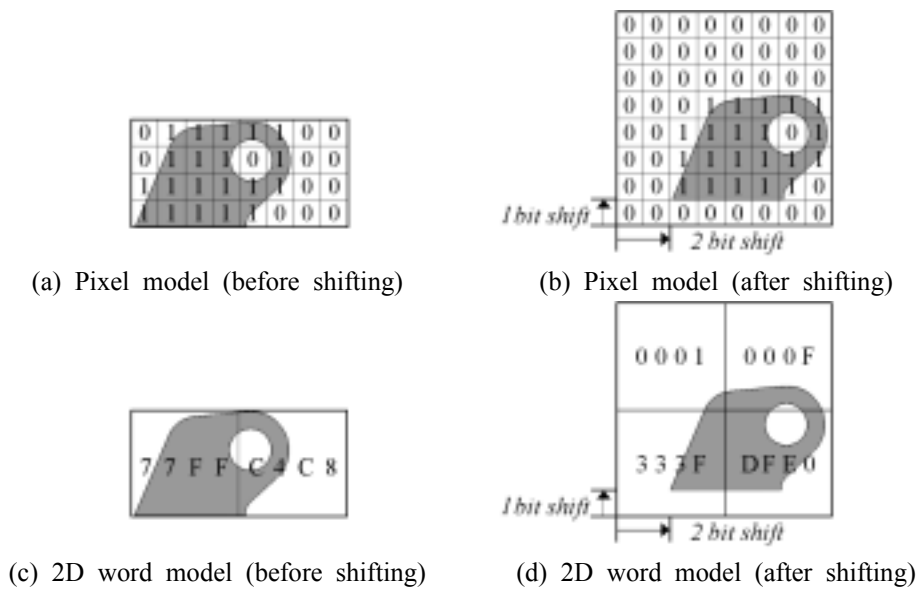
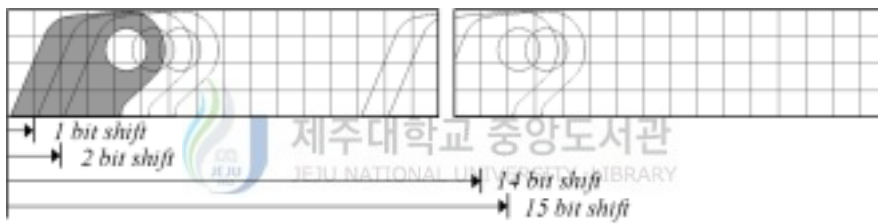


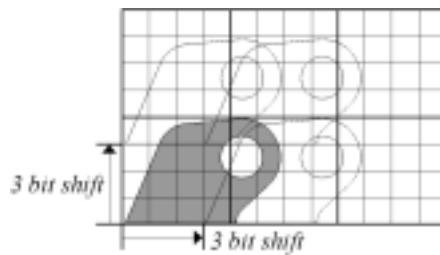
Fig. 17 Shifted pixel and 2D word model

하지만 shifted 워드모델을 부재의 배치, 삭제시 필요할 때마다 생성하여 사용하는 것은 비효율적이므로 본 연구에서는  $n \times m$  비트로 구성된 워드에 대해서  $(0, 0), (0, 1), (0, 2) \dots (n-1, m-2), (n-1, m-1)$ 인  $n \times m$  개의 shifted 워드모델을 생성하여 저장하고 있다가 필요시 이를 사용하도록 하였다.

예를 들어 Fig. 18(a)와 같이 16 bits/word로 구성된 1D 워드모델은  $(0, 0), (1, 0), (2, 0) \dots (15, 0)$ 인 16개의 shifted 워드모델을 생성하여 저장하고, (b)와 같이  $4 \times 4$  bits/word로 구성된 2D 워드모델은  $(0, 0), (0, 1), (0, 2) \dots (3, 2), (3, 3)$ 인 16개의 shifted 워드모델을 생성하여 저장한다.



(a) 1D word model



(b) 2D word model

Fig. 18 Shifted 1D word models and 2D word models of part

### 3. 비트네스팅 연산자

최적배치 과정에서 수반되는 부재의 배치, 삭제, 중첩검사를 위해 기존 대부분의 네스팅 프로그램에서는 형상을 이루는 직선과 곡선들 간의 교차점을 구하는 과정이 필요하게 된다. 형상이 복잡하고 많은 형상을 배치시킬 때에는 이에 따라 수많은 교차점을 구하는 과정이 필요하다. 이러한 교차점을 구하지 않고 모든 형상들을 픽셀화 시켜서 픽셀값의 증감이나 픽셀의 정수값을 비교하여 부재의 배치, 삭제, 중첩검사를 수행하는 방법이 제시되었으나 분해능 증가에 따라 처리 시간이 길어지는 단점을 가지고 있다.

위 단점을 보완하기 위해서 본 연구에서는 워드모델내 개개의 비트를 조작하는 비트연산을 적용한다.

비트연산은 두 피연산자에 대응되는 여러 비트들의 이진값들을 동시에 처리하여 결과값을 리턴하는 것으로 비트연산을 수행하는 비트연산자는 'bit AND', 'bit OR', 'bit XOR', 'bit SHIFT' 등이 있다. Table 1은 본 연구에서 도입한 비트연산 기능 및 해당기호를 나타낸다.

**Table 1** Bit Operators

Name	Symbol	Usage
bit AND	&	overlap check
bit OR		part allocation
bit XOR	^	part de-allocation
bit SHIFT	<<, >>	word shifting



### 3.1 부재 배치 비트연산

부재 배치(part allocation)는 원자재의 간극에 적절한 부재를 위치시키는 일련의 과정을 말한다.

본 연구에서는 워드모델로 표현된 원자재와 부재들 간의 비트연산을 이용하여 부재의 배치를 수행한다. 이 과정은 배치될 위치의 원자재 워드모델들과 대응되는 부재 shifted 워드모델들 간의 워드별 'bit OR' 연산으로 표현할 수 있다. 'bit OR' 연산자는 두 피연산자 사이의 비트 대 비트를 비교하여 대응하는 비트 중 어느 하나라도 1이면 결과 비트도 1이 됨을 나타내고 식 (3)과 같이 표현된다.

$$\begin{array}{r}
 11110000 \\
 | 11001100 \\
 \hline
 11111100
 \end{array} \quad (3)$$

위 과정을 부재 배치에 적용하면 원자재내 워드모델의 각 비트는 아래와 같이 정의되어 해당 비트가 갱신이 된다.

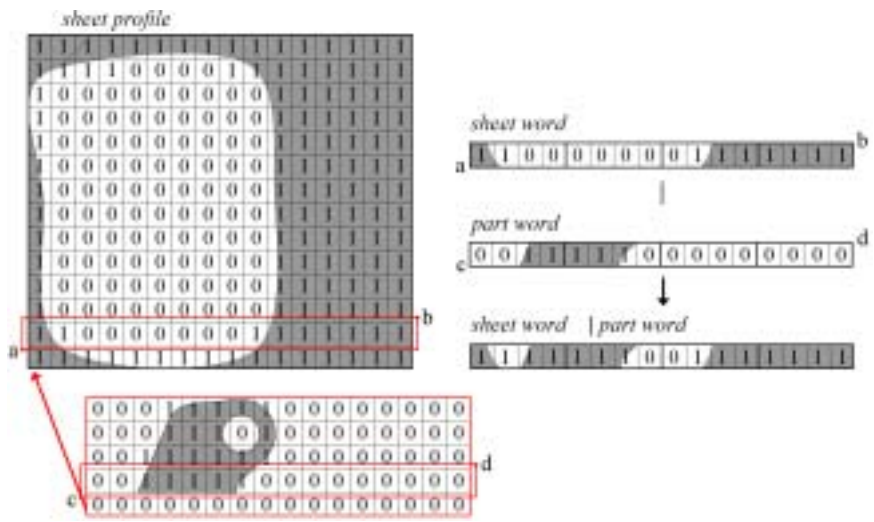
- 부재가 배치된 워드모델 내의 비트 → 1
- 배치가 불가능한 결함부위를 나타내는 워드모델 내의 비트 → 1
- 부재가 배치되지 않은 워드모델 내의 비트 → 0

Fig. 19(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 1D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*의 해당 비트패턴이 'bit OR' 연산을 거쳐 배치된 결과를 나타낸 것이고, Fig. 19(b), (c)는 1D 워드모델로 표현된 원자재와 부재에서 부재를 배치함에 따른 원자재에 대응되는 워드들이 갱신된 결과를 나타낸 것이다.

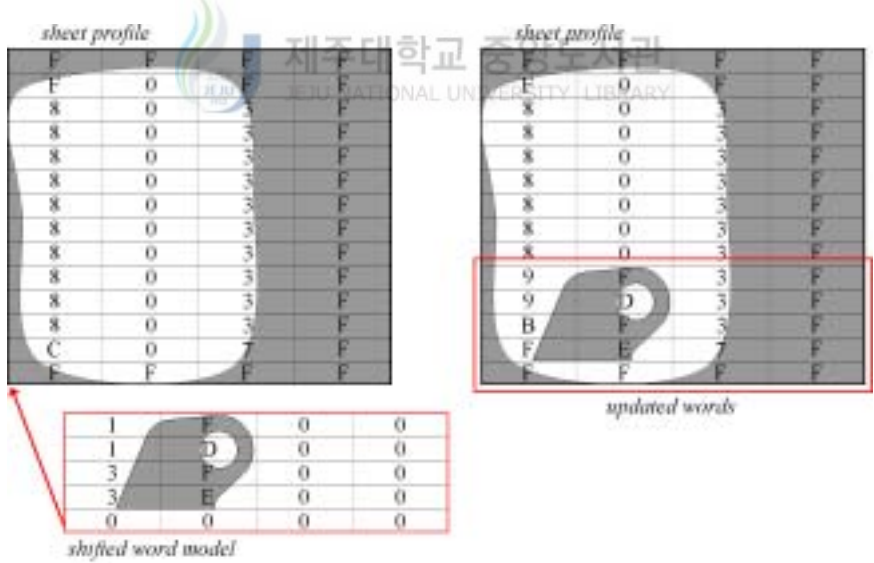
그리고 Fig. 20(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 2D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*가 배치되는 과정을 나타낸 것이고, Fig. 20(b), (c)는 2D 워드모델로 표현된 원자재와 부재에서 부재가 배치됨에 따라 대응되는 원자재 2D 워드와의 비트연산 결과를 나타낸 것이다.

즉, 픽셀을 기반으로 한 네스팅에서는 Fig. 19, 20의 경우를 표현하려면 최소 원자재의 6×4 픽셀들의 값을 하나하나 갱신해야 하지만 1D 워드모델에서는 이를 5개의 워드 비트연산만으로 표현이 가능하고 2D 워드모델에서는 4개의 워드 비트연산만으로 단순화 시킬수 있다.





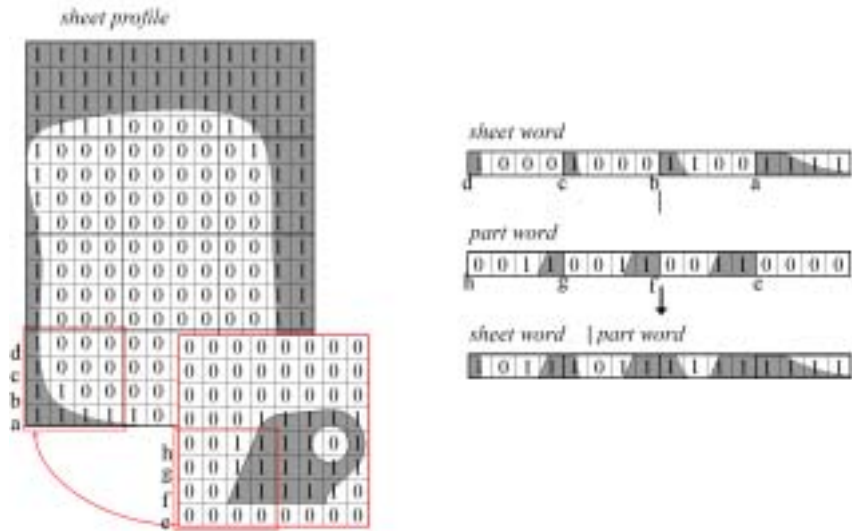
(a) Allocation procedure of bit patterns



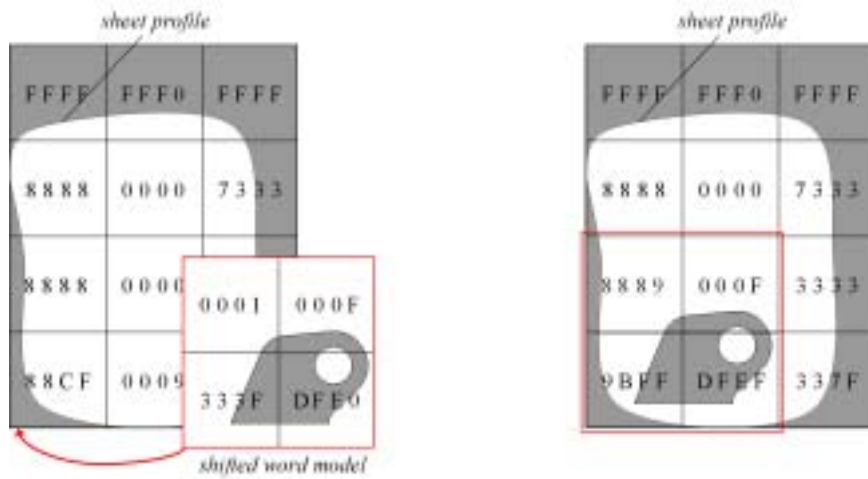
(b) Before part allocation

(c) After part allocation

Fig. 19 1D word based part allocation



(a) Allocation procedure of bit patterns



(b) Before part allocation

(c) After part allocation

Fig. 20 2D word based part allocation

### 3.2 부재 삭제 비트연산

부재 삭제(part de-allocation)는 배치상황을 갱신하거나 중첩이 발생할 경우 부재가 배치되기 이전단계로 되돌아가는 과정이며 부재의 배치 위치에 대응되는 원자재의 워드모델과 부재의 shifted 워드모델 간의 'bit XOR' 연산으로 표현할 수 있다. 'bit XOR' 연산자는 두 피연산자 사이의 비트 대 비트를 비교하여 대응하는 비트가 모두 1이거나 0이면 결과 비트를 0으로 리턴하고 대응하는 비트가 어느 하나만 1이면 결과 비트를 1로 리턴한다. 따라서 이 과정을 표현하면 식 (4)와 같다.

$$\begin{array}{r}
 11110000 \\
 \wedge 11001100 \\
 \hline
 00111100
 \end{array} \quad (4)$$

위 과정을 부재 삭제에 적용하면 원자재내의 워드모델의 각 비트는 아래와 같이 정의되어 해당 비트가 갱신이 된다.

- 결합부위에 배치된 부재를 삭제한 워드모델 내의 비트 → 1
- 배치된 부재가 삭제된 워드모델 내의 비트 → 0
- 부재가 배치되지 않은 워드모델 내의 비트 → 0

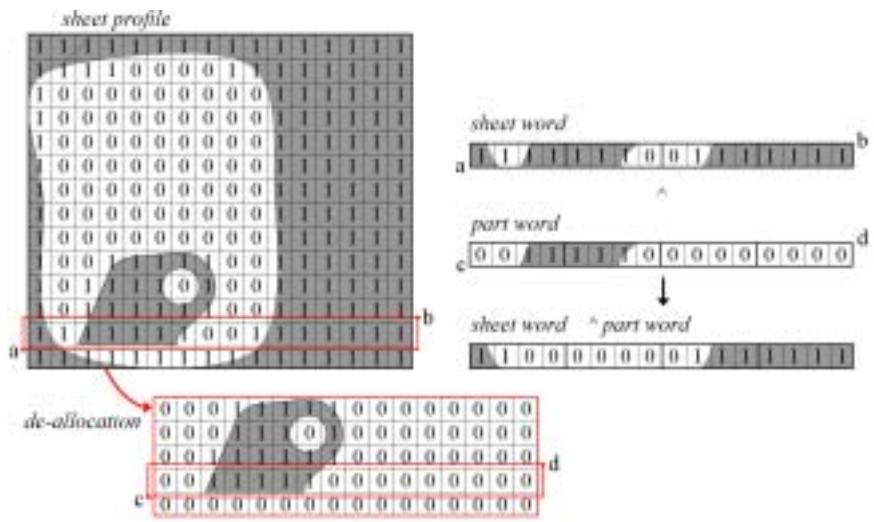
Fig. 21(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 1D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*가 삭제되는 과정을 나타낸 것이고, Fig. 21(b), (c)는 1D 워드모델로 표현된 원자재와 부재에서 부재를 삭제함에 따른 원자재에 대응되는 워드들이 갱신된 결과를 나타낸 것이다.

Fig. 22(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 2D 워드에 해당하

는 원자재에 부재가 삭제되는 과정을 나타낸 것이고, Fig. 22(b), (c)는 2D 워드 모델로 표현된 원자재와 부재에서 부재가 삭제됨에 따라 대응되는 원자재 2D 워드와의 비트연산 결과를 나타낸 것이다.

즉, 픽셀을 기반으로 한 네스팅에서는 Fig. 21, 22의 경우를 표현하려면 최소 원자재의 6×4 픽셀들의 값을 하나하나 갱신해야 하지만 1D 워드모델에서는 이를 5개의 워드 비트연산만으로 표현이 가능하고 2D 워드모델에서는 4개의 워드 비트연산만으로 단순화 시킬수 있다.





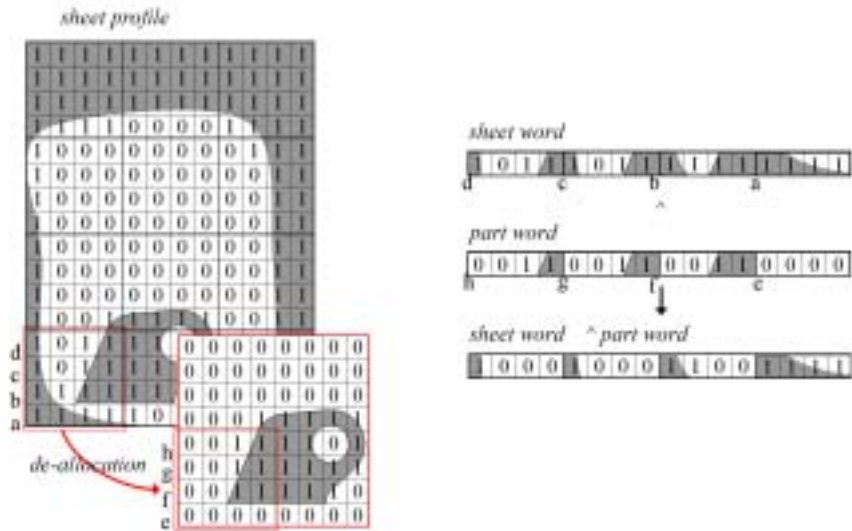
(a) De-allocation procedure of bit patterns



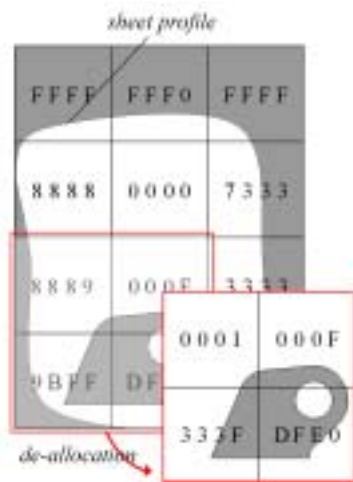
(b) Before part de-allocation

(c) After part de-allocation

Fig. 21 1D word based part de-allocation



(a) De-allocation procedure of bit patterns



(b) Before part de-allocation



(c) After part de-allocation

Fig. 22 2D word based part de-allocation



### 3.3 중첩 검사 비트연산

중첩 검사는 원자재 경계를 벗어나 배치되는 경우, 원자재에 배치가 불가능한 결합부위에 배치되는 경우, 배치된 부재 위에 다시 배치되는 경우를 겹치는 부분으로 판단하고 이를 검사하는 일련의 과정이며 부재가 배치될 위치에 대응하는 원자재의 워드모델들과 부재의 워드모델들 간 'bit AND' 연산으로 표현할 수 있다. 'bit AND' 연산자는 두 피연산자 사이의 비트 대 비트를 비교하여 대응하는 비트가 모두 1이면 결과 비트를 1로 리턴하는 것이며 이 과정을 표현하면 식 (5)와 같다.

$$\begin{array}{r} 11110000 \\ \& 11001100 \\ \hline 11000000 \end{array} \quad (5)$$

위 과정을 중첩검사에 적용하면 원자재내 워드모델의 각 비트는 아래와 같이 정의되어 해당 비트가 갱신이 된다.

- 중첩이 발생하면 해당 워드내 비트의 연산결과 → 1
- 중첩이 발생하지 않은 경우는 해당 워드내 비트의 연산결과 → 0

Figs. 23 ~ 26은 1D 워드모델 혹은 2D 워드모델로 표현된 원자재에 부재를 배치하는 과정에서 중첩 여부를 판단하는 과정을 나타낸 것이다.

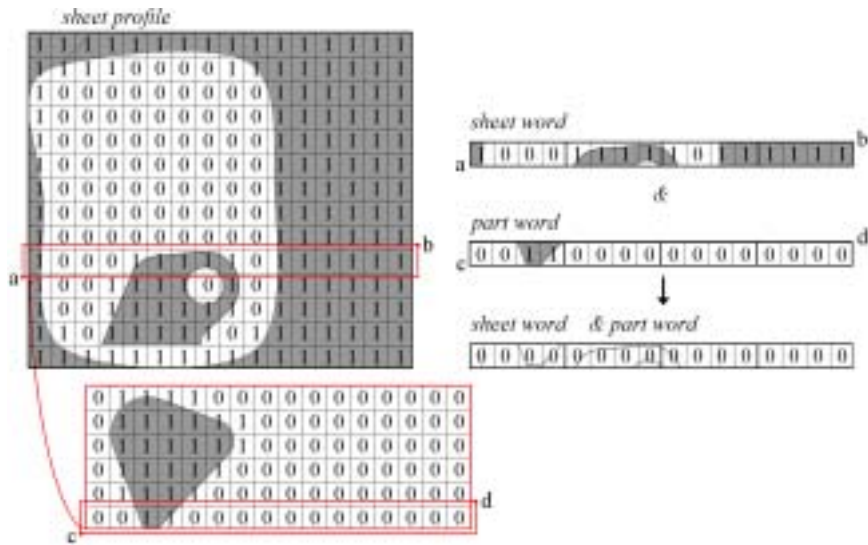
Fig. 23(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 1D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*를 배치되는 과정에서 대응되는 모든 비트의 연산결과가 0으로 중첩이 발생하지 않은 경우를 나타낸 것이고, (b), (c)는 1D 워드모델로 표현된 원자재와 부재에서 부재가 배치됨에 따라 대응되는 원자

재 2D 워드와의 연산결과가 모두 '0000' 임에 따라 중첩이 발생하지 않는 경우를 나타낸 것이다.

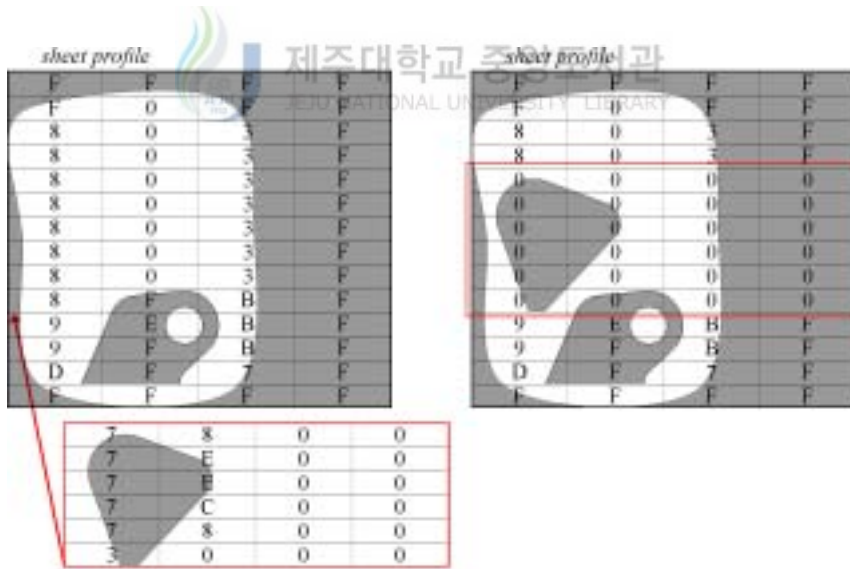
반면에 Fig. 24(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 1D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*가 배치되는 과정에서 대응되는 비트의 연산결과가 0이 아니므로 중첩이 발생한 경우를 나타낸 것이고, (b), (c)는 1D 워드모델로 표현된 원자재와 부재에서 부재가 배치됨에 따라 대응되는 원자재 2D 워드와의 연산결과가 모두 '0C00' 임에 따라 중첩이 발생한 경우를 나타낸 것이다.

Fig. 25(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 2D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*가 배치되는 과정에서 대응되는 모든 비트의 연산결과가 0으로 중첩이 발생하지 않는 경우를 나타낸 것이고, (b), (c)는 2D 워드모델로 표현된 원자재와 부재에서 부재가 배치됨에 따라 대응되는 원자재 2D 워드와의 연산결과가 모두 '0000' 임에 따라 중첩이 발생하지 않는 경우를 나타낸 것이다.

반면에 Fig. 26(a)는 비트패턴으로 표현된 원자재와 부재에서 1개의 2D 워드에 해당하는 원자재 *sheet word*에 부재 *part word*가 배치되는 과정에서 대응되는 비트의 연산결과가 0이 아니므로 중첩이 발생한 경우를 나타낸 것이고, (b), (c)는 2D 워드모델로 표현된 원자재와 부재에서 부재가 배치됨에 따라 대응되는 원자재 2D 워드와의 연산결과가 모두 '000C' 임에 따라 중첩이 발생한 경우를 나타낸 것이다.



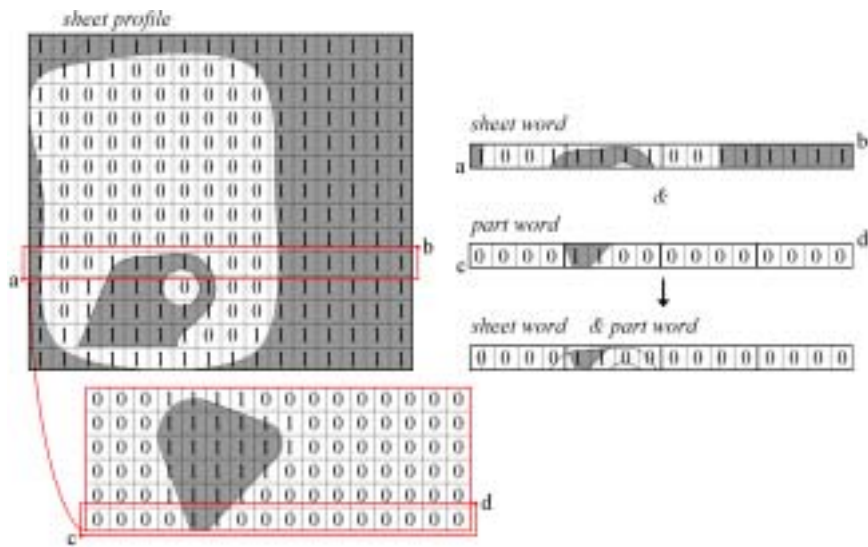
(a) Overlap test procedure of bit patterns(non-overlapped)



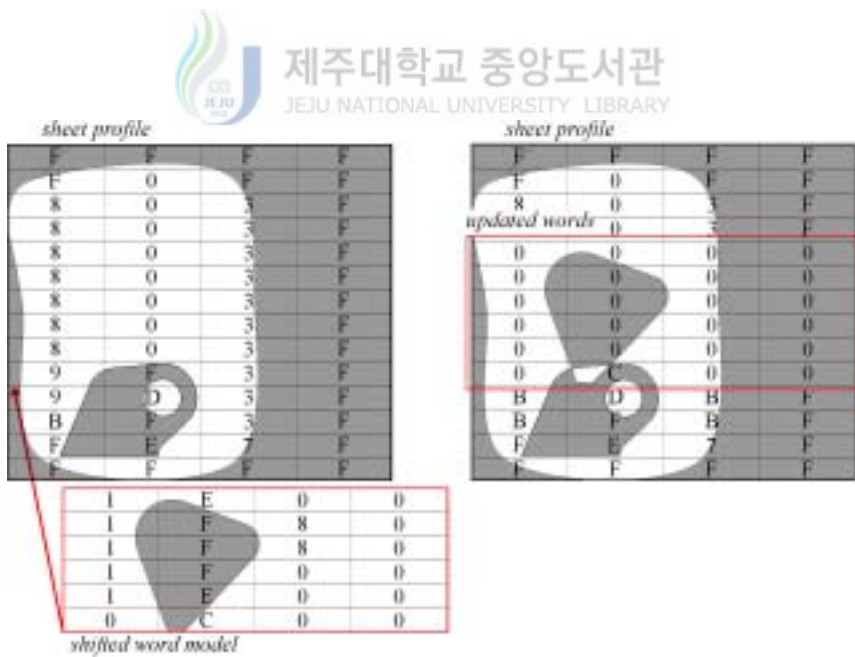
(b) Before overlap test

(c) After overlap test(non-overlapped)

Fig. 23 1D word based overlap test(non-overlapped)



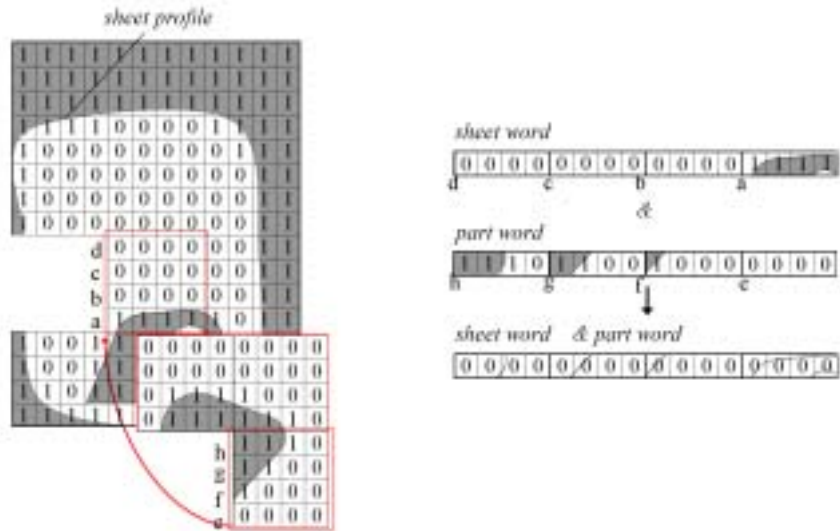
(a) Overlap test procedure of bit patterns(overlapped)



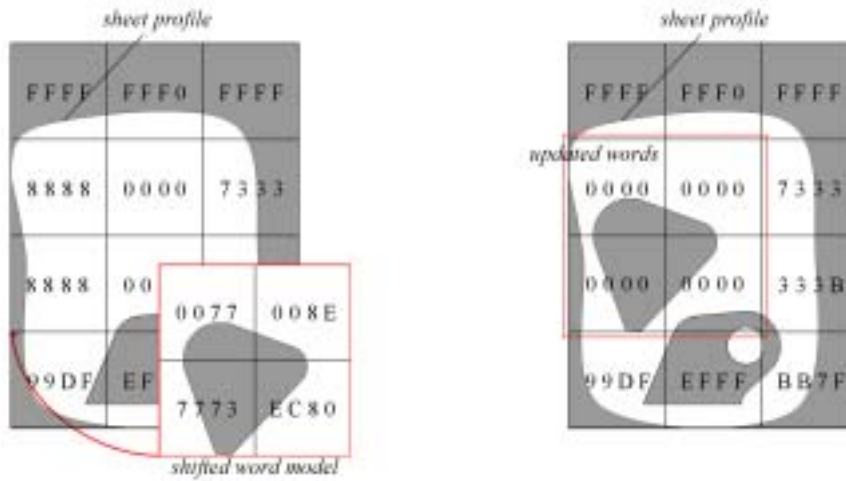
(b) Before overlap test

(c) After overlap test(overlapped)

Fig. 24 1D word based overlap test(overlapped)



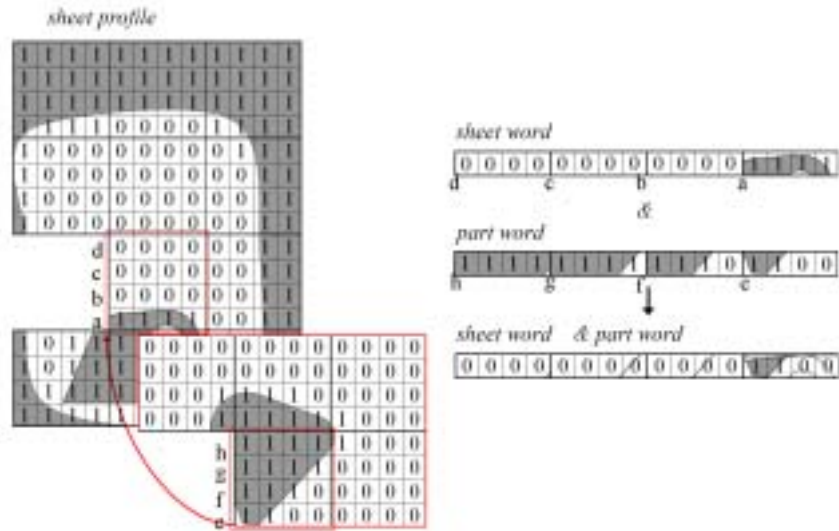
(a) Overlap test procedure of bit patterns(non-overlapped)



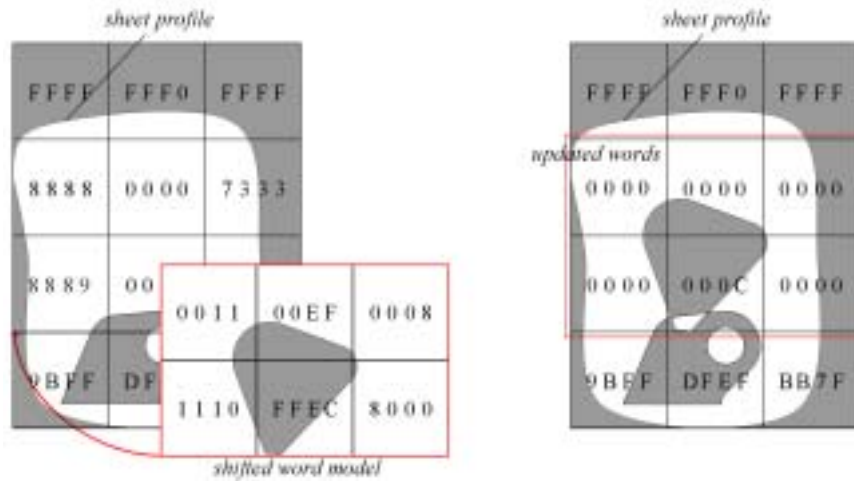
(b) Before overlap test

(c) After overlap test(non-overlapped)

Fig. 25 2D word based overlap test(non-overlapped)



(a) Overlap test procedure of bit patterns(overlapped)



(b) Before overlap test

(c) After overlap test(overlapped)

Fig. 26 2D word based overlap test(overlapped)

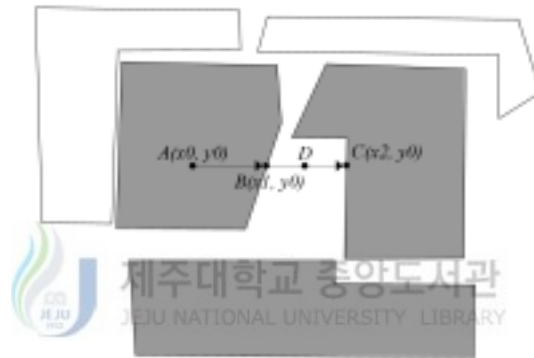
#### 4. 쿼드트리를 이용한 간극탐색

원자재에 부재를 배치한 후 자투리 영역의 일부를 나타내는 직사각형들을 간극(gap)이라 하고, 이를 찾는 일련의 과정을 간극탐색(gap search)이라 한다. 1993년 조경호[6]는 작은 형상의 효율적인 배치를 위해 국부적인 풀림을 통한 간극 채우기(gap filling with local annealing)를 제한한 바 있으며 아래는 이에 따른 간극 탐색 수행과정을 간략히 소개한 것이다.

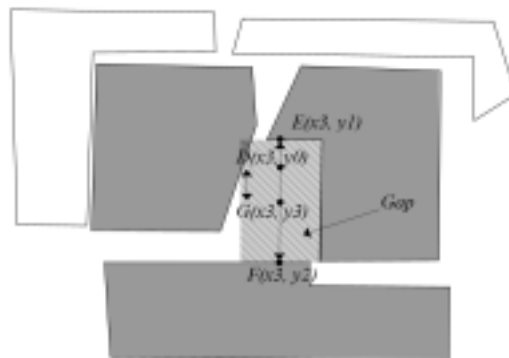
- 과정 1) Fig. 27(a)처럼 시스템에 의해 임의로 선택된 원자재 위의 픽셀  $A(x_0, y_0)$ 로부터 픽셀값을 검사하여 A의 값이 1인 경우 x만 증가시켜 scrap의 경계  $B(x_1, y_0)$ 를 찾고, 0인 경우 x만 감소시켜 scrap의 경계  $B(x_1, y_0)$ 를 찾는다.
- 과정 2)  $B(x_1, y_0)$ 로부터 x만 증가시켜 scrap의 경계  $C(x_2, y_0)$ 를 찾는다.
- 과정 3) 동일한 y에서 B, C의 중점  $D(x_3, y_0)$ 를 scrap의 좌우 경계의 중점이라 하고 D로부터 y값만 변화시켜 scrap의 상하경계  $E(x_3, y_1)$ ,  $F(x_3, y_2)$ 를 찾는다.(Fig. 27(b) 참조)
- 과정 4) 동일한 x에서 E, F의 중점  $G(x_3, y_3)$ 를 scrap의 상하 경계의 중점이라 하고 이로부터 y값만 변화시켜 새로운 D를 구한다.
- 과정 5) 과정 3과 4에서 구해진 scrap의 좌우 경계의 중점과 상하 경계의 중점을 'D → G → D → G ...'를 지정된 횟수 이내로 반복한다.
- 과정 6) 지정된 횟수 이내의 반복으로 두 중점 사이의 거리가 허용치 내로 수렴하면 이때의 새로운 경계점 B, C, E, F를 포함하는 가장 작은 직사각형을 간극이라 정의한다.

이때 사용된 간극탐색법은 픽셀을 기반으로 한 네스팅을 근간으로 하기 때문에 본 연구에서 제안하고 있는 워드표현법을 기반으로 하는 네스팅에 그대로 사용하는 것은 부적절하다.

본 연구에서는 비교적 효율적이고 간편한 계층적 자료구조의 하나인 쿼드트리 를 이용한 간극탐색법을 제안한다.



(a)



(b)

Fig. 27 Gap searching



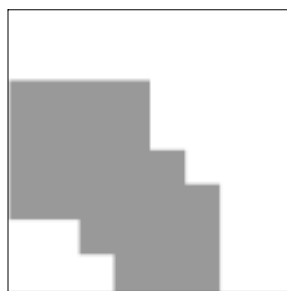
#### 4.1 쿼드트리 개요

쿼드트리(quad-tree)는 주어진 형상 혹은 영역을 네 개의 같은 크기를 갖는 직사각형으로 균일한 값(uniform value)이 나올때까지 계속 분할하여 표현하는 계층적인 자료 구조로서 이때 생성되는 직사각형을 셀(cell)이라 하며 셀은 노드(node)로 연결된다.[11] 여기서 각 노드들은 레벨(level), 상태(status), 위치(location) 등의 정보를 가진다. 레벨은 원자재 전체 형상을 포함하는 셀(root cell) 노드의 레벨을 0으로 하고 셀이 4등분될 때마다 1씩 증가하며, 상태는 형상의 영역과 셀 영역의 포함관계에 따라 full, partial, empty로 세분화된다. 위치는 각 셀이 4분할될 때 좌측 하단 셀을 시작점으로 해서 반시계방향으로 0, 1, 2, 3과 같이 정의된다.

예를 들면 Fig. 28(a)의 형상은 (b)와 같이  $2^3 \times 2^3$ 개의 픽셀로 근사화하여 표현할 수 있는데 여기서 1로 표시된 픽셀은 형상의 내부를 나타내고, 0으로 표시된 픽셀은 형상의 외부를 나타낸다. Fig. 28(c)는 형상을 표현하고 있는 셀 넓이와 (b)에서 1로 표시된 모든 픽셀의 넓이를 비교하여 상태가 full 또는 empty가 될 때까지 4분할된 결과를 나타낸 것이다. 셀을 4분할하는데 4분할시 나누어지는 셀을 부모셀(father cell)이라 하고 나누어져 생성된 셀은 자식셀(son cell)이라 한다. 자식셀은 Fig. 28(c)에서 □으로 표시된 영역과 같이 좌측 하단 셀을 시작으로 해서 반시계방향으로 0, 1, 2, 3의 위치 정보를 가진다. Fig. 28(d)는 위와 같이 생성된 형상을 쿼드트리로 나타낸 것이다. 이와 같이 쿼드트리는 부모셀의 상태가 empty이거나 full일 때까지 네 개의 영역으로 분할하여 자식셀을 표현하는 과정을 반복하여 형상을 표현하게 된다.[12]

이러한 쿼드트리는 형상의 2차원적인 동질성을 이용하여 형상을 압축하여 표

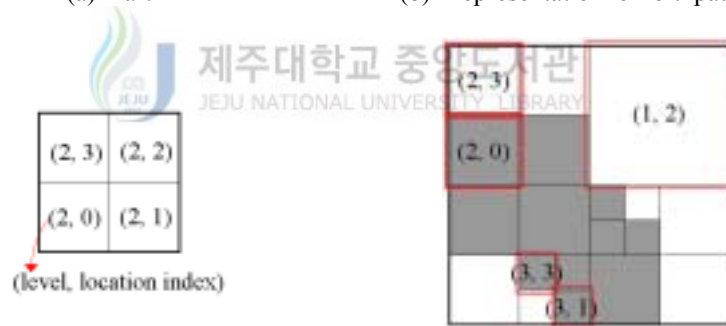
현하는 방법으로 메모리 용량을 절약할 수 있으며, 형상 정보를 경계 표현 방식이 아닌 영역 표현 방식으로 표현하기 때문에 형상 정보를 추출하기가 용이하다. 또한 트리 형태의 자료구조이기 때문에 데이터의 효율적인 접근이 가능하다는 장점을 가지고 있다.



(a) Part

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	1	0	0

(b) Representation of bit patterns



(c) Representation of cells



(d) Representation of quad-tree

Fig. 28 Quad-tree data structure

## 4.2 쿼드트리 형성과정

일반적으로 쿼드트리는 형상을 표현하기 위해 각 셀들의 full 노드를 검색하여 이웃관계에 있는 셀들의 노드를 서로 병합하여 표현하지만 본 연구에서는 이와 상반된 간극 탐색이 목적이기 때문에 full 노드가 아닌 empty 노드를 검색하여 쿼드트리를 나타내어야 한다. 아래는 본 연구에서 적용되는 쿼드트리 표현과정을 나타낸 것이다.

과정 1) Fig. 29(a)와 같이 원자재 형상을 포함하는 직사각형 □abcd, 즉 루트 셀을 생성하고 해당노드를 아래와 같이 정의한다.

루트 셀의 레벨(level) : Level 0

루트 셀의 위치정보(location index) : 0

루트 셀의 넓이(root cell area) :  $R0(w \times h)$

원자재 영역(sheet area) :  $S0$

과정 2) 루트 셀의 상태(status)를 정의한다.

$R0 < (\text{허용치})$  인 경우 : status = empty

$R0 > (S0 - \text{허용치})$  인 경우 : status = full

이외의 경우 : status = partial

(본 연구에서 허용치는 셀 넓이의 5%라 정의함)

과정 3) 과정 2에서 셀의 상태가 partial일 경우 셀을 크기가 같은 영역으로 사분할(quadrant)한 직사각형 □aa'od', □a'bb'o, □ob'cc', □d'oc'd를 생성한다. 이때 루트 셀은 부모 셀이 되고 4개로 나누어진 셀은 자식 셀이 되는데 부모 셀은 자식 셀을 연결하는 4개의 노드를 가지고 있다.

과정 4) 과정 3에서 생성된 각각의 셀에 해당하는 노드를 아래와 같이 정의한다.

셀의 레벨 : Level 0 + 1

셀의 위치정보 : □aa'od' = 0, □a'bb'o = 1

□ob'cc' = 2, □d'oc'd = 3

새로 생성된 셀 넓이 :  $R1(w' \times h')$

새로 생성된 셀 영역내의 원자재 넓이 : S0, S1, S2, S3

과정 4) 과정 3에서 생성된 4개 셀의 상태를 정의한다.

$R1 < (\text{허용치})$  인 경우 : status = empty

$R1 > (S0 - \text{허용치})$  인 경우 : status = full

$R1 > (S1 - \text{허용치})$  인 경우 : status = full

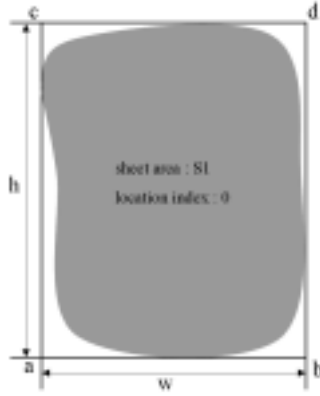
$R1 > (S2 - \text{허용치})$  인 경우 : status = full

$R1 > (S3 - \text{허용치})$  인 경우 : status = full

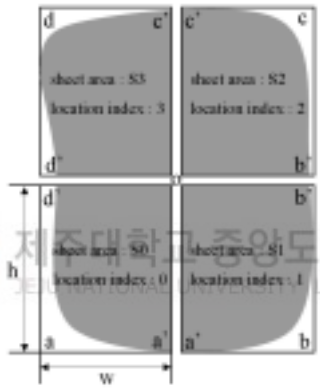
이외의 경우 : status = partial

과정 5) 모든 셀의 상태가 full 또는 empty일때 까지 과정 1에서 4를 반복한다.

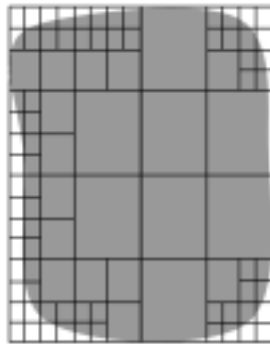
Fig. 29는 원자재 형상을 쿼드트리로 표현하는 과정을 나타낸 것으로 (a)는 초기 원자재 형상을 포함하는 루트 셀과 루트 셀 노드의 정보를 나타낸 것이고, (b)는 셀의 상태가 partial일 경우 4등분된 셀과 해당 셀 노드의 정보를 나타낸 것이다. 그리고 Fig. 29(c)는 분할된 모든 셀의 상태가 full 또는 empty가 될 때 까지 분할된 상태를 나타낸 것이다.



(a) Level 0



(b) Level 1



(c) Level n

Fig. 29 Quad-tree level

### 4.3 간극탐색

앞 절에서 언급된 퀴드트리를 이용하여 간극을 탐색하는 과정은 레벨이 0인 초기단계의 최상위 노드부터 최하위 노드까지 empty 노드들을 검색하여 이를 연결한 간극리스트를 생성한 후 empty 노드들을 수평, 수직으로 적절하게 병합해 나가는 것이다. 아래는 퀴드트리를 이용하여 간극을 탐색하는 과정을 나타낸 것이다.

- 과정 1) 퀴드트리에서 empty 노드들을 수평, 수직으로 병합(merge)한 간극리스트(gap list)를 생성한다. 수평접합은 같은 길이(h)를 가지는 인접셀들을 검색하여 해당 셀들을 병합하는 것이고(Fig. 30(b) 참조), 수직접합은 같은 폭(w)를 가지는 인접셀들을 검색하여 해당 셀들을 병합하는 것이다.(Fig. 30(c) 참조)
- 과정 2) 과정 1에서 생성된 간극들 중에서 다른 크기의 간극이라도 허용 범위에서 수평, 수직 연결이 가능한 것끼리 병합한다. 즉 과정 1에서 생성된 간극을 크기순으로 내림차순 정렬하여 가장 큰 간극이 나머지 간극과 확장연결이 가능한 것을 찾아 확장하여 새로운 간극을 생성한다.(Fig. 30(d) 참조)
- 과정 3) 과정 1과 과정 2에서 생성된 간극들과 미배치 부재를 크기순으로 내림차순 정렬하여 Fig. 30(e)와 같이 미배치 부재  $P_1$ 을 간극의 중심에 배치한 후 간극의 좌측하단으로 이동시킨다. 이때  $P_1$ 이 간극 주위에 이미 배치되어 있는 형상  $P_j$ , ( $j=1, 2, 3 \dots n$ )들과 중첩을 일으키면 다음 단계를 수행한다.
- 과정 4) Fig. 30(e)처럼 부재  $P_1$ 과 주위의 기존 배치 형상  $P_j$  들만으로 국부적

인 재배치를 시뮬레이티드 어닐링 기법으로 수행한다. 이는 Fig. 30(f)와 같이 형상  $P_1$ 과 기존의 배치 형상  $P_j$ 간의 중첩뿐만 아니라 형상  $P_j$ 들과 이를 둘러싸는 기배치 형상들 사이에서도 중첩이 일어나지 않을 때까지 계속된다. 이때  $P_j$ 들은 병진 이동만이 허용되고  $P_1$ 은 회전 이동까지 할 수 있도록 하였다. 이는 과정 3에 이르면 형상 사이의 간격이 매우 가까워져서 큰 형상의 조그만 회전도 쉽게 주위의 다른 형상과 중첩을 유발할 수 있기 때문이다. 만약 지정된 횟수 이상으로 재배치가 시도되어도 중첩이 해소되지 않으면 그 위치에 대한 형상  $P_1$ 의 간극 채워넣기는 취소된다. 이 경우 기배치 형상  $P_j$ 들은 본래의 배치상태로 되돌아가고, 형상  $P_1$ 를 위한 새로운 간극탐색이 과정 1부터 다시 시도된다.[6]

과정 5) 부재 배치후 원자재의 쿼드트리를 새로 갱신한 후 과정 1 ~ 4를 반복한다.

즉 Fig. 30(a)는 부재배치 후 쿼드트리로 나누어진 empty 노드들을 나타낸 것이고 (b), (c)는 각각 같은 크기의 주변 empty 노드들을 수평, 수직으로 병합한 영역을 나타낸 것이다. Fig. 30(d)는 (b), (c)의 결과를 수평, 수직으로 확장하여 얻어진 현 배치상태에서의 가용한 최종 간극들을 나타낸 것이고 (e)는 (d)의 간극들 중 일부에 어느 한 부재가 배치된 나타낸 것이다. Fig. 30(f)는 (e)에서 발생한 중첩을 해소하기 위해서 국부적 풀림을 적용한 경우를 나타낸 것이다.



(a) Empty nodes of quad-tree



(b) Horizontally merging equal size empty nodes



(c) Vertically merging equal size empty nodes



(d) Available gaps



(e) Gap fill by positioning



(f) Result after local annealing

**Fig. 30** Gap searching and filling



## IV. 적용사례

본 연구에서는 형상을 워드모델로 단순화하여 워드별 비트연산을 이용하여 부재의 배치, 삭제, 중첩검사를 수행하였으며, 부재가 배치될 수 있는 간극을 쿼드 트리를 이용하여 탐색하였다. Fig. 31은 본 연구에서 개발된 방법론을 바탕으로 구현된 네스팅 프로그램의 개략적인 흐름도를 나타낸 것이다.

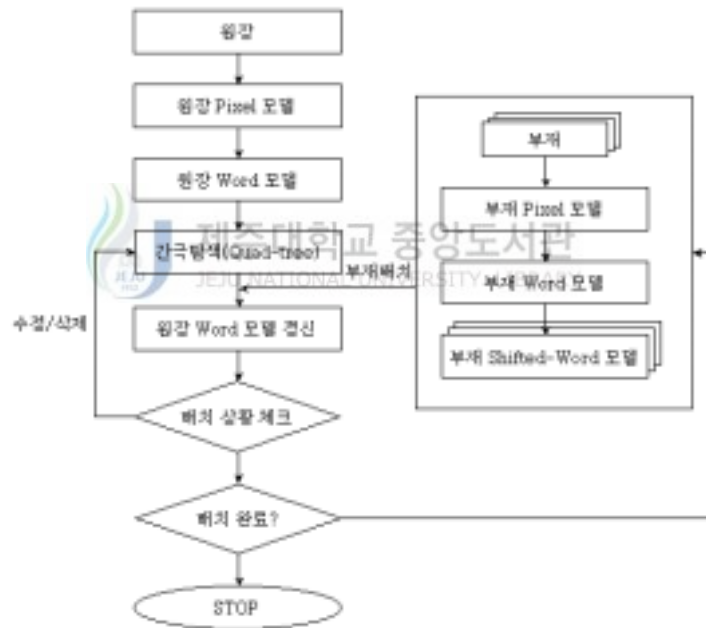
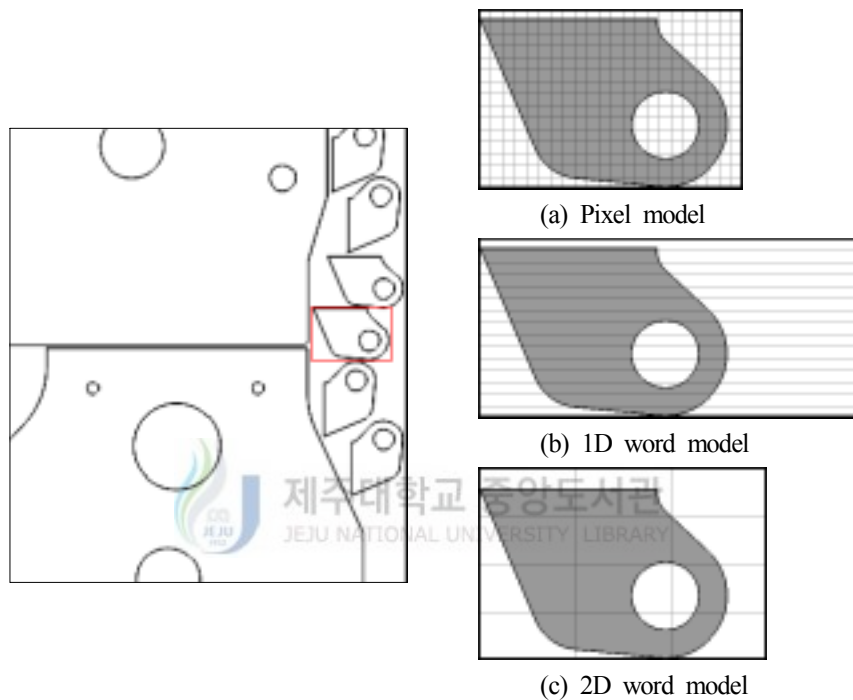


Fig. 31 Schematic flow diagram of the nesting

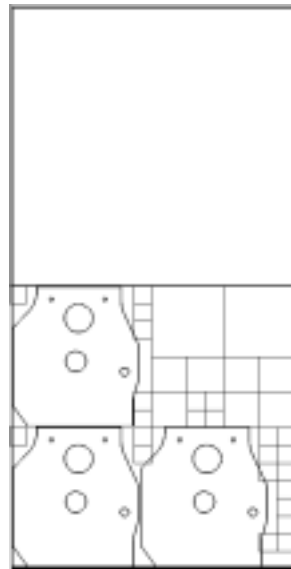
Fig. 32는 개발된 네스팅 프로그램에서 32 bits/word 기준으로 배치된 일부 영역을 보이는 것으로서 (a)의 수많은 격자들은 해당부재를 픽셀모델로 표현할 때 처리해야 할 격자들을 1 mm 단위로 픽셀화 시켜 나타낸 것이다.

반면에 (b)는 해당 부재의 픽셀모델을 1D 워드모델로 표현할 때 처리해야 하는 워드를 격자로 나타낸 것이고, (c)는 2D 워드모델로 표현할 때 처리해야 하는 워드를 격자로 나타낸 것이다.



**Fig. 32** Comparison of the grid size between the pixel model and the word model

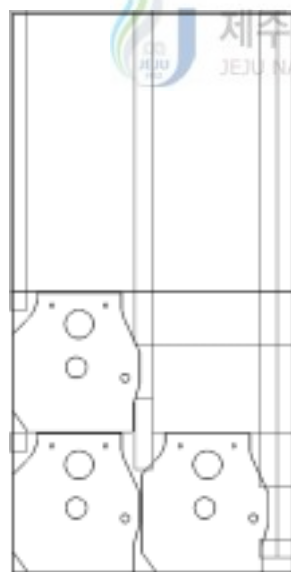
Fig. 33은 개발된 네스팅 프로그램에서 간극을 탐색하는 과정으로 (a)는 부재 배치 후 쿼드트리로 나누어진 empty 노드들을 나타낸 것이고 (b)는 같은 크기의 주변 empty 노드들이 병합된 영역을 나타낸 것이다. Fig. 33(c)는 (b)의 결과를 수평, 수직으로 확장하여 얻어진 현 배치상태에서의 가용한 최종 간극들을 나타낸 것이고 (d)는 (c)의 간극들 중 일부에 어느 한 부재가 배치되고, 그 상황에서의 간극들을 갱신하기 위해 다시 (a)와 같은 과정이 반복되는 것을 보이는 것이다.



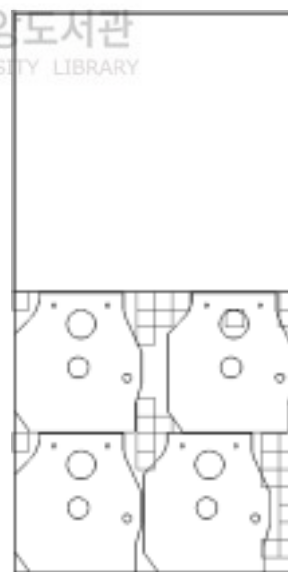
(a) Empty nodes of quad-tree



(b) Merging equal size empty nodes



(c) Available gaps



(d) After part allocation at a gap

**Fig. 33** Gap searching using quad-tree and part allocation

Figs. 34 ~ 36은 픽셀을 기반으로 한 네스팅과 본 연구에서 개발된 프로그램을 적용하여 PentiumIV 3.0 GHz PC에서 워드범위 32 bits/word로 항공기, 중장비, 조선용 샘플을 배치한 결과이다. 워드범위를 32 bits/word로 정의한 것은 워드를 구성하는 픽셀들이 정수값으로 분해되었기 때문에 부동소수점(64 bits/word)을 가진 데이터형은 본 연구에 적용될 수 없기 때문이다.

Fig. 34는 원장크기 1200×2500에 항공기 부품 부재 103개를 배치한 결과이고, Table 2는 픽셀을 기반으로 한 네스팅과 개발된 방법에서 자투리 비율 및 처리시간을 나타낸 것이다. 본 연구에서 자투리 비율(waste ratio)은 식 (6)과 같이 계산되었으며 여기서  $W$ ,  $L$ 은 원자재의 폭 및 사용길이이며  $\sum A_i$ 는 배치 형상의 면적 총합을 나타낸다.

$$waste\ ratio = \frac{(W \times L) - \sum A_i}{W \times L} \times 100 \quad (\%) \quad (6)$$

또한 처리시간은 부재의 배치, 삭제, 중첩검사 등에 소요된 순수한 시간외에 배치전략과 관련한 목적함수의 계산, 배치위치의 탐색, 그래픽 처리, 데이터베이스 입출력 등의 처리에 소요된 시간도 포함된 것이다.

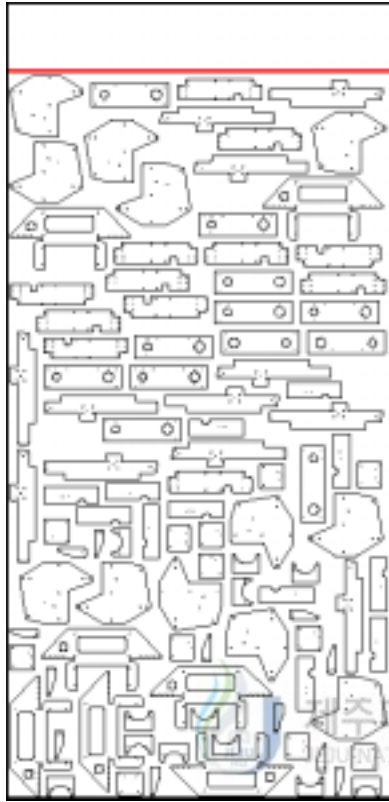
Fig. 35는 크기 3000×6000인 4장의 원자재에 중장비 부품 부재를 배치한 예이며 수행결과는 Table 3에 나타내었다. Table 3에서 보는 바와 같이 픽셀을 기반으로 한 네스팅은 처리시간이 84초가 소요된 반면에 개발된 방법에서는 처리시간이 51초가 소요되어 픽셀을 기반으로 한 네스팅보다 약 1.6배 정도 처리시간을 단축시킬 수 있었다.

Fig. 36은 크기 2500×8000인 3장의 원자재에 조선 부품 부재 배치한 예이며 수

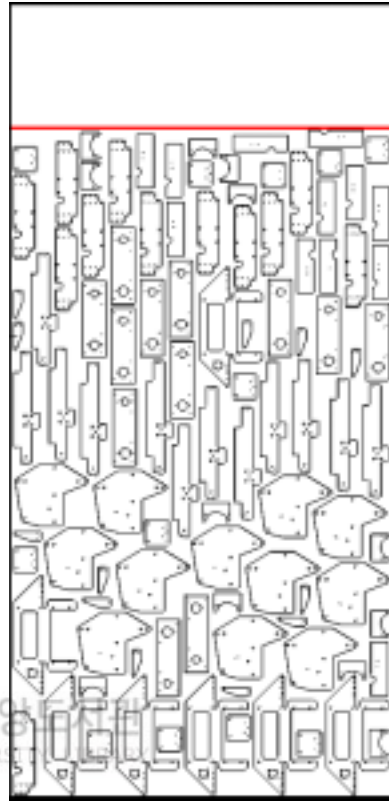
행결과는 Table 4에 나타내었다. Table 4에서 보는 바와 같이 픽셀을 기반으로 한 네스팅은 배치효율이 81.26%, 처리시간이 57초가 소요된 반면에 개발된 방법에서는 배치효율이 85.83%, 처리시간이 48초가 소요되었다.

이와 같이 본 연구에서 개발된 방법으로 원자재와 부재를 표현하여 네스팅에 적용하였을 때 원활히 수행됨을 확인하였으며, 적용되는 분야에 따라 약간의 차이는 있지만 기존 픽셀을 기반으로 한 네스팅에 비해 처리시간이 향상됨을 확인하였다.





(a) Pixel based nesting

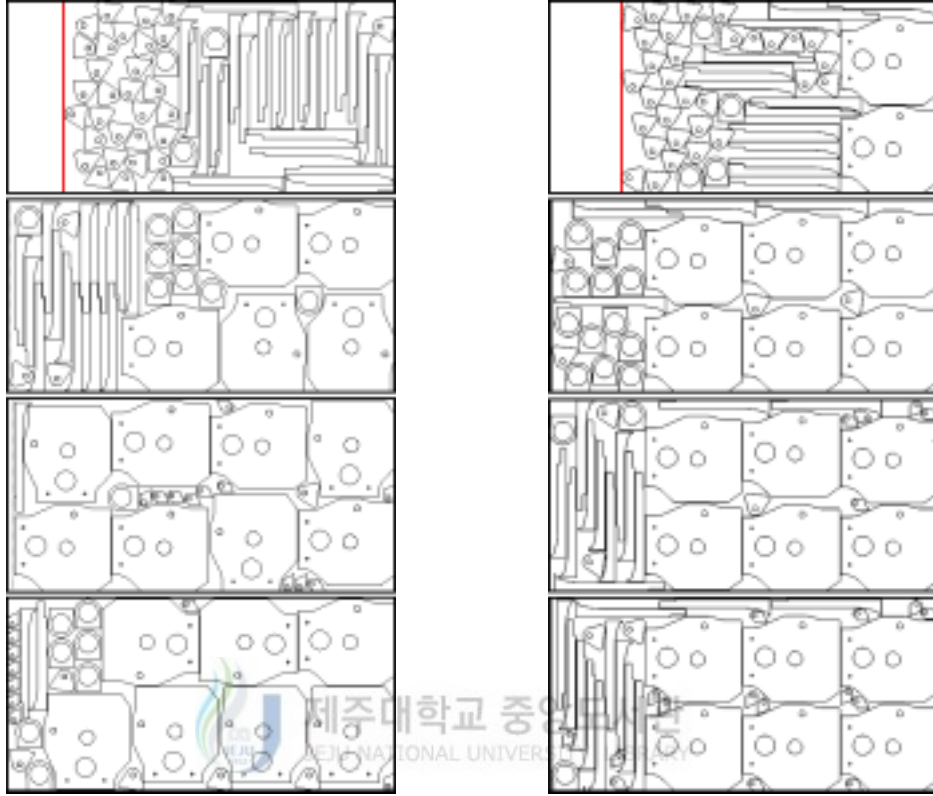


(b) 2D word based nesting

**Fig. 34** Nesting examples of aircraft parts

**Table 2** nesting results of aircraft parts

Fig. 34	used size(mm)	Allocated parts(EA)	Actual efficiency(%)	Waste ratio(%)	Process time(sec)
(a)	1200×2290	103	53.74	46.26	21
(b)	1200×2150		55.70	44.30	18



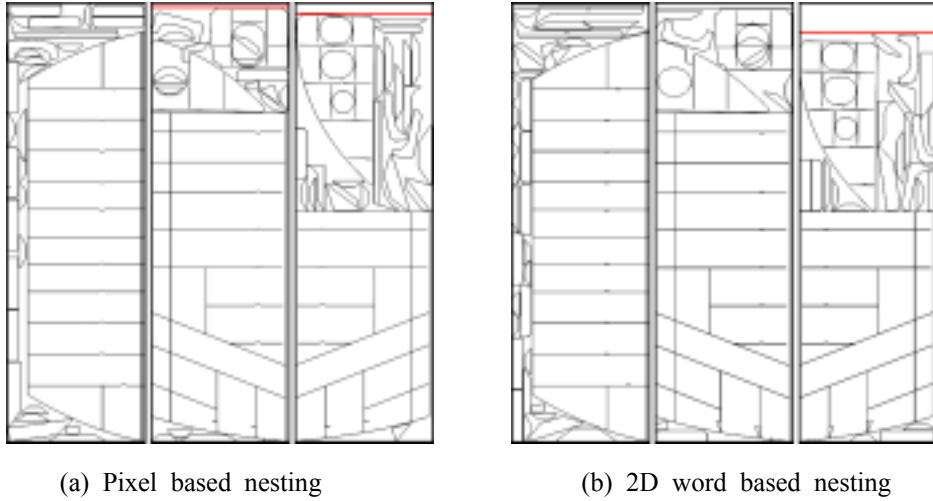
(a) Pixel based nesting

(b) 2D word based nesting


**Fig. 35** Nesting examples of heavy machinery parts

**Table 3** nesting results of aircraft parts

Fig. 35	used size(mm)	Allocated parts(EA)	Actual efficiency(%)	Waste ratio(%)	Process time(sec)
(a)	3000×5744	140	76.19	23.81	84
(b)	3000×5699		77.20	22.80	51



**Fig. 36** Nesting examples of shipbuilding parts


**제주대학교 중앙도서관**  
 JEJU NATIONAL UNIVERSITY LIBRARY

**Table 4** nesting results of shipbuilding parts

Fig. 36	used size(mm)	Allocated parts(EA)	Actual efficiency(%)	Waste ratio(%)	Process time(sec)
(a)	2500×7898	69	81.26	18.74	57
(b)	2500×7765		85.83	14.17	48



## V. 결론

기존에 연구된 배치 알고리즘 중에서 경계가 불규칙하거나 내부 결함이 있는 원자재의 표현이 용이하고 간극탐색 및 중첩검사시 수학적 연산이 필요하지 않는 픽셀을 기반으로 한 네스팅은 많은 장점을 가지고 있지만 원자재와 부재의 정확한 모사를 위해서 픽셀의 분해능을 증가시킬 경우 처리해야 될 픽셀의 수가 기하급수적으로 증가하여 처리시간이 길어지는 단점을 가지고 있다.

본 연구에서는 픽셀을 기반으로 한 네스팅의 단점을 보완하기 위해서 픽셀모델로 표현된 원자재와 부재를 비트의 집합으로 구성된 워드로 형상을 단순화 시킨 워드모델로 변환한 후 워드모델로 변환된 원자재와 부재들 간의 대응되는 워드간 비트연산으로 부재의 배치, 삭제, 중첩검사를 수행하는 방법을 처음으로 개발하였다. 또한 원자재내 부재를 배치할 수 있는 공간을 탐색하기 위해서 비교적 효율적이고 탐색시간을 단축할 수 있는 쿼드트리를 이용한 간극탐색법을 제시하였다. 이를 통해서 기존의 픽셀을 기반으로 한 네스팅과 비교하여 다음과 같은 효과를 확인하였다.

- 픽셀모델을 워드모델로 단순화하여 워드간 비트연산을 수행함에 따른 연산횟수의 감소 효과
- 연산횟수 감소에 따른 실행시간 단축 효과

따라서 본 연구에서 제시된 방법이 기존 픽셀을 기반으로 한 네스팅보다 정밀한 배치를 수행할 수 있으며, 픽셀 분해능 증가에 따른 메모리 문제와 처리시간 증가 문제 등을 해소 할 수 있는 방안임을 확인하였다.

## VI. 참고 문헌

- [1] Adamowicz, M. and Albano, A., 1976, "Nesting Two-Dimensional Shapes in Rectangular Modules," Computer Aided Design, Vol. 8, No. 1, pp. 27~33
- [2] Albano, A. and Sapuppo, G., 1980, "Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods," IEEE Transaction on Systems, Man and Cybernetics, Vol. SMC-10, No. 5
- [3] Kirkpatrick, S., Gelatt Jr., C. D., and Vecchi, M. P., 1983, "Optimization by simulated annealing," Science, Vol. 220, No. 4598, pp. 671~680
- [4] Jain P., Fenyves P. and Richter R., 1991, "Optimal Blank Nesting Using Simulated Annealing," J. of ASME, pp. 109~116
- [5] 방기범, 1990, "판재소재를 최소화하는 이차원 형상의 최적배치," 석사학위 논문, 서울대
- [6] 조경호, 1993, "판재부품의 가공 자동화를 위한 CAD/CAM 통합시스템," 박사 학위논문, 서울대
- [7] 조경호, 이건우, 1995, "임의 형상의 여러 원자재위에서의 효과적인 배치방안," 대한기계학회논문집, 제 19권, 제 8호, pp. 1854~1868
- [8] Ismail, H. S. and Hon, K. K .B., 1992, "New approaches for the nesting of two-dimensional shapes for press tool design," International Journal of Production Research, Vol. 30, No. 4, pp. 825~837
- [9] K. Fujita, S. Akagi and N. Hirokawa, 1993, "Hybrid Approach for Optimal Nesting Using a Genetic Algorithm and a Local Minimization Algorithm," in Proc. of the 19th Annual ASME Design Automation Conference, Vol 1,

pp. 477~484

- [10] 김영근, 양경부, 조경호, 2006, "최적배치를 위한 비트연산자 개발," 한국 CAD/CAM 학회 학술발표회 논문집, pp. 38~44
- [11] H. Samet, "Region representation : quadtrees from boundary codes," 1980, Comm. ACM 23, pp 163~170
- [12] 조준홍, 1991, "Quadtree를 이용한 불규칙한 형상을 갖는 패턴의 최적배치에 관한 연구," 석사학위논문, 한국과학기술원



## 감사의 글

“순간의 쾌락을 위해 너의 밝은 미래를 버리겠는가?”

너의 밝은 미래를 위해 순간의 쾌락을 버리겠는가?”

많은 우려곡절 끝에 입학한 대학원이라 가시밭길의 연속이었지만 이제 그 가시밭길이 비단길이 되어 나의 마음속에 펼쳐져 있습니다. 그동안 갈고 닦은 이 길을 후회 없이 걷고자 죽을힘을 다 하고자 합니다. 지금까지 아쉬워했던 순간들이 앞으로는 다시 오지 않기를 매 순간마다 최선을 다 하겠습니다.

그동안 부족한 저를 제자로 맞아 지금까지 이끌고 지도하여 주신 조경호 교수님께 감사드립니다. 교수님께서 베풀어 주신 가르침 정말 소중하게 간직하며 인생에 좋은 지표로 삼겠습니다. 그리고 논문 심사 기간 중 조언과 더불어 지혜를 주신 권기린 교수님, 김귀식 교수님, 허종철 교수님, 현명택 교수님, 정동원 교수님, 박윤철 교수님께도 감사드립니다. 또한 이 순간이 있기까지 격려와 도움을 주신 임종환 교수님, 최경현 교수님, 강철웅 교수님께 이 자리를 빌어 감사드립니다.

학위과정을 밟을 수 있도록 도움과 격려를 주신 아이스텔의 김가람 사장님과 같은 연구실에 언제나 한결같이 묵묵히 연구하는 카리스마 양경부 실장님께 감사드립니다. 그리고 얼큰한 목소리의 주인공 도성이형, 원조 꽃미남(?) 동훈이형, 키다리 아저씨처럼 멀리서 도움을 주는 대현이형과 형훈이형, 학과 조교로써 지원을 아끼지 않고 도와준 경조형, 병찬이형, 상율이, 친구이자 연애상담사 정근이, 언어술사 문종이, 불량감자(?) 광수, 축구매니아 병기, 유리, 창종이, 형찬이,

수정이, 실험실 막둥이이자 민무늬 빈대 종육에게 감사드리고 앞으로도 학과 발전을 위해 더욱더 힘써주시기를 부탁드립니다.

그리고 해외(?) 원정단으로 많은 지원을 해준 영원한 친구 민수, 성수, 덕균이와 안티영근의 선두주자 인환이형, 도현이형, 용민이형, 성원이형, 그리고 리더 진욱이, 지옥훈련 속에서 정을 맺은 승우, 승은외 해병대 ROTC 동기와 변치않는 우정을 함께하고픈 윤호, 경은, 민경, 매련외 컴퓨터 동아리 JUCC 15기 동기에게 감사드립니다.

마지막으로 불초 소생을 항상 올바른 길로 인도하여 주시고 잘 되기만을 기도하시는 아버지, 어머니, 그리고 항상 걱정해주고 사랑해주고 격려해주고 위로를 해준 하나밖에 없는 나의 형아에게 이 논문을 바칩니다.



푸르른 바다가 보이는 아늑한 연구실 한켠에서 김영근  
제주대학교 중앙도서관  
JEJU NATIONAL UNIVERSITY LIBRARY