타부서치를 이용한 2차원 직사각 적재문제에 관한 연구

이상헌* · 이정민**

Applying a Tabu Search Approach for Solving the Two-Dimensional Bin Packing Problem

Sang-Heon Lee* · Jeong-Min Lee**

Abstract

The 2DBPP(Two-Dimensional Bin Packing Problem) is a problem of packing each item into a bin so that no two items overlap and the number of required bins is minimized under the set of rectangular items which may not be rotated and an unlimited number of identical rectangular bins. The 2DBPP is strongly NP-hard and finds many practical applications in industry. In this paper we discuss a tabu search approach which includes tabu list, intensifying and diversification strategies. The HNFDH(Hybrid Next Fit Decreasing Height) algorithm is used as an internal algorithm. We find that use of the proper parameter and function such as maximum number of tabu list and space utilization function yields a good solution in a reduced time. We present a tabu search algorithm and its performance through extensive computational experiments.

Keyword : Two-Dimensional Bin Packing Problem, Tabu Search, Meta Heuristic, Cutting

1. 서 론

2차원 직사각 적재문제(2DBPP : Two-Dimensional Bin Packing Problem)는 물품/물류관리에서 발생하는 일상적인 적재 활동을 기술하는 용어로써 작은 개체(items)를 고정된 공간(bins or objects)에 적재하는 행위를 포함한다. 본 문제는 규정된 크기의 유리나 금속 원판을 직사각 형태로 자

논문접수일 : 2004년 2월 24일 논문게재확정일 : 2005년 4월 30일
* 국방대학교 운영분석학과
** 국방부

르는 물품 절단(stock cutting)과 동일한 의미로 사용된다. 절단된 조각의 형상은 원판 자체보다 크지 않은 무작위 크기의 직사각 형태로써, 여기서 제기되는 문제는 요구되는 조각 개수를 맞추기 위해서 표준 크기인 원판을 얼마나 최소로 사용하느냐는 것이다. 만일 효과적인 절단 계획이 수립되지 않는다면 낭비되는 조각이 많아져 결국 원 재료비 상승을 초래할 것이다. 또한 같은 문제로써 도심에 위치한 공장에서 효과적인 적재 계획을 수립하지 않는다면 저장시설 확충에 따른 부수비용이 필요할 것이며, 트럭 수송의 경우 불필요한 운송으로 인해 비용이 낭비될 것이다. 위 예와 같이 절단(cutting)과 적재(packing) 문제는 상이한 제약 상황 및 목적으로 서로 다른 산업 시설에서 흔히 볼 수 있다. 예를 들어 나무, 유리, 종이 산업은 일반적으로 표준화된 형상의 절단 작업이 요구되며 수송, 선박배치, 재직, 가죽산업 등에서는 비표준 및 무작위 형상의 적재작업이 요구된다(Dyckhoff[10]).

이러한 유형의 문제는 1960년대 이후 지속적으로 연구되어 1차원 및 2차원 이상의 최적 알고리듬이 Gilmore and Gomory[13-15]에 의해 최초로 연구되었다. 보다 세밀한 연구로써 Golden[18]은 2차원 절단(cutting stock)문제에 대해 실험하였고, Hinxman[19]은 2차원 조각손실(trim loss) 및 배치(assortment) 문제에 대한 해법을 실 공정에 적용시켰으며, Rayward-Smith and Shing[25]에 이르러 '1차원 및 2차원 적재 문제'라는 용어가 사용되었다. 3차원 직사각 적재문제에 관한 연구는 Dowsland[6]가 시초로 Dowsland, K and Dowsland, W[7]에 의해 확장되었다. Whelan and Batchelor [28]는 불규칙한 개체에 대한 적재를 연구하였으며 Dowsland, K and Dowsland, W[8]에 의해 상대적으로 우수한 해법이 개발되었다.

위와 같은 기법들은 모두가 최적해를 구하기란 거의 불가능한 NP-complete문제로써 지역 최적해에 빠지는 단점이 있으므로 이를 극복하기 위해 메타휴리스틱(meta-heuristic) 기법에 대한 연구가 이루어져 왔다.

메타휴리스틱 기법은 단순 휴리스틱으로 초기해를 구한 후 이를 개선하는 형태로 발전하였는데, 예를 들어 Smith[27]는 초기해를 슬라이드(slide) 알고리듬으로 구한 후 유전자 알고리듬(Genetic Algorithm)을 이용하여 최적해를 구했으며 Jacobs [22]와 Liu and Teng[24]은 개체가 적재되는 순서를 조합(permutation)의 형태로 표현하고 적재 위치를 Bottom-Left 알고리듬으로 구현하였다. 이러한 표현 기법은 교차(crossover) 연산자와 돌연변이(mutation) 연산자의 효과적 활용에 근거한 유전자 알고리듬을 통하여 유용한 기법임이 밝혀졌다.

Kapmke[23]는 1차원 적재문제에 시뮬레이티드 어닐링(Simulated Annealing)기법을 적용하였으며, Dowsland[6]에 의해 동일 개체 및 다른 크기의 개체를 팔렛트(pallet)에 적재하는 실험으로 확장되었다. Dowsland는 2차원 무한 적재 문제(strip packing)를 다루었는데, 적재 가능해(feasible solution)와 불가능해(overlapping solution)를 동시에 고려하므로, 탐색간 목적함수를 총 겹치는 영역과 수직 또는 수평 개체 이동에 대응하는 모든 해를 포함하는 이웃으로 선정하였다. 이를 통하여 현재 해를 개선시키는 새로운 가능 해가 발견되면 상한기준(upper bound) 높이를 갱신하여 해를 발전시켰다.

타부서치(Tabu Search)에 관한 연구는 Blazewicz 등[4]이 최초로 시도하였다. Blazewicz 등은 단순한 적재 절차로써 가능해를 생성한 후 타부서치를 이용하여 기존 레이아웃(layout)을 개선시키는 방법을 취하였는데, 개체 하나를 임의로 선택한 후 가능한 수많은 목표지점을 선별하여 가장 좋은 곳에 적재한다. 여기서 이동(move)은 해당 개체가 다른 개체와 겹치지 않으면서 적재됨을 의미하며 최근에 이동한 개체가 타부리스트(tabu list) 원소가 된다. Blazewicz 등[4]의 이론적 연구와는 달리 Lodi 등[2]은 현실에서 사용가능한 직사각 적재 알고리듬을 개발하였다. Lodi 등은 두 가지 제한사항으로써 고정된 개체와 초기해로 생성된 레이아웃을 생성하였다. 위 가정에 적용된 타부서치의 중요한 특징은 특별한 적재문제에 독립적인 탐색 기법

(search scheme)과 이웃(neighborhood)을 적용하였다. 가장 우수한 해를 구하기 위한 절차로써 단순 휴리스틱 알고리듬으로 초기해(feasible solution)를 구한 후 이동을 통하여 적재된 개체들을 다시 분배하는 과정을 취한다. 이동은 k개의 특정 저장소에 적재된 부분 개체집합과 목표 저장소(target bin, 비우기 쉬운 저장소)에 적재된 단일 개체의 교환을 통해 이루어진다. 알고리듬은 자동으로 지역 최적해(local optima)로부터 벗어나기 위하여 탐색간 k를 지속적으로 갱신한다. 따라서 높은 수의 k는 보다 강력한 재결합을 의미하지만 동시에 이웃 탐색간 상당한 계산 시간을 요구한다. Lodi 등[2] 이 사용한 알고리듬은 유한공간 최초 적재(Finite First Fit)와 유한 공간 최적 적재(Finite Best Fit)이며 초기해를 개선하기 위하여 열거된 저장소중 비울 수 있는 가능성이 가장 높은 저장소를 해에서 삭제시키기 위해 개체 및 저장소의 크기(dimensions)와 임의상수(user defined constant)로 계산되는 채우기 함수(FF : Filling Function)를 활용하고, 타부서치의 비교 기준으로써 이웃수를 조절하는 최대 타부목록 개수를 모든 개체군에 동일하게 적용하였다.

본 논문에서는 2차원 직사각 적재문제의 최적해를 혼합 우측 적재 알고리듬(HNFDH : Hybrid Next Fit Decreasing Height Algorithm, Frenk and Galambos[12])으로 보다 최적해에 근접한 좋은 초기해를 생성한 후 타부서치로 해를 개선하기 위하여 공간 활용도 함수(space utilization function)를 사용하여 각 개체군에 적합한 최대 타부목록 개수를 동적(dynamic)으로 산정하고 강화 및 다양화전략을 구사함으로써 2차원 직사각 적재문제(2DBPP)에 대한 보다 효율적인 최적해 기법을 제시하고자 한다.

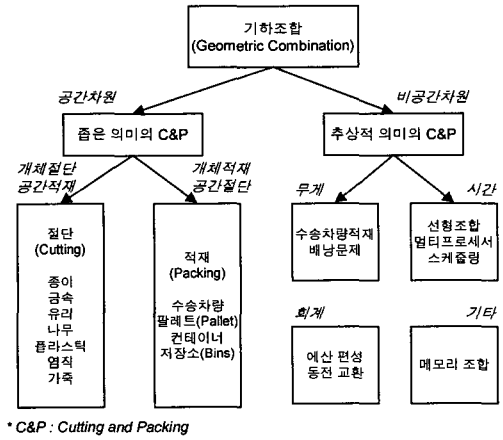
2. 문제의 모형화

2차원 적재문제는 여러 개체들의 적절한 조합을 고려하여 큰 저장소에 적재하는 최적화 문제로써 수많은 산업 시설에서 응용되었으며 단순히 작업 공정의 한 부분으로 제한되지 않고 OR(Operations Research)과 회계 분야 등을 통해 보다 추상적인 형태로 발전되었다. 적재문제는 유사한 문제의 다양성과 응용분야로 인해 동일한 분야이지만 다른 이름의 형태를 갖고 있다. 따라서 비록 이러한 문제들이 다른 응용분야에서 발견된다 할지라도 결국 같은 논리 구조를 갖게 된다. 상이한 학문분야 사이의 정보 교환을 통해서 Dyckhoff[10]는 공통의 특성을 발견하여 <그림 1>과 같은 분류구조(classification system)를 제안했다. 분류구조는 크게 공간 차원(spatial dimension) 및 비공간 차원(non-spatial dimension)과 관련된 문제로 나눌 수 있다. 첫 번째 그룹은 3차원까지 정의되는 유클리디안(euclidian) 공간 속에서 다발(bundle) 절단문제, 차량 로딩(loading) 문제, 팔레트(pallet) 로딩 문제 등과 같이 '절단과 적재' 또는 '로딩 문제'로 구성된다. 또 다른 그룹은 메모리 배치, 재정 회계, 동전 교환, 라인 밸런싱(line balancing) 등의 무게, 시간, 화폐 등과 같은 비 공간적 차원을 다루는 추상적인 '절단과 적재' 문제를 포함한다.

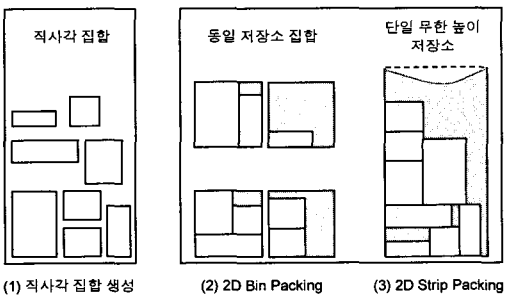
본 논문의 모형은 직사각 개체들을 동일한 형태의 저장소에 적재하고 각 저장소는 레벨(level) 형태를 이루는 2차원 직사각 적재문제로써 가정사항은 다음과 같다. 먼저 기하학적인 동질성을 보장하기 위해 모든 개체와 저장소는 직사각형태로 고정된 폭과 높이로 이루어져 있고 각 개체는 기타 개체 및 저장소의 모서리와 겹치지 않으며, 접합 공간은 0이다. 또한 수평으로 적재되고 회전하지 않으며 폭과 높이는 저장소의 폭과 높이보다 크지 않은 정수이다. 다음으로 레벨적재를 위한 가정사항으로써 각 레벨에서 최좌측 개체는 해당 레벨에서 높이가 가장 크고 단일 저장소의 최아래 레벨은 다른 레벨의 높이보다 크며 각 개체들은 높이에 따른 내림차순으로 정렬된 후 번호가 다시 부여된다.

<그림 1>과 같이 분류되는 적재문제는 2차원의 경우 <그림 2>와 같이 2차원 직사각 적재문제(2DBPP : 2-Dimensional Bin Packing Problem)

와 2차원 무한 적재문제(2DSPP : 2-Dimensional Strip Packing Problem)로 구분되며, 표준화된 크기의 직사각 형태로 적재 또는 절단된다.



〈그림 1〉 적재 및 절단에 대한 분류 구조

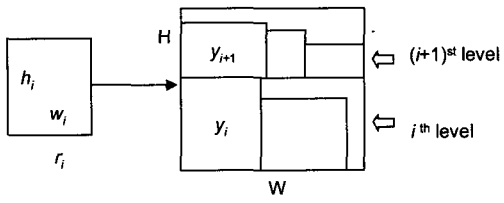


〈그림 2〉 2DBPP 분류

신문지 섹션 분할, 차량 적재 효율성 제고 등에 이용되는 <그림 2-(2)>는 동일한 크기의 저장소 또는 그렇지 않은 저장소로의 적재를 다루며, 일반적인 목적함수는 최소한의 저장소 개수를 찾는 것이다. 본 연구에서 고려하는 모형은 동일한 크기의 저장소를 다룬다. <그림 2-(3)>은 종이 및 염직 공정에서의 롤(roll) 형태인 단일 원 재료(raw material)에 대한 활용도를 최대한 높이기 위해 사용하며, 일반적인 목적함수는 원재료에 대한 사용을 최대화 하는 것으로서 결국 롤의 높이를 최소화 시킨다.

2.1 변수 정의

개체를 r_i라 정의하고 y_i를 i번째 레벨에 적재되는 개체라고 하자. 여기서 r_i가 최좌측에 위치한다면 y_i는 1이며 그렇지 않은 경우 0값을 갖는다. 개체 i에 의해서 초기화되는 i번째 레벨의 잠재적인 개수를 n, 레벨 k에 의해서 초기화되는 k번째 저장소의 개수를 마찬가지로 n이라 가정할 경우 <그림 3>을 참조할 때, 수리모형에서 사용되는 변수는 다음과 같다.



〈그림 3〉 레벨 적재

y_i : $i \in J$ (J는 직사각 개체집합)

q_k : $k \in J$ 레벨 k가 저장소 k를 초기화하면 1, 그렇지 않을 경우 0

H : 저장소의 높이

W : 저장소의 폭

h_i : 개체 i의 높이

w_i : 개체 i의 폭

2.2 수리 모형

2.1에서 정의한 변수를 이용하여 수리모형을 구성하면 목적함수인 식 (1)과 제약식인 식 (2)부터 식 (6)과 같이 수식화 할 수 있다(Frenk and Galambos[12]).

$$\text{Minimize} \sum_{k=1}^{n} q_k \tag{1}$$

Subject to

$$\sum_{i=1}^{j-1} x_{ij} + y_j = 1, \quad \forall j \tag{2}$$

$$\sum_{j=i+1}^{n} w_j x_{ij} \le (W - w_i) y_i, \quad (i = 1, \cdots, n-1) \qquad (3)$$

$$\sum_{k=1}^{i-1} z_{ki} + q_i = y_i, \quad \forall i \qquad (4)$$

$$\sum_{j=k+1}^{n} h_i z_{ki} \le (H - h_k) q_k, \quad (k = 1, \cdots, n-1) \qquad (5)$$

$$y_i, x_{ij}, q_k, z_{ki} \in \{0, 1\}, \quad \forall i, j, k \qquad (6)$$

위에서 x_{ij}와 z_{ki}는 각각 개체와 레벨을 나타내며 개체 j가 레벨 i에 적재될 때 x_{ij}는 1이고 그렇지 않을 경우 0이며, z_{ki}는 레벨 i가 저장소 k에 위치할 때 1이며 기타의 경우 0이다.

본 모형의 근본적인 목적은 개체 적재에 사용되는 총 저장소 개수(n)를 최소화하는 것이므로 식 (1)이 목적함수가 된다. 유효한 적재 상태를 구성하기 위한 제약식으로는 식 (2)~식 (6)이 사용되며, 식 (2)~식 (5)에서 i, j, k의 크기는 레벨 적재 가정사항에 의해 $j > i$와 $i > k$이 성립한다. 식 (2)는 각 개체가 정확히 한 번만 적재됨을 의미하며, 식 (4)는 각 레벨이 하나의 저장소에만 위치한다는 것을 뜻한다. 식 (3)과 식 (5)는 각각 사용된 레벨의 폭과 저장소의 높이를 제한한다.

3. 개체군에 적합한 최대타부목록 개수를 적용한 타부서치알고리듬

타부서치의 일반적인 구성요소로는(Golver[16, 17], Rolland 등[26]) '단기(short term)메모리와 장기(long term)메모리', '통제조건(tabu constraint)과 열망(aspiration)조건', '강화(intensifying)전략과 다양화(diversification) 전략' 등을 포함한다. 열망조건은 비록 타부목록에 들어있는 금지된 이동이라 할지라도 지금껏 발견된 최고해를 능가하는 이동이라면 타부목록에서 그 이동을 빼주어 다음 반복의 개선해로 채택되도록 한다. 강화전략은 단기메모리 함수를 사용하여 현재 상태보다 좋은 해를 탐색하도록 환경을 맞춘다. 이러한 전략은 제한된 영역에서 최상의 해를 적극적으로 탐색하는 데 초점을 두며, 다양화 전략은 장기메모리 함수를 이용하여 해 공간에서 방문하지 않았던 영역을 탐색하도록 한다. 단기메모리 함수는 타부목록과 열망조건으로 구현된다. 타부목록은 최근에 방문했던 곳의 이동을 기억하여 다시 탐색하지 않도록 한다. 이동은 일정기간 목록에 남아 있어서 보다 좋은 해를 적극적으로 탐색하도록 한다.

3.1 기본 알고리듬

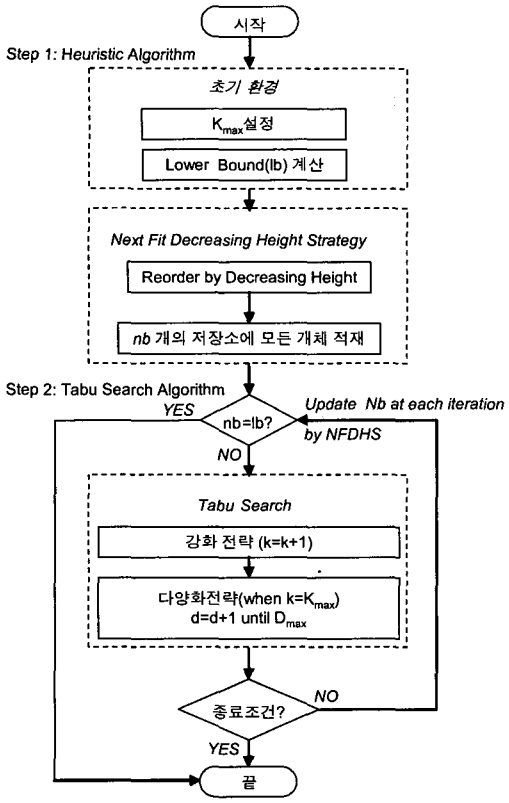
적재문제에 타부서치를 적용하기 위한 기본 알고리듬은 <그림 4>와 같으며 Step 1은 단순 휴리스틱 절차이고 Step 2는 타부서치 절차를 나타낸다. 본 절차는 유한공간 최초 적재(Finite First Fit)로 초기해를 구한 후 고정된(fixed) 최대 타부목록 개수와 채우기 함수(Filling Function)로 초기해를 개선한 Lodi 등[2]의 연구와 달리 적재되는 개체를 입력 받은 후, 이를 단순 휴리스틱 알고리듬으로 높이에 따라 내림차순으로 정렬하여 우측적재를 실시하며, 타부서치기법으로써 개체군에 적합한 최대 타부목록 개수를 산정하고 공간 활용도 함수로써 최적해를 산출한다.

순서도에서 $K_{\max}$는 타부서치 기법에서 금지되는 이동 목록을 저장하는 타부목록(tabu list)의 최대개수를 의미하고 k는 강화전략간 갱신되는 타부목록 개수를 의미한다. 식 (7)의 하한값(lb : lower bound)은 모든 개체의 총 면적을 단일 저장소 면적으로 나누어 올림한 이상적인 최적해이다. 따라서 하한값은 어떠한 최종해 보다 반드시 작거나 같기 때문에 종료 조건이 된다. 또한 다양화 전략의 $D_{\max}$는 허용되는 다양화 전략의 최대반복횟수이며 d는 갱신되는 다양화 전략의 횟수이다.

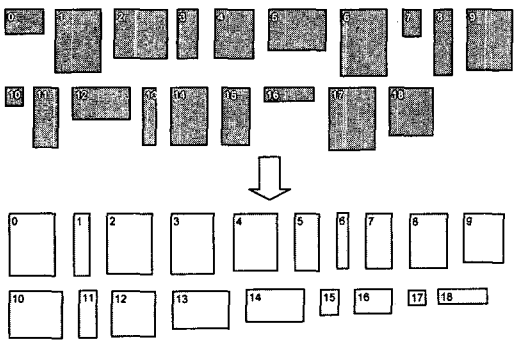
$$lb = \left\lceil \frac{\sum_{i=1}^{n} (w_i \times h_i)}{W \times H} \right\rceil \qquad (7)$$

Step 1에서 적용된 단순 발견 알고리듬은 초기해 생성을 위해 이용하고 타부서치 구동시 매 단

계마다 반복되며, 모든 개체를 정렬한 후 이용하는 오프라인(off-line) 알고리듬이다. 계산시간은 $O(n \log n)$이며 특정 문제에 대해 이상적으로 사용할 수 있는 저장소 개수를 M이라 할 때 $2M$개 이하만을 사용한다.

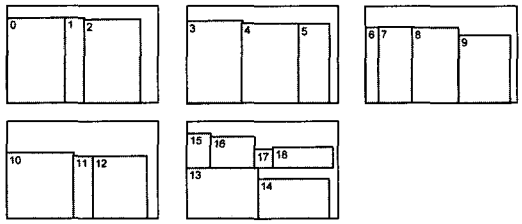


〈그림 4〉 순서도



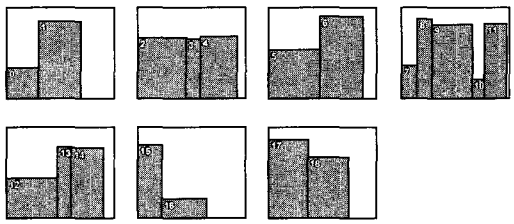
〈그림 5〉 내림차순 정렬 예

<그림 5>는 무작위로 생성된 개체(random items)를 높이에 따라 내림차순으로 정렬한 상태(reordered items)를 의미하며 <그림 6>은 정렬된 개체를 우측 적재 알고리듬으로 적재한 결과(level packing)를 나타낸다.



〈그림 6〉 우측 적재 예

<그림 6> 예제의 결과 총 저장소는 5개 소요된 반면, 내림차순으로 정렬하지 않은 개체를 적재할 경우(normal packing) 일반적으로 <그림 7>과 같이 필요한 저장소 개수는 늘어난다.



〈그림 7〉 일반 적재 예

Step 2의 타부서치 절차는 초기해가 lb와 같지 않을 경우 작동하며 강화 전략과 다양화 전략을 반복하여 초기해를 개선시킨다.

3.2 공간 활용도 함수

타부서치를 반복할 때마다 식 (8)의 각 저장소의 활용도 함수(S_U)가 가장 낮은 저장소(wb : the weakest bin)를 선별하여 개체들을 다른 저장소에 적재시키면 wb를 기존해(nb)에서 삭제시킬 수 있다. 이때 구한 해($nb-1$)가 종료 조건($nb = lb$)을 만족하지 않는다면 각 저장소의 S_U값은 갱신된 상

태에서 다음 타부서치의 비교 기준이 된다.

$$S_U = \frac{W \times H - S_S}{W \times H} \quad (8)$$

식 (9)의 k는 해당 저장소의 총 개체수이며, 여유 공간(S_S : slack area)을 계산한다.

$$S_S = W \times H - \sum_{i=1}^{k} w_i \times h_i \quad (9)$$

본 알고리듬의 종료 조건은 하한값에 도달한 경우, 제한시간(time limit)이 초과된 경우, 그리고 최대 다양화 반복 횟수에 도달한 경우이다.

3.3 단기 메모리를 이용한 강화 전략

초기해가 하한값(lb)과 같지 않을 경우 최초로 강화전략을 취하는데, 이는 타부목록의 크기를 k라 할 때, 최초 타부서치가 끝난 뒤($k=1$) 얻어진 최적해가 하한값과 같지 않을 경우 이웃의 크기를 동적으로 변환($k=k+1$)시켜 최대 이웃 개수에 도달($k=K_{\max}$)할 때까지 반복한다. 이를 요약하면 다음과 같다.

Procedure Intensifying Strategy :
  If nb is greater or equal to lb **then**
    $k=k+1$;
  End If
Return.(Until $k=K_{\max}$)

3.4 장기 메모리를 이용한 다양화 전략

다양화 전략은 해공간에서 방문하지 않았던 새로운 영역을 탐색하기 위한 방법으로써 지역 최적해를 벗어나기 위해 사용되며 보다 넓은 해 영역을 탐색하도록 구현한다. 이의 구현은 더 이상 강화전략만으로 해를 개선시킬 수 없을 경우 총 저장소(nb) 중 가장 작은 공간 활용도 함수 값을 가진 저장소의 1/2을 해에서 제거한 후 다시 일련의 과정으로 적재 행위를 시행하여 그 동안 탐색하지 못한 다른 영역을 탐색하도록 한다. 이 과정은 다음과 같다.

Procedure Diversification Strategy :
  If $d \le nb$ and $d < D_{\max}$ **then**
    $d=d+1$;
    let t be the bin with d-th smallest value of S_U ;
  Else
    remove from the solution the $\lfloor nb/2 \rfloor$
      bins with smallest S_U value;
    pack into a separate bin each item
      currently packed in a removed bin;
    reset all tabu lists to empty;
    $d=1$;
Return.

4. 알고리듬 실행 및 분석

프로그램은 Visual C++6.0으로 작성하여, Pentium III 930Mhz, 메모리 256M인 환경에서 구동했다. 총 21,000개의 개체에 대해 $K_{\max}$를 변화시키면서 최적해와 CPU 계산 시간을 측정했고, 기본 환경으로 다양화 전략을 2회까지 허용하였다.

4.1 실험 계획

본 실험을 위한 개체는 <표 1>과 <표 2>의 생성 기준으로 Visual C++ 6.0 seed를 적용하였다. Class Ⅰ~Ⅲ은 Berkey and Wang[3]이 제시한 개체군이며, Class Ⅳ~Ⅶ은 Lodi 등[2]이 Class Ⅰ~Ⅲ을 확장시킨 것으로써 <표 2>의 형(Type) 기준에서 저장소의 폭과 높이 비율을 임의로 한정하여 보다 다양한 형태의 개체를 생성하였다.

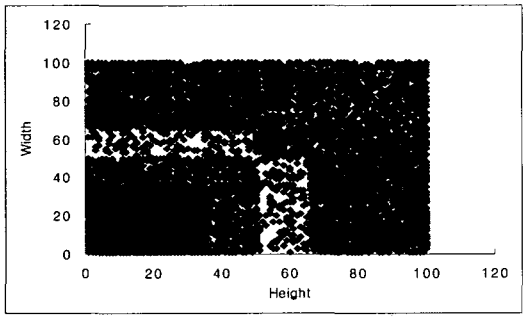
각 Class는 20, 40, 60, 80, 100개의 개체로 구성되며, 각 개체군은 통계를 위해 10개의 인스턴트를 생성한다. 따라서 하나의 Class는 3,000개의 개체로 구성된다. <그림 8>은 <표 1>과 <표 2>의 기준에 의해 생성된 21,000개의 개체 분포로써 높이와 폭이 모두 0~100이내에 위치한다.

〈표 1〉 개체 생성 기준 I

Class	저장소 크기	각 개체
I	$W_1 = H_1 = 10$	구간 [1, 10]에서 무작위로 개체생성
II	$W_2 = H_2 = 40$	구간 [1, 35]에서 무작위로 개체생성
III	$W_3 = H_3 = 100$	구간 [1, 100]에서 무작위로 개체생성
IV	$W_4 = H_4 = 100$	Type 1 확률 70%, Type 2,3,4 확률 각각 10%
V	$W_5 = H_5 = 100$	Type 2 확률 70%, Type 1,3,4 확률 각각 10%
VI	$W_6 = H_6 = 100$	Type 3 확률 70%, Type 1,2,4 확률 각각 10%
VII	$W_7 = H_7 = 100$	Type 4 확률 70%, Type 1,2,3 확률 각각 10%

〈표 2〉 개체 생성 기준 II

형(Type)	각 개체
Type 1	w_j는 $\left[\frac{2}{3}W, W\right]$, h_j는 $\left[1, \frac{1}{2}H\right]$의 구간에서 무작위 개체생성
Type 2	w_j는 $\left[1, \frac{1}{2}W\right]$, h_j는 $\left[\frac{2}{3}H, H\right]$의 구간에서 무작위 개체생성
Type 3	w_j는 $\left[\frac{1}{2}W, W\right]$, h_j는 $\left[\frac{1}{2}H, H\right]$의 구간에서 무작위 개체생성
Type 4	w_j는 $\left[1, \frac{1}{2}W\right]$, h_j는 $\left[1, \frac{1}{2}H\right]$의 구간에서 무작위 개체생성



〈그림 8〉 개체 분포

4.2 실험 결과

본 실험은 Lodi 등[2]의 기존 연구(CODE : FORTRAN 77, HARDWARE : Silicon Graphics INDY R4000sc 100MHz)와 동등한 환경에서 테스트 할 수 없는 제한사항과 단순 발견 알고리듬으로써 'Finite First Fit/Finite Best Fit' 알고리듬을 동일하게 구축할 수 없는 한계로 인하여, 초기해 생성을 위한 혼합 우측적재 알고리듬을 Lodi 등[2] 연구와 본 논문에 동일하게 적용하고 타부서치 수행간 취약 저장소를 선별한 후 최종 저장소 개수를 줄이는 방법으로써 공간 활용도 함수를 사용하여 CPU 계산시간을 향상시키며 각 개체군에 적합한 최대타부목록 개수를 산정하여 보다 효율적인 최적해를 유도하여 비교한다.

<표 3>은 3장에서 제시한 알고리듬을 7개의 개체군에 적용한 계산결과를 나타낸다. 표 안의 각 행은 개체군에 대한 세부 결과이며 첫 번째 열은 개체군(class), 두 번째 열은 개체수, 세 번째 열은 하한값(*lb*), 네 번째 열은 단순 발견 알고리듬(HNFDH)에 의한 초기 해를 의미한다. 다섯 번째 열은 Lodi 등[2]에 의한 기존 연구결과로써 HNFDH으로 초기해를 구한 후 $K_{\max}$를 고정($K_{\max} = 3$)시켜 채우기 함수(FF)로 취약 저장소를 선별한 결과이다. 마지막 열은 본 논문에서 제시하는 알고리듬의 산출결과로써 마찬가지로 HNFDH으로 초기해를 구한 후 개체군에 따라 $K_{\max}$를 동적으로 변화시키면서 공간활용도 함수로 취약 저장소를 선별한 것이다.

이에 대한 결과로 전반적으로 향상된 CPU 계산시간을 유도할 수 있었으며, 특히 개체수가 40 이하 일 때, 보다 우수한 해가 생성되었다. 각 Class 별 발견된 우수한 해의 분포는 <표 4>와 같으며, 개체 수에 대한 $K_{\max}$와 저장소 개수의 변화는 <그림 9>와 같다.

<표 4>에서 두 번째 열은 Filling Function으로 취약저장소를 선별한 경우의 CPU 계산시간이며 세 번째 열은 공간활용도 함수로 취약 저장소를 해에서 삭제시킨 경우이다. 개체수가 20~40인 경우

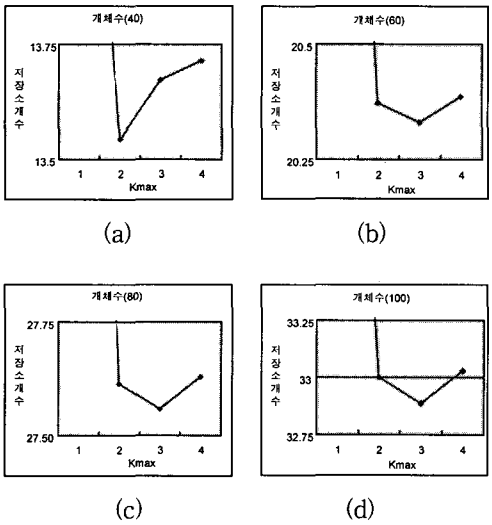
〈표 3〉 Class Ⅰ~Ⅶ 실험결과 비교

개체군 (Class)	개체수	하한값	초기해	HNFDH/ Fixed K_{max}		HNFDH/ Dynamic K_{max}	
				최종해 (평균)	평균 CPU 시간(sec)	개선된 해 (평균)	평균 CPU 시간(sec)
Ⅰ	20	6.4	9.5	7.3	0.41	7.3	0.32
Ⅰ	40	12.0	18.1	13.9	4.34	13.7	2.02
Ⅰ	60	18.5	28.4	20.6	15.48	20.6	15.37
Ⅰ	80	25.3	38.6	28.3	24.59	28.3	23.96
Ⅰ	100	30.5	47.1	33	57.55	33	54.27
Ⅱ	20	4.4	7.2	5.6	0.43	5.6	0.40
Ⅱ	40	8.2	13.9	10	2.70	9.8	1.22
Ⅱ	60	12.5	22	14.8	8.90	14.8	8.82
Ⅱ	80	17.3	30.2	20.5	12.18	20.5	11.90
Ⅱ	100	20.5	36.2	23.9	25.57	23.9	24.13
Ⅲ	20	5.4	8.3	6.8	0.46	6.8	0.37
Ⅲ	40	10.1	16.9	12.4	3.43	12.4	1.39
Ⅲ	60	15.7	26.4	18.6	11.61	18.6	11.51
Ⅲ	80	21.5	36.5	25.6	23.51	25.6	22.98
Ⅲ	100	25.9	43.9	29.7	30.49	29.7	28.96
Ⅳ	20	4.7	6.5	5.8	0.35	5.8	0.30
Ⅳ	40	9.7	14.4	11.6	3.43	11.5	1.44
Ⅳ	60	14.0	21.3	16.5	10.38	16.5	10.27
Ⅳ	80	19.7	30.6	23.8	25.05	23.8	24.50
Ⅳ	100	23.8	36.2	28.1	37.25	28.1	35.15
Ⅴ	20	4.8	6.9	5.9	0.49	5.9	0.42
Ⅴ	40	9.6	14.6	11.7	4.66	11.6	1.88
Ⅴ	60	14.1	20.9	17	21.64	17	21.48
Ⅴ	80	19.5	29.3	23.1	42.54	23.1	41.34
Ⅴ	100	24.1	35.8	28.6	95.94	28.6	91.70
Ⅵ	20	9.8	16	14.8	2.85	14.8	0.98
Ⅵ	40	18.0	31	27.8	18.25	27.7	3.84
Ⅵ	60	27.6	47.9	43.7	70.30	43.7	69.67
Ⅵ	80	37.1	62.9	57.6	178.41	57.6	173.51
Ⅵ	100	45.0	77.2	69.6	302.32	69.6	293.47
Ⅶ	20	3.8	5.4	4.6	0.29	4.6	0.27
Ⅶ	40	6.9	10.9	8.3	2.09	8.2	1.23
Ⅶ	60	9.4	14.9	11.1	5.87	11.1	5.81
Ⅶ	80	12.2	19	14	12.30	14	12.05
Ⅶ	100	15.3	23.6	17.3	11.53	17.3	11.03

계산 시간의 개선 비율이 대폭적으로 증가한 이유는 K_{max}가 2이기 때문이며, 개체수 60이상일 때 점차적으로 향상됨을 알 수 있다.

〈표 4〉 향상된 CPU 계산 시간

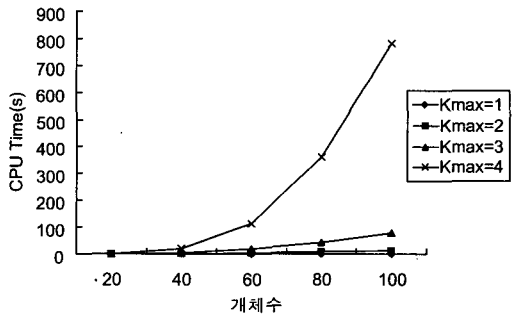
개체수(Item)	Time (Lodi *et al.*)	Improved Time	Improved Ratio(%)
20	0.75	0.44	70.45
40	5.56	1.86	198.92
60	20.60	20.42	0.88
80	45.51	44.32	2.69
100	80.30	76.96	4.34



〈그림 9〉 K_{max}와 개체 수에 따른 저장소 관계

<그림 9>에서 보듯이 K_{max}가 1일 때 저장소를 가장 많이 사용하여 항상 나쁜 해가 생성되며, 4인 경우 역시 2와 3에 비해 상대적으로 우수하지 못한 해가 생성됨을 보여준다. 이는 타부목록이 너무 작으면 해의 순환이 일어나 효율적인 탐색이 어려우며, 반대로 클 경우 너무 많은 움직임이 금지되기 때문에 해의 질이 나빠지기 때문이다[1].

<그림 10>은 K_{max}에 따라 소요되는 CPU 계산 시간을 나타내며, K_{max}가 커질수록 개체 수에 따라 지수적으로 증가함을 알 수 있다. <그림 10>을 통해 K_{max}가 1~3인 경우 전반적으로 빠른 계산 속도를 확인 할 수 있으나, 4인 경우에는 개체수가 60 이상일 때 상당한 과부하가 걸림을 알 수 있다. 특히 <표 5>에서 알 수 있듯이 "Class VI, K_{max} = 4, 개체수 = 100"인 조건에서의 평균 계산시간은 1시간 이상이 소요된다.



〈그림 10〉 개체 수와 K_{max}에 따른 계산 시간

〈표 5〉 Class VI 산출결과(평균값)

개체수	하한값	초기해	$K_{max}=1$		$K_{max}=2$		$K_{max}=3$		$K_{max}=4$	
			저장소개수	시간(초)	저장소개수	시간(초)	저장소개수	시간(초)	저장소개수	시간(초)
20	9.8	16	15	0.259	14.8	0.98	14.8	2.84	14.8	7.27
40	18.0	31	28.7	0.507	27.7	3.84	27.8	18.18	27.8	92.01
60	27.6	47.9	44.7	0.786	43.7	9.68	43.7	69.67	43.7	590.11
80	37.1	62.9	58.9	1.014	57.6	17.54	57.6	173.51	57.7	1718.51
100	45.0	77.2	70.6	1.101	69.6	29.38	69.6	293.47	69.8	3817.88

주) 각 수치는 전체 평균임.

5. 결 론

물류이동 및 적재의 필수 요소인 적재시스템은 모든 산업시설에서 고려되는 사안으로써 잘못된 적재 배치 및 적재차량 소요 판단 그리고 적정 용량을 초과한 저장시설 판단 등은 막대한 처리 비용과 재설치 비용이 소요되기 때문에 신중한 결정이 요구되는 분야이다.

본 연구에서는 2차원 직사각 적재문제로써 길로틴(guillotine) 형태를 보이는 레벨 적재(level packing) 알고리듬을 다루었으며 균등한 크기의 저장소에 무작위로 생성된 직사각 개체들을 적재하여 총 사용되는 저장소 개수를 최소화 하였다. 본 논문에서 제시한 알고리듬은 보다 빠르고 만족할 만한 수준의 해를 얻을 수 있도록 단순 발견 알고리듬인 혼합우측 적재 알고리듬과 상위 수준의 발견적 알고리듬인 타부서치를 이용하여 2차원 적재 문제를 적용하였다.

본 연구에서 제시한 발견적 알고리듬은 특정 공정에 쓰이는 단순 발견 알고리듬으로 초기해를 산출하고 국부탐색능력이 우수한 타부서치를 이용하여 최고해를 산출함으로써 비교적 빠른 수렴 속도와 만족할 만한 좋은 해를 동시에 얻을 수 있었다. 또한 취약한 저장소 선정을 위해 공간 활용도 함수를 적용함으로써 향상된 CPU 계산시간을 유도 할 수 있었다.

적재문제로의 활용을 위한 타부서치 기법의 적용은 현실적인 모형 구축을 위해 추가적인 제약 사항들을 고려하여 알고리듬을 개발함은 물론 타부서치 기법의 일반성을 입증하기 위해서 보다 많은 조합 최적화 문제에 적용해 보아야 할 것이며, 이때 각 파라메타(parameter)의 설정에 대한 세밀한 연구가 수행되어져야 할 것으로 판단된다.

참 고 문 헌

[1] 김여근, 윤복식, 이상복, 「메타휴리스틱」, 영지문화사, 2000.

[2] Andrea Lodi, Silvano Martello and Daniele Vigo, "Approximation Algorithms for the Oriented Two-Dimensional Bin Packing Problem," *European Journal of Operational Research*, Vol.112(1999), pp.158-166.

[3] Berkey, J.O. and P.Y. Wang, "Two Dimensional Finite Bin Packing Algorithms," *Journal of the Operational Research Society*, Vol.38(1987), pp.423-429.

[4] Blazewicz, J., P. Hawrylak and R. Walkowiak, "Using a Tabu Search Approach for Solving the Two-Dimensional Irregular Cutting Problem," *Annals of Operations Research*, Vol.41(1993), pp.313-327.

[5] Burke, E. and G. Kendall, "Applying Simulated Annealing and the No Fit Polygon to the Nesting Problem," *In Proceedings of the World Manufacturing Congress*, Durham, UK, (1999), pp.27-30.

[6] Dowsland, K., "Some Experiments with Simulated Annealing Techniques for Packing Problems," *Operational Research*, Vol. 68(1993), pp.389-399.

[7] Dowsland, K.A. and W.B. Dowsland, "Packing Problems," *European Journal of Operations Research*, Vol.56(1992), pp.2-14.

[8] Dowsland, K.A. and W.B. Dowsland, "Solution Approaches to Irregular Nesting Problems," *European Journal of Operations Research*, Vol.84(1995), pp.506-521.

[9] Dowsland, W.B., "Two and Three Dimensional Packing Problems and Solution Methods," *New Zealand Journal of Operational Research*, Vol.13(1985), pp.1-18.

[10] Dyckhoff, H., "A Typology of Cutting and Packing Problems," *European Journal of Operational Research*, Vol.44(1990), pp.145-159.

[11] Fowler, R.J., M.S. Paterson and S.L. Tatimoto, "Optimal Packing and Covering in the Plane are NP-Complete," *Information Processing Letters*, Vol.12(1981), pp. 133-137.

[12] Frenk, J.B. and G.G. Galambos, "Hybrid Next-fit Algorithm for the Two Dimensional Rectangle Bin-Packing Problem," *Computing*, Vol.39(1987), pp.201-217.

[13] Gilmore, P.C. and R.E. Gomory, "A Linear Programming Approach to the Cutting-Stock Problem Part I," *Operational Research*, Vol.9(1961), pp.724-746.

[14] Gilmore, P.C. and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem Part II," *Operational Research*, Vol.11(1963), pp.863-888.

[15] Gilmore, P.C. and R.E. Gomory, "Multi Stage Cutting Stock Problems of Two and More Dimensions," *Operational Research*, Vol.13(1965), pp.94-112.

[16] Glover, F., "Tabu Search," *ORSA Journal on Computing*, No.3(1989), pp.190-206.

[17] Glover, F. and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[18] Golden, B., "Approaches to the Cutting Stock Problem," *AIIE Transactions*, Vol.8 (1976), pp.265-274.

[19] Hinxman, A.I., "The Trim Loss and Assortment Problems," *European Journal of Operational Research*, Vol.5(1980), pp.8-18.

[20] Hopper, E. and B.C.H. Turton, "A Genetic Algorithm for a 2D Industrial Packing Problem," *Computers in Engineering*, Vol.37 (1999), pp.375-378.

[21] Hopper, E. and B.C.H. Turton, "An Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D Packing Problem," *European Journal of Operational Research*, Vol.128, No.1(2000), pp.34-57.

[22] Jacobs, S., "On Genetic Algorithms for the Packing of Polygons," *European Journal of Operational Research*, Vol.88(1996), pp. 165-181.

[23] Kampke, T., "Simulated Annealing : Use of a New Tool in Bin-Packing," *Annals of Operations Research*, Vol.16(1988), pp.327 -332.

[24] Liu, D. and H. Teng, "An Improved BL-Algorithm for Genetic Algorithm of the Orthogonal Packing of Rectangles," *European Journal of Operational Research*, Vol.112(1999), pp.413-419.

[25] Rayward-Smith, V.J. and M.T. Shing, "Bin Packing," *Bulletin of the Institute of Mathematics and Its Applications*, Vol.19(1983), pp.142-146.

[26] Rolland, E., H. Pirkul and F. Glover, "Tabu Search for Graph Partitioning," *Annals of Operations Research*, Vol.63(1996), pp.290 -232.

[27] Smith, D., "Bin-Packing with Adaptive Search," *In Grefenstette (ed), Proceedings of an International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, (1985), pp.202-206.

[28] Whelan, P.F. and B.G. Batchelor, "Automated Packing Systems : Review of Industrial Implementations," *SPIE, Machine Vision Architectures, Integration and Applications* 2064, (1993), pp.358-369.