Journal of Korea Multimedia Society Vol. 19, No. 2, February 2016(pp. 443-450)
http://dx.doi.org/10.9717/kmms.2016.19.2.443

First-fit 전략을 사용하는 템플릿 패킹 문제를 위한 근사 알고리즘

송하주†, 권오흠††

An Approximation Algorithm based on First-fit Strategy for Template Packing Problem

Ha-Joo Song†, Oh-Heum Kwon††

ABSTRACT

This paper deals with a kind of packing problem of which the goal is to compose one or more templates which will be used to produce the items of different types. Each template consists of a fixed number of slots which are assigned to the different types of items and the production of the items is accomplished by printing the template repeatedly. The objective is to minimize the total number of produced items. This problem is known to be NP-hard. We present a polynomial time approximation algorithm which has a constant approximation ratio. The proposed algorithm is based on the well-known first-fit strategy.

Key words: NP-complete, Approximation Algorithm, Template Packing Problem, Approximation Ratio

1. 서 론

패킹(packing) 문제는 여러 개의 아이템(item)들을 일정한 제약조건하에서 빈(bin)에 담는 것에 관한 문제이다[4]. 구체적인 응용 사례에 따라서 수많은 변형들이 존재하는 매우 고전적인 최적화 문제의 하나이다. 가장 기본 형태의 패킹 문제에서는 각각의 아이템이 하나의 스칼라(scalar) 값으로 표현되고, 모든 빈은 고정된 용량(capacity)을 가진다. 각 빈에 수용된 아이템들의 합은 그 빈의 용량을 초과할 수 없으며, 모든 아이템들을 수용하기 위해서 필요한 빈의 최소 개수를 구하는 것이 목적이다. 2차원 빈 패킹 문제에서는 각각의 아이템들이 너비와 높이를 가지는 2차원 사각형으로 표현되며, 각각의 빈 역시 하나의 2차원 사각형으로 표현된다. 아이템들은 서로 겹치지 않게 패킹되어야 한다. 2차원 스트립(strip) 패킹 문제는 높이가 무한대인 하나의 빈에 모든 아이템을 패킹하는 것이며 이 경우 목적함수는 사용된 빈의 높이를 최소화하는 것이다. 이외에도 선반(shelf) 패킹 문제나 병렬 태스크 스케쥴링(parallel task scheduling), 일반화된 할당 문제(generalized assignment problem)등도 일종의 빈 패킹 문제의 변형으로 볼 수 있다[1,2,3,5,7].

이 논문에서는 패킹 문제의 또 다른 한 형태인 템플릿(template) 패킹 문제를 다룬다. 템플릿 패킹 문제는 제품에 부착되는 라벨을 생산하는 응용을 통해서 가장 잘 설명될 수 있다. 모양과 크기는 동일하지만 제품의 종류에 따라 조금씩 다른 내용의 라벨들이 서로 다른 수량으로 생산되어야 한다고 가정하자. 이 라벨들을 생산하기 위해서 먼저 템플릿을 제작하는

※ Corresponding Author : Oh-Heum Kwon, Address: (608-737) Daeyeon-dong 599-1, Nam-gu, Busan, Korea, TEL : +82-10-9359-9568, E-mail : ohkwn@pknu.ac.kr
Receipt date : Jan. 22, 2016, Approval date : Feb. 3, 2016

† Dept. of IT Convergence and Application, Pukyong National University (E-mail : hajoosong@pknu.ac.kr)
†† Dept. of IT Convergence and Application, Pukyong National University
※ This work was supported by Pukyong National University Research Grant CD20140326.

데, 각 템플릿에는 고정된 개수의 라벨 원형들이 배치된다. 제작된 템플릿들을 각각 필요한 수량만큼 인쇄하여 각 제품별로 요구된 수량의 라벨들을 생산한다. 각각의 템플릿에는 서로 다른 종류의 라벨들이 서로 다른 수량으로 배치될 수 있지만 하나의 라벨 유형은 하나의 템플릿에만 배치된다.

예를 들어 4종류의 라벨들 A, B, C, D가 있고, 주문 수량이 각각 $q_A=8$, $q_B=15$, $q_C=24$, 그리고 $q_D=30$라고 하고, 2개의 템플릿을 사용하며, 각 템플릿에는 최대 8개의 라벨을 배치할 수 있다고 가정하자. 첫 번째 템플릿 T_1에 라벨 A와 C를 각각 $x_A=2$, $x_C=6$장 배치하고, 두 번째 템플릿 T_2에는 라벨 B와 D를 각각 $x_B=3$, $x_D=5$장 배치한다. 이제 T_1을 4장, 그리고 T_2를 6장 인쇄하면 총 80개의 라벨이 생산되며 (A가 8개, B가 18개, C가 24개, D가 30개), 라벨 타입별 요구량을 모두 충족한다. 필요한 수량은 총 8 + 15 + 24 + 30 = 77개이므로 낭비된 라벨의 개수는 3개이다.

템플릿 패킹 문제를 정형화하면 다음과 같다. n종류의 아이템이 있고, 각 아이템별 주문 수량은 q_i, $i=1,2,...,n$,이다. 하나의 템플릿에 배치될 수 있는 아이템의 최대 개수, 즉 템플릿의 용량은 C이다. m개의 템플릿을 사용했다고 가정하고, 템플릿 T_j, $j=1,2,...,m$,상에 배치될 타입 i인 아이템의 개수를 x_{ij}로 표시한다. 이때 $x_{i1},x_{i2},...,x_{im}$ 중 하나만이 양의 정수이고 나머지는 모두 0이어야 한다. 하나의 템플릿에는 최대 C개의 아이템만이 배치될 수 있으므로 $\sum_{i=1}^{n} x_{ij} \le C$, $j=1,2,...,m$,이어야 한다. 아이템 i가 템플릿 j에 배치될 때, 즉 $x_{ij}>0$일 때 "템플릿 j는 아이템 i를 커버(cover)한다"라고 말한다.

주문량을 충족하기 위한 각 템플릿의 인쇄 수량은 이러한 배치 $\{x_{ij} \mid i=1,...,n,\ j=1,...,m\}$로부터 유일하게 결정된다. 템플릿 T_j는 자신이 커버하는 모든 아이템의 주문 수량을 충족해야 하므로 템플릿 T_j의 인쇄 수량 s_j는 $s_j \ge \lceil q_i/x_{ij} \rceil$, $i=1,...,n,\ x_{ij}\neq 0$, 이어야 한다. 즉 다음과 같이 결정된다.

$$s_j = \max_{i\ with\ x_{ij}\neq 0} \lceil \frac{q_i}{x_{ij}} \rceil$$

알고리즘의 목적은 주문량을 충족하기 위한 인쇄 수량의 총합 즉 $\sum_{j=1}^{m} s_j$를 최소로 만드는 것이다. 본 논문에서는 이 문제를 m-MTP (Minimum Template Packing) 문제라 부른다[6].

m-MTP 문제는 템플릿의 개수 m이 고정되어 있는 경우에 관한 것이다. 템플릿의 개수 m을 고정하는 대신 하나의 템플릿을 추가 제작하는 비용 α가 주어지고 $\sum_{j=1}^{m} s_j + m\alpha$를 최소화하는 m과 x_{ij}를 결정하는 좀 더 일반화된 문제를 생각할 수 있다. 이 문제는 $m=1,2,...,n$에 대해서 m-MTP 문제를 풀어서 목적함수를 최소화하는 m을 구하면 해결할 수 있다. 즉 m-MTP 문제는 이 일반화된 문제의 하나의 부분 문제로 생각할 수 있으며, 본 논문에서는 m-MTP 문제에 대해서만 다룬다.

m-MTP는 대부분의 패킹 문제들과 마찬가지로 NP-hard에 속하며, 이 문제에 대한 선행 연구로 탐욕적 기법(greedy method)이나 분기한정법에 기초한 근사 알고리즘들이 제시되어 있다[6]. 하지만 이러한 알고리즘들에 대한 실험적인 성능평가만이 제시되어 있을 뿐이다. 근사 알고리즘의 성능을 분석하는 한 가지 방법은 알고리즘의 근사율(approximation ratio)을 분석하는 것이다. 최소화(minimization) 문제의 경우 임의의 입력 I에 대해서 근사 알고리즘의 해 $S(I)$와 최적해 $S_{OPT}(I)$간에 $S(I) \le \rho S_{OPT}(I) + c$의 관계가 성립할 때 근사 알고리즘의 근사율은 ρ라고 말한다. 여기서 c는 임의의 상수이다. 본 논문에서는 m-MTP 문제에 대해서 m이 2 이상의 상수일 때 근사율이 3인 근사 알고리즘을 제시한다. 제시된 알고리즘은 first-fit 전략을 사용하며, 이 문제에 대해서 처음으로 근사율을 증명한 것이다.

논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제시하는 알고리즘을 기술하고, 3장에서는 결론을 제시한다.

2. 알고리즘

본 절에서는 먼저 m-MTP 문제의 완화된(relaxed) 버전인 m-FMTP (Fractional-MTP) 문제를 해결하는 근사 알고리즘을 제시한다. 이 완화된 버전에서는 m-MTP 문제의 조건이 다음과 같이 완화된다:

$x_{ij}>0$인 모든 (i,j)에 대해서 $s_j \ge \frac{q_i}{x_{ij}}$

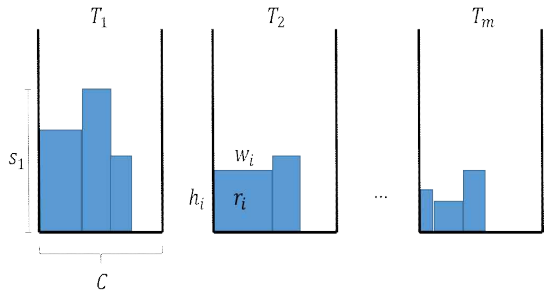

Fig. 1. Two-dimensional bins and items.

즉 s_{ij}가 정수가 아닐 수도 있다. m-FMTP 문제의 해는 s_j들의 올림(ceiling)을 취함으로써 항상 m-MTP 문제의 해로 변환될 수 있다. 따라서 m-FMTP 문제에 대해서 목적함수의 값이 S인 해가 존재한다면 m-MTP 문제에 대해서는 목적함수의 값이 $S+m$인 해가 존재한다.

1장에서 언급한 바와 같이 m-FMTP 문제는 빈 패킹 문제와 닮은 점이 있다. 각각의 템플릿은 용량이 C인 빈(bin)으로 볼 수 있고 각각의 라벨 타입은 빈에 패킹될 아이템으로 볼 수 있다. 템플릿 T_j의 x_{ij}개의 슬롯이 라벨 타입 i에게 할당되었다면 빈 j의 용량을 x_{ij}만큼 소비한 것이며 또한 $s_j \geq \frac{q_i}{x_{ij}}$이어야 한다. 우리는 이 상황을 Fig. 1과 같이 각각의 빈을 폭(width)이 C인 2차원 스트립(strip)으로 표현하고, 각각의 아이템을 폭이 x_{ij}이고 높이가 $\frac{q_i}{x_{ij}}$인 사각형으로 표현한다. 하나의 빈에 할당된 사각형들의 너비의 합은 C를 초과할 수 없다. 그리고 하나의 빈에 할당된 사각형들의 높이의 최대값이 s_j가 된다. 각 사각형의 너비 x_{ij}는 고정된 값이 아니며 사각형의 면적을 보존하는 한 변화될 수 있다.

이하에서는 각각의 템플릿들을 빈이라고 부르고 각각의 라벨 타입들을 단순히 아이템이라고 부르겠다. 각각의 아이템은 하나의 순서쌍 (w_i,h_i)로 표현되는데 여기서 $w_ih_i=q_i$이고 w_i는 C 이하의 양의 정수이다. 너비 w_i가 1인 아이템들을 singular 아이템이라고 부른다. 처음에는 모든 아이템이 singular 아이템이다. 즉 처음에는 아이템들의 집합 $\{(q_i,1) \mid i=1,...,n\}$으로 시작한다. 알고리즘이 진행되는 동안 각각의 아이템들은 자신의 면적을 유지하면서 다음 절에서 기술할 Expand-To-Fit과 Contract-To-Fit이라는 두 가지 기본 알고리즘에 의해서 변형(deform)되어 진다.

2.1 Expand-To-Fit과 Contract-To-Fit 알고리즘

아이템들의 집합 $I=\{(w_i,h_i) \mid i=1,...,k\}$에 속한 임의의 두 아이템 i와 j에 대해서 $h_j \leq 2h_i$가 성립할 때, 즉 어떤 아이템의 높이도 다른 아이템의 높이의

Algorithm 1: Expand-To-Fit(I, C_{upper})

Input: $I=\{(w_i,h_i) \mid i=1,2,...,k\}$ and $C_{upper} \geq \sum_{i=1}^{k} w_i$;

1: let $w_{\sum} = \sum_{i=1}^{k} w_i$;
2: **while** $w_{\sum} < C_{upper}$ **do**
3: let (w_i,h_i) be the tallest and (w_j,h_j) be the second tallest item with $w_i < C$ and $w_j < C$;
4: let δ be the smallest integer such that $\frac{w_ih_i}{w_i+\delta} < h_j$;
5: **if** $\delta > \min\{C-w_i, C_{upper}-w_{\sum}\}$ **then**
6: let $\delta = \min\{C-w_i, C_{upper}-w_{\sum}\}$;
7: **end.**
8: replace (w_i,h_i) with a new item $(w_i+\delta, \frac{w_ih_i}{w_i+\delta})$;
9: let $w_{\sum} = w_{\sum}+\delta$;
10: **end.**
11: **return** I;

2배 이내일 때 이 아이템들의 집합 I는 적절하다(proper)라고 부른다. 또한 만약 $1 < w_i < C$이고 $w_j < C$인 임의의 두 아이템 i와 j에 대해서 $h_j \le 2h_i$가 성립하면 I는 "거의 적절하다(semi-proper)"라고 부른다. 즉 너비가 1이거나 혹은 C인 예외적인 아이템들을 제외하고는 적절하다는 의미이다.

Expand-To-Fit 알고리즘의 입력은 하나의 거의 적절한 아이템들의 집합 $\{(w_i,h_i) \mid i=1,...,k\}$와 목표 용량 $C_{upper} \ge \sum_{i=1}^{k} w_i$이다. 알고리즘의 목적은 모든 혹은 일부 아이템들의 너비를 확대하여 아이템들의 너비의 합이 주어진 목표 용량 C_{upper}가 되면서 아이템들의 최대 높이가 최소가 되도록 하는 것이다. 이것은 간단한 탐욕적 알고리즘을 적용하여 가능하다. 각 스텝에서 높이가 최대인 아이템을 선택한다. 그런 다음 그것이 두 번째로 높은 아이템보다 낮아지거나, 너비가 C에 도달하거나, 혹은 아이템들의 총 너비가 C_{upper}가 될 때까지 너비를 확대한다. 이 과정을 총 너비가 C_{upper}가 될 때 까지 반복한다. 알고리즘1은 이 과정을 의사코드의 형태로 기술한다.

알고리즘1의 라인 4에서는 가장 높은 아이템을 두 번째로 높은 아이템보다 낮게 만들기 위해서 너비를 얼마나 확장할 것인지를 결정한다. 라인 5와 6에서는 선택된 아이템의 너비가 C보다 커지거나 혹은 아이템들의 너비의 총 합이 C_{upper}가 넘지 않도록 한다.

보조정리 1: 만약 거의 적절한 아이템들의 집합이 Expand-To-Fit 연산에게 입력으로 주어진다면 출력도 또한 거의 적절하다.

증명: 그렇지 않다고 가정해보자. 즉 알고리즘1의 출력에 $1 < w_i < C$이고 $w_j < C$이면서 $h_j > 2h_i$인 두 아이템 (w_i,h_i)와 (w_j,h_j)가 존재한다고 가정해보자. 두 가지 경우로 나누어서 생각한다. 첫 번째 경우는 아이템 i가 알고리즘이 진행되는 동안 적어도 한 번 확장된 경우이다. 아이템 i가 가장 긴 아이템으로 선택된 마지막 순간을 생각해보자. 그 시점에서 두 아이템의 높이를 각각 h'_i와 h'_j로 나타내자. 그러면 $h'_i \ge h'_j$일 것이다. 어떤 아이템도 알고리즘의 진행 과정에서 높이가 늘어나지는 않으므로 $h_j \le h'_j \le h'_i$이다. 이제 라인 4에서 선택된 정수를 δ라고 하자. 다시 세 가지 경우로 나뉜다. 첫 번째 경우는 $\delta \le \min\{C-w_i, C_{upper}-w_{\sum}\}$인 경우이다. 이 경우 δ값은 변하지 않고 그대로 사용된다. 그러면 $h_i = \frac{w'_i h'_i}{w'_i+\delta}$가 되고 $\frac{w'_i h'_i}{w'_i+\delta-1} \ge h'_j$가 만족된다.

$\frac{w'_i h'_i}{w'_i+\delta-1} \le \frac{2w'_i h'_i}{w'_i+\delta}$가 성립하므로 $h_j \le 2h_i$가 성립한다. 이것은 모순이다. 두 번째 경우는 $\delta > C_{upper}-w_{\sum}$이고 $C-w_i > C_{upper}-w_{\sum}$인 경우이다. 이 경우 아이템 i는 마지막으로 확장된 아이템일 것이므로 따라서 $h_j = h'_j$일 것이다. 더구나 δ는 아이템 i를 두 번째로 높은 아이템보다 짧게 만드는 최소의 정수이므로 당연히 $h'_j \le h_i$일 것이다. 그러므로 모순이다. 세 번째 경우는 $\delta > C-w_i$이고 $C-w_i \le C_{upper}-w_{\sum}$인 경우이다. 이 경우 아이템 i의 너비 w_i는 C가 되므로 역시 모순이다.

이제 아이템 i가 알고리즘이 진행되는 동안 한 번도 확장되지 않은 경우를 고려해보자. (w_i^0,h_i^0)와 (w_j^0,h_j^0)을 각각 입력에서의 아이템 i와 j라고 하자. 그러면 $h_i = h_i^0$이고 $h_j \le h_j^0$이고 $h_j^0 \le 2h_i^0$이다. 따라서 증명이 완료된다. ■

Contract-To-Fit 알고리즘의 입력은 다음과 같은 세 가지 조건을 만족하는 적절한(proper) 아이템들의 집합 $\{(w_i,h_i) \mid i=1,...,k\}$이다.

(1) $1 < w_i < C,\ i=1,2,...,k$;

(2) $w_i \ge w_{i+1},\ i=1,2,...,k-1$;

(3) $\sum_{i=1}^{k-1} w_i < C$이고 $\sum_{i=1}^{k} w_i > C$이다.

Contract-To-Fit은 일부 혹은 전체 아이템들의 너비를 줄여서 총 너비가 C가 되도록 아이템들을 변형한다. 이 연산은 사실상 Expand-To-Fit의 역변환에 가깝다. 먼저 너비가 1이 아닌 아이템들 중에서 가장 짧은 아이템을 선택한다. 그런 다음 두 번째로 짧은 아이템보다 더 길어지도록 너비를 축소한다. 이 과정을 너비의 총합이 C가 될 때 까지 반복한다. 아래의 알고리즘2가 이 과정을 의사코드의 형태로 기술한다.

아래의 보조정리2는 알고리즘2가 진행되는 동안 어떤 아이템도 다른 아이템보다 2배 이상 길어지지 않는다는 것을 증명한다. 또한 이것은 알고리즘2의

라인4에서 그런 조건을 만족하는 δ가 항상 존재함을 의미한다.

보조정리2: $I=\{(w_i,h_i) \mid i=1,2,...,k\}$를 알고리즘2의 while 반복문에서 임의의 반복의 끝에서의 아이템의 집합라고 하자. 그러면 집합 I는 적절하다.

증명: $h_i > 2h_j$라고 가정해보자. $\{(w_i^0,h_i^0) \mid i=1,2,...,k\}$이 알고리즘2에 대한 입력이었다고 가정하자. $h_i^0 \le 2h_j^0$이고 $h_j \ge h_j^0$이므로 따라서 $h_i > h_i^0$이다. 이것은 아이템 i가 적어도 한 번 가장 짧은 아이템으로 선택된 적이 있음을 의미한다. h_i'과 h_j'을 각각 그 시점에서의 아이템 i와 j의 높이라고 하자.

그 시점에서 아이템 i가 non-singular 아이템들 중에서 가장 짧은 아이템이므로 다음과 같은 두 가지 경우가 가능하다: (1) $h_i' \le h_j'$이거나 혹은 (2) $h_i' > h_j'$ 이면서 아이템 j가 singular한 경우이다. 먼저 경우 (1)을 생각해보자. δ를 라인 4에서 선택된 정수라고 하고, 그것이 라인 5와 6에서 바뀌지 않았다고 가정해보자. 그러면 $h_i = \frac{w_i h_i}{w'_i - \delta}$이고 $\frac{w_i h_i}{w'_i - (\delta-1)} \le h'_j$이다. $1 \le \delta \le w'_i$일 때 $\frac{w_i h_i}{w'_i - \delta} \le 2 \cdot \frac{w_i h_i}{w'_i - (\delta-1)}$이 성립하므로 따라서 $h_i \le 2 \cdot \frac{w_i h_i}{w'_i - (\delta-1)} \le 2h'_j \le 2h_j$이다. 이것은 모순이다. 이번에는 $\delta > w_{\sum} - C$ 여서 라인 6에서 갱신되었다고 가정해보자. 이것은 아이템 i가 알고리즘의 진행 과정에서 가장 마지막으로 축소(contracted)된 아이템이라는 의미이다. 따라서 $h_j = h'_j$이고, 또한 δ는 아이템 i를 두 번째로 작은 아이템보다 높게 만드는 최소값이므로 결과적으로 $h_i \le h_j$이다. 이것 역시 모순이다.

이제 경우 (2)를 생각해보자. $h_j^0 \ge 2$이고 $h'_j = 1$이므로 $h'_j \ge 2h_j^0$이고 $w_j = w'_j \le w_j^0/2$이다. $h'_j = h_j$이고 $h_i^0 \le 2h_j^0$이므로 $h_i > 2h_j = 2h'_j \ge 4h_j^0 \ge 2h_i^0$이고 따라서 $w_i < w_i^0/2$이다. 그러므로 $\sum_{p=1}^{k} w_p$

$< \sum_{p \ne i \text{ and } p \ne j} w_p^0 + (w_i^0 + w_j^0)/2 = \sum_{p=1}^{k} w_p^0 - (w_i^0 + w_j^0)/2$

$\le \sum_{p=1}^{k} w_p^0 - w_k^0 = \sum_{p=1}^{k-1} w_p^0 < C$이다. 이것은 모순이다. ■

2.2 Packing 알고리즘

패킹 알고리즘은 초기 singular 아이템들의 집합 $I_{init} = \{(1,q_i) \mid i=1,...,n$에서 시작한다. 여기서 아이

Algorithm 2: Contract-To-Fit(I)

Input: $I=\{(w_i,h_i) \mid i=1,2,...,k\}$ satisfying conditions (1), (2) and (3);

1: let $w_{\sum} = \sum_{i=1}^{k} w_i$;
2: **while** $w_{\sum} > C$ **do**
3: let (w_i,h_i) be the shortest non-singular item and (w_j,h_j) be the shortest, possibly singular, item among those no shorter than (w_i,h_i);
4: let $\delta < w_i$ be the smallest integer such that $\frac{w_i h_i}{w_i - \delta} > h_j$;
5: **if** $\delta > w_{\sum} - C$ **then**
6: let $\delta = w_{\sum} - C$;
7: **end.**
8: replace (w_i,h_i) with a new item $(w_i - \delta, \frac{w_i h_i}{w_i - \delta})$;
9: let $w_{\sum} = w_{\sum} - \delta$;
10: **end.**
11: **return** I;

템들은 수량에 대해서 내림차순으로 정렬되어 있다고 가정한다. 즉 $q_i \geq q_{i+1}$, $i=1,...,n-1$,이다. 첫 번째 단계는 이 아이템들에 Expand-To-Fit 알고리즘을 적용하는 것이다. 이때 $C_{upper}=mC$로 한다. 즉 아이템들의 너비의 합이 m개의 템플릿에 의해서 제공되는 전체 용량인 mC가 되도록 만드는 것이다. 이제 $I=\{r_i=(w_i,h_i) \mid i=1,...,n$를 그 결과라고 하자. I_{init}은 거의 적절하므로(semi-proper) 보조정리1에 의해서 I 역시 거의 적절하다. 또한 I에서 아이템들은 너비에 대해서 내림차순으로 정렬되어 있고 너비가 동일한 경우에는 높이에 대해서 내림차순으로 정렬되어 있다. 즉 $w_i > w_{i+1}$이거나 만약 $w_i = w_{i+1}$이라면 $h_i \geq h_{i+1}$이다.

보조정리3: $h_{\max}$를 I에 속한 아이템들의 높이의 최대값이라고 하자. 그러면 $S_{OPT} \geq h_{\max}$이다. 여기서 S_{OPT}는 $m-FMLP$ 문제에 대한 최적해를 나타낸다.

증명: Expand-To-Fit(I_{init}, mC)는 아이템들의 너비의 합이 mC를 초과하지 않는 한도 내에서 아이템의 최대 높이를 최소화한다. 따라서 적어도 하나의 템플릿 T_j에 대해서 $s_j \geq h_{\max}$일 수 밖에 없다. 따라서 이 보조정리가 성립한다. ■

이제 m개의 템플릿들을 하나씩 순차적으로 구성한다. 첫 번째 템플릿 T_1은 I에 속한 아이템들을 처음으로 너비의 합이 C보다 크거나 같아질 때까지 순서대로 추가하여 구성한다. 즉 T_1은 아이템 집합 $I_1=\mathrm{r}_1,\mathrm{r}_2,...,\mathrm{r}_k$으로 구성한다. 여기서 k는 $\sum_{i=1}^{k} w_k \geq C$가 되는 최소값이다. 만약 $\sum_{i=1}^{k} w_k > C$이면 집합 I_1에 속한 아이템들을 Contract-To-Fit 알고리즘을 적용하여 너비의 합이 C가 되도록 만든다. 이 경우 I_1은 적절(proper)하므로 Contract-To-Fit 알고리즘을 적용하기 위해 필요한 3가지 조건을 모두 만족한다.

이제 남은 아이템들의 너비의 합은 $(m-1)C$ 이하이다. 만약 $(m-1)C$ 미만이면 이 아이템들에 대해서 다시 Expand-To-Fit을 적용하여 너비의 합이 정확히 $(m-1)C$가 되도록 조정한다. 그런 후 이번에는 두 번째 템플릿 T_2를 동일한 방법으로 구성한다. 이 과정을 모든 아이템들이 커버될 때까지 반복한다. 알고리즘 3은 이 과정을 의사코드의 형태로 표현한 것이다.

2.3 근사율의 증명

템플릿 T_i에 대해서 알고리즘 3의 라인 6에서 선택된 아이템들의 집합을 I_i로 나타내고, 라인 7에서 Contract-To-Fit 알고리즘을 적용한 결과를 I_i'이라고 표시하자. T_s를 I_s가 적어도 하나의 singular 아이템을 포함하는 최초의 템플릿이라고 하자. 아이템들은 너비에 대해서 내림차순으로 고려되므로 I_s의 마지막 아이템은 singular일 것이고 따라서 I_s에 속한 아이템들의 너비의 합은 C를 초과하지 않을 것이다. 또한 라인 9에서 Expand-To-Fit은 아무 일도 하지

Algorithm 3: Greedy-Fit(m, C)

Input: $I_{init}=\{(1,q_i) \mid i=1,2,...,n\}$ be the initial items with $q_i \geq q_{i+1}, i=1,...,n-1$;

1: I = Expand-To-Fit(I_{init}, mC);
2: Let $I=\{r_i=(w_i,h_i) \mid i=1,...,n\}$;
3: begin = 1;
4: for $j=1$ to m **do**
5: let $k \geq begin$ be the first index with $\sum_{i=1}^{k} w_k \geq C$;
6: let $I_j=\{r_{begin},...,r_k\}$ and let $I=I-I_j$;
7: I_j' = Contract-To-Fit(I_j);
8: for each item $(w'_i,h'_i) \in I'_j$, assign w'_i slots of T_j to the labell type i;
9: I = Expand-To-Fit(I, $(m-1)C$);
10: $begin=k+1$;
11: **end.**
12: **return** $T_1,...,T_m$;

않을 것이고 남아있는 모든 아이템들도 역시 singuar일 것이다. 따라서 템플릿 $T_{s+1},...,T_m$은 오직 singular 아이템들로만 구성될 것이다. $I_j', j<s$,에는 알고리즘 3의 라인 7의 Contract-To-Fit의 결과로 singular 아이템이 포함될 수 있다.

이하에서는 템플릿 $T_1,...,T_{s-1}$을 non-singular 템플릿, T_s를 mixed 템플릿, 그리고 $T_{s+1},...,T_m$을 singular 템플릿이라고 부르겠다. $h_i^{\max}$와 $h_i^{\min}$을 각각 I_i'에 속한 아이템들의 높이의 최대값과 최소값이라고 하자.

보조정리4: 임의의 non-singular 템플릿 T_i에 대해서 $h_i^{\min} \geq \frac{1}{2}h_i^{\max}$이다.

증명: 임의의 non-singular 템플릿 T_i에 대해서 아이템 집합 I_i는 너비가 C인 하나의 아이템만으로 구성되거나 혹은 Contract-To-Fit 알고리즘의 입력이 되기 위한 3가지 조건을 만족한다. 따라서 두 가지 경우 모두 이 보조정리는 자명하게 성립한다. ■

보조정리5: $\sum_{j=1}^{m} h_j^{\min} \leq \frac{1}{C}\sum_{i=1}^{n} q_i$ 이다.

증명:

$\sum_{i=1}^{n} q_i = \sum_{j=1}^{m} \sum_{(w_i,h_i)\in I_j'} w_i h_i \geq \sum_{j=1}^{m} \sum_{(w_i,h_i)\in I_j'} w_i h_j^{\min}$

$= \sum_{j=1}^{m} h_j^{\min} C = C\sum_{j=1}^{m} h_j^{\min}$ 이다. ■

S_{GR}와 S_{OPT}를 각각 $m-$FMTP에 대한 우리의 알고리즘과 최적 알고리즘의 해라고 하자. $S_{GR} = \sum_{i=1}^{m} h_i^{\max}$이고, S_{OPT}에 대해서는 다음과 같이 2가지 하한(lower bound)이 성립한다: (1) $S_{OPT} \geq \sum_{i=1}^{n} q_i / C$; (2) $S_{OPT} \geq \max_{i=s}^{m} h_i^{\max}$. 하한 (1)이 성립하는 것은 자명하다. 하한 (2)는 보조정리 3에 의해서 역시 자명하다.

이제 3가지 경우로 나누어서 고려한다: (1) mixed 템플릿이 존재하지 않는 경우; (2) mixed 템플릿 T_s가 존재하고 $h_s^{\max} > 2h_s^{\min}$인 경우; 그리고 (3) mixed 템플릿 T_s가 존재하고 $h_s^{\max} \leq 2h_s^{\min}$인 경우이다. 우선 경우 (1)을 고려해보자. 보조정리 4와 5에 의해서 $S_{GR} = \sum_{j=1}^{m} h_j^{\max} \leq 2\sum_{j=1}^{m} h_j^{\min} \leq 2S_{OPT}$가 성립한다. 경우 (2)를 생각해보자. 보조정리 1에 의해서 mixed 템플릿에서 최소 높이를 가진 아이템은 singular일 수밖에 없다. 모든 singular 아이템은 높이에 대해서 내림차순으로 정렬되어 있으므로 $h_j^{\min} \geq h_{j+1}^{\max}, j=s,...,m-1$,이다. 따라서 $S_{GR} = \sum_{j=1}^{m} h_j^{\max}$

$= \sum_{j=1}^{s-1} h_j^{\max} + h_s^{\max} + \sum_{j=s+1}^{m} h_j^{\max} \leq 2\sum_{j=1}^{s-1} h_j^{\min} + h_s^{\max}$

$+ \sum_{j=s}^{m-1} h_j^{\min} \leq 2\sum_{j=1}^{m} h_j^{\min} + h_s^{\min} \leq 3S_{OPT}$이다. 이제 경우 (3)을 생각해보자. 이 경우 $h_j^{\min} \geq h_{j+1}^{\max}$, $j=s+1,...,m-1$,은 여전히 성립한다. 따라서 $S_{GR} = \sum_{j=1}^{m} h_j^{\max} = \sum_{j=1}^{s} h_j^{\max} + \sum_{j=s+1}^{m} h_j^{\max} = 2\sum_{j=1}^{s} h_j^{\min}$

$+ h_{s+1}^{\max} + \sum_{j=s+1}^{m-1} h_j^{\min} \leq 2\sum_{j=1}^{m} h_j^{\min} + h_{s+1}^{\max} \leq 3S_{OPT}$가 성립한다.

이제 $S'_{GR} = \sum_{j=1}^{m} \lceil h_j^{\max} \rceil$ 은 $m-$MTP 문제에 대한 하나의 해가 된다. S'_{OPT}를 $m-$MTP 문제에 대한 최적해라고 하면 다음과 같은 정리가 성립한다.

정리1: $S'_{GR} \leq S_{GR} + m \leq 3S_{OPT} + m \leq 3S'_{OPT} + m$ 이다.

3. 결 론

본 논문에서는 템플릿 패킹 문제에 대해서 처음으로 상수 근사율을 가지는 다항시간 알고리즘을 제시하였다. 제안된 알고리즘은 잘 알려진 휴리스틱들 중 하나인 best-fit 전략을 사용하였다. 본 논문에서 증명한 근사율이 타이트(tight)한지는 아직 밝히지 못했으며, 또한 근사율을 향상하는 것이 추후 연구과제이다.

REFERENCE

[1] R. Cohen, L. Katzir, and D. Raz, "An Efficient Approximation for the Generalized Assignment Problem," *Information Processing Letters*, Vol. 100, No. 4, pp. 162-166, 2006.

[2] J. Du and J.Y. Leung, "Complexity of Scheduling Parallel Task Systems," *SIAM Journal of Discrete Mathematics*, Vol. 2, No.

4, pp. 473-487, 1989.

[3] L. Fleischer, M.X. Goemans, V.S. Mirrokni, and M. Sviridenko, "Tight Approximation Algorithms for Maximum General Assignment Problems," *Proceedings of Symposium on Discrete Algorithms*, pp. 611-620, 2006.

[4] M. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the NP-Completeness*, W.H. Freemac and Company, New York, 1979.

[5] A. Lodi, S. Martello, and D. Vigo, "Recent Advances on Two-dimensional Bin Packing Problems," *Discrete Applied Mathematics*, Vol. 123, No. 3, pp. 379-396, 2002.

[6] O.H. Kwon and H.J. Song, "Improved Approximation Algorithm Based on Branch-and-Bound Technique for Label Packing Problem," *Journal of Korean Institute of Next Generation Computing,* Vol. 10, No. 10, pp. 13-24, 2014.

[7] H.M. Kwon, "Two Level Bin-Packing Algorithm for Data Allocation on Multiple Broadcast Channels," *Journal of Korea Multimedia Society,* Vol. 14, No. 9, pp. 1165-1174, 2011.

송 하 주

1993년 서울대학교 컴퓨터공학과 졸업
1995년 서울대학교 대학원 컴퓨터공학과 졸업(공학석사)
2001년 서울대학교 대학원 전기컴퓨터공학부 졸업 (공학박사)
2003년 8월 ㈜아이티포웹 부장
2003년 9월~현재 부경대학교 교수
관심분야 : 데이터베이스 시스템, RFID/USN

권 오 흠

1988년 서울대학교 컴퓨터공학과 졸업
1991년 KAIST 전산학과 졸업 (공학석사)
1996년 KAIST 전산학과 졸업 (공학박사)
1997년~현재 부경대학교 교수
관심분야 : 알고리즘 설계 및 분석, 무선 센서 네트워크