DOI: 10.3745/KIPSTB.2009.16B.5.403

적합성 함수를 이용한 2차원 저장소 적재 문제의 휴리스틱 알고리즘

연 용 호[†] · 이 선 영[††] · 이 종 연[†††]

요 약

2차원 저장소 적재는 NP-hard 문제로서 그 문제의 정확한 해를 구하는 것이 어려운 것으로 알려져 있으며, 이의 더 좋은 해를 얻기 위해 유전자(genetic) 알고리즘, 시뮬레이티드 어닐링(simulated annealing), 타부서치(tabu search)등과 같은 근사적 접근법이 제안되어 왔다. 하지만 분지한계(branch-and-bound)나 타부서치 기법들을 이용한 기존의 대표적인 근사 알고리즘들은 휴리스틱 알고리즘의 해에 기반을 둠으로 효율성이 낮고 반복수행에 의한 계산시간이 길다. 따라서 본 논문에서는 이러한 근사 알고리즘의 복잡성을 간소화하고, 알고리즘의 효율성을 높이기 위해 적재가능성을 판단하는 적합성 함수(fitness function)를 정의하고 이를 이용하여 어떤 특정 개체의 적재영역을 판단하는데 영향을 주는 적재영역의 수를 계산한다. 또한, 이들을 이용한 새로운 휴리스틱 알고리즘을 제안하였다. 끝으로 기존의 휴리스틱 또는 메타휴리스틱 기법과의 비교실험을 통해 기존의 휴리스틱 알고리즘인 *FFF* 와 *FBS* 에 비해 97%의 결과가 같거나 우수하였으며, 타부서치 알고리즘에 비해 86%의 결과가 같거나 우수한 것으로 나타났다.

키워드 : 절단, 저장소 적재, 휴리스틱 방식, 메타휴리스틱 방식, 타부서치

A Heuristic Algorithm for the Two-Dimensional Bin Packing Problem Using a Fitness Function

Yong Ho Yon[†] · Sun Young Lee[††] · Jong Yun Lee[†††]

ABSTRACT

The two-dimensional bin packing problem(2D-BPP) has been known to be NP-hard, and it is difficult to solve the problem exactly. Many approximation methods, such as genetic algorithm, simulated annealing and tabu search etc, have been also proposed to gain better solutions. However, the existing approximation algorithms, such as branch-and-bound and tabu search, have shown the low efficiency and the long execution time due to a large of iterations. To solve these problems, we first define the fitness function to simplify and increase the utility of algorithm. The function decides whether an item is packed into a given area, and as an important information for a packing strategy, the number of subarea that can accommodate a given item is obtained from the variant of the fitness function. Then we present a heuristic algorithm *BP* for 2D bin packing, constructed by the fitness function and subarea. Finally, the effectiveness of the proposed algorithm will be expressed by the comparison experiments with the heuristic and the metaheuristic of the literatures. As comparing with existing heuristic algorithms and metaheuristic algorithms, it has been found that the packing rate of algorithm BP is the same as 97% as existing heuristic algorithms, *FFF* and *FBS*, or better than them. Also, it has been shown the same as 86% as *tabu* search algorithm or better.

Keywords : Cutting, Bin Packing, Heuristic Algorithm, Metaheuristic Algorithm, Rabu Search

1. 서 론

저장소 적재(bin packing) 문제는 오랜 기간 동안 적재/절단(packing/cutting) 문제로 연구되어 왔으며 컨테이너나 운송수단의 물품 적재, 유리나 금속판 등의 물품 절단 문제로부터 스케줄 문제에 이르기 까지 폭넓은 응용 분야에서 사용되어 왔다. 특히 절단 문제나 운송문제에 있어 원재료나 운송장비의 효율적인 사용은 비용절감에 중요한 영향을 미치며 이로 인해 저장소 적재 문제의 좋은 해결 방법이 더욱 필요하게 되었다. 이러한 폭넓은 산업현장에서의 응용으로 지난 수십 년 동안 여러 가지 형태의 개체를 자르거나 적재하는 적재/절단 문제에 있어 많은 발전을 가져왔다. 2차원

※ 이 논문은 2007년도 지식경제부 성장동력기술 개발사업의 일환으로 (주)코리아컴퓨터의 위탁과제로 수행되었음.
† 정 회 원 : 목원대학교 공학교육혁신센터 전임강사
†† 준 회 원 : 충북대학교 컴퓨터교육과 박사과정
††† 종신회원 : 충북대학교 컴퓨터교육과 교수(교신저자)
논문접수 : 2009년 4월 27일
수 정 일 : 1차 2009년 8월 6일
심사완료 : 2009년 8월 11일

저장소 적재 문제(two- dimensional bin packing problem, 2D-BPP)에서 각각 w_j, h_j를 너비와 높이로 갖는 n개의 직사각 개체(item)와 너비와 높이가 W, H로 일정한 무한수의 저장소(bin)가 주어진다. 이 문제의 목적은 모든 개체를 겹치지 않도록 저장소에 적재하되 저장소의 수가 최소가 되도록 적재하는 방법을 찾는 것이다.

2D-BPP의 연구는 Gilmore와 Gomory[12]에 의해 처음으로 소개되었으며 지금까지 많은 적재 기법들이 소개되어 왔다. 2차원 저장소 적재문제는 강한 NP-hard 문제로 이 문제들의 정확한 해를 구하는 것은 어렵다. 따라서 기존의 많은 연구들은 더 좋은 해를 얻기 위해 휴리스틱(heuristic) 또는 메타휴리스틱(meta-heuristic)등과 같은 근사 알고리즘들[4, 6, 11, 15-17]을 제안한 바 있다. 하지만 분지한계나 타부서치 기법들을 이용한 기존의 대표적인 근사 알고리즘들은 휴리스틱 알고리즘의 해에 기반을 둠으로 효율성이 낮고 반복수행에 의한 계산시간이 길다.

따라서 본 논문은 휴리스틱 알고리즘과 타부서치 등과 같은 근사 알고리즘의 복잡함을 간소화하고, 효율성을 높이기 위해 적재가능성을 판단할 수 있는 적합성 함수(fitness function) 를 정의하고 이를 이용하여 개체를 적재하는 새로운 휴리스틱 알고리즘 *BP*(Bin Packing)를 제안한다. 알고리즘 *BP*는 1단계 알고리즘이고 회전가능하지 않은 개체에 적용되며 실험을 통하여 기존의 휴리스틱 알고리즘인 *FFF*와 *FBS*([4])에 비해 97%의 결과가 같거나 우수하였으며, 타부서치 알고리즘 [16]에 비해 86%의 결과가 같거나 우수함을 입증하였다.

본 논문의 구성은 다음과 같다. 제 3장에서 저장소의 부분집합인 부분영역 s를 정의하고, 개체 j가 부분영역 s에 적재가능한지의 여부를 판단할 수 있는 적합성 함수 $\phi(s,j)$를 정의한다. 4장에서 회전가능하지 않은 개체를 적재하는 알고리즘으로 주어진 저장소에 개체를 적재하는 알고리즘 *BP*를 소개한다. 5장은 본 논문에서 제안한 알고리즘을 다양한 예로 실험하고 기존의 휴리스틱 또는 메타휴리스틱 기법에서 제안한 알고리즘과 비교 분석한다. 끝으로 6장은 논문의 결론을 요약한다.

2. 관련연구

적재/절단 문제에서 자주 고려되는 두 가지의 제약 사항으로 (1) 개체를 90°만큼 회전 할 수 있는 지의 여부에 따른 지향성(orientation), (2) 개체가 저장소의 모서리와 평행하게 자르는 방법으로 얻어 질 수 있는지의 여부에 따른 평행절단(guillotine cut)이 있다. Lodi, Martello 와 Vigo[17]는 이 두 가지의 제약 사항에 따라 문제의 유형을 다음과 같이 분류할 수 있다.

(i) 2BP|O|G: 개체를 회전할 수 없고(O), 평행절단을 사용(G);
(ii) 2BP|R|G: 개체를 회전할 수 있고(R), 평행절단을 사용(G);
(iii) 2BP|O|F: 개체를 회전할 수 없고(O), 평행절단을 사용하지 않음(F);
(iv) 2BP|R|F: 개체를 회전할 수 있고(R), 평행절단을 사용하지 않음(F).

회전할 수 없는 개체를 위한 근사 알고리즘은 [6], [4]와 [16]에서 연구된 바 있으며, 회전 가능한 개체를 위한 근사 알고리즘들은 [1], [11], [17]와 [15]에서 연구되었다. 정확한(exact) 알고리즘에 대한 연구가 Martello, Vigo와 Dell'Amico([8], [19], [21])에 의해 이루어졌으며 회전되지 않는 개체에 대한 정확한 알고리즘은 [19]과 [21]에서 제안되었다. 이 알고리즘들은 분지한계 기법을 사용하였고, 주어진 모든 개체의 개체 저장을 위해 필요한 저장소의 하한(lower bound)을 분석하였다. Martello와 Vigo([21])는 연속하한(continuous lower bound)

$$L_0 = \left\lceil (\sum_{j=1}^{n} w_j h_j) / WH \right\rceil$$

에 대한 최악인 경우의 실행율(worst-case performance ratio)을 결정하였고, [7]과 [20]에서 제안한 1차원 적재 문제의 하한을 일반화하여 2차원 문제의 하한 L_1과 이를 개선한 하한으로 L_2, L_3, L_4을 유도하였다. 회전 가능한 개체에 대한 하한은 Dell' Amico, Martello와 Vigo([8])에 의해 소개되었다. Lodi, Martello와 Vigo([18])는 적재 알고리즘을 적재 기법에 따라 다음과 같이 두 가지로 분류하였다.

(i) 1단계(one-phase) 알고리즘: 크기가 유한인 저장소에 개체를 직접 적재하는 방법
(ii) 2단계(two-phase) 알고리즘: 너비가 W이고, 높이가 무한인 스트립(strip)에 개체를 적재하고, 여기에서 얻어진 가상개체(pseudo-item)들을 크기가 유한인 저장소에 적재하는 방법

대부분의 적재 기법은 행으로 레벨(level)을 형성하며 "left-most downward strategy"([13])로 개체를 적재한다. 일반적으로 2단계 알고리즘은 두 단계로 구성되며 1단계에서는 왼쪽에서 오른쪽으로 레벨을 형성해 가며 개체를 적재하고, 각 레벨을 2단계에서 필요한 가상개체로 생성한다. 2단계에는 1단계에서 생성한 가상개체들을 개체로 하여 저장소에 적재한다. 2단계 알고리즘은 Chung, Garey와 Johnson([6])에 의해 처음으로 제안되었으며 Berkey와 Wang([4])에 의해 효과적인 휴리스틱으로 발전하였다. 이들이 제안한 두 가지 알고리즘 *FFF*와 *FBS*는 기존의 논문에서 자주 언급되고 있으며, 이 두 가지 휴리스틱 알고리즘의 시간복잡도는 각각 $O(n\log n)$(*FBS*)과 $O(n^2)$(*FFF*)으로 알려져 있다([17]). 위의 두 알고리즘은 모두 개체를 높이에 따른 내림차순으로 정렬하고 적재방법은 다음과 같다.

(i) Finite Best Strip(*FBS*): 1단계에서 현재 개체를 적재했을 때 나머지 수평 공간이 최소화되는 레벨에 이 개체를 적재한다. 존재하는 레벨에 현재의 개체를 적재할 수 없다면 새로운 레벨을 초기화한다. 2단계에서는

1단계에서 만들어진 레벨을 저장소에 적재한다. 이 때, 현재의 레벨(나머지 중 높이가 가장 큰 것)을 저장소에 적재했을 때 나머지 수직 공간이 최소인 저장소에 적재한다. 현재의 레벨을 적재할 수 있는 저장소가 없는 경우 새로운 저장소에 적재한다.

(ii) Finite First Fit(FFF): 현재의 개체를 적재할 수 있는 첫 번째 저장소에 적재한다. 이때, 저장소에 레벨 방식으로 적재하며 현 개체를 적재할 수 있는 첫 번째 레벨에 적재한다. 적재할 수 있는 레벨이 없는 경우 새로운 레벨을 초기화한다. 개체를 적재할 수 있는 저장소가 없는 경우 새로운 저장소에 적재한다.

적재/절단 문제의 유형 분류와 일반적인 조사는 [9], [10], [14], [17]과 [18]등에서 찾아볼 수 있으며 Dyckhoff([9])는 문제의 차원과 적재방법 그리고 저장소의 모양과 개체의 모양에 따라 문제의 유형을 분류하였다. 본 논문에서 다루는 문제는 [9]에서의 분류에 따르면 2/V/I/M이고, [17]에서의 분류에 따르면 2BP|O|F에 속한다.

3. 부분영역과 적합성 함수의 정의

너비가 W이고 높이가 H인 저장소와 n 개의 개체로 구성된 집합 $X=\{1,2,\cdots,n\}$ 가 주어져 있다고 가정하자. 각각의 개체 i ($i=1,2,\cdots,n$)는 너비 w_i ($w_i \le W$)와 높이 h_i ($h_i \le H$)를 갖는 2차원 직사각 개체로 2차원 벡터 $i=(w_i,h_i)$로 나타낸다. 주어진 저장소에 개체를 적재할 때, 저장소를 하나의 직교좌표계로 생각할 수 있다. 이 때, 저장소의 왼쪽 아래 꼭지점을 원점 $O(0,0)$, 아래 가로모서리를 x-축, 왼쪽 세로모서리를 y-축으로 보고, 각 개체의 왼쪽아래 꼭짓점을 저장소의 한 좌표 (x,y)에 위치시킨다.

임의의 실수 x, y, l ($0 \le x < W$, $0 \le y < H$, $0 < l < H-y$)에 대하여 부분영역을 다음의 부분집합 식(1)로 정의하고, $s=(x,y;l)$로 나타낸다(그림1).

$$\{(u,v) \mid x \le u \le W \text{ and } y \le v \le y+l\} \qquad (1)$$

여기에서 (x,y) 는 저장소에 위치한 부분영역의 왼쪽아래 꼭짓점의 좌표이고 l 은 부분영역의 높이이다.

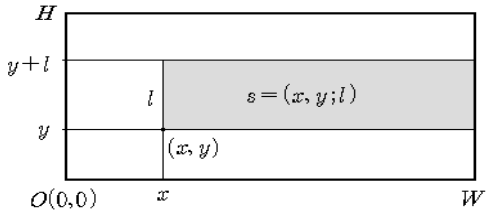
저장소에 개체를 적재할 때 부분영역의 집합 S를 $S=\{(0,0;H)\}$로 초기화 하고, 각각의 개체를 적재하는 반복과정에서 개체 $j=(w,h)$ 를 S 안에서 x 값이 가장 작은 부분영역 $s=(x,y;l)$에 적재한다. 개체 j가 s 에 적재될 수 있다면 개체의 왼쪽아래 꼭짓점을 부분영역의 좌표 (x,y)에 위치시키고 부분영역 $s=(x,y;l)$는 S에서 제거되고 다음의 새로운 부분영역이 생성된다.

$h < l$ 이면 $(x, y+h; l-h)$ 와 $(x+w, y; h)$(그림 2)(a)
$h = l$ 이면 $(x+w, y; l)$ (그림 2)(b)

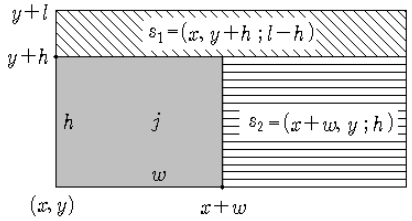
남아있는 어떤 개체도 부분영역 $s=(x,y;l)$에 적재될 수 없다면 더 넓은 적재공간을 확보하기 위해 이를 다른 부분영역과 병합한다(그림 3).

적재하는 저장소에 더 이상의 개체를 적재할 수 없다면 새로운 저장소를 다시 부분영역으로 나누어 가며 개체를 적재하고, 모든 개체가 적재될 때 까지 같은 과정을 반복한다.

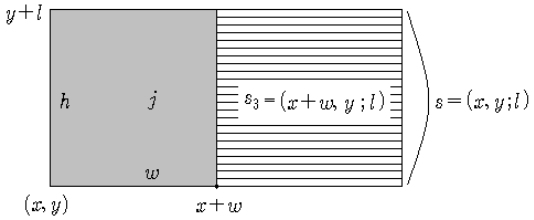
개체 $j=(w,h)$ 가 저장소안의 한 부분영역 $s=(x,y;l)$에 적재될 수 있는지를 판단하려면 두 가지 조건 $h \le l$ 와 $x+w \le W$를 조사하여야 한다. 또한 한 부분영역에 얼마나 많은 개체가 적재될 수 있는 지를 판단하는 것은 개체를 효율적으로 적재할 수 있는 정보를 제공한다. 이러한 과정을 좀 더 용이하게 처리하기 위해 한 개체가 주어진 부분영역에 적재될 수 있는지를 판단할 수 있는 함수를 다음과 같이 정의한다.



(그림 1) 저장소내의 부분영역 s

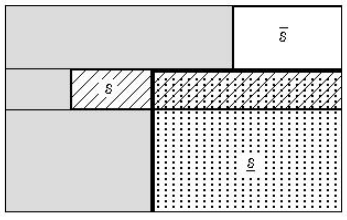


(a) $h < l$ 일 때 생성된 부분영역 s_1, s_2



(b) $h = l$ 일 때 생성된 부분영역 s_3

(그림 2) 부분영역 s 에 $h \le l$ 인 개체가 적재될 때, 새롭게 생성된 부분영역



(그림 3) 부분영역 s 에 적재 가능한 개체가 없을 때, s 와 $\underline{s}$ 를 병합

[정의] 개체의 집합 $J=\{1,2,\cdots,n\}$ 와 부분영역의 집합 S에 대하여 적합성(fitness) 함수 $\phi : S\times J \rightarrow \{0,1\}$를 다음과 같이 정의한다.

$$\phi(s,j)=\left\lfloor \frac{W-x-w}{M}+1 \right\rfloor \times \left\lfloor \frac{l-h}{M}+1 \right\rfloor$$

여기에서 $s=(x,y;l)\in S$, $j=(w,h)\in J$이고, 저장소의 너비 W와 높이가 H에 대하여$M=\max\{W,H\}$이다. □

임의의 $s=(x,y;l)\in S$, $j=(w,h)\in J$에 대하여 $0<l$, $h\le M=\max\{W,H\}$, $0<w\le W\le M$이고, $0<x\le W$이므로 $0\le W-x\le W\le M$이다. 이들로부터 다음의 식 (2)을 얻을 수 있다.

$$-M<l-h<M,\ -M\le W-x-w\le M \qquad (2)$$

$\phi_1=\frac{W-x-w}{M}+1$, $\phi_2=\frac{l-h}{M}+1$ 이라 하면 $0<\phi_1$, $\phi_2<2$이다. 특히, ϕ_1 과 ϕ_2 는 다음의 성질을 갖는다.

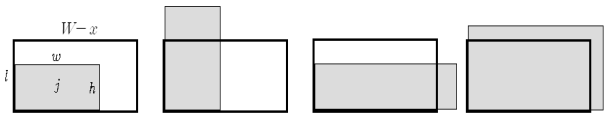
$w\le W-x$ 이면 $1\le\phi_1<2$, 그렇지 않으면 $0\le\phi_1<1$
$h\le l$ 이면 $1\le\phi_2<2$, 그렇지 않으면 $0<\phi_2<1$

따라서 ϕ_1 과 ϕ_2 는 식 (3)을 만족하며, 이로부터 식 (4)를 얻을 수 있다.

$$\lfloor\phi_1\rfloor=\begin{cases}1, & \text{if } w\le W-x\\ 0, & \text{if } w>W-x\end{cases},\quad \lfloor\phi_2\rfloor=\begin{cases}1, & \text{if } h\le l\\ 0, & \text{if } h>l\end{cases} \qquad (3)$$

$$\phi(s,j)=\lfloor\phi_1\rfloor\times\lfloor\phi_2\rfloor=\begin{cases}1, & w\le W-x \text{ and } h\le l\\ 0, & otherwise\end{cases} \qquad (4)$$

즉, (그림 4)(a)와 같이 $j=(w,h)$ 가 $s=(x,y;l)$ 에 적재될 수 있으면 $\phi(s,j)=1$ 이고, (그림 4)(b)-(그림 4)(d)의 경우에 $\phi(s,j)=0$이다.



(a) (b) (c) (d)

(그림 4) 개체 $j=(w,h)$(회색영역)와 부분영역 $s=(x,y;l)$ (굵은 선의 직사각형);
(a)$h\le l$, $w\le W-x$, (b)$h>l$, $w\le W-x$,
(c)$h\le l$, $w>W-x$, (d)$h>l$, $w>W-x$.

[정리] 저장소의 너비와 높이가 각각 W, H이고, 개체의 집합과 부분영역의 집합이 각각 $J=\{1,2,\cdots,n\}$ 와 S라 하자. 부분영역 $s=(x,y;l)\in S$에 개체 $j=(w,h)\in J$가 적재될 수 있다면 $\phi(s,j)=1$이고, 적재될 수 없다면 $\phi(s,j)=0$이다. □

4. 회전 가능하지 않은 개체를 위한 휴리스틱 알고리즘

이 장에서는 회전가능하지 않은 개체를 위한 알고리즘 BP를 소개한다. 먼저 (그림 5)와 같이 크기가 10×8인 저장소와 두 개의 부분영역 $s_1=(2,5;3)$, $s_2=(5,0;5)$, 그리고 두 개의 개체 $j_1=(5,3)$, $j_2=(8,3)$ 을 갖는 예를 생각하자. 개체 j_1이 부분영역 s_1에 적재되면 개체 j_2가 적재될 여유 공간이 없어 2개의 저장소가 필요하게 되지만 개체 j_1이 부분영역 s_2, 개체 j_2가 부분영역 s_1에 적재되면 1개의 저장소만으로 모든 개체를 적재할 수 있다. 여기에서 개체 j_1이 적재될 수 있는 부분영역의 수는 2개이고 j_2가 적재될 수 있는 부분영역의 수는 1개로 개체 j_2는 부분영역 s_2에 적재될 수밖에 없다. 이와 같이 어떤 개체를 적재할 수 있는 부분영역의 수는 개체를 적재하는데 중요한 정보를 제공한다.

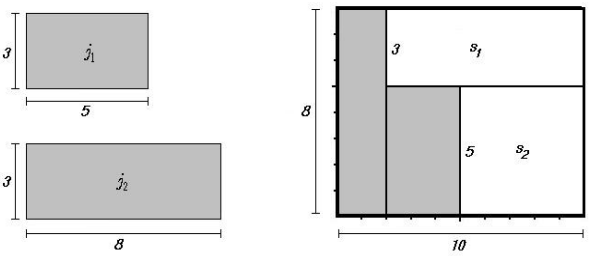
주어진 개체 j와 부분영역의 집합 S에 대하여 j를 적재할 수 있는 S안의 부분영역들의 집합을 S_j라 할 때, 이 집합의 원소의 수 $|S_j|$는 적합성 함수를 이용하여 다음과 같이 구할 수 있다.

$$|S_j|=\sum_{s\in S}\phi(s,j)$$

또한 개체 집합 J안의 원소 중 주어진 부분영역 $s=(x,y;l)$ 에 적재될 수 있는 개체들의 집합을 J_s라 할 때, 이 집합의 원소의 수 $|J_s|$를 다음과 같이 구할 수 있다.

$$|J_s|=\sum_{j\in J}\phi(s,j)$$

다음의 (그림 6)은 본 논문에서 제안한 BP알고리즘이다. 알고리즘 BP는 1단계 알고리즘으로 크기가 $W\times H$인 저장소에 개체를 직접 적재하는 알고리즘이다. 부분영역의 집합을 $S=\{(0,0;H)\}$로 초기화하고, 한 부분영역 s에 개체가 적재될 때, 가장 작은 $|S_j|$값을 갖는 개체 j가 선택되도록 설계하였다. 또한 최소의 $|S_j|$값을 갖는 개체가 많을 경우에



(그림 5) 크기가 10×8인 저장소에 두 개의 부분영역 $s_1=(2,5;3)$, $s_2=(5,0;5)$와 두 개 개체 $j_1=(5,3)$, $j_2=(8,3)$ 을 갖는 예

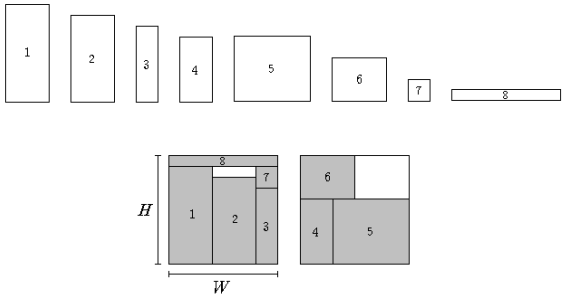
Algorithm BP : $W\times H$인 저장소에 개체를 직접 적재

```
input  W, H, J={1,2,…,n}
output 개체의 저장위치, 저장소의 수
0  begin;
1  sort items by non-increasing height;
2  while J≠∅ do
3     S:={(0,0;H)};
4     while S≠∅ do
5        select the subarea (x, y; l) in S which has the smallest x-value;
6        if there are many subareas which have the smallest x-value then
7           select the subarea with the smallest y-value;
8        endif;
9        if |J_s|≠0 then
10          select the item k=(w_k,h_k) in J_s that has the smallest |S_k| value;
11          if there are many items then
12             select one that has the largest area;
13          endif;
14          place the item k at the coordinate (x,y);
15          J=J-{k};
16          if h_k=l then
17             S:=(S-{(x, y; l)})∪{(x+w_k, y; l)};
18          else
19             S:=(S-{(x, y; l)})∪{(x, y+h_k; l-h_k),(x+w_k, y; h_k)};
20          endif;
21       else   //|J_s|=0
22          if S-{(x,y; l)}=∅ then
23             S:=∅;
24          else
25             find subareas (x̄, ȳ; l̄) and (x̲, y̲; l̲) in S-{(x, y; l)} satisfying
26                y+l=ȳ and y=y̲+l̲;
27             if x̄ < x̲ then
28                S=(S-{(x, y; l),(x̄, ȳ; l̄)})∪{(x̄, y; l+l̄)};
29             endif;
30             if x̄ > x̲ then
31                S=(S-{(x, y; l), (x̲, y̲; l̲)})∪{(x̲, y̲; l+l̲)};
32             endif;
33             if x̄ = x̲ then
34                S=(S-{(x, y; l), (x̲, y̲; l̲), (x̄, ȳ; l̄)})∪{(x̲, y̲; l+l̲+l̄)};
35             endif;
36          endif;
37       endif;
38    endwhile;
39 endwhile;
40 end;
```

(그림 6) BP 적재 알고리즘

면적이 가장 큰 개체를 선택하여 적재한다. 알고리즘의 단계 5-7은 현존하는 부분영역 중 x 값이 가장 작은 부분영역을 선택하고, 그와 같은 부분영역이 많은 경우에 y 값이 가장 작은 부분영역을 선택한다. 단계 9-20은 선택된 부분영역 s 에 적재될 수 있는 개체가 있는 경우에 실행된다. 단계 10-13은 가장 작은 $|S_k|$값을 갖는 개체를 선택하며 그러한 개체가 많은 경우에 면적이 가장 큰 개체를 선택한다. 단계 16-20은 개체 $j=(w, h)$ 를 S 안의 선택된 부분영역 $s=(x,y;l)$ 에 적재할 수 있을 때, 실행되어 $h=l$ 이면 부분영역 $(x+w, y; l)$, $h<l$ 이면 부분영역 $(x, y+h; l-h)$ 와 $(x+w, y; h)$를 생성하여 부분영역의 집합 S에 추가하는 과정이다. 단계 21-37은 선택된 부분영역 s 에 적재될 수 있는 개체가 없는 경우에 실행되고, 단계 22-23은 집합 S 안에 더 이상 개체를 적재할 수 있는 부분영역이 없는 경우로 현재 while문을 벗어나는 조건이 설정된다. 단계 24-36은 집합 S 안에 현재의 부분영역 s 에 적재될 수 있는 개체가 없고 s 이외의 부분영역이 있을 때, 적재 공간을 효율적으로 사용하기 위해 현재의 부분영역을 위쪽이나 아래쪽에 있는 다른 부분영역과 병합하는 과정이다.

(그림 7)은 적재 알고리즘 BP을 이용하여 크기가 $W\times H$인 저장소에 6개의 개체를 적재한 예이다. 알고리즘 BP의 반복과정에서 최대의 시간 복잡도를 갖는 예는 개체수가 n 이고 모든 개체의 높이가 H/n 보다 작을 때로 이 때 k 번째 개체가 적재될 때 나타나는 부분영역의 최대 수는 $n-k$로 단계 10에서 $|S_k|$를 계산하여 비교하는 $(k+1)(n-k)$



(그림 7) 알고리즘 BP로 8개의 개체를 크기 $W\times H$인 저장소에 적재한 예

번의 연산이 발생하여 시간 복잡도는 $n(n+1)(2n+1)/6$ 으로 $O(n^3)$ 이다.

5. 실험 결과

본 논문에서 소개한 BP 알고리즘은 java6.0으로 프로그래밍 되었고, PC pentium 4 2.4GHz Memory RAM 1 GB에서 실험되었다.

실험결과의 비교 대상은 휴리스틱 알고리즘 *FFF*, *FBS* ([4])와 이웃검색(neighborhood search)기법을 사용한 타부서치 *TS* ([16]), 그리고 floor-ceiling기법의 휴리스틱 알고리즘 FC_{OG}, FC_{OF} ([17])와 1차원 배낭문제(knapsack problem)의 해법을 적용한 알고리즘 KP_{OG}([17]), Lodi, Martello와 Vigo([17])에 의해 새롭게 제안된 양방향(alternate directions) 알고리즘 AD_{OF}, 그리고 알고리즘 KP_{OG}, AD_{OF}의 해를 초기해로 이용한 타부서치 알고리즘이다. 대부분의 타부 서치 알고리즘은 두 단계로 나뉘며 1단계에서 휴리스틱 알고리즘 *FC*, *KP* 또는 *AD* 등을 이용하여 그 초기해를 구하고 2단계에서 그 초기해와 채우기 함수를 이용하여 최악의 저장소를 선택하고 재 적재하는 과정을 필요한 만 큼 반복한다. 이 때 1단계에서 사용되는 휴리스틱알고리즘 *FC*와 *AD* 의 시간 복잡도는 모두 $O(n^3)$ 으로 알려져 있다([17-18]).

<표1>은 관련연구의 문헌상에서 얻어진 120개의 예로부터 얻어진 결과로 beng1~beng8은 Bengtsson([1]), cgcut1~cgcut3은 Christofides와 Whitlock([5]), gcut1~gcut13과 ngcut1~ngcut12는 Beasley([2], [3])에 의해 제안된 예이다. 제 2열의 n 은 각 예의 개체 수이고, 3열의 LB는 저장소 수의 하한(lower bound) L_4 ([21])이다. 휴리스틱 또는 메타휴리스틱 알고리즘으로 얻은 해를 SOL 이라 할 때, LB 는 하한이므로 항상 $SOL \geq LB$ 이다. 4열과 5열은 알고리즘 *FFF* 와 *FBS* ([4])에 의해 얻어진 해이고, 6열은 타부서치 알고리즘 *TS* ([16])로부터 얻어진 해이다. 7열의 *BP*는 본 논문의 알고리즘에 의해 얻어진 해이다. 마지막 열의 * 표시는 알고리즘 *BP* 와 *TS* 의 비교에서 $BP \leq TS$ 인 경우이다. 표1의 실험결과로 휴리스틱 알고리즘인 *FFF* 와 *FBS* 에 비해 97%의 결과가 같거나 우수하였으며, 타부서치 알고리즘 *TS* 에 비해 86%의 결과가 같거나 우수한 것으로 나타났다.

<표2>에서는 무작위로 생성된 10개 클래스의 예로 첫 번째 6개의 클래스는 Berkey와 Wang([4])에 의해 제안된 크기의 예로 다음과 같은 크기를 갖는다.

클래스 I: w_j 와 h_j 를 구간 [1,10] 에서 uniformly random하게 선택, $W = H = 10$;
클래스 II: w_j 와 h_j 를 구간 [1,10] 에서 uniformly random하게 선택, $W = H = 30$;
클래스 III: w_j 와 h_j 를 구간 [1,35] 에서 uniformly random하게 선택, $W = H = 40$;

〈표 1〉 주어진 예에 대한 알고리즘 *FFF*, *FBS*, *TS*, *BP*의 실험 결과

Name	n	*LB*	*FFF*	*FBS*	*TS*	*BP*	
beng1	20	4	4	4	4	4	*
beng2	40	6	7	7	7	7	*
beng3	60	9	10	9	9	10	
beng4	80	11	12	12	12	12	*
beng5	100	14	16	15	14	15	
beng6	40	2	2	2	2	2	*
beng7	80	3	3	3	3	3	*
beng8	120	5	5	5	5	5	*
cgcut1	16	2	2	2	2	2	*
cgcut2	23	2	3	3	2	2	*
cgcut3	62	23	26	26	23	24	
gcut1	10	4	5	5	5	5	*
gcut2	20	6	7	7	6	7	
gcut3	30	8	9	8	8	8	*
gcut4	50	13	15	15	14	14	*
gcut5	10	3	4	4	4	3	*
gcut6	20	6	8	8	7	7	*
gcut7	30	10	12	12	12	11	*
gcut8	50	12	15	14	14	14	
gcut9	10	3	3	3	3	3	*
gcut10	20	7	8	8	8	8	*
gcut11	30	8	10	10	9	9	*
gcut12	50	16	17	17	16	16	*
gcut13	32	2	2	2	2	2	*
ngcut1	10	2	3	3	3	3	*
ngcut2	17	3	4	4	4	4	*
ngcut3	21	3	4	4	4	4	*
ngcut4	7	2	2	2	2	2	*
ngcut5	14	3	4	4	3	3	*
ngcut6	15	2	3	3	3	3	*
ngcut7	8	1	2	2	1	1	*
ngcut8	13	2	2	2	2	2	*
ngcut9	18	3	4	4	4	4	*
ngcut10	13	3	3	3	3	3	*
ngcut11	15	2	3	3	3	3	*
ngcut12	22	3	4	4	4	4	*

클래스 IV: w_j 와 h_j 를 구간 [1,35] 에서 uniformly random하게 선택, $W = H = 100$;
클래스 V: w_j 와 h_j 를 구간 [1,100] 에서 uniformly random하게 선택, $W = H = 100$;
클래스 VI: w_j 와 h_j 를 구간 [1,100] 에서 uniformly random하게 선택, $W = H = 300$;

나머지 클래스 7-10은 Martello와 vigo([21])에 의해 제안된 예로 다음과 같은 타입으로 구성되었다.

타입 1: w_j 는 구간 $\left[\frac{2}{3}W, W\right]$, h_j 는 $\left[1, \frac{1}{2}H\right]$ 에서 uniformly random하게 선택;
타입 2: w_j 는 구간 $\left[1, \frac{1}{2}W\right]$, h_j 는 $\left[\frac{2}{3}H, H\right]$ 에서 uniformly random하게 선택;
타입 3: w_j 는 구간 $\left[\frac{1}{2}W, W\right]$, h_j 는 $\left[\frac{1}{2}H, H\right]$ 에서 uniformly random하게 선택;

〈표 2〉 Martello와 Vigo에 의해 제안된 크기의 무작위 추출된 예에 대한 휴리스틱 알고리즘들의 실험 결과

Class	n	$\frac{FFF_{OG}}{LB}$	$\frac{FBS_{OG}}{LB}$	$\frac{FC_{OG}}{LB}$	$\frac{KP_{OG}}{LB}$	$\frac{FC_{OF}}{LB}$	$\frac{AD_{OF}}{LB}$	$\frac{TKP_{OG}}{LB}$	$\frac{TAD_{OF}}{LB}$	$\frac{BP}{LB}$
I	20	1.17	1.14	1.14	1.13	1.12	1.12	1.11	1.06	1.08
	40	1.12	1.09	1.09	1.10	1.08	1.09	1.08	1.06	1.08
	60	1.10	1.07	1.07	1.07	1.07	1.07	1.05	1.04	1.05
	80	1.08	1.06	1.06	1.06	1.06	1.06	1.04	1.05	1.04
	100	1.07	1.06	1.06	1.05	1.06	1.05	1.05	1.04	1.04
Average		1.108	1.084	1.084	1.082	1.078	1.078	1.066	1.050	1.060
II	20	1.10	1.10	1.10	1.00	1.10	1.00	1.00	1.00	1.00
	40	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
	60	1.15	1.15	1.15	1.15	1.10	1.10	1.15	1.10	1.10
	80	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07
	100	1.06	1.06	1.03	1.03	1.03	1.03	1.03	1.03	1.03
Average		1.096	1.096	1.090	1.070	1.080	1.060	1.070	1.060	1.060
III	20	1.20	1.18	1.18	1.18	1.18	1.20	1.18	1.20	1.18
	40	1.18	1.14	1.14	1.15	1.14	1.15	1.12	1.11	1.14
	60	1.14	1.11	1.11	1.12	1.11	1.13	1.07	1.05	1.12
	80	1.13	1.10	1.10	1.10	1.10	1.10	1.08	1.08	1.08
	100	1.12	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09
Average		1.154	1.124	1.124	1.128	1.124	1.134	1.108	1.106	1.120
IV	20	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	40	1.10	1.10	1.00	1.10	1.00	1.00	1.10	1.00	1.00
	60	1.20	1.20	1.10	1.20	1.10	1.15	1.20	1.15	1.15
	80	1.10	1.10	1.10	1.13	1.10	1.10	1.10	1.10	1.10
	100	1.10	1.10	1.10	1.10	1.10	1.03	1.10	1.03	1.07
Average		1.100	1.100	1.060	1.106	1.060	1.056	1.100	1.056	1.060
V	20	1.14	1.14	1.14	1.13	1.14	1.14	1.13	1.11	1.11
	40	1.11	1.11	1.11	1.09	1.11	1.11	1.09	1.04	1.09
	60	1.11	1.10	1.10	1.10	1.10	1.10	1.07	1.06	1.08
	80	1.12	1.09	1.09	1.09	1.09	1.09	1.08	1.06	1.07
	100	1.12	1.09	1.09	1.09	1.09	1.09	1.09	1.08	1.08
Average		1.120	1.106	1.106	1.100	1.106	1.106	1.092	1.070	1.090
VI	20	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	40	1.40	1.40	1.40	1.50	1.40	1.40	1.50	1.40	1.40
	60	1.10	1.10	1.10	1.10	1.10	1.05	1.10	1.05	1.05
	80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	100	1.13	1.10	1.10	1.10	1.10	1.07	1.10	1.07	1.10
Average		1.126	1.120	1.120	1.140	1.120	1.104	1.140	1.104	1.110
VII	20	1.10	1.10	1.10	1.10	1.08	1.10	1.08	1.04	1.14
	40	1.11	1.11	1.11	1.07	1.09	1.10	1.07	1.06	1.07
	60	1.08	1.08	1.08	1.06	1.07	1.07	1.05	1.05	1.06
	80	1.07	1.06	1.06	1.06	1.06	1.06	1.05	1.04	1.05
	100	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.03	1.04
Average		1.080	1.078	1.078	1.066	1.068	1.074	1.058	1.044	1.070
VIII	20	1.17	1.16	1.16	1.12	1.16	1.13	1.12	1.06	1.08
	40	1.09	1.08	1.08	1.07	1.07	1.08	1.04	1.03	1.06
	60	1.06	1.06	1.06	1.06	1.06	1.06	1.03	1.02	1.04
	80	1.07	1.06	1.06	1.05	1.06	1.06	1.03	1.02	1.03
	100	1.06	1.06	1.06	1.04	1.06	1.06	1.04	1.04	1.04
Average		1.090	1.084	1.084	1.068	1.082	1.078	1.052	1.034	1.050
IX	20	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.00	1.01
	40	1.02	1.02	1.02	1.02	1.02	1.02	1.01	1.01	1.02
	60	1.02	1.02	1.02	1.01	1.02	1.02	1.01	1.01	1.01
	80	1.02	1.02	1.02	1.02	1.02	1.02	1.01	1.01	1.02
	100	1.02	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
Average		1.018	1.016	1.016	1.014	1.016	1.016	1.008	1.008	1.010
X	20	1.14	1.14	1.14	1.16	1.14	1.10	1.14	1.10	1.12
	40	1.14	1.09	1.09	1.10	1.09	1.09	1.09	1.06	1.09
	60	1.15	1.12	1.10	1.10	1.08	1.11	1.08	1.07	1.10
	80	1.15	1.13	1.12	1.12	1.11	1.10	1.10	1.06	1.09
	100	1.14	1.10	1.10	1.08	1.09	1.10	1.07	1.08	1.08
Average		1.144	1.116	1.110	1.112	1.102	1.100	1.096	1.074	1.100

타입 4: w_j 는 구간 $\left[1, \frac{1}{2}W\right]$, h_j 는 $\left[1, \frac{1}{2}H\right]$ 에서 uniformly random하게 선택;

각 클래스에서 저장소의 크기는 $W = H = 100$ 으로 고정하고 타입별로 개체 비율을 다르게 선택하였다.

클래스 VII: 타입 1을 70%, 타입 2, 3, 4를 각각 10%
클래스 VIII: 타입 2를 70%, 타입 1, 3, 4를 각각 10%
클래스 IX: 타입 3을 70%, 타입 1, 2, 4를 각각 10%
클래스 X: 타입 4를 70%, 타입 1, 2, 3을 각각 10%

각 클래스마다 개체 수 n 이 20, 40, 60, 80, 100인 예를 10개씩 생성하고, 표 2에 나타난 값은 각 개체 수에 해당되는 10개의 예에 대한 알고리즘의 해 SOL과 하한의 비율 $\frac{SOL}{LB}$의 평균값으로 그 값이 작을 수 록 좋은 성능을 갖는다고 볼 수 있다. 여기에서 LB는 표1과 같은 하한 L_4 이고 제 2열의 두 값은 각각 FFF와 FBS의 해와 LB에 대한 비율의 평균값이고, 3열과 4열은 알고리즘 FC_{OG}, KP_{OG}, FC_{OF}, AS_{OF}([17])의 결과이고, 5열은 같은 논문 [17]의 타부서치 알고리즘에서 얻어진 해이다. 마지막 열은 본 논문의 알고리즘 BP에 의한 결과로 각 클래스의 평균값을 비교할 때, 모든 예에서 FFF, FBS와 평행절단 문제의 알고리즘인 FC_{OG}, KP_{OG}와 같거나 우수한 결과를 얻었으며, 평행절단을 적용하지 않은 알고리즘인 FC_{OF}, AS_{OF}와 97%가 같거나 우수한 것으로 나타나 전반적으로 기존의 휴리스틱 알고리즘에 비해 우수한 성능을 보였다.

6. 결 론

산업의 발전으로 인하여 많은 물량의 이동과 물품들의 제조량이 증가함에 따라 운송 수단의 절감과 원재료의 절감은 비용절감에 중요한 영향을 줄 수 있다. 이러한 관점에서 적재와 절단 문제의 발전은 그 중요성이 증대되고 있는 반면 그 문제의 난해함으로 인하여 획기적이 발전이 어렵다. 본 논문에서는 적재 패턴을 판단하는 적합성 함수 제안으로 알고리즘을 간소화하였고 계산을 용이하게 처리할 수 있었으며, 기존의 연구결과와 비교하여 좋은 결과를 얻을 수 있었다.

본 논문에서 제안한 휴리스틱 알고리즘 BP는 타부서치 알고리즘에 비해 그 알고리즘이 간결하며 기존의 휴리스틱 알고리즘에 비해 좋은 성능을 보이고 있어 이 알고리즘의 해를 초기해로 적용하는 타부서치 기법은 기존의 근사 알고리즘에 비해 좋은 결과를 얻을 것으로 예상된다. 또한 회전 가능 개체에 대한 2차원 적재문제와 2차원 적재기법을 이용한 3차원 적재 문제에 이 함수를 확장하여 적용하면 원재료의 절단 문제나 컨테이너 또는 팔레트 적재 문제 등 산업현장에 실제로 적용되는 예에서 좋은 결과를 얻을 수 있을 것으로 기대된다.

참 고 문 헌

[1] B. E. Bengtsson, Packing rectangular pieces-a heuristic approach, The Computer Journal Vol.25, No.3, pp.353-357, 1982.

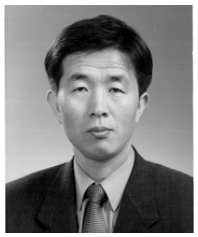
[2] J. E. Beasley, Algorithms for unconstrained two-dimensional guillotine cutting, Journal of Operational Research Society, Vol.36, pp.297-306.
[3] J. E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research, Vol.33, pp.49-64, 1985.
[4] J. O. Berkey and P. Y. Wang, Two dimensional finite bin packing algorithms Journal of the Operational Research Society 38, pp.423-429, 1987.
[5] N. Christofides and C. Whitlock, An algorithm for two-dimensional cutting problrems, Operations Research, Vol.25, pp.30-44, 1977.
[6] F. K. R. Chung, M. R. Garey and D. S. Johnson, On Packing two-dimensional bins, SIAM Journal of Algebraic and Discrete Methods, Vol.3, No.1, pp.66-76, 1982.
[7] M. Dell'Amico and S. Martello, Optimal scheduling of tasks on identical parallel processors, ORSA Journal on Computing, Vol.7, No.2, pp.191-200, 1995.
[8] M. Dell'Amico, S. Martello and D. Vigo, A lower bound for the non-oriented two-dimensional bin packing problem, Discrete Applied Mathematics, Vol.118, pp.13-24, 2002.
[9] H. Dyckhoff, A typology of cutting and packing problems, European Journal of Operational Research, Vol.44, No.2, pp. 145-159, 1990.
[10] H. Dyckhoff and G. Finke, Cutting and packing in production and distribution, Physica Verlag, Heidelberg, 1992.
[11] A. El-Bouri, N. Popplewell, S. Balakrishnan and A. Alfa, A search based heuristic for the two-dimensional bin-packing problem, INFOR, Vol.32, No.4, pp.265-274, 1994.
[12] P. C. Gilmore and R. E. Gomory, Multistage cutting problems of two and more dimensions, Operations Research, Vol.13, No.1, pp.94-120, 1965.
[13] E. Hadjiconstantinou and N. Christofides, An Exact algorithm for general, orthogonal, two-dimensional knapsack problem, European Journal of Operational Research, Vol.83, pp.39-56, 1995.
[14] A. Lodi, S. Martello and M. Monaci, Two-dimensional packing problems: a survey, European Journal of Operational Research, Vol.141, pp.241-252, 2002.
[15] A. Lodi, S. Martello, and D. Vigo, Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem, in: S. Martello, I. H. Osman, C. Roucairol (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, pp.125-139, 1998.
[16] A. Lodi, S. Martello and D. Vigo, Approximation algorithms for the oriented two-dimensional bin packing problem, European Journal of Operational Research, Vol.112, 158-166, 1999.
[17] A. Lodi, S. Martello and D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal of Computing, Vol.11, No.4, pp. 345-357, 1999.
[18] A. Lodi, S. Martello, D. Vigo, Ricent advances on two-dimensional bin packing problems, Discrete Applied Mathematics 123, pp.373-390, 2002.
[19] S. Martello, M. Monaci and D. Vigo, An exact approach to the strip packing problem, Informs Journal on Computing, Vol.15, 310-319, 2003.
[20] S. Martello and P. Toth, Lower bounds and reduction procedures for the bin packing problem, Discrete Applied Mathematics, Vol.28, pp.59-70, 1990.
[21] S. Martello and D. Vigo, Exact solution of the two dimensional finite bin packing problem, Management Science, Vol.44, pp. 388-399, 1998.

연 용 호

e-mail : yhyon@mokwon.ac.kr
1988년 충북대학교 수학과 졸업(이학사)
1990년 충북대학교 수학과 졸업(이학석사)
1997년 충북대학교 수학과 졸업(이학박사)
2006~현 재 목원대학교 공학교육혁신센터 전임강사
관심분야 : Lattice Theory, Fuzzy Theory, RFID 보안

이 선 영

e-mail : elesun@nate.com
2001년 충북대학교 전기공학과 졸업(공학사)
2005년 충북대학교 전기공학과 (공학석사)
2007년 충북대학교 컴퓨터교육과(박사수료)
관심분야 : 데이터베이스, Bioinformatics, Approximate Query Answering, 유비쿼터스 컴퓨팅



이 종 연

e-mail : jongyun@chungbuk.ac.kr
1985년 충북대학교 전자계산기공학과 (공학사)
1987년 충북대학교 전자계산기공학과(공학석사)
1999년 충북대학교 전자계산학과(이학박사)
1989년 비트컴퓨터(주) 개발부
1990년~1994년 현대전자산업(주) 소프트웨어연구소 주임연구원
1994년~1996년 현대정보기술(주) CIM사업부 책임연구원
1999년~2003년 삼척대학교 정보통신공학과 조교수
2003년~현 재 충북대학교 컴퓨터교육과 부교수
관심분야 : 질의 처리 및 최적화, 시공간 데이터베이스, 데이터 마이닝, 국제물류, RFID 보안, u-Learning 및 평가, GIS