Journal of the Korean Institute of Industrial Engineers
Vol. 34, No. 4, pp. 470-480, December 2008.

철강 코일제품 수송 팔레트의 설계 최적화

박종호 · 임경국† · 이정원

포항공과대학교 산업경영공학과

On Optimal Design Methods for Steel Product Pallets

Jongho Park · Kyungkuk Lim · Jeongwon Lee

Department of Industrial and Management Engineering, POSTECH

Loading steel coil products on a specialized packing case called pallet can be represented as a bin-packing problem with the special constraint where objects should be loaded on designated positions of bins. In this paper, under assuming that there exist only two types of objects, we focus on finding the optimum number of positions in a bin which minimizes the number of bins needed for packing a collection of objects. Firstly, we propose a method to decide the number of positions and prove that the method is optimum. Finally, for the packing problem using bins designed by the method, we show that the well-known algorithm, First-Fit Decreasing(FFD), is the optimum algorithm.

Keywords: Bin-Packing, Pallet Design, Steel Coil

1. 서 론

대부분의 제조 산업은 운반 도중에 발생하는 제품 손상을 줄이기 위하여 제품의 특성에 맞게 고정시킬 수 있는 운반도구를 개발하여 사용하고 있다. 이런 특별한 운반도구가 필요한 가장 대표적인 제조업은 철강 산업이다. 철강 산업에서는 코일 제품의 출하 공정 시에는 코일(Coil) 제품을 적재하기 위해 팔레트(Pallet)라는 도구를 사용한다. 현재 일반적으로 사용되고 있는 팔레트는 <Figure 1>에서 보듯이 둥근 코일이 움직이지 못 하도록 만든 골이라고 부르는 홈이 있으며 하나의 팔레트에는 9개의 골이 있도록 설계되어 있다.

이 팔레트는 다음과 같은 제약 조건을 가지고 있다.

- 적재 중량 100톤을 넘어서는 안 된다.

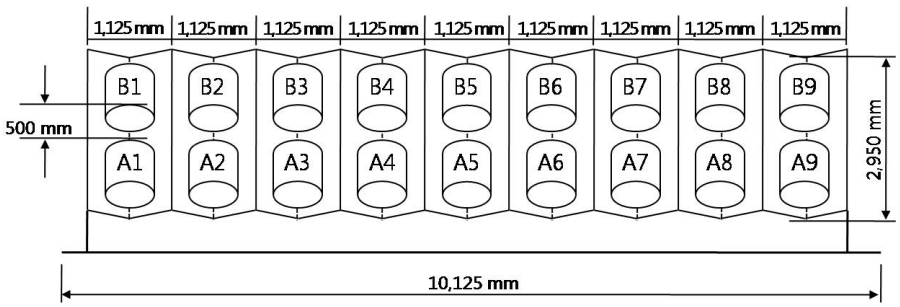

Figure 1. 팔레트의 사양

이 논문은 교육인적자원부의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(연구 과제명 : 3차년도 미래형 기계 국고지원금, 과제번호 : 4.0002619.02).

† 연락저자 : 임경국, 790-784 경북 포항시 남구 효자동 산 31번지 포항공과대학교 산업경영공학과, Fax : 054-279-2870,
E-mail : lkkuk@postech.ac.kr

2008년 8월 28일 접수; 2008년 10월 12일 게재 확정.

- 최대 적재 가능한 코일 수는 18개이다.
- 각 골에서의 코일 간 간격 500㎜ 이상을 유지해야 한다.
- 전후 중량 차 10톤 이내이고 좌우 중량 차 5톤 이내를 유지해야 한다.
- 골의 중심에 코일의 중심이 놓여야 하고 각 골에 놓인 코일들은 서로 겹쳐서는 안 된다.
- 골의 폭은 최대 코일외경의 1/2이상이어야 한다.

이처럼 팔레트는 여러 제약을 갖고 있으며 비용이 고가이기 때문에 현장에서는 최소의 팔레트로 모든 코일을 패킹(packing)하는 문제가 대두되고 있다. 그래서 이번 연구에서는 빈의 개수를 최소화하기 위하여 골을 어떻게 디자인해야 하는지 다루도록 한다.

코일을 아이템(item)으로 보고 팔레트를 빈(Bin)으로 생각하자. 특히 팔레트의 장 축 길이를 빈 크기(Bin size)로 보고 코일의 외경을 아이템 크기(Item size)로 본다면 우리 문제를 *Bin Packing Problem* 으로 생각할 수 있다. 일반적으로 Bin Packing Problem은 임의의 수용력(capacity)을 가진 빈(Bin)으로 모든 아이템을 패킹하면서 빈의 개수를 최소화하는 문제이다. 이 문제가 NP-Complete라는 것은 잘 알려져 있는 사실이며 많은 연구가 이루어져 왔다(Johnson D. S. *et al*. (1974)). 지금까지 제안된 대표적인 휴리스틱 알고리즘 중 하나는 First-Fit Decreasing (FFD)이다. FFD는 아이템 리스트(item list)를 비증가 순서 리스트(non-increasing order list)로 변형한 후, 리스트 순서의 아이템을 빈에 패킹하는 알고리즘이다. 이 때, 가장 낮은 인덱스(index)를 갖는 빈부터 아이템을 패킹하고 알맞은 빈이 없는 경우에는 새로운 빈에 아이템을 패킹시키는 방법이다(Baker, B. S.(1983)와 Johnson, D. S.(1973)).

일반적으로 *Multiprocessing Timing Anomalies* 에서는 작업들의 순서(order), 작업들의 리스트(list)와 작업 시간(processing time) 등이 동일하지만 작업자(processor)의 수가 늘어났음에도 불구하고 모든 작업들의 최소 완료 시간(least completion time)이 늘어나는 경우가 존재한다(Graham, R. L.(1969)). 이와 같이 일반적으로 기대하는 것과는 다른 결과를 초래하는 경우를 *Anomalies*라고 부른다. 우리 문제에서도 *Anomalies Problem*이 발생할 수 있는데 이는 각각의 아이템들이 빈 내의 정해진 위치에만 놓이기 때문이다.

우리 문제의 경우, 빈 크기가 고정되어 있고 빈 내의 골수가 늘어났다면 빈의 수용력이 증가되었다고 생각할 수 있다. 하지만 일반적으로 골의 개수가 증가했다고 해서 빈의 소요 개수가 감소하는 것이 아니다. <Figure 2>는 크기가 1인 아이템이 4개 있고, 크기가 4인 빈의 골이 4개인 경우와 5개인 경우에 대한 예제이다. <Figure 2>에서 보는 바와 같이 골이 4개일 경우 1개의 빈이 소요되지만 골이 5개인 경우 2개의 빈이 소요됨을 볼 수 있다.

본 논문의 구성은 다음과 같다. 제 2절에서는 철강 산업의 출하공정에서의 코일의 세 가지 특성을 기술하였고 제 3절에서는 문제를 정의하고 수학적 모델을 제시하였다. 제 4절에서는 제 3절에서 정의한 문제에 대한 최적해 기법을 보이고 이를 증명하였으며 제 5절에서는 제안된 최적해 기법으로 실험한 결과를 분석하고 토의하였다. 마지막으로 제 6절에서는 연구의 내용을 요약하고 앞으로의 과제에 대해서 기술하였다.

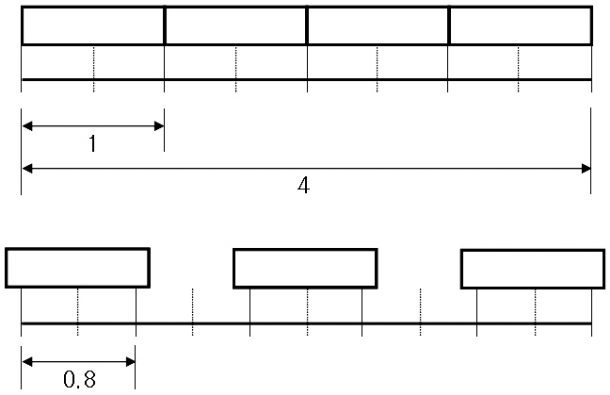

Figure 2. Anomalies을 나타내는 예제

2. 철강 산업의 출하 공정에서의 코일 분석

원래 팔레트의 최적 골수를 결정하기 위해서는 코일의 세 가지 주요 속성(폭, 외경, 무게) 모두를 고려해야하지만 이는 결코 쉬운 일이 아니다. 그래서 이 논문에서는 세 가지 속성들의 관계를 분석하고 이 중에서 가장 중요한 속성 하나만을 고려한다.

3개월 분량의 출하 공정 데이터들을 살펴보면 다음의 두 가지 사실을 알 수 있었다.

첫째, 팔레트의 한 골에는 대부분 두 개의 코일을 패킹할 수 있었다. 한 골에 두 개의 코일을 패킹하기 위해서는 두 코일 폭의 합에 500㎜를 더하여 2950㎜를 초과하면 안 된다. 실제 3개월 동안의 데이터 43952개를 살펴본 결과, 평균 폭은 1108.96㎜이었으며 이 중에서 71%가 1225㎜ 이하의 폭을 지니고 있었다. 이는 대부분 한 골에 두 개의 코일을 패킹할 수 있으며 폭이 1225㎜를 초과하는 코일조차도 폭이 좁은 코일과 함께 같은 골에 패킹 가능하다고 볼 수 있다. 이러한 이유로 이 논문에서는 코일의 속성 중에서 코일의 폭을 제외하였다.

둘째, 코일의 무게는 외경의 반지름을 제곱한 값과 폭의 곱에 비례한다. 다시 말하면, 코일의 외경이 커지면 코일의 무게는 제곱으로 증가한다. 이러한 이유로 코일의 외경이 큰 몇 개의 코일을 패킹하고 팔레트에 공간적 여유가 있어도 중량적인 제한으로 패킹이 불가능하다고 말할 수 있다. 그리고 외경이 작은 코일의 경우는 무게가 적기 때문에 팔레트에 최대 적재 코일 수인 18개 모두를 패킹하여도 중량 제한을 초과하지 않는다고 말할 수 있다. 따라서 이 논문에서는 외경에 의한 팔레트 내의 공간 활용이 더 중요하다고 판단되어 코일의 무게도 고려해야 할 속성에서 배제하였다.

3. 문제 정의

실제 코일의 외경은 600㎜~2057㎜의 다양한 값을 가지고 있다. 하지만 여기에서는 코일들의 외경 평균값보다 큰 외경을 가지는 것과 그렇지 않은 것의 두 가지로 분류하여 우리 문제를 단순화하였다.

<Figure 3>에서 큰 직사각형은 큰 아이템을 나타내고 작은 직사각형은 작은 아이템을 나타낸다. 가로 줄은 빈을 표현하고 세로 실선은 골의 경계를 나타내며 세로 점선은 골의 중심을 뜻한다. <Figure 3>에서는 하나의 빈에 4개의 골이 있고, 각 골의 중심에 아이템의 중심이 놓이도록 하여 3개의 큰 아이템과 1개의 작은 아이템이 패킹된 경우이다.

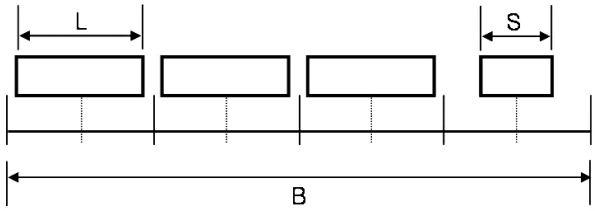

Figure 3. 문제 설명에 대한 시각 자료

이 논문에서는 큰 아이템과 작은 아이템이 각각 N_L과 N_S개 있고, 빈 크기가 B라고 가정한다. 이 경우 모든 아이템을 패킹하는데 필요한 빈의 수를 최소화하는 골의 수(G)를 찾고자 한다. 여기에서 주의할 사항은 골의 폭이 큰 아이템 크기의 반 이상이어야 한다는 제약이 있기 때문에 빈 내의 골의 개수는 계속 증가시킬 수 없다.

빈의 골에 아이템이 놓일 수 있는 가능한 모든 경우는 다음과 같이 4가지 경우로 분류 가능하다(<Figure 4> 참조).

Case[1] : 큰 아이템과 작은 아이템이 모두 한 골에 들어가는 경우

Case[2] : 큰 아이템은 한 골에 들어가지 못 하지만 큰 아이템이 놓이는 골의 옆에 작은 아이템이 들어가는 경우

Case[3] : 큰 아이템이 한 골에 들어가지 못하며 큰 아이템이 놓이는 골의 옆에도 작은 아이템이 들어가지 못 하고, 작은 아이템은 한 골에 놓일 수 있는 경우

Case[4] : 골의 폭 제약을 만족하면서 큰 아이템과 작은 아이템 모두가 한 골에 놓이지 않는 경우

지금까지 제시한 문제를 명확히 정의하기 위해 수학적 모델을 수립하면 다음과 같다.

Index

큰 아이템 인덱스(Large item index) : i, $(i=1, \cdots, m)$

작은 아이템 인덱스(Small item index) : j, $(j=1, \cdots, n)$

빈 인덱스(Bin index) : k, $(k=1, \cdots, l)$

Parameter

큰 아이템 크기(Large item size) : L

작은 아이템 크기(Small item size) : S

빈 크기(Bin size) : B

큰 아이템 개수(Large item 개수) : N_L

작은 아이템 개수(Small item 개수) : N_S

Variables

G : 골수($G \geq 0$인 정수)

$$x_{ik} = \begin{cases} 1 : \text{큰 아이템 } i\text{가 빈 } k\text{에 편성된 경우} \\ 0 : \text{그 외} \end{cases}$$

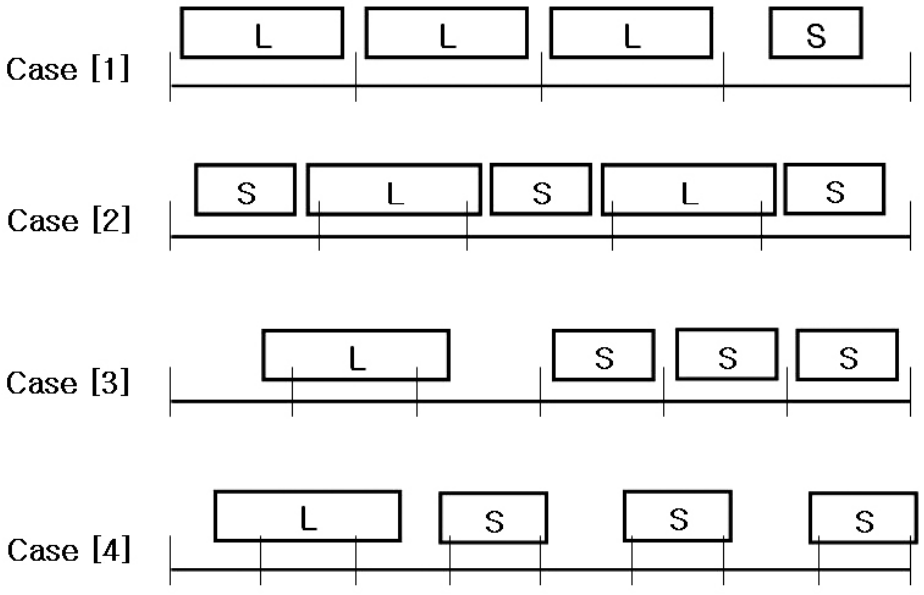

Figure 4. 빈의 이웃한 골간에 아이템들이 놓을 수 있는 경우

y_{jk} = $\begin{cases}1 : \text{작은 아이템 } j\text{가 빈 } k\text{에 편성된 경우}\\0 : \text{그 외}\end{cases}$

z_k = $\begin{cases}1 : \text{빈 } k\text{가 편성된 경우}\\0 : \text{그 외}\end{cases}$

t = $\begin{cases}1 : S > L/2\text{인 경우}\\0 : S \leq L/2\text{인 경우}\end{cases}$

a_k = $\begin{cases}1 : \text{빈에 큰 아이템으로만 채워진 경우}\\0 : \text{그 외}\end{cases}$

Objective

$$Min \sum_k z_k \tag{1}$$

Constraints

$$G \leqq \frac{2B}{L} \tag{2}$$

$$x_{ik} \leqq z_k \quad \forall i, k \tag{3}$$

$$y_{jk} \leqq z_k \quad \forall j, k \tag{4}$$

$$\sum_{k=1}^{l} x_{ik} = 1 \quad \forall i \tag{5}$$

$$\sum_{k=1}^{l} y_{jk} = 1 \quad \forall \text{ j} \tag{6}$$

$$\sum_{k=1}^{l}\sum_{i=1}^{m} x_{ik} = N_L \tag{7}$$

$$\sum_{k=1}^{l}\sum_{j=1}^{n} y_{jk} = N_S \tag{8}$$

$$2S - L \leqq Mt \tag{9}$$

$$L - 2S \leqq M(1-t) \tag{10}$$

$$G \leq \left\lfloor \frac{B}{L} \right\rfloor + M(1-w_1) \tag{11}$$

$$\sum_i x_{ik} + \sum_j y_{jk} \leq G + M(1-w_1) \quad \forall k \tag{11-1}$$

$$G \leq \left\lfloor \frac{2B}{L+S} \right\rfloor + M(1-w_2) \tag{12}$$

$$\sum_i x_{ik} + \sum_j y_{jk} \leq G + M(1-w_2) \quad \forall k \tag{12-1}$$

$$2\sum_i x_{ik} \leq G + 1 + M(1-w_2) \quad \forall k \tag{12-2}$$

$$G \leq \left\lfloor \frac{2B}{L} \right\rfloor + M(1-w_3) + Mt \tag{13}$$

$$G \leq \left\lfloor \frac{B}{S} \right\rfloor + M(1-w_3) + M(1-t) \tag{13-1}$$

$$\sum_j y_{jk} \geq 1 - Ma_k \quad \forall k \tag{13-2}$$

$$2\sum_i x_{ik} + \sum_j y_{jk} \leq G + M(1-w_3) + Ma_k \quad \forall k \tag{13-3}$$

$$\sum_j y_{jk} \leq M(1-a_k) \quad \forall k \tag{13-4}$$

$$2\sum_i x_{ik} \leq G + 1 + M(1-w_3) + M(1-a_k) \quad \forall k \tag{13-5}$$

$$G \leq \left\lfloor \frac{2B}{L} \right\rfloor + M(1-w_4) + M(1-t) \tag{14}$$

$$2\sum_i x_{ik} + 2\sum_j y_{jk} \leq G + 1 + M(1-w_4) + M(1-t) \quad \forall k \tag{14-1}$$

$$w_4 \leqq t \tag{15}$$

$$w_1 + w_2 + w_3 + w_4 = 1 \tag{16}$$

G is non-negative integer variable.

x_{ik}, y_{jk}, z_k, a_k, t, w_1, w_2, w_3, w_4 are binary integer variables. $\forall i, j, k$

목적 식 (1)은 아이템이 들어있는 빈 개수의 합으로 이것을 최소화하는 것이 이 모델의 목적이다. 조건 식 (2)는 골의 폭이 큰 아이템 크기의 반 이상이 되어야 함을 나타낸다. 조건 식 (3)과 식 (4)에서 x_{ik}와 y_{jk}가 1의 값을 가지면 이진 변수 z_k는 반드시 1이 된다. 즉, 큰 아이템 i 또는 작은 아이템 j 중 일부가 빈 k 로 편성되면 z_k는 1이라는 것을 알 수 있다. 조건 식 (5)과 식 (6)은 특정 큰 아이템 i와 작은 아이템 j는 오직 하나의 빈 k에만 놓일 수 있다는 것을 의미한다. 조건 식 (7)과 식 (8)은 빈에 패킹된 큰 아이템과 작은 아이템의 개수의 총합이 각각 N_L와 N_S개 임을 의미한다. 조건 식 (9)과 식 (10)은 파라미터 값인 L과 S의 관계에 대한 표현으로써 $S \leq L/2$인 경우에는 $t = 0$의 값을 갖고 $S > L/2$인 경우에는 $t = 1$이 되도록 하여 조건 식 (13) ~ 식 (15) 을 선택적으로 취할 수 있도록 하였다.

조건 식 (11)은 Case[1]에서 가능한 골수를 나타내고 조건 식 (11-1)은 아이템들이 빈에 패킹될 수 있는 개수를 표현한 것이다. 조건 식 (11)과 식 (11-1)은 Case[1]의 최소 빈 수를 갖는 골수(G)를 찾는 조건이다. 조건 식 (12)는 Case[2]에서 가능한 골수를 나타내고 조건 식 (12-1)과 식 (12-2)는 아이템들이 빈에 패킹될 수 있는 개수를 나타낸 것이다. 조건 식 (12), 식 (12-1)과 (12-2)는 Case[2]에서의 최소 빈 수를 갖는 골수(G)를 찾는 조건이다. 조건 식 (13)과 식 (13-1)은 Case[3]이 L과 S의 관계에 따라 골수(G)의 상한 값이 달라짐을 뜻한다. 조건 식 (13-2)과 식 (13-4)는 빈 k에 작은 아이템이 있는지 없는지를 표현한 것이고 조건 식 (13-3)과 식 (13-5)는 아이템들이 빈에 패킹될 수 있는 개수를 나타낸 것이다.

조건 식 (14)는 Case[4]에서 가능한 골수를 나타내고 조건 식 (14-1)은 아이템들이 빈에 패킹될 수 있는 개수를 표현한 것이다. 조건 식 (14)와 식 (14-1)은 Case[4]에서의 최소 빈 수를 갖는 골수(G)를 찾는 조건이고 조건 식 (14)와 식 (14-1)은 $S > L/2$인 경우에만 성립한다. 만약 $S \leq L/2$이라면, 즉 $t = 0$이라면, Case[4]가 고려되지 않음을 표현한 식이 조건 식 (15)이다. 조건 식 (16)은 Case[1], Case[2], Case[3]과 Case[4] 중에서 하나의 Case에 대한 제약 식만이 유효한 제약식이라는 것을 나타낸 조건이다. 이상의 모델은 정수계획법(Integer Programming)의 형태이며 이 모델로부터 해를 도출하는 것은 현실적으로 불가능하다. 따라서 해를 도출하기 위하여 다른 방법을 모색할 필요가 있다.

4. 문제의 특성 및 최적해 기법 제안

앞에서 정의한 문제가 기존의 Bin Packing Problem과 구별되는 특징은 다음과 같다.

첫째, 아이템들이 빈 내의 정해진 위치(골)에 놓인다. 이 때 골의 중심과 아이템의 중심이 일치해야 한다.

둘째, 아이템들이 빈의 수용력을 초과하더라도 빈 내의 골의 폭이 큰 아이템 크기의 절반 이상이라면 아이템들은 빈 내의 어느 골에도 놓일 수 있다.

먼저 골의 폭 범위에 따라 가장 좋은 골수를 찾고 이러한 골수들 중에서 빈 수를 최소화하는 최적의 골수를 찾겠다. 이렇게 함으로써 우리 문제가 최적해(최적 골 수)를 갖는다는 것을 보이도록 하겠다.

4.1 4가지 Case 각각에서 Dominant 골수 찾기

골에 대한 폭의 제약 ($B/G \geq L/2$)을 만족하면서 골의 폭 범위에 따라 빈의 골에 아이템들이 놓일 수 있는 경우와 각 경우의 dominant 골수를 찾아보자.

Case[1] : 큰 아이템과 작은 아이템이 모두 한 골에 들어가는 경우

골의 폭(B/G)이 $L \leq B/G \leq B$을 만족하면 모든 아이템들이 한 골에 놓인다. 이 식을 골 수(G)에 관한 식으로 정리해 보면 다음과 같다.

$$1 \leq G \leq \left\lfloor \frac{B}{L} \right\rfloor \tag{17}$$

식 (17)에서 빈의 골수가 $\left\lfloor \frac{B}{L} \right\rfloor$ 인 경우에 가장 많은 아이템들을 패킹할 수 있기 때문에 최소 빈이 사용되는 $G_1 = \left\lfloor \frac{B}{L} \right\rfloor$ 이 dominant 골수가 된다.

Case[2] : 큰 아이템은 한 골에 들어가지 못 하지만 큰 아이템이 놓이는 골의 옆에 작은 아이템이 들어가는 경우

골의 폭(B/G)이 $(L+S)/2 \leq B/G < L$을 만족하는 경우이고 이 식을 골수에 관해 표현하면 다음과 같다.

$$\left\lfloor \frac{B}{L} \right\rfloor < G \leq \left\lfloor \frac{2B}{L+S} \right\rfloor \tag{18}$$

빈의 골수가 $\left\lfloor \frac{2B}{L+S} \right\rfloor$ 인 경우는 가장 많은 아이템들을 패킹할 수 있기 때문에 최소 빈이 사용되는 $G_2 = \left\lfloor \frac{2B}{L+S} \right\rfloor$ 이 dominant 골수가 된다.

Case[3] : 큰 아이템이 한 골에 들어가지 못하며 큰 아이템이 놓이는 골의 옆에도 작은 아이템이 들어가지 못 하고, 작은 아이템은 한 골에 놓일 수 있는 경우

골의 폭(B/G)이 $S \leq B/G < (L+S)/2$을 만족하는 경우이다. 그런데 골의 제약 폭 ($B/G \geq L/2$)을 만족해야 하기 때문에 작은 아이템 크기의 범위를 $S > L/2$인 경우와 $S \leq L/2$인 경우로 구분하여 살펴보자.

1) $S > L/2$인 경우

이 경우에는 골수에 관한 식을 다음과 같이 나타낼 수 있다.

$$\left\lfloor \frac{2B}{L+S} \right\rfloor < G \leq \left\lfloor \frac{B}{S} \right\rfloor \tag{19}$$

빈의 골수가 $\left\lfloor \frac{B}{S} \right\rfloor$ 인 경우는 가장 많은 아이템들을 패킹할 수 있기 때문에 최소 빈이 사용되는 $G_3 = \left\lfloor \frac{B}{S} \right\rfloor$ 이 dominant 골수가 된다.

2) $S \leq L/2$인 경우

골의 폭 제약 ($B/G \geq L/2$)을 만족하기 위해서는 $L/2 \leq B/G < (L+S)/2$을 만족해야 하고 이 식을 골수에 관한 식은 다음과 같이 나타낼 수 있다.

$$\left\lfloor \frac{2B}{L+S} \right\rfloor < G \leq \left\lfloor \frac{2B}{L} \right\rfloor \tag{20}$$

빈의 골수가 $\left\lfloor \frac{2B}{L} \right\rfloor$ 인 경우가 가장 많은 아이템들을 패킹할 수 있기 때문에 최소 빈이 사용되는 $G_3 = \left\lfloor \frac{2B}{L} \right\rfloor$ 이 dominant 골수가 된다.

Case[4] : 골의 폭 제약을 만족하면서 큰 아이템과 작은 아이템 모두가 한 골에 놓이지 않는 경우

$S > L/2$인 경우만 해당되고 골의 폭(B/G)이 $L/2 \leq B/G < S$을 만족하는 경우이고 이 식을 골수에 관한 식은 다음과 같이 나타낼 수 있다.

$$\left\lfloor \frac{B}{S} \right\rfloor < G \leq \left\lfloor \frac{2B}{L} \right\rfloor \tag{21}$$

빈의 골수가 $\left\lfloor \frac{2B}{L} \right\rfloor$ 인 경우는 가장 많은 아이템들을 패킹

할 수 있기 때문에 최소 빈이 사용되는 $G_4 = \left\lfloor \frac{2B}{L} \right\rfloor$ 이 dominant 골수가 된다.

위의 case[1]~case[4]의 dominant 골수를 정리한 결과는 <Table 1>과 같다.

4.2 골의 폭 범위를 고려하여 찾아낸 dominant 골수들 중에서 최적 골수 찾기

앞에서는 4가지 Case로 나누고 각 경우에서 dominant 골수를 찾았다. 여기에서는 각각의 Case가 아닌 모든 Case에서 빈의 개수를 최소화하는 골 수(G^*)를 찾도록 한다. 하지만 각 Case의 dominant 골수 중에서 어느 것이 최적 골수인지는 큰 아이템의 개수와 작은 아이템의 개수에 따라 다르다. 이러한 이유로 모든 아이템들을 빈에 패킹하기 위해서는 각 Case의 dominant 골수를 가진 빈이 몇 개가 필요한지 비교해야 한다.

이 논문에서는 아이템들을 빈의 골에 패킹하기 위해서 FFD Algorithm을 사용한다. 본 논문에서는 우리 문제에 대해 이 알고리즘이 최적임을 보이고 일반적으로 큰 아이템과 작은 아이템이 각각 N_L개와 N_S개가 있을 경우의 소요되는 빈의 개수를 유도하였다.

Lemma 1

큰 아이템과 작은 아이템의 수가 각각 N_L개와 N_S개가 있으며 빈의 골수가 $G_1 = \left\lfloor \frac{B}{L} \right\rfloor$ 이라고 가정하자. 이 때, 모든 아이템을 빈에 패킹하기 위해 필요한 최소 빈의 개수(B_1)는 $B_1 = \left\lceil \frac{N_L + N_S}{G_1} \right\rceil$ 이다.

Proof. FFD Algorithm에 의해서 큰 아이템을 빈에 패킹시킨 후 작은 아이템을 빈에 패킹시키면, 큰 아이템과 작은 아이템이 모두 한 골에 놓이게 되므로 빈 내의 골의 손실이 전혀 없이 최적해와 동일한 수의 빈을 사용하게 된다. 이웃한 골에 놓인 아이템들 사이의 공간적인 손실은 있을 수 있겠지만 문제의 특성상 이러한 공간에는 아이템들이 놓일 수 없다. 그러므로 Bin Packing Problem과 달리 공간적인 손실이 아닌 골의 손실이 있는지와 없는지에 따라 최적해와 비교 가능할 것이다. 최소 빈의 개수에 관한 일반식을 구해 보면 하나의 빈에는 개의 아이템들이 놓일 수 있으므로 전체 아이템의 개수를 골 수(G_1)로 나누면 된다. □

Lemma 2

큰 아이템과 작은 아이템의 수가 각각 N_L개와 N_S개가 있으며 빈의 골수가 $G_2 = \left\lfloor \frac{2B}{L+S} \right\rfloor$ 라고 가정하자. 이 때, 모든 아이템을 빈에 패킹하기 위해 필요한 최소 빈의 개수(B_2)는 다음과 같다.

$$B_2 = \max\left\{ \left\lceil \frac{N_L}{\left\lceil \frac{G_2}{2} \right\rceil} \right\rceil, \left\lceil \frac{N_L + N_S}{G_2} \right\rceil \right\}$$

Proof. G_2을 홀수인 경우와 짝수인 경우로 나누어 생각해 보자.

Case 1. G_2가 짝수인 경우

큰 아이템과 작은 아이템은 각각 최대 $G_2/2$개와 G_2개를 하나의 빈에 패킹할 수 있다. 그리고 큰 아이템이 놓인 골의 옆 골에 작은 아이템이 놓일 수 있으므로 $N_L \geq N_S$인 경우와 $N_L < N_S$인 경우로 구분하여 B_2을 생각해 보자.

Table 1. 골의 폭 범위에 따라 아이템들이 놓일 수 있는 경우와 dominant 골수

Position Width	Number of Positions	Dominant No. of Positions	Case		
			SS	SL	LL
$L \leq \frac{B}{G} \leq B$	$1 \leq G \leq \left\lfloor \frac{B}{L} \right\rfloor$	$G_1 = \left\lfloor \frac{B}{L} \right\rfloor$	○	○	○
$\frac{L+S}{2} \leq \frac{B}{G} < L$	$\left\lfloor \frac{B}{L} \right\rfloor < G \leq \left\lfloor \frac{2B}{L+S} \right\rfloor$	$G_2 = \left\lfloor \frac{2B}{L+S} \right\rfloor$	○	○	×
$S \leq \frac{B}{G} < \frac{L+S}{2}$	$S > \frac{L}{2}$: $\left\lfloor \frac{2B}{L+S} \right\rfloor < G \leq \left\lfloor \frac{B}{S} \right\rfloor$	$G_3 = \left\lfloor \frac{B}{S} \right\rfloor$	○	×	×
	$S \leq \frac{L}{2}$: $\left\lfloor \frac{2B}{L+S} \right\rfloor < G \leq \left\lfloor \frac{2B}{L} \right\rfloor$	$G_3 = \left\lfloor \frac{2B}{L} \right\rfloor$			
$\frac{L}{2} \leq \frac{B}{G} < S$	$\left\lfloor \frac{B}{S} \right\rfloor < G \leq \left\lfloor \frac{2B}{L} \right\rfloor$	$G_4 = \left\lfloor \frac{2B}{L} \right\rfloor$	×	×	×

SS : Whether a small item can be loaded adjacent to a small item.
SL : Whether a small item can be loaded adjacent to a large item.
LL : Whether a large item can be loaded adjacent to a large item.

Sub-Case 1. $N_L \geq N_S$

FFD Algorithm에 의하여 큰 아이템을 패킹하고 나면 작은 아이템은 큰 아이템들이 놓인 빈에 모두 놓을 수 있다. 그리고 큰 아이템이 작은 아이템보다 같거나 많기 때문에 작은 아이템이 하나도 놓이지 않는 빈에는 큰 아이템이 최대 $G_2/2$개가 놓일 수 있고 큰 아이템들만이 놓인 빈이 있다면 아무런 아이템도 놓이지 않는 골이 있을 것이다. 하지만 이 경우는 작은 아이템들도 없으며 큰 아이템이 놓인 이웃한 골에 다른 큰 아이템이 놓이지 못 하기 때문에 골의 손실이 없이 최소 빈의 개수 $B_2 = \left\lceil \frac{N_L}{\frac{G_2}{2}} \right\rceil$ 가 사용되었다고 볼 수 있다.

Sub-Case 2. $N_L < N_S$

FFD Algorithm에 의하며 큰 아이템을 패킹하고 나면 작은 아이템은 큰 아이템이 놓인 빈에 $N_S - N_L$개를 놓을 수 있다. 그리고 작은 아이템들은 최대 G_2개를 빈에 패킹할 수 있다. 그러므로 빈의 모든 골에는 하나의 아이템이 놓여있는 경우이기 때문에 골의 손실이 없이 최소 빈의 개수 $B_2 = \left\lceil \frac{N_L + N_S}{G_2} \right\rceil$ 가 사용되었다고 볼 수 있다.

Sub-Case 1과 Sub-Case 2의 B_2을 하나로 묶어서 정리하면 다음과 같다.

$$B_2 = \left\lceil \frac{\max\left\{N_L, \frac{N_L+N_S}{2}\right\}}{\frac{G_2}{2}} \right\rceil = \left\lceil \frac{\max\{2N_L, N_L+N_S\}}{G_2} \right\rceil$$

Case 2. G_2가 홀수인 경우

큰 아이템은 최대 $\frac{G_2+1}{2}$개를 빈에 패킹할 수 있고 작은 아이템은 최대 G_2개를 빈에 패킹할 수 있다. 그리고 큰 아이템이 놓인 골의 옆 골에는 작은 아이템을 놓을 수 있기 때문에 패킹할 때, FFD Algorithm을 적용하면 큰 아이템과 작은 아이템의 개수의 비율이 $\frac{G_2+1}{2} : G_2 - \frac{G_2+1}{2} (= \frac{G_2-1}{2})$인 빈이 존재하게 된다. 큰 아이템 $\frac{G_2+1}{2}$개를 빈에 패킹하고 나머지 골에 작은 아이템을 패킹하게 되므로 $\frac{G_2-1}{2}N_L \geq \frac{G_2+1}{2}N_S$ 인 경우와 $\frac{G_2-1}{2}N_L < \frac{G_2+1}{2}N_S$인 경우로 구분하여 B_2을 생각해 보자.

Sub-Case 1. $\frac{G_2-1}{2}N_L \geq \frac{G_2+1}{2}N_S$

FFD Algorithm에 의하여 큰 아이템을 패킹한 후, 작은 아이템은 큰 아이템들이 놓인 빈에 모두 놓을 수 있다. 그리고 큰 아이템이 많이 있기 때문에 작은 아이템이 하나도 놓이지 않은 빈에는 큰 아이템이 최대 $\frac{G_2+1}{2}$개 놓일 수 있고 큰 아이템들만이 놓인 빈이 있다면 아무 아이템도 놓이지 않는 골이 존재할 것이다. 하지만 작은 아이템도 없고 이웃한 골의 큰 아이템이 있어서 큰 아이템을 놓을 수 없다. 그래서 골의 손실이 없이 최소 빈의 개수 $B_2 = \left\lceil \frac{N_L}{\frac{G_2+1}{2}} \right\rceil$ 가 사용되었다고 볼 수 있다.

Sub-Case 2. $\frac{G_2-1}{2}N_L < \frac{G_2+1}{2}N_S$

FFD Algorithm에 의해 큰 아이템을 패킹한 후 작은 아이템은 큰 아이템들이 놓은 빈에 $\frac{G_2+1}{2}N_S - \frac{G_2-1}{2}N_L$개가 놓일 수 있다. 그리고 남아있는 작은 아이템들은 최대 G_2개를 빈에 패킹할 수 있기 때문에 골의 손실이 없이 최소 빈의 개수 $B_2 = \left\lceil \frac{N_L+N_S}{G_2} \right\rceil$ 가 사용되었다.

Sub-Case 1과 Sub-Case 2의 B_2을 하나로 묶어서 정리하면 Sub-Case 1의 $B_2 = \left\lceil \frac{2N_L}{G_2+1} \right\rceil$ 는 $\left\lceil \frac{\frac{2G_2}{G_2+1}N_L}{G_2} \right\rceil$ 로 바꾸어 쓸 수 있고 Sub-Case 2의 B_2가 성립하는 조건 $\frac{G_2-1}{2}N_L < \frac{G_2+1}{2}N_S$은 $N_S > \frac{G_2-1}{G_2+1}N_L$으로 바꿀 수 있다. 특히 Sub-Case 2의 B_2가 성립하는 조건으로부터 $N_L + N_S > \frac{2G_2}{G_2+1}N_L$ 을 이끌 수 있으므로 B_2는 다음과 같다.

$$B_2 = \left\lceil \frac{\max\left\{\frac{2G_2}{G_2+1}N_L,\ N_L+N_S\right\}}{G_2} \right\rceil$$

이제 G_2가 홀수인 경우와 짝수인 경우를 통합하면 B_2는 다음과 같다.

$$\left\lceil \frac{\max\left\{\frac{G_2}{\left\lceil \frac{G_2}{2} \right\rceil}N_L,\ N_L+N_S\right\}}{G_2} \right\rceil$$
$$= \max\left\{ \left\lceil \frac{N_L}{\left\lceil \frac{G_2}{2} \right\rceil} \right\rceil, \left\lceil \frac{N_L+N_S}{G_2} \right\rceil \right\} \qquad \square$$

Lemma 3

큰 아이템과 작은 아이템의 수가 각각 N_L개와 N_S개가 있으며 빈의 골수가 $S > \frac{L}{2}$이면 $G_3 = \left\lfloor \frac{B}{S} \right\rfloor$ 이고 $S \le \frac{L}{2}$이면 $G_3 = \left\lfloor \frac{2B}{L} \right\rfloor$ 이라고 가정하자. 그리고 빈의 골수가 $S > \frac{L}{2}$ 일 때, $\left\lfloor \frac{2B}{L+S} \right\rfloor < G_3 \le \left\lfloor \frac{B}{S} \right\rfloor$ 을 만족하고 $S \le \frac{L}{2}$일 때, $\left\lfloor \frac{2B}{L+S} \right\rfloor < G_3 \le \left\lfloor \frac{2B}{L} \right\rfloor$ 을 만족한다고 가정하자. 이 때, 모든 아이템들을 빈에 패킹하기 위하여 필요한 최소 빈의 개수는 $B_3 = \left\lceil \frac{N_L}{\left\lceil \frac{G_3}{2} \right\rceil} + \frac{N_S}{G_3} \right\rceil$ 이다.

Proof : 의 범위에 따라 골수만 달라질 뿐 아이템들이 빈의 골에 놓이는 Case는 동일하다. 이러기 때문에 모든 아이템들을 빈에 패킹하기 위하여 필요한 최소 빈의 개수는 S의 범위와 관계없이 동일한 식으로 표현 가능하다. 이제 G_3을 짝수인 경우와 홀수인 경우로 나누어 생각해 보자.

Case 1 : G_3가 짝수인 경우

큰 아이템은 최대 $G_3/2$개를 빈에 패킹할 수 있고 작은 아이템은 최대 G_3개를 패킹 할 수 있으며 큰 아이템이 놓인 옆 골에 작은 아이템을 놓을 수 없다. FFD Algorithm에 의하여 큰 아이템을 빈에 모두 패킹시킨 후 작은 아이템을 빈에 패킹시키는 것이 빈의 개수를 최소화한다. 이는 큰 아이템이 놓인 골과 이웃한 골만 빈 골로 남아 있는데 여기에는 어떤 아이템도 놓일 수 없기 때문에 골의 손실이 없다고 볼 수 있으며 이때 B_3는 $\left\lceil \frac{2N_L + N_S}{G_3} \right\rceil$ 이 된다.

Case 2 : G_3가 홀수인 경우

큰 아이템은 최대 $(G_3+1)/2$개, 작은 아이템은 최대 G_3개를 하나의 빈에 패킹할 수 있으며 큰 아이템이 놓인 옆 골에 작은 아이템을 놓을 수 없다. FFD Algorithm에 의하여 먼저 큰 아이템을 빈에 모두 패킹한 후에 작은 아이템을 패킹하는 것은 소요되는 빈의 개수를 최소화한다. 큰 아이템이 놓인 골과 이웃한 골만 비어있게 되는데 이곳에는 어떤 아이템도 놓일 수 없기 때문이다. 이 때, B_3는 다음과 같다.

$$B_3 = \left\lceil \frac{N_L}{\frac{G_3+1}{2}} + \frac{N_S}{G_3} \right\rceil = \left\lceil \frac{\frac{2G_3}{G_3+1} N_L}{G_3} + \frac{N_S}{G_3} \right\rceil = \left\lceil \frac{\frac{2G_3}{G_3+1} N_L + N_S}{G_3} \right\rceil$$

G_3이 홀수인 경우와 짝수인 경우의 B_3을 통합하여 정리하면 다음과 같다.

$$B_3 = \left\lceil \frac{\frac{G_3}{\left\lceil \frac{G_3}{2} \right\rceil} N_L + N_S}{G_3} \right\rceil = \left\lceil \frac{N_L}{\left\lceil \frac{G_3}{2} \right\rceil} + \frac{N_s}{G_3} \right\rceil \quad \square$$

Lemma 4.

큰 아이템과 작은 아이템의 수가 각각 N_L개와 N_S개가 있으며 $S > \frac{L}{2}$인 경우에 빈의 골수가 $G_4 = \left\lfloor \frac{2B}{L} \right\rfloor$ 라고 가정하자. 만약, $\left\lfloor \frac{B}{S} \right\rfloor < G_4 \le \left\lfloor \frac{2B}{L} \right\rfloor$ 을 만족한다면, 모든 아이템들을 빈에 패킹하기 위하여 필요한 최소 빈의 개수는 $B_4 = \left\lceil \frac{N_L + N_S}{\left\lceil \frac{G_4}{2} \right\rceil} \right\rceil$ 이다.

Proof : 이 경우는 큰 아이템과 작은 아이템 각각이 한 골에는 놓일 수 없으나 아이템들의 크기가 골 폭의 2배를 넘지 않는다. 이제 G_4을 짝수인 경우와 홀수인 경우로 구분하여 생각한다.

Case 1 : G_4가 짝수인 경우

큰 아이템과 작은 아이템 모두 최대 $G_4/2$개를 하나의 빈에 패킹할 수 있다. 한 아이템이 놓은 골과 이웃한 골은 비어있지만 여기에는 어떤 아이템도 놓일 수 없기 때문에 골의 손실이 없이 최소 개수의 빈의 개수 $B_4 = \left\lceil \frac{N_L + N_S}{\frac{G_4}{2}} \right\rceil$ 이 사용되었다고 볼 수 있다.

Case 2 : G_4가 홀수인 경우

큰 아이템과 작은 아이템 모두 최대 $(G_4+1)/2$개를 하나의 빈에 패킹할 수 있다. 한 아이템이 놓은 골과 이웃한 골은 비었지만 여기에는 어떤 아이템도 놓일 수 없기 때문에 골의 손실이 없이 최소 개수의 빈의 개수 $B_4 = \left\lceil \frac{N_L + N_S}{\frac{G_4+1}{2}} \right\rceil$ 이 사용되었다고 볼 수 있다.

G_4가 짝수인 경우와 홀수인 경우를 통합하면 $B_4 = B_4 = \left\lceil \frac{N_L + N_S}{\left\lceil \frac{G_4}{2} \right\rceil} \right\rceil$ 이 된다. $\square$

Lemma 1～Lemma 4에 의하면 아이템들을 FFD Algorithm으로 패킹한다면 골의 손실이 없이 빈의 골에 놓을 수 있다. 그래서 이 문제에서는 FFD Algorithm이 최적의 알고리즘이라 할 수 있다.

Table 2. 실제 출하분에 대한 데이터 통계 값과 최적해 기법에 의해 구한 최적 골수

코일 Set	외경평균	L	S	총 코일수	N_L	N_S	G_1	G_2	G_3	G_4	B_1	B_2	B_3	B_4	G^*
980	1223.29	1624.88	1040.97	980	306	674	6	7	9	12	164	140	137	164	9
402	1796.11	1875.08	1472.71	107	86	21	5	6		10	22	29		22	5, 10
403	1647.63	1876.96	1200.58	233	154	79	5	6	8	10	47	52	49	47	5, 10
104	1430.59	1811.23	1039.15	572	290	282	5	7	9	11	115	82	90	96	7
406	1231.22	1663.54	937.196	625	253	372	6	7	10	12	105	90	88	105	10
410	1421.31	1668.18	1162.68	215	110	105	6	7	8	12	36	31	41	36	7
414	978.182	1022.47	962.882	148	38	110	9	10		19	0	15		0	10, 19
415	1416.27	1696.27	1136.27	228	114	114	5	7	8	11	5	33	5	4	7
421	1231.23	1803.42	1027.64	583	153	430	5	7	9	11	0	84	0	0	9
425	1703.12	1721.74	1607.38	258	216	42	5	6		11	0	0		0	11
427	1280.31	1666.96	1030.65	785	308	477	6	7	9	12	0	113	0	0	7
502	1518.24	1789.69	1236.35	371	189	182	5	6	8	11	0	0	0	0	11
504	1382.34	1837.06	1076.83	535	215	320	5	6	9	11	0	90	0	0	9
505	1180.06	1510.79	948.043	439	181	258	6	8	10	13	0	55	0	0	8
506	1044.6	1335.58	915.958	274	84	190	7	8	11	15	6	35	7	6	11
513	1757.61	1896.01	1643.37	230	104	126	5		6	10	10		12	10	5, 10
518	1720.26	1877.23	1061.7	239	193	46	5	6	9	10	0	0	0	0	9
519	1367.06	1863.12	1144.23	271	84	187	5	6	8	10	0	46	0	0	8
520	996.01	1885.89	886	509	56	453	5	7	10		0	73	0		10
524	1413.77	1793.7	1139.37	465	195	270	5	6	8	11	0	78	0	0	6, 11
525	1859.32	1904.18	1814.47	34	17	17	5			10	0			0	5, 10
526	1909.18	1922.21	1876.24	60	43	17	5			10	0			0	5, 10
527	1662.72	1883.51	1232.18	590	390	200	5	6	8	10	0	0	0	0	5, 10
528	1438.94	1756.31	1218.88	591	242	349	5	6	8	11	0	99	0	0	6, 11
530	1198.5	1233.96	1156.59	48	26	22	8			16	0			0	8, 16
531	1092.29	1365.24	993.55	463	123	340	7	8	10	14	2	58	2	2	8
602	1560.38	1733.65	1130.66	348	248	100	5	7	8	11	16	10	15	13	11
603	1359.85	1867.68	953.586	126	56	70	5	7	10		0	18	0		7
607	1189.59	1424.01	1026.13	813	334	479	7	8	9	14	0	102	0	0	8
612	1644.06	1825.68	1213.77	283	199	84	5	6	8	11	29	24	27	24	11
616	1185.08	1425.85	1042.77	393	146	247	7	8	9	14	0	50	0	0	8
619	1753.07	1903.48	1666.34	175	64	111	5		6	10	15		19	15	5, 10
624	1758.28	1899.24	1536.03	621	380	241	5		6	10	0		0	0	5, 10
625	1370.86	1807.75	986.318	816	382	434	5	7	10	11	0	117	0	0	7
626	1329.59	1769.78	1071.43	963	356	607	5	7	9	11	0	138	0	0	7
627	1858.06	1891.32	1768.39	207	151	56	5			10	0			0	5, 10
628	1270.55	1636.75	1019.82	716	291	425	6	7	9	12	0	103	0	0	7
630	1213.02	1502.45	975.45	437	197	240	6	8	10	13	0	55	0	0	8
701	1884.35	1927.05	1798.95	63	42	21	5			10	10			10	5, 10
550	1498.72	1649.92	1343.06	550	279	271	6		7	12	0		0	0	6, 12

Theorem 1.

큰 아이템과 작은 아이템의 수가 각각 N_L개와 N_S개가 있다면 이들을 모두 패킹하는데 필요한 빈의 개수(B^*)와 최적 골의 개수(G^*)는 다음과 같다.

1. $S > \frac{L}{2}$인 경우, $1 \le G_1 \le \left\lfloor \frac{B}{L} \right\rfloor$, $\left\lfloor \frac{B}{L} \right\rfloor < G_2 \le \left\lfloor \frac{2B}{L+S} \right\rfloor$, $\left\lfloor \frac{2B}{L+S} \right\rfloor < G_3 \le \left\lfloor \frac{B}{S} \right\rfloor$ 와 $\left\lfloor \frac{B}{S} \right\rfloor < G_4 \le \left\lfloor \frac{2B}{L} \right\rfloor$ 을 만족하는 $G_i (i = 1, 2, 3, 4)$의 집합을 I라고 하자.
2. $S \le \frac{L}{2}$인 경우, $1 \le G_1 \le \left\lfloor \frac{B}{L} \right\rfloor$, $\left\lfloor \frac{B}{L} \right\rfloor < G_2 \le \left\lfloor \frac{2B}{L+S} \right\rfloor$ 와 $\left\lfloor \frac{2B}{L+S} \right\rfloor < G_3 \le \left\lfloor \frac{2B}{L} \right\rfloor$ 을 만족하는 G_i $(i = 1, 2, 3)$의 집합을 I라고 하자.

그러면 $B^* = \min\{B_i \mid i \in I\}$이고 $G^* = \{G_i \mid B_i = B^*, i \in I\}$이다.

Proof. $S > L/2$인 경우의 집합 I는 큰 아이템과 작은 아이템이 빈에 놓일 수 있는 가능한 Case의 수를 나타낸다. 이러한 경우들에서 집합 I에 속하는 Case에 해당하는 Lemma에 의하여 모든 아이템들을 빈에 패킹하기 위해 필요한 빈의 개수를 구하여 이 중에서 가장 작은 $B_i (i \in I)$가 B^*이 되고 $B^* = B_i$에 해당하는 $\{G_i \mid B_i = B^*, i \in I\}$가 최적 골 수 G^*가 된다. 그리고 $S \le L/2$인 경우에는 Case[4]와 Lemma 4를 고려하지 않는다는 것 이외에는 $S > L/2$인 경우와 동일하다. □

5. 실험 및 결과

위에서 제시한 최적해 기법의 수행 능력을 평가하기 위하여 3개월 분량의 실제 데이터를 가지고 실험을 해 보았다. 편의상 큰 아이템과 작은 아이템을 분류하기 위하여 대상 코일 외경의 평균을 구하였고, 평균값을 초과하는 값을 갖는 코일과 그렇지 않는 코일로 분류하였다. 그리고 B는 10125㎜, 전체 아이템의 평균을 초과하는 외경 값을 지닌 코일들의 외경 평균값과 개수를 각각 L과 N_L이라 하고 그 이외의 코일들의 외경 평균값과 개수를 각각 S와 N_S라고 하였다. 여기에서 평균값을 이용한 이유는 이웃하는 골에 놓인 아이템들의 외경 반지름 합이 골의 폭 이하가 되면 아이템을 패킹할 수 있고 편차의 합이 0이 되는 성질을 활용하기 위함이다.

실제 자료의 일일 출하대상코일의 각종 통계 값 및 최적 골 수 값은 <Table 2>와 같다.

여기서 코일 세트(980)을 살펴보면 $S > L/2$인 경우이므로 4가지 Case를 모두 고려해야 한다.

$$G_1 = \left\lfloor \frac{B}{L} \right\rfloor = 6,\ G_2 = \left\lfloor \frac{2B}{L+S} \right\rfloor = 7,$$
$$G_3 = \left\lfloor \frac{B}{S} \right\rfloor = 9,\ G_4 = \left\lfloor \frac{2B}{L} \right\rfloor = 12$$

그리고 Lemma 1에 의하면 $B_1 = 164$이고, Lemma 2에 의하면 $B_2 = 140$이다. Lemma 3에 의하면 $B_3 = 137$이며 Lemma 4에 의하면 $B_4 = 164$이다. 그러므로 $B^* = B_3 = 137$이고 $G^* = G_3 = 9$이 된다. 코일 세트(980)에 대해서 최적해 기법을 적용한 결과를 살펴보면 현재 사용되는 팔레트의 골수와 같은 값인 9골이 최적 골수임을 알 수 있다. 그러나 큰 아이템 크기와 작은 아이템 크기가 다르거나 큰 아이템과 작은 아이템의 개수의 비율이 달라지면 최적의 골수가 달라질 수 있다(<Table 2> 참조).

6. 결 론

이번 연구에서는 기존의 정해진 수의 아이템을 임의의 수용력을 가진 빈의 특정 위치인 골에 모두 패킹하면서 빈의 개수를 최소화하는 문제를 다루었다. 그리고 이 논문에서는 FFD Algorithm이 우리 문제에서는 최적의 알고리즘임을 보였고 우리 문제에 대해서 최적해를 보장하는 최적해 기법을 제안하였다.

주어진 문제를 파악 분석하고 단순화한 문제에 대한 해결 방법은 제안하였지만 아직까지 미진한 부분들을 보완하려면 다음과 같은 연구가 추진되어야 할 것이다.

첫째, 주어진 원래 문제에서는 코일의 세 가지 특성인 폭, 외경, 무게를 고려하여야 하는데 여기서는 그 중 외경만을 고려하였다. 그러므로 세 가지 특성을 모두 고려하였을 때도 최적해를 보장할 수 있는 최적해 기법의 제안이 요구된다.

둘째, 본 논문에서 제안한 최적해 기법은 아이템 크기가 두 종류라는 가정을 바탕으로 한 것인데 아이템 크기가 더 다양한 경우에도 최적 해를 보장하는 최적해 기법의 제안이 요구된다.

참고문헌

Baker, B. S. (1983), A New Proof for the First-Fit Decreasing Bin-Packing Algorithm, *Technical Memorandum*, Bell Laboratories, Murray Hill, NJ 07974.

Coffman, E. G., Garey, M. R., and Johnson, D. S. (1983), Approximation Algorithms for Bin-Packing-An Updated Survey, Bell Laboratories, Murray Hill, NJ, 49-106.

Coffman, E. G., Garey, M. R., and Johnson, D. S. (1996), Approximation Algorithm for Bin-Packing : A Survey, Approximation Algorithms for NP-Hard Problem, D. Hochbaum, PWS Publishing,

Boston, 46-93.

Graham, R. L. (1969), Bounds on Multiprocessing Timing Anomalies, *SIAM Journal Appl. Math.*, **17**(2).

Johnson, D. S. (1973), Near-Optimal Bin Packing Algorithms, *Technical Report MACTR-109*, Project MAC, Massachusetts Institute of Technology, Cambridge, Mass.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., and Graham, R. L. (1974), Worst-Case Performance Bound for Simple One-Dimensional Packing Algorithms, *SIAM Journal of Computing*, **3**(4), 299-326.