한국해양공학회지 제26권 제4호, pp 8-14, 2012년 8월 (ISSN 1225-0767)
http://dx.doi.org/10.5574/KSOE.2012.26.4.008

휴리스틱 탐색 기법을 이용한 네스팅 전문가 시스템

신동목*
*울산대학교 조선해양공학부

Nesting Expert System using Heuristic Search

Dongmok Sheen*
School of Naval Architecture and Ocean Engineering, Univerty of Ulsan, Ulsan, Korea

KEY WORDS: Nesting 네스팅, Expert system 전문가 시스템, Pixel 픽셀, Steel cutting 강재절단, Heuristic search 휴리스틱 탐색

ABSTRACT: *Two dimensional nesting is a common problem in industries such as the shipbuilding, automotive, clothing, shoe-making, and furniture industries, in which various parts are cut off from stock or packed in a flat space while minimizing waste or unoccupied space. Nesting is known as an NP-complete problem, which has a solution time proportional to the superpolynomial of the input size. It becomes practically impossible to find an optimal solution using algorithmic methods as the number of shapes to nest increases. Therefore, heuristic methods are commonly used to solve nesting problems. This paper presents an expert system that uses a heuristic search method based on an evaluation function for nesting problems, in which parts and stock are represented by pixels. The system is developed in CLIPS, an expert system shell, and is applied to four different kinds of example problems to verify its applicability in practical problems.*

1. 서 론

네스팅(Nesting)은 다양한 모양의 여러 형상들을 평판 소재에서 잘라내거나 평면 상에 배치할 때 낭비되는 소재 또는 공간이 최소화 되도록 형상들을 배치하는 문제라 정의할 수 있다. 조선소의 강재절단, 자동차 및 기계부품 공장 등에서 블랭킹 공정, 의복 공장, 구두공장, 가구공장 등에서의 소재절단 공정에서 형상들을 배치하는 문제는 대표적인 네스팅 문제로서 가능한 한 공간적으로 빼곡히 배치하면 버려지는 부분을 작게 함으로써 재료비를 절약할 수 있다. 또한 조선소 야적장에서의 블록 배치 문제도 네스팅 문제로 볼 수 있다. 이러한 네스팅 문제는 각 형상별로 자세 및 배치될 위치 등을 찾아야 하는데 NP-complete (Non determimistric polynomial complete) 문제로서(Bennell and Oliveira, 2008) 알고리즘적인 방법으로 최적해를 찾을 수 없어 통상적으로 휴리스틱 방법을 이용한다. 상업용 CAD 시스템들에서 사용하고 있는 방법 또한 휴리스틱 방법에 기반을 두고 있는데(한국찬과 나석주, 1994), 부재들 간의 순서를 정한 후 배치할 부재들을 볼록 다각형으로 단순화하여 기존에 배치된 부재에 NFP(Nofit polygon) 알고리즘(Burke et al., 2007) 등을 적용하여 밀집 배치하는 단계적 방법을 사용하고 있다.

본 연구에서는 지식기반 추론 시스템인 전문가 시스템을 이용하여 배치할 형상이나 배치될 소재 경계 형상 등에 대한 제약 없이 네스팅 문제를 해결하고자 하며, 제안된 방법을 이용하여 각각 다른 특성을 갖는 네스팅 문제에 적용하여 그 적용 가능성을 예시하고자 한다.

2. 관련 연구

네스팅 기법은 형상표현 방법에 따라 픽셀로 형상을 표현하는 방법과 형상을 근사화한 다각형으로 표현하는 방법으로 나눌 수 있다. 픽셀 형태를 이용할 경우 표현이 간단하고 형상간 중첩여부를 쉽게 판단할 수 있는 반면, 정밀한 형상 표현을 위해서는 많은 양의 메모리와 계산시간이 소요된다. 반면 다각형으로 형상을 표현하는 경우 다각형의 경계를 정의하는 점과 선들의 정보만을 처리하게 되므로 상대적으로 적은 메모리를 이용하나, 형상간 중첩 여부를 판단하는 알고리즘이 복잡해진다.

형상을 표현하기 위하여 픽셀형태를 이용하는 경우 형상을 표현하기 위한 해상도를 정한 후 형상의 내부 및 경계선이 걸쳐지는 픽셀은 '1', 그 외의 공간은 '0'을 할당하는 Fig. 1에 예시

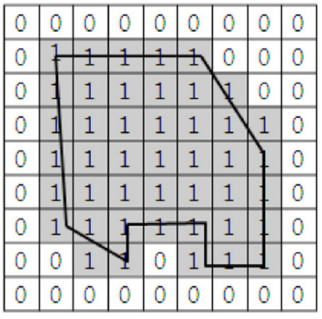


Fig. 1 Pixel representation

교신저자 신동목: 울산광역시 남구 무거동 산 29, 052-259-2152, dmsheen@ulsan.ac.kr

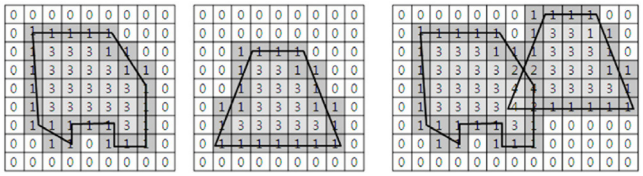


Fig. 2 A variation of pixel representation and overlapping

한 방법이 일반적이나(Weng and Kuo, 2011), 내부와 경계를 다르게 코딩하여 중첩 시 경계 픽셀 정보를 얻거나 우에서 좌로 빈 공간의 픽셀값을 1씩 증가시켜 할당함으로써 중첩 시 빈 공간까지의 상대적인 위치를 얻는 방법도 있다(Bennell and Oliveira, 2008; Babu and Babu, 2001). Fig. 2와 같이 코딩하여 두 형상을 배치할 경우 두 픽셀의 합이 4 이상인 픽셀은 두 형상이 겹치는 중첩 부위를 의미하며 합이 2인 픽셀은 접촉 부위, 1 또는 0은 중첩되지 않는 부위를 의미하게 된다. 픽셀을 이용하여 형상을 표현하는 경우 형상을 배치할 때 BLF(Bottom-left filling) 방법이 흔히 사용된다(Babu and Babu, 2001; Weng and Kuo, 2011). BLF 방법에서는 평판 소재 상에서 빈 곳 중 좌측 하단 끝 픽셀을 형상의 기준 픽셀이 놓일 위치로 선정하여 중첩여부를 검사하고 중첩이 될 때는 한 픽셀씩 위로 이동하면서 빈 곳을 찾고 최상단까지 배치할 빈 곳이 없을 때에는 우측의 다음 픽셀 열의 가장 아래 빈 픽셀부터 검사해 나가는 방법이다.

다각형을 이용하는 방법은 각 형상을 둘러싼 다각형으로 근사화하며 NFP를 이용하는 방법(Burke et al., 2007)이 대표적이다(Fig. 3 참조). 한 도형(A)의 모서리를 따라 다른 한 도형(B)이 회전 없이 움직일 때 도형 B의 한 기준점이 움직이는 궤적(N_{AB})을 나타내는 NFP를 정의하고 나면 두 다각형의 중첩 여부는 B의 기준점이 NFP의 내부에 속하는 지를 판단하면 된다. NFP를 구하는 방법은 A의 모서리는 반시계방향으로 정의하고, B의 모서리는 시계 방향으로 정리한 후 두 도형의 모서리의 합

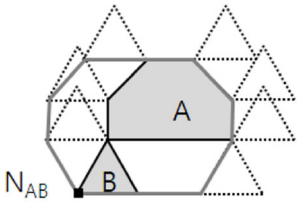


Fig. 3 Nofit polygon, N_{AB}

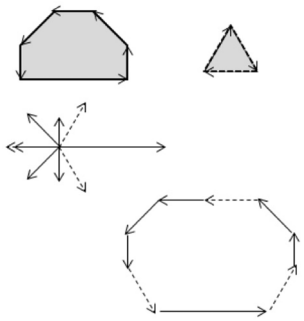


Fig. 4 Generation of a nofit polygon

집합에 대하여 반시계 방향으로 정렬하면 NFP가 구해진다(Fig. 4). 오목한 형상의 경우나 형상에 구멍이 있는 경우 NFP 알고리즘은 복잡해진다. NFP는 두 형상간의 중첩여부를 판단하는 효과적인 도구이나 여러 부재들 간의 배치 순서를 제시하지는 못하며 따라서 흔히 큰 부재부터 순차적으로 NFP를 적용한다.

한편 조선 관련 연구(이철수와 박광렬, 1996; 한창봉과 박제웅, 1999)에서는 같은 폭을 갖는 부재들을 그룹화 한 후 각 그룹별로 부재들과 폭이 같은 긴 평판 소재에 일렬로 배치하는 문제로 접근하였다. 이 경우 일반적인 2차원 네스팅 문제보다 단순화 할 수 있다. 즉, 폭 방향으로는 하나의 부재만 배치되며 한 부재는 0^{o},180^{o}회전 형상, 그리고 각각에 대하여 뒤집은 형상의 4 가지 자세만 고려하면 된다. 기존 연구(이철수와 박광렬, 1996; 한창봉과 박제웅, 1999)에서는 큰 부재부터 순차적으로 연속 배치하는 방법을 사용하였으며 따라서 단계별로 추가하는 형상에 대하여 4 가지 자세 중 최적의 배치를 찾았다.

이밖에 평판 소재 형상이 직사각형이 아닌 비정형 형상인 경우에 형상들을 배치하는 문제도 주요한 연구주제들로 다루어지고 있다(Tay et al., 2002; Lee et al., 2008; Burke et al., 2007).

3. 네스팅 전문가시스템

3.1 시스템 구성

전문가 시스템은 C언어나 Fortran 언어 등을 이용하여 개발되는 전통적 프로그램과 구별되는 프로그램이다. Fig. 5는 대표적인 전문가 시스템인 규칙기반 시스템(Rule-based system)의 구성을 보여준다.

각 실행 단계에서 확인된 사실들(Facts)로부터 IF<conditions> THEN<action> 형태의 규칙을 순차적으로 적용하여 새로운 사실을 생성해가면서 궁극적으로 원하는 해를 얻는 추론적 방법이다. 실행과정을 살펴보면 전역데이터베이스(Global data base, GDB)에 저장된 사실로부터 해석기(Interpreter)는 규칙기반(Rule base)에 있는 규칙들 중 적용 가능한 규칙들을 찾아내고, 충돌해결기구(Conflict resolution mechanism)에서 제어기준(Control strategy)을 적용하여 하나의 규칙을 선택하며, 선택된 규칙은 추론엔진(Inference engine)에 의해 실행된다. 대표적인 제어기준으로는 가장 최근에 추가된 사실과 부합하는 규칙을 선택하는 방법(Recency ordering), 조건부가 복잡한 규칙을 선택하는 방법(Specificity ordering)등이 있다. 실행 결과에 따라 GDB가 갱신되고 전체 과정이 반복된다. 실행 과정에서 동일한 사실에 기반

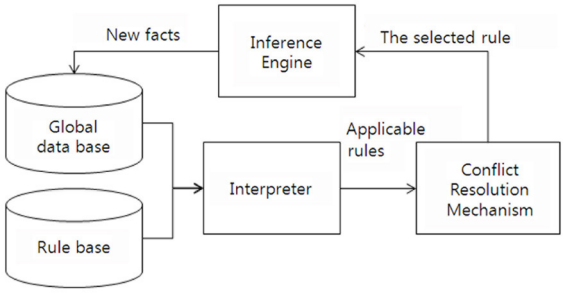


Fig. 5 A Schematic diagram of expert system

하여 한 번 실행된 규칙은 다시 실행되지 않으며 해를 얻거나 더 이상 적용 가능한 규칙이 없을 때 시스템은 종료된다.

해를 찾는 과정은 탐색 트리(Tree)로 표현할 수 있는데 각 노드는 단계별 상태를, 각 노드에서 나온 가지들은 각 상태에서 적용 가능한 규칙들을 나타낸다. 하나의 규칙이 적용되면 새로운 사실이 추가된 새로운 상태가 되며 새로운 가지치기를 할 수 있게 된다. 문제에 대한 사전 정보가 없을 때 문제를 해결하는 일반적인 탐색 트리는 크게 깊이 방향을 먼저 탐색하는 깊이우선(Depth-first) 탐색 트리와 같은 깊이에서 가능한 경우의 수를 먼저 모두 탐색하는 폭우선(Breadth-first) 탐색 트리로 나눌 수 있다. 문제의 해를 찾는 탐색 방향에 관한 정보가 있을 때는 각 노드에서 다음 탐색할 노드들에 대하여 해에 근접한 정도와 관련된 평가함수(Evaluation function)를 계산하여 그 값이 큰 노드부터 탐색하는 휴리스틱(Heuristic) 탐색 방법을 사용한다.

본 연구에서는 미 항공우주국(NASA)에서 개발한 전문가 시스템 쉘(Shell)인 CLIPS(http://clipsrules.sourceforge.net/)를 이용하여 네스팅을 위한 전문가 시스템을 휴리스틱 탐색 방법을 이용하여 구축하며 이를 블록 맞추기 퍼즐, 일반 네스팅, 조선 분야 네스팅, 이형 경계 형상 소재에의 배치 문제에 적용하여 그 적용 가능성을 예시하고자 한다.

3.2 데이터 구조

본 논문에서는 네스팅 문제를 휴리스틱 기법을 이용한 깊이우선 트리 탐색 문제로 정의한다. Fig. 6은 휴리스틱 탐색 트리를 보여준다. 각 노드에서 다음에 탐색할 아래 계층 노드들은 평가함수 값에 따라 재정렬되어 평가함수 값이 가장 큰 노드부터 탐색을 한다. 따라서 Fig. 6에서 제일 좌측 가지들은 각 노드에서 평가함수 값이 가장 큰 노드들이다. 본 연구에서 평판 소재 및 부재들은 픽셀 형태로 표현한다. 흑백 픽셀 데이터는 기본적으로 이진 영상(Binary image) 이므로 형상 간 중첩을 검증하기 쉬우며 상업용 CAD 시스템과의 연동도 쉽게 할 수 있다는 장점이 있다. Fig. 6에서 시작 노드는 하나의 부재도 배치되지 않은 평판 소재의 상태를 나타내며, 하나의 계층이 내려갈 때마다 하나의 부재가 배치된다. 모든 부재가 다 배치된 계층에 이르면

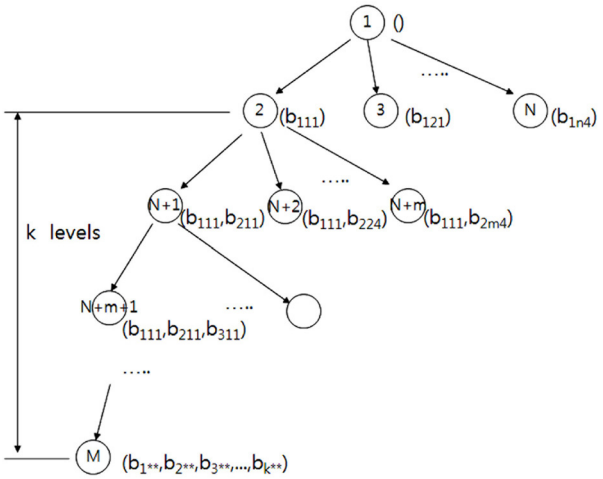


Fig. 6 A heuristic depth-first search tree

평판소재에는 모든 부재가 다 배치된 상태로서 문제의 해를 나타낸다. 하나의 노드는 (b_{111},b_{211},…)와 같이 정의되는데 여기서 b_{rst}는 r번째 부재의 s번째 위치, t번째 자세를 의미한다. 하나의 부재는 회전각에 따라 무한한 가짓수의 자세를 가질 수 있으나 본 연구에서는 0°,90°,180°,270°회전 형상, 그리고 각각에 대한 미러(Mirror) 이미지 등 최대 총 8 가지 자매 형상을 고려한다.

회전이나 미러링(Mirroring) 결과 동일한 형상은 생성하지 않는다. 따라서 각 노드에는 식(1)과 같이 n_r개의 가지가 생성된다. 여기서, $n(r,i,B_{r-1})$은 현재 (r-1)개의 부재가 배치된 평판 소재의 상태 B_{r-1}에서 r번째 부재의 i번째 자세에서 배치 가능한 위치에 대한 경우의 수를 의미한다.

$$n_r = \sum_{i=1}^{8} n(r,i,B_{r-1}) \tag{1}$$

다음은 CLIPS로 나타낸 노드에 대한 데이터 템플리트(Template)를 나타낸다. Node-id와 Parent 슬롯(Slot)은 자신의 ID와 부모 노드의 ID를, Tile-list와 Tile-orient-list 슬롯은 현재 노드까지 배치되어 있는 부재들과 각각의 자세를, Tile-position-row-list와 Tile-position-col-list 슬롯은 각 부재들의 기준점 위치를 행 번호와 열번호 리스트로 나타낸 것이다. 기준점은 부재들을 둘러싼 최소 사각형(Enclosing rectangle, ER)의 좌 상단 픽셀이다.

```
(Deftemplate MAIN::node
  (Slot node-id (type NUMBER))
  (Slot parent (type NUMBER))
  (Multislot tile-list (type FACT-ADDRESS))
  (Multislot tile-orient-list (type FACT-ADDRESS))
  (Multislot tile-position-row-list (type NUMBER))
  (Multislot tile-position-col-list (type NUMBER)))
```

전역 데이터베이스(GDB)에는 노드들에 관한 정보 이외에 부재들의 배치 순서를 저장하는 Lay-tile-list, 다음 번에 배치할 확장할 순서에 따라 노드들을 리스트로 표현한 Open-list가 있다.

3.3 규칙

지식 베이스는 Initialize, Open-node, Goal 등의 규칙과 관련 함수들로 구성되어 있다. Initialize 규칙에서는 각 부재를 먼저 크기 순서로 정렬한 후 이를 Lay-tile-list에 저장한다. 또한 각각의 부재에 대하여 8 가지 자세를 생성하며 적용 분야에 따라 자세의 가짓수를 조정한다. 예를 들어 의복제조 분야와 같아 소재의 앞 뒷면을 구별해야 하는 경우 거울 이미지는 고려할 필요가 없고, 조선 블록 생산 시 소요되는 보강재류와 같이 긴 소재에 폭이 같은 부재들을 배치하는 경우 90°,270°의 회전 형상 및 그에 대한 미러(Mirror) 형상은 고려할 필요가 없다.

Open-node 규칙은 해에 근접한 노드를 찾아 다음 순서에 따라 탐색트리를 확장한다.

```
Rule: open-node
n = first(open-list). where, open-list={n, REST}   ; Get the
```

```
        first node from the set open-list and call it n
{N} = expand(n)          ; Expand the node n and save the
        result set as {N}
{Ns} = usort(N)          ; Sort N in descending order of
        evaluation function value and save it as set {Ns}
open-list = {Ns, REST}   ; Update the set open-list with
        Ns front
```

다음은 CLIPS로 작성된 Open-node 규칙의 조건부를 일부 보여준다. CLIPS에서는 패턴 매칭 방법에 의하여 규칙의 조건부와 매칭된 사실들을 찾는다.

```
(defrule MAIN::open-node
  ?op <- (open-list (nodes ?nid $?rest-n))
  ?cn <- (node (node-id ?nid)
               (parent ?p)
               (tile-list $?tlist)
               (tile-orient-list $?torlist)
               (tile-position-row $?prlist)
               (tile-position-col $?pclist))
  (lay-tile-list $?tlist ?t $?rest-t)
```

여기서, 가지를 새로 생성할 노드는 Open-list의 Nodes 슬롯에 저장된 리스트의 맨 앞에 있는 노드(?nid)이며 이 노드에 대한 정보는 GDB에 있는 노드들 중 패턴이 일치하는 (Node (Node-id ?nid) …)를 찾아 관련정보를 가져오게 된다. 현재 노드의 Tile-list 슬롯 값인 $?tlist는 현재까지 배치된 부재들을 나타내며 이는 Lay-tile-list와 매칭되어 리스트의 요소 ?t가 다음에 배치할 부재를 나타내게 된다. 이 규칙은 탐색 트리에서 한 노드로부터 가지를 확장하는 규칙으로 현재 노드에서 가지를 생성할 때 즉, 현재까지 배치가 끝난 부재들에 다음 부재를 추가할 때 부재의 8개 자세에 대하여 배치 가능한 n_r개의 모든 위치(식 (1) 참조)를 좌 상단부터 차례로 찾아서 자식(Children) 가지로 확장한다. 새로 생성된 노드들은 평가함수 $u(B_{r-1},b_{ir**})$값이 큰 순서대로 정렬하여 Open-list에 저장함으로써 평가함수 값이 큰 노드부터 탐색이 이루어질 수 있도록 한다. 여기서, B_{r-1}은 현재까지 배치된 부재들을 포함한 평판 소재의 상태를 나타낸다. 식 (2)는 본 연구에서 적용한 평가함수를 나타낸다. 식에서 count는 픽셀 수를 세는 함수이며, $u(B_{r-1},b_{ir**})$는 r 번 째 부재를 b_{ir**}에 배치할 때 현재 배치가 완료된 (r-1)개의 부재들을 둘러싼 ER들과 부재 r의 ER을 1:1로 비교했을 때 겹쳐지는 픽셀 수들을 합한 것이다.

$$u(B_{r-1},b_{r**}) = \sum_{b_{k**} \in B_{r-1}} count(ER(b_{k**}) \cap ER(b_{r**})) \quad (2)$$

평가함수는 탐색트리에서 노드의 확장 순서를 결정하는 기준이 되며 함수 값이 클수록 조밀하게 형상이 배치됨을 나타내므로 원하는 해를 도출할 가능성이 높다. 식 (2)의 평가함수는 각 ER의 좌상단과 우하단의 두개의 꼭지점 좌표를 이용하여 쉽게 계산할 수 있다. 각 ER의 두 개의 꼭지점의 상대좌표는 8 개의 자세별로 사전에 정해져 있으므로 절대좌표는 기준점 좌표만큼 더하기만 하면 된다. 다음은 Count 함수를 나타낸다. 여기서 (xi,yi)는 Fig. 7에 표시한 바와 같이 형상 i의 좌상단 픽셀 위치를 나타내며, (xip,yip)는 우하단 픽셀 위치를 나타낸다. Fig. 7은 두 ER 사이 겹쳐지는 부분을 짙은 회색으로 나타내고 있으며, 여기서 두 ER간 겹쳐지는 픽셀 수, 즉 Count 값은 49이다.

```
Function: count
  if (xj ≦ xip) ∧ (xi ≦ xjp) ∧ (yj ≦ yip) ∧ (yi ≦ yjp)
    width = min(xip,xjp) - max(xi,xj) + 1
    height = min(yip,yjp) - max(yi,yj) + 1
    count = width * height
  else
    count = 0
```

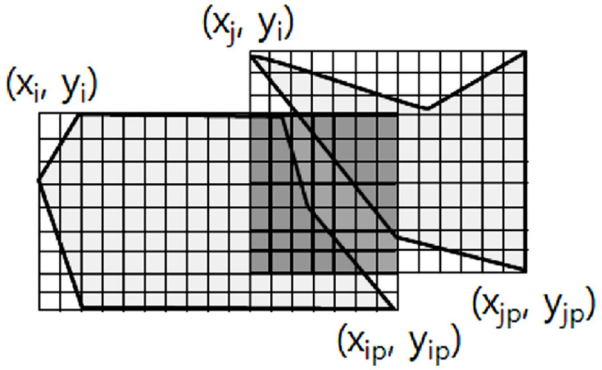


Fig. 7 Graphical representation of the evaluation function

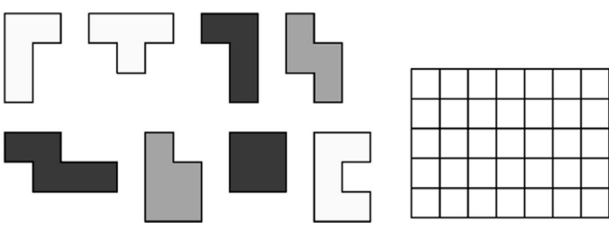
Goal 규칙은 해가 구해졌음을 판단하는 규칙으로 전체 부재의 위치가 다 정해지면 실행된다. 다음은 CLIPS로 나타낸 goal 규칙의 조건부의 일부를 보여준다. 여기서, Lay-tile-list의 값 $?tlist는 배치해야 할 부재 전체 리스트이며 Node의 Tile-list 슬롯 값과 일치하면 모든 부재가 배치되었음을 나타내므로 Goal 규칙은 노드가 해에 이르렀을 때 실행됨을 알 수 있다.

```
(defrule MAIN::goal
   ?open <- (open-list (nodes $?list))
   (lay-tile-list $?tlist)
   (node      (node-id ?nid)
              (tile-list $?tlist)
              (tile-orient-list $?torlist)
              (tile-position-row $?prlist)
              (tile-position-col $?pclist))
```

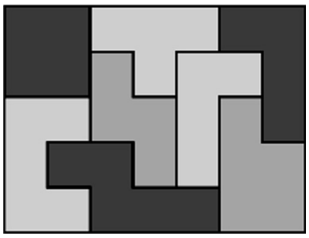
4. 적용 예

4.1 블록 퍼즐

블록 퍼즐은 Fig. 8(a)와 같은 블록들과 빈 보드를 이용하여 (b)와 같이 채우는 게임이며, 블록들의 회전은 허용되지 않는다.



(a) Blocks and board



(b) a solution

Fig. 8 An example of block puzzle

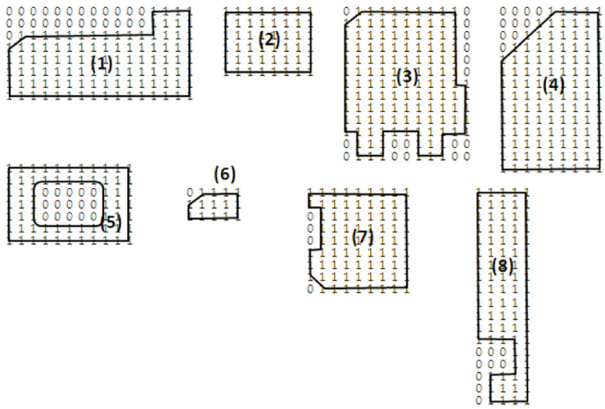
```
11   111   11   10   1100   10   11   11
10   010   01   11   0111   11   11   10
10         01   01          11        11
```

(a) Patterns

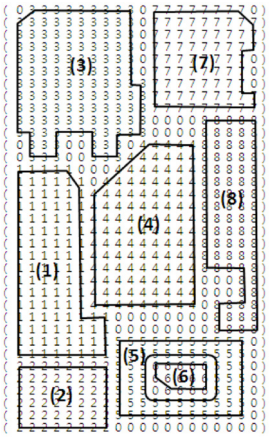
```
(7 7 2 2 2 3 3)
(7 7 4 2 1 1 3)
(8 8 4 4 1 6 3)
(8 5 5 4 1 6 6)
(8 8 5 5 5 6 6)
```

(b) A solution

Fig. 9 Patterns and a solution generated by the system



(a) Patterns



(b) A solution

Fig. 10 Patterns and a solution

따라서, 하나의 블록을 배치할 때 경우를 나타내는 식(1)은 식(3)과 같이 된다.

$$n_r = n(r, i, B_{r-1}) \tag{3}$$

일반적인 네스팅 문제와 달리 빈 공간을 최소화 하는 최적화 문제가 아니라 사각평판에 형상들이 빈 곳 없이 채워져야 하는 정답이 있는 문제로서 NFP등을 이용한 일반적인 네스팅 알고리즘으로는 풀기 어렵다.

본 연구에서 Fig. 8(a)와 같은 형상들은 Fig. 9(a)와 같이 이진(Binary) 패턴으로 ER안에 표시한다. 각 형상을 구별하기 위하여 1~8의 숫자로 패턴을 표현하였으며, Fig. 8(b)와 동일한 배치를 나타낸다.

4.2 일반 네스팅

일반 네스팅 문제에서는 회전 및 대칭을 통하여 한 형상당 8개의 자매 형상이 만들어 진다. 퍼즐 문제와 달리 평판을 완전히 채우는 것인 일반적으로 불가능하며 따라서 배치 결과 여백을 최소로 하는 해를 찾는다. 본 시스템에서는 주어진 평판의 크기에 맞추어 해를 찾는다. Fig. 10(a)는 여러 형상들을 ER 안에 픽셀 형태로 표현한 것이다. 부재가 픽셀에 가득 차 있거나 걸쳐져 있는 곳은 '1', 빈 곳은 '0'으로 표현된다. 시스템이 실행되면 먼저 각각의 형상에 대하여 회전 및 미러링에 의하여 자매형상들 간에 중복 없이 최대 7 개의 추가 자매 형상들이 생성된다. 네스팅은 각각의 자매 형상 그룹에서 하나씩 골라 이루어진다. 탐색 트리에서 한 노드에 하나의 형상을 추가할 때마다 배치 가능한 모든 경우의 수를 생성하며 이들은 식(2)의 평가 함수 값이 큰 순서로 정렬하여 Open-list의 맨 앞에 저장함으로써 그 순서대로 탐색이 이루어지도록 하도록 한다. Fig. 10(b)는 35행 x 20열 크기의 평판에 네스팅한 결과를 보여주며 해 위에 실제 형상을 중첩하여 나타내었다.

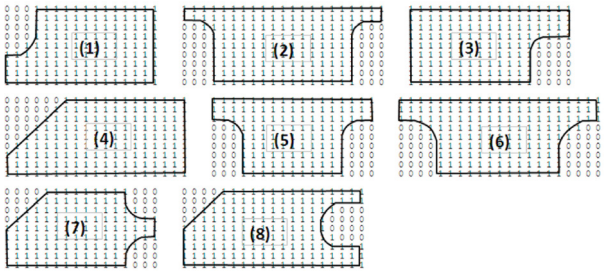


Fig. 11 Patterns

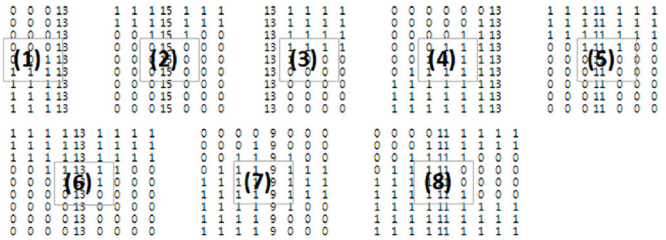


Fig. 12 The patterns after initialization

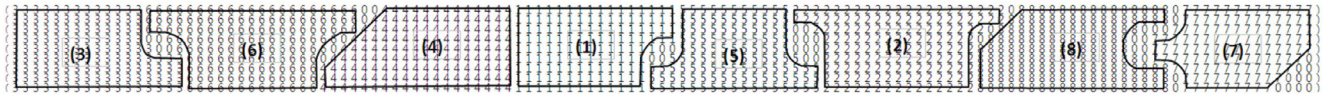


Fig. 13 A Solution

일반적인 NFP 알고리즘을 이용한 네스팅에서는 구멍이 있는 형상이나 오목한 형상의 경우 보통 구멍이나 오목한 부분을 채워 단순한 볼록 형상을 만든 후 배치를 한다. 본 시스템에서는 그러한 제약이 없으며 Fig. 10(b)에서 형상 (5)와 같은 형상의 구멍 안에 다른 형상 (6)이 제약 없이 배치가 됨을 확인할 수 있다.

4.3 조선 강재류 네스팅

조선산업에서 블록 보강재류 네스팅 문제는 먼저 폭이 같은 부재들끼리 분류한 후 각 그룹에 대하여 동일 폭을 갖는 긴 판재에 부재들을 배치를 한다. 따라서, 0^o,180^o회전 형상, 그리고 각각에 대한 미러 이미지의 4개 형상이 하나의 형상 가족이 된다. N 개의 형상 가족에 대하여 배치 위치를 배제하고 정렬 순서만을 고려해 보면 4^NN!가지의 경우의 수가 있다. 예를 들어 8 개의 부재를 배치할 때 약 26억가지 이상의 경우의 수가 발생한다. 따라서 기존의 연구들에서는 큰 부재부터 한 쪽 끝부터 순차적으로 연속 배치하는 방식을 사용하였으나 본 연구에서는 각 부재를 배치할 때 이전 부재에 이어서 배치하는 것이 아니라 제약 없이 전 평판을 활용하여 배치를 한다.

두 형상 간 중첩 여부는 양 단의 경계선 형상만 고려하면 되므로 Fig. 11의 형상은 Initialize 규칙에 의해서 Fig. 12와 같이 코딩된다. 형상 내에서 세로 폭이 평판의 세로 폭과 같은 부분은 해당 부분의 가로 폭만을 기록하는데, 예를 들어 (1)번 형상의 경우 Fig. 11에서 전체 열이 '1'로 코딩된 이웃한 열들은 모두 합하여 13 개의 열이므로 Fig. 12에서 하나의 대표 열에 13으로 코딩 되어 있음을 알 수 있다.

각 형상들이 Fig. 12와 같이 코딩된 후에는 형상들이 배치될 평판 소재도 그 만큼 가로 폭을 줄인 상태에서 배치하며 해를 찾은 후에는 평판 소재 및 각 형상들은 원래 가로 폭으로 복원한다. 형상을 배치할 때 기본적으로 그 형상이 중첩 없이 놓일 수 있는 모든 위치를 고려하게 되므로 이와 같은 코딩을 통하여 각 형상들이 배치될 위치에 대한 경우의 수를 줄일 수 있다. Fig. 13은 배치된 결과를 형상들과 중첩시켜서 보여주고 있다.

4.4 이형 경계형상을 갖는 소재에 네스팅

각 형상을 배치할 소재가 직사각형 형상이 아닌 경우 기존의

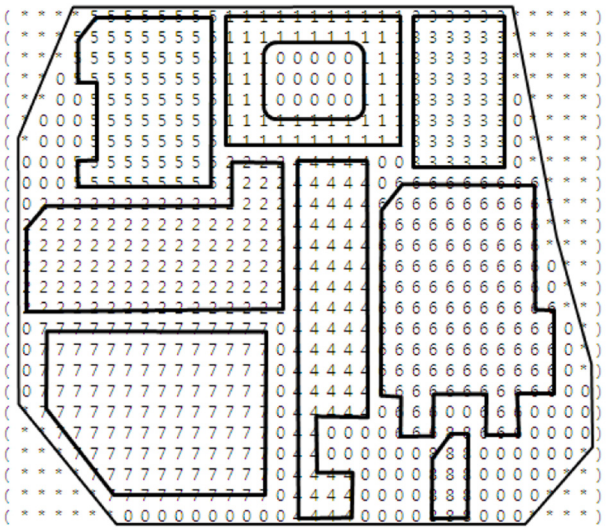


Fig. 14 A solution for nesting on a irregular shaped stock

알고리즘의 경우 어려움이 있어 주요 연구주제가 되어 왔다 (Tay et al., 2002; Lee et al., 2008; Burke et al., 2007). 본 시스템에서는 사각부재에 잘려나간 부분을 사전에 배치된 형상으로 처리함으로써 간단히 처리할 수 있다. Fig. 14는 이형 경계형상을 갖는 소재에 형상들을 배치한 실행 결과를 보여준다.

5. 결 론

본 연구에서 개발한 네스팅 전문가 시스템은 평가함수를 활용한 휴리스틱 탐색을 통하여 해를 비교적 짧은 시간(예제들의 경우 수천에서 수만개 이내의 노드 탐색)에 제시할 수 있었다. 퍼즐문제, 일반 네스팅 문제, 그리고 조선 강재류 네스팅 예제에 적용하여 그 적용 가능성을 보였으며, 각 문제별로 특성을 고려하여 문제를 효과적으로 해결할 수 있음을 설명하였다. 또한 형상들을 배치할 평판 소재의 형상이 사각 판넬이 아닐 경우에도 가상의 사각 판넬을 경계로 실제 경계 형상부분까지 이미 다른 형상이 배치되어 있는 상태로 처리하면 되므로 평판 소재의 형

상도 문제가 되지 않음을 예제를 통하여 보였다. 강재 네스팅 문제 등에서 고려해야 할 형상 간 공구 경로 확보 문제는 각 형상에 공구경로 폭에 해당하는 Offset을 추가하는 방법으로 해결될 수 있다.

전문가 시스템은 일반 프로그램과 달리 시스템에 대한 전체적 수정 없이 새로운 규칙을 지식베이스에 추가하는 방법으로 시스템 성능을 쉽게 개선할 수 있으며 따라서 본 시스템도 경험 지식 등에 의한 새로운 규칙을 추가하므로써 지속적 성능 개선이 가능할 것으로 기대된다. 또한, 본 연구에서 사용한 픽셀을 이용한 형상표현 방법을 이용할 경우 상업용 CAD 시스템과 쉽게 연계할 수 있으며, 오목한 형상이나 구멍이 있는 형상들을 배치하는데 도형의 근사형상을 이용한 방법과 달리 볼록 형상으로 근사화해야 하는 등의 제약이 없다.

본 시스템에서는 정해진 원소재 크기에 맞추어 배치를 하게 되므로 원하는 수율에 맞추어 평판 크기를 정하면 해를 구할 수 있다. 다만 휴리스틱에 의한 탐색을 하므로 해를 찾는 시간(또는 탐색 노드 개수)을 제한하면 해를 찾지 못하는 경우가 발생할 수 있으며, 그 경우 탐색 제한 시간을 늘리거나 평판 소재 크기를 키우면 해를 비교적 쉽게 찾을 수 있다.

일반적인 네스팅 문제는 NP-complete 문제로서 형상 개수가 늘어날수록, 표현의 정밀도를 높이기 위하여 픽셀의 갯수가 늘어날수록 최적해를 구하는 것은 거의 불가능하다. 본 연구에서 개발한 시스템도 평가함수를 활용한 휴리스틱 탐색 기법으로 탐색 시간을 줄였으나 형상의 개수가 늘어나거나 형상 표현 정밀도를 높이기 위하여 픽셀 수를 늘릴 경우 탐색 시간이 길어진다. 따라서 이러한 경우에는 픽셀 수를 줄여 해상도를 낮추거나 형상들을 일정 단위로 묶어 그룹별로 네스팅 한 후 그룹 간 네스팅하는 방법 등을 강구할 수 있을 것이다.

참 고 문 헌

이철수, 박광렬 (1996). “선박용 플랫바의 자동네스팅 및 가스/플라즈마에 의한 NC 절단”, 산업공학, 제9권, 제3호, pp 283-297.

한국찬, 나석주 (1994). “레이저 절단공정에서 절단부재의 최적 배치를 위한 네스팅 알고리즘”, 대한용접학회지, 제12권, 제2호, pp 11-19.

한창봉, 박제웅 (1999). “파라메트릭 매크로를 이용한 부재생성 및 네스팅에 관한 연구“, 한국해양공학회지, 제13권, 제2호, pp 176-185.

Babu, A.R. and Babu, N.R. (2001). “A Generic Approach for Nesting of 2-D Parts in 2-D Sheets using Genetic and Heuristic Algorithms”, Computer-Aided Design, Vol 33, pp 879-891.

Bennell, J.A. and Oliveira, J.F. (2008). “The Geometry of Nesting Problems: A Tutorial”, European Journal of Operational Research, Vol 184, pp 397–415.

Burke, E.K., Hellier, R.S.R., Kendall, G. and Whitwell, G. (2007). “Complete and Robust No-fit Polygon Generation for the Irregular Stock Cutting Problem”, European Journal of Operational Research, Vol 179, pp 27-49.

Lee, W.-C., Ma, H. and Cheng, B.-W. (2008). “A Heuristic for Nesting Problems of Irregular Shapes”, Computer-Aided Design, Vol 40, pp 625-633.

Tay, F.E., H. Chong, T.Y. and Lee, F.C. (2002). “Pattern Nesting on Irregular-shaped Stock using Genetic Algorithms”, Engineering Applications of Artificial Intelligence, Vol 15, pp 551-558.

Weng, W.C. and Kuo, H.C. (2011). “Irregular Stock Cutting System Based on AutoCAD”, Advances in Engineering Software, Vol 42, pp 634-643.

2012년 5월 17일 원고 접수
2012년 7월 10일 심사 완료
2012년 7월 12일 게재 확정