Journal of The Korea Society of Computer and Information
Vol. 22 No. 9, pp. 81-89, September 2017
www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.09.081

# An In-depth Analysis and Performance Improvement of a Container Relocation Algorithm

Hyung-Bong Lee*, Ki-Hyeon Kwon**

## Abstract

The CRP(Container Relocation Problem) algorithms pursuing efficient container relocation of wharf container terminal can not be deterministic because of the large number of layout cases. Therefore, the CRP algorithms should adopt trial and error intuition and experimental heuristic techniques. And because the heuristic can not be best for all individual cases, it is necessary to find metrics which show excellent on average. In this study, we analyze GLAH(Greedy Look-ahead Heuristic) algorithm which is one of the recent researches in detail, and propose a heuristic metrics HOB(sum of the height differences between a badly placed container and the containers prohibited by the badly placed container) to improve the algorithm. The experimental results show that the improved algorithm, GLAH', exerts a stable performance increment of up to 3.8% in our test data, and as the layout size grows, the performance increment gap increases.

▸Keyword: Allocation, CPMP, CRP, B&B, GLAH, Heuristic

## I. Introduction

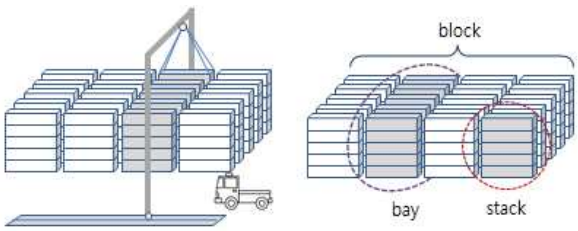


Fig. 1. Overview of General Container Terminal Yard

항만 야적장은 수·출입을 위한 컨테이너로 늘 분빈다. 이 컨테이너들은 보통 최종 목적지와 관계없이 일단 도착순서대로 임시 보관되는데, 이들을 쌓는 형태는 그림 1과같이 컨테이너 이동 장치인 크레인을 따라 블록(block) 단위로 구성되고, 블록은 다시 여러 줄의 베이(bay)로 구성된다. 크레인은 레일 위를 좌우로 움직이며 특정 베이 옆에 정지한 상태에서 컨테이너를 원하는 스택 탑(stack top) 위에 올려놓거나. 탑 컨테이너를 꺼낼 수 있다. 이와 같이 무작위로 쌓여있는 컨테이너 더미에서의 문제는 특정 기준에 따라 컨테이너를 차례로 꺼내야 할 경우 어떤 순서를 따르면 효율적이겠는가 하는 점이다. 즉, 중간에 끼어 있는 컨테이너를 먼저 꺼내야 한다면 그 위의 컨테이너들을 주변의 다른 스택으로 이동시켜야 하는데, 이런 작업이 특정 순서에 따라 연속적으로 필요하다면 이동작업의 최소화가 작업 전체의 효율성을 좌우한다. 이를테면 수출을 위해 컨테이너를 선박에 적재할 때는 무겁거나 목적지가 먼 컨테이너를 바닥부분에 먼저 선적해야 할 것이고, 수입된 컨테이너들은 육상 수송을 위해 도착한 트럭들의 행선지 순서에 맞게 꺼내 실어야 한다. 이를 위해서는 애초부터 순서에 맞게 쌓거나 사후에 정렬 작업을 실시해야 하는데, 스택에 이미 놓여 있는 컨테이너를 다른 위치로 이동시켜 원하는 배치로 전환하는 작업을 총체적으로 재배치 또는 재취급(rehandling)이라 한다.

항만에서의 컨테이너 재배치 작업은 크게 3 단계에 걸쳐 이

• First Author: Hyung-Bong Lee, Corresponding Author: Ki-Hyeon Kwon
*Hyung-Bong Lee (hblee@gwnu.ac.kr), Dept. of Computer Science & Engineering, Gangneung-Wonju National University
**Ki-Hyeon Kwon (kweon@kangwon.ac.kr), Dept. of Electronics, Information & Communication Engineering, Kangwon National University
• Received: 2017. 07. 04, Revised: 2017. 07. 19, Accepted: 2017. 09. 07.

루어진다[1,2]. 첫 번째 단계는 수입을 위해 선박에서 막 하역되거나 수출을 위해 트럭 등 육상 운송 수단에 의해 도착한 컨테이너들이 터미널 야적장에 처음 적치될 때 이루어지는데, 이 경우 컨테이너들의 정렬 기준이 결정되어 있다는 전제 하에 컨테이너가 도착할 때마다 재배치 작업의 연속으로 정렬 상태를 유지하도록 한다. 그러나 앞으로 도착할 컨테이너들에 대한 정보가 사전에 정확히 알려지기는 현실적으로 어렵기 때문에 비효율적인 재배치 작업이 이루어질 가능성이 높다. 두 번째 단계는 컨테이너가 선박에 선적되거나 육상으로 배송되기 며칠 전에, 적치된 컨테이너 전체를 대상으로 재배치 작업이 이루어진다. 이 경우는 더 이상의 변동 사항이 일어나지 않는다는 전제가 필요하고, 이 과정에서의 문제 해결 영역을 CPMP( Container Pre-Marshalling Problem)이라 부른다. 특히, CPMA-A 영역에서는 초기 배치를 주어진 정렬 기준에 따라 정렬하는 방법을 다루고, CPMP-B에서는 초기 배치를 임의의 주어진 배치로 재배치하는 방법을 다룬다. 마지막 세 번째 단계는 선박 선적이나 육상 트럭으로의 배송작업을 하면서 순서에 따른 반출을 위해 이루어진다. 보통 선적이나 배송 직전까지는 여러 가지 변동 사항으로 인해 컨테이너들의 순서가 결정되지 않으므로 이 단계에서의 재배치 작업이 가장 활발하다. 보통 CRP(Container Relocation Problem)라 함은 세 번째 단계에서의 주어진 순서 기준에 따른 재배치 및 반출 방법을 의미한다.

선적 및 배송을 위한 컨테이너 재배치 작업의 효율성은 곧 해당 항만 전체의 효율성과도 직결되고, 터미널 설비 규모가 지속적으로 증가하고 있는 최근 추세에 따라 그 중요성 또한 크게 높아지고 있다[3]. 이 논문에서는 CRP의 최근 연구인 GLAH(Greedy Look-ahead Heuristic) 알고리즘[1]에 대한 성능 평가와 함께 개선점을 모색한다. 이를 위해 2장에서 CRP 문제의 내용과 관련 연구를 살펴보고, 3장에서 GLAH 알고리즘을 자세히 분석하고 성능특성을 조명한다. 4장에서는 GLAH 알고리즘 개선점을 제시하고, 마지막 5장에서 결론으로 맺는다.

## II. Problem Description and Related Works

### 1. Problem Description

그림 1의 컨테이너 터미널 설비는 스택의 탑 컨테이너에 대한 접근만이 허용된다는 당연한 제한을 가지며 이러한 대명제하에, 쌓여있는 컨테이너들을 최소 비용으로 정렬하는 방안 즉, 최소 재배치 순서(minimum relocation sequence)를 찾는 것이 CRP 알고리즘의 목표이다. 여기서 비용이란 목적을 이룰 때까지의 소요시간을 의미하는 것으로 컨테이너의 무게나 에너지 소모 등은 고려하지 않는다. 컨테이너 이동 장치인 크레인은 베이 사이를 이동하는 시간을 크게 소모하므로 베이 내에서의 정렬 방안이어야만이 의미가 있고, 스택의 높이에 따른 상하 이동 시간차는 아주 미미하므로 무시할 수 있기 때문에, 결국 재배치 비용은 컨테이너 재배치(이동) 횟수로 정의해도 무방하다[4]. 즉, CRP 알고리즘은 주어진 초기 배치($L^0$: initial layout)로부터 목적(target) 컨테이너를 순서에 따라 차례로 하나씩 반출(retrieve)하는 과정에서, 목표 컨테이너의 접근을 방해하고 있는 상위의 컨테이너들을 다른 스택으로 옮기기 위한 전략의 전개로 보면 타당한데, 이 전략은 제한 이동(restricted move)과 무제한 이동(unrestricted move) 등 크게 두 가지로 분류된다[2]. 제한 이동에서는 현재 목표 컨테이너 위에 있는 것들만 이동이 가능하고, 무제한 이동에서는 어떤 스택의 컨테이너라도 이동이 가능하다. 그림 2에서 보는 바와 같이 제한 이동과 무제한 이동 중 어느 쪽이 유리한가는 단정 지을 수 없다.

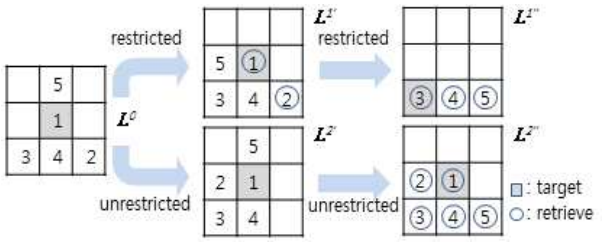


Fig. 2. Comparison between Restricted and Unrestricted Move

CRP에 대한 최적의(optimal) 해는 현재 주어진 배치로부터 어느 스택의 컨테이너를 어느 스택으로 옮겨서 다음 배치($L$)로 전환하는 것이 최선인가에 대한 명확한 기준이 존재한다면 다항시간 내에 간단하게 구해질 수 있지만 그 기준이 알려져 있지 않기 때문에 기본적으로 이동 가능한 모든 경우의 수에 대한 예측 재배치 비용을 비교하여 결정하는 비결정적(non- deterministic) 알고리즘을 따를 수밖에 없다. 그런데 배치의 크기 즉, 베이의 규모에 비례하여 경우의 수가 거의 무한대로 증가하므로 NP 알고리즘의 현실적 적용이 불가능하다. 따라서 현재까지의 모든 CRP 알고리즘 관련 연구는 확률론이나 시행착오 경험에 기초한 최선책(heuristic method) 탐색에 모아져 있다.

### 2. Container Relocation Related Works

초기의 컨테이너 재배치 관련 연구는 항만을 설계하거나 항만의 처리 능력을 평가하기 위한 하나의 척도로서 적치된 컨테이너 더미에서 하나의 컨테이너 반출을 위해 과연 얼마만큼의 재배치 횟수가 필요할 것인가를 계량화하는 데에서 출발하였고, 그 방법으로 확률론을 도입하였다[4]. 이를테면 그림 3의 배치도에서 임의의 컨테이너에 대한 반출이 요구된다면, 각 컨테이너가 반출될 확률은 1/10이고, 컨테이너 1이 선택될 경우는 2번의 이동 작업이 필요하다. 이를 모든 컨테이너에 적용하면 예상되는 평균 재배치 이동 횟수 $Nr$은 식 (1)과 같이 구할 수 있다.

| | | | | | |
|---|---|---|---|---|---|
| 10 | | | | | |
| 7 | 8 | 9 | | | |
| 1 | 2 | 3 | 4 | 5 | 7 |

Fig. 3. A Sample Layout for Probability-based Cost Estimation

$$Nr = 0.1*2 + 0.1*1 + 0.1*1 + 0.1*0 + 0.1*0 + 0.1*0 \cdots\cdots\cdots (1)$$

식 (1)을 확장하여 수출을 위해 터미널에 도착한 컨테이너를 무게에 따라 그 적치 위치를 결정하는 방안들이 제안되었다[5, 6]. 이미 도착하여 쌓여있는 컨테이너들을 선박 선적이나 육상 배송 전에 외부 공간을 사용하지 않고 베이 내부 이동만으로 정렬하는 CPMP 알고리즘들[7-9]도 제안되었다. [7,8]은 그림 4와 같이 컨테이너 재배치에 의해 흐르는 타임 라인(시계열 혹은 타임 슬롯)에 따라 변화하는 배치도를 네트워크 모델로 전개하고, 가능한 모든 이동 즉, 간선(sp, i, o, u, d, r, dm의 7 가지)에 독립된 정수 변수를 대응시켜 그 값이 1이면 선택, 0이면 버림으로 정의한다. 여기에 컨테이너 정렬 조건(constraint) 26 가지를 적용하여 ILP(Integer Linear Programming) 문제[10]로 변환한다. 이를테면 모든 컨테이너는 바닥이나 다른 컨테이너 위에 존재한다는 스택의 기본 요건은 식 (2)와 같이 표현된다.

$$i(t, s, h) \geq i(t, s, h+1),\ \forall t \in TIME,\ s \in STACK,\ h \in HEIGHT \cdots\cdots (2)$$

이 기법에서는 타임 라인의 길이가 해의 최적성과 소모시간 사이의 상충 요소로 작용하여 그 설정에 어려움이 있다.

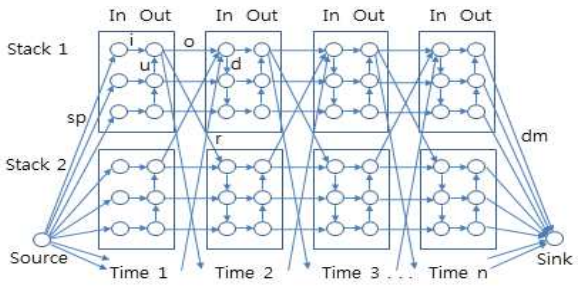


Fig. 4. The Network Model for CPMP

컨테이너를 주어진 순서대로 반출하기 위한 본격적인 CRP 알고리즘들은 트리기반 방법론으로 집중되고 있다[1-3]. 트리기반 방법론은 주어진 배치 $L$에서 어느 한 컨테이너의 재배치 혹은 반출에 의해 변환된 후속 배치 $L'$을 자식 노드로 대응시키고 배치의 축약이 완료될 때까지의 경로를 트리로 표현하여 그 중 최소비용 경로를 탐색한다. 그런데, $n$개의 스택이 있다면 $n*(n-1)$개의 재배치가 존재하여 생성 경로가 너무 많으므로 가능성이 낮은 경로는 전지하고 그 중 일부 경로만을 선택하여 탐색해야 한다. 이를 위해서는 현재 주어진 배치에서 가능한 모든 재배치 $<s_i \rightarrow s_j>$(스택 $i$에서 스택 $j$로의 이동) 중 최선의 (i, j) 쌍에 대한 판단 기준이 필요하다. 이는 재배치 $<s_i \rightarrow s_j>$에 의해 변환된 배치 $L''$의 축약 비용 예측 문제로 귀결된다.

B&B(Bound & Branch)[11]는 배치 $L$의 축약 비용을 식 (3)의 재배치 하한선(RLWB: Relocation Low Bound)과 식 (4)의 ENAR( Expected Number of Additional Relocations) 두 가지를 제안하여 트리기반 방법론[1-3]의 토대가 되었다. RLWB는 어떤 축약 방법을 적용하더라도 결코 더 줄일 수 없는 명백한 최소 재배치 횟수를 의미하는 데, 그 수치의 도출 방법은 연구관점에 따라 상이하다.

$RLWBofB\&B$ = 단말 노드까지의 재배치 횟수(깊이) + 각 스택에서 우순 순위가 가장 높은 컨테이너의 접근을 방해하는 컨테이너 수 ····· (3)

ENAR은 주어진 배치를 축약하기 위해 필요한 전체 재배치 횟수를 확률론에 근거하여 예측한다. 이를테면 1~20의 20 개 컨테이너가 적치된 베이에서 그림 5의 스택에서, 1~5(5 종류)의 컨테이너가 탑으로 온다면 더 이상의 재배치가 필요 없지만 7~20(12 종류)의 컨테이너가 온다면 이들에 대해서는 반드시 재배치가 필요하므로 이 스택에 대한 ENAR은 식 (4)로 구할 수 있다.

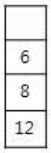


Fig. 5. A Sample Stack for ENAR Concept

$$ENARofFig.5 = 5/17*0 + 12/17*1 \cdots\cdots (4)$$

도출된 예측 비용을 트리 탐색에 적용하는 방법도 다양하다. B&B는 그림 6과 같이, 일단 주변의 자식노드들을 적절한 깊이(3~5)까지 깊이우선 탐색(DFS: Depth First Search)한 뒤 각 경로의 탐색 단말노드에 재배치 하한선을 적용하여 선택된 자식노드를 다음 단계의 루트로 삼는다. ENAR은 바로 아래의 자식노드들에 적용되어 다음 루트 선정에 활용된다.

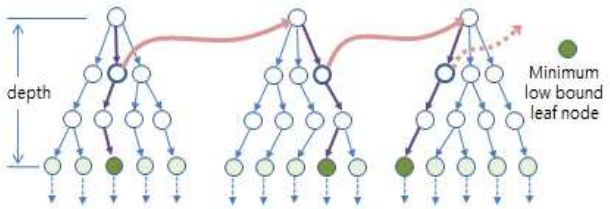


Fig. 6. B&B Approach with DFS and Back-tracking

IDA*(Iterative Deepening A*) 알고리즘[2, 12]]은 휴리스틱 방법을 적용하여 나름대로 최선의 초기 해를 다항 복잡도 알고리즘으로 구한 후 B&B와 유사한 단계를 거치며 주어진 시간 동안 해를 최적화해 나간다. IDA*의 초기 해 탐색 휴리스틱은 현재 주어진 배치에서 후속 배치로의 신속한 전이를 위해 제한 이동법을 적용하고, 그 이동의 수용 스택 선정을 위한 아래의 4 가지 기준을 도입한다.

- PR1: 높이가 가장 낮은 스택 선택(동일하다면 가장 앞쪽 스택 선택)
- PR2: 스택별 반출순위가 가장 빠른 컨테이너를 비교하여, 그 순위가 가장 낮은 스택 선택
- PR3: 스택별 반출순위가 가장 빠른 컨테이너 중, 이동 예정 컨테이너 우순순위보다 낮은 것이 있다면 그 중에서 우선순위 차이가 가장 작은(반출 우선순위가 가장 높은) 스택을 선택하고(경우 a), 없다면 마찬가지로 우선순위 차이가 가장 작은(반출 우선순위가 가장 낮은) 스택을 선택(경우 b)
- PR4: PR3와 동일하되, 경우 b이고 이동 예정 컨테이너가

해당 스택에서 두 번째로 높은(반출 예정 컨테이너의 바로 다음) 순위가 아니며, 선택된 수용 스택에 오직 하나의 빈자리만 있다면 해당 스택을 포기하고 두 번째로 낮은 스택을 선택

위 각 기준에 따라 얻어진 해는 주어진 배치의 형태에 따라 그 우수성이 일정치 않으므로 네 가지 해 중 가상 우수한 쪽을 초기 해로 선정한다. 이와 같이 거칠게 도출된 초기 해를 전역적 최소화 대상($S_{best}$)으로 설정한 후 B&B 형태인 깊이우선 탐색을 진행한다. 여기서는 주어진 배치를 모두 비우기 위해 명백하게 필요한 최소한의 재배치 횟수의 하한선을 식 (1, 3, 4)를 보완하여 식 (5)와 같이 정의하고 이를 바탕으로 탐색 경로에 대한 비용을 예측한다.

$LWBofIDA^*$ = 각 스택별 역전되어 있는 컨테이너 수+ 각 스택별 반출 예정 컨테이너 위에 위치한 컨테이너 수 ····· (5)

TSP(Tree Search Procedure) 알고리즘[3]은 깊이우선 탐색과정에서 해당 경로에 대한 비용 예측을 위한 재배치 하한선을 식 (6)으로 정의하여 전지 정확성을 개선하였고, 식을 구성하는 항목은 아래와 같다.

• $n'_{REM}$ : 배치 L에 존재한 각 컨테이너에 대한 총 반출 횟수로서 n(L) 즉, L에 존재하는 컨테이너 수와 동일

• $n'_{BG}$ : 반드시 Bad→Good 유형의 재배치가 요구되는 총 횟로서 오치 컨테이너의 총 수와 동일

• $n'_{non\text{-}BG}$ : Bad→Good 유형 이외의 재배치가 최소한 한 번 필요한가를 지시(1 혹은 0). 이를 위한 조건은 빈 스택이 없고, 각 스택의 탑 컨테이너의 반출순위 중 가장 높은 것이 각 스택별 최우선 반출순위 중 가장 낮은 것보다 낮은 경우이다.

$$LWBofTSP = n'_{REM} + n'_{BG} + n'_{non\text{-}BG} \cdots\cdots (6)$$

B&B와 $IDA^*$, 그리고 TSP 알고리즘에 이어 초기 해의 정확도를 개선하여 이를 DFS 의 전지 지표로 활용함으로써 성능을 획기적으로 개선한 GLAH 알고리즘[1]이 제안되었고, 이를 바탕으로 컨테이너들이 시간대별로 반출 되는 경우를 확률적으로 접근하는 TW(Time Window) 알고리즘[13] 등이 연구되고 있다.

## III. Analysis on Greedy Look-ahead Heuristic Algorithm

B&B, $IDA^*$, TSP 알고리즘 등은 DFS 과정을 축약이 완료될 때까지 반복하여 최적 해를 구하기 때문에 이를 위한 충분한 시간이 주어져야 한다. 이를 보완하기 위해 가장 최근의 연구결과 중의 하나인 GLAH 알고리즘은 초기 해를 간단하면서도 적절한 휴리스틱으로 구한 후, 이를 토대로 DFS를 통해 더 우수한 해를 탐색하는 방법을 사용한다. 이 때 초기 해는 DFS 과정에서 확실한 전지 기준이 되어 동일한 시간 내 더 많은 경로탐색을 가능하게 한다.

### 1. Identification of well-placed and badly-placed container, and priority of relocation type

GLAH는 스택에 쌓여있는 각 컨테이너의 상태를 반출 순서에 접합한지 여부에 따라 정치(well placed)와 오치(badly placed)로 분류한다. 정치는 해당 컨테이너 아래에 더 빠른 반출 순서의 컨테이너가 없는 경우이고, 오치는 더 빠른 컨테이너가 하나라도 존재하는 경우이다. 그림 7의 예에서 음영 처리된 것들이 오치 컨테이너들이다. 정치 컨테이너들만 쌓여있는 스택을 청결(clean), 그렇지 않은 스택을 불결(dirty)이라 정의한다. 이에 따라 재배치의 유형은 자연스럽게 BG(Bad→Good), BB(Bad→Bad), GG(Good→Good), GB(Good→Bad) 등 네 가지로 분류되는데, 선택 우선순위 또한 직관에 의해 같은 순서로 부여한다. 여기서 BG는 오치 컨테이너가 이동 후 정치가 되는 위치로 재배치한다는 의미이다. 그런데 오치 컨테이너가 반출 목적 컨테이너 바로 위에 존재한다면 이 컨테이너의 재배치 후 곧바로 목적 컨테이너를 반출(free a target )할 수 있으므로 다른 경우보다 더 높은 우선순위를 부여한다. 이를 위해 BG와 BB 유형을 $B_{ft}G$, $B_{ft}B$, $B_{nf}G$, $B_{nf}B$ 등 네 가지 유형으로 세분하여 우선순위에 차등을 둔다. 동일한 재배치 유형이 다수이면 아래와 같이 해당 유형별 2차 기준을 적용하여 선택한다.

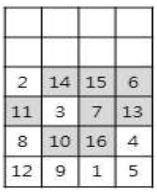

Fig. 7. Well(□) and Badly(■) Placed Container

• $B_{ft}G$, $B_{nf}G$ : 재배치 컨테이너와 이 컨테이너가 위에 놓일 컨테이너(수용 스택의 탑(top)) 반출 순위 차이가 가장 작은 스택을 선택. 만약 빈 스택으로 재배치해야 한다면 재배치 컨테이너 반출순위가 가장 높은 것을 선택

• $B_{ft}B$, $B_{nf}B$ : 반출 목표 컨테이너가 존재하는 스택 중 하나를 반출 스택으로 선택하되, 해당 컨테이너 반출순위와 스택별 최고 반출순위와의 차가 가장 크도록 선택

### 2. Heuristics for initial solution of GLAH

GLAH는 주어진 배치에서 순서에 따른 반출 목적 컨테이너 하나를 긴급 컨테이너로(urgent target) 지정하고, 그 위에 놓인 오치 컨테이너들을 모두 재배치한 후 긴급 컨테이너를 반출하는 과정을 축약이 완료될 때까지 반복한다. 이 때 긴급 컨테이너는 그림 8과 같이 반출하기 전까지의 재배치 횟수가 가장 작은 것 즉, 위에 놓인 컨테이너 수가 가장 작은 것으로 선택한다. 그림 9에 긴급 컨테이너 가 있는

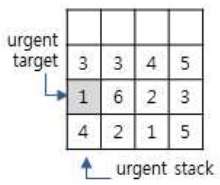

Fig. 8. An Example of Urgent Target

스택 $s^*$를 결정하는 Get_Urgent_Target() 프로시저를 보였다. 긴급 컨테이너가 결정되면 오치 컨테이너들이 옮겨갈 수용 스택을 아래의 우선순위로 결정하여 재배치한다.

```
Get_Urgent_Target(layout L, int  s*)
{
   s* = stack including the target container
        which have the fewest number of badly
        placed containers above it;
}
```

Fig. 9. Determining Urgent Target Procedure of GLAH Algorithm

• B*G 재배치가 가능한 스택 중, 스택별 최고 반출 우선순위 컨테이너 기준으로 가장 높은 스택 선택($S_1$ : 그림 10의 a)

• 스택($s$')의 탑($c$')이 정치이고, 이 컨테이너를 보조 스택($s_a$)으로 GG 재배치한 후, 그 곳으로 긴급 컨테이너의 BG 재배치가 가능한 스택 중, BG 재배치 컨테이너의 반출 우선순위가 가장 낮은 스택 선택($S_2$ : 그림 10의 b). 보조 스택으로는 스택별 최고 반출 우선순위 기준으로 가장 높은 스택 선택

• BB 재배치가 가능한 스택 중, 스택별 최고 반출 우선순위 기준으로 가장 낮은 스택 선택($S_4$ : 그림 10의 d). 단, 해당 스택의 빈 공간이 하나이고, 재배치 대상 컨테이너의 반출 우순순위가 재배치 대상 컨테이너들 중 최고가 아니라면 두 번째로 낮은 스택 선택($S_3$ : 그림 10의 c)

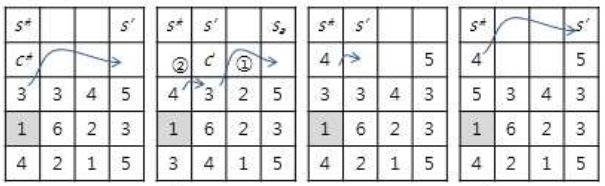


Fig. 10. Determining Destination Stack in GLAH Algorithm

재배치가 결정되어 컨테이너 $c^*$를 $c'$ 위로 재배치하기 전, $c'$~$c^*$ 사이에 BG 재배치 유형으로 끼워 넣어 정렬이 가능한 컨테이너들이 다른 스택에 존재할 경우 이들을 미리 옮겨 정렬한다. 그림 11에 끼워 넣기 정렬 프로시저 Gap_Utilize()를 보였는데, 여기서 f(s) 는 해당 스택에서 탑을 제외한 아래의 컨테이너 중 최고 반출순위를 의미하고, 식 top(s) > f(s)는 스택 s의 탑 컨테이너가 오치임을 의미한다. 이들 프로시저를 활용하여 초기 해를 구하는 Get_Initial_Sol() 프로시저는 그림 12와 같다.

```
Gap_Utilize(layout L, moves sol, move(s*→s'))
{
   while (height(s') ≦ T -2) {
      S = {(s ≠ s* ,s' | top(s) > f(s),
                top(s*) ≦ top(s) ≦ top(s') };
      if (S = ∅) break;
      s'' = min_{s∈S}(top(s'));
      L = L ⊕ move(s'' →s');
      sol = sol + move(s'' →s');
   }
}
```

Fig. 11. Gap Utilization Procedure of GLAH Algorithm

```
Get_Initial_Sol(layout L,  moves sol)
{
   sol = ∅;
   while (|L| ≠ 0) {
      Get_Urgent_Target(L, s*, rn); ····· ①
      while (rn > 0) {
         if (S1 ≠∅)
            s' = max_{s∈S1}(spri_max(s));
         else if (S2 ≠∅) {
            s' = min_{s∈S2}(cpri_max(top(s)));
            sa = max_{s∈Sa(s')}(spri_max(s));
            c' = top(s');
            Gap_Utilize(L, sol, move<s' →sa>);
            L = L ⊕ move<s' →sa>);
            sol = sol + move<s' →sa>);
            Gap_Utilize(L, sol, move<s* →s'>);
         }
         else
            s' = element(S3∪S4);
         Gap_Utilize(L, sol, move<s*→s'>);
         L = L ⊕ move<s* →s'>);
         sol = sol + move<s* →s'>);
      }
   }
}
```

Fig. 12. Determining Initial Solution Procedure of GLAH Algorithm

## 3. Look-ahead DFS of GLAH

GLAH의 미리보기(look-ahead) DFS 탐색은 경로별 비용 예측을 제한된 깊이 범위 내에서 탐색하고, 권고(advice)된 최상의 경로에 대하여 다음 단계로 진행함에 있어서 그 경로 전체를 반영하여 전진하지 않고, 그 경로가 시작되는 첫 1 단계(바로 아래 자식노드)까지만 전진한다는 점에서는 B&B와 유사하다. 탐색 단말노드에 대한 비용 예측을 위한 RLWB는 식 (7)로 정의한다.

• $n'_{REM}$ : 배치 $L$에 존재한 각 컨테이너에 대한 총 반출 횟수로서 $n(L)$ 즉, $L$에 존재하는 컨테이너 수와 동일

• $n'_{BG}$ : Bad→Good 유형의 재배치가 반드시 요구되는 전체 횟수로서 오치 컨테이너 수와 동일

• $n'_{non\text{-}BG}$ : Bad→Good 유형 이외의 재배치가 최소한 한 번 필요한가를 지시(1 혹은 0). 이를 위한 조건은 빈 스택이 없고, 각 스택의 탑의 반출순위 중 가장 높은 것이 각 스택별 최우선 반출순위 중 가장 낮은 것보다 낮은 경우이다.

$$RLWBofGLAH = n'_{REM} + n'_{BG} + n'_{non\text{-}BG} \quad \cdots\cdots (7)$$

GLAH는 DFS 과정에서 깊이 제한에 의해 탐색 단말노드(예:그림 13의 노드 $c$와 $z$)에 이르면 RLWB에 의한 평가 비용이 한계를 벗어난 경로는 전지하고, 이 한계를 통과한 나머지 경로에 대하여 초기 해를 구하는 그림 12의 Get_Initial_Sol() 프로시저를 이용하여 이 노드에서 최종 단말 노드(빈 배치)까지의 해를 구하여 비용에 합산한다.

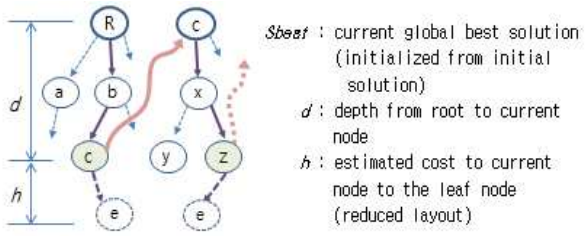


Fig. 13. DFS Process of GLAH Algorithm

즉, 그림 13의 $h$ 부분에 대한 비용을 예측 대신 실측하여 반영한다. 이는 초기 해 구하는 프로시저가 상당히 최적화 되었을 것이라는 전제 하에 가능하다. 후속 노드의 너비는 BftG, BnfG, BftB, BnfB, GG, GB 등 6 가지 가능한 재배치를 순서대로 고려하되 각각의 최대 개수를 제한한다. 그림 14의 Look_Ahead() 프로시저는 DFS 결과로부터 탐색 루트 노드 이하의 모든 자식 노드 중 비용이 가장 적은 것 (예 그림 13의 노드 $b$와 $x$) 하나를 추천한다. Look_Ahead()의 추천을 받은 그림 15의 Greedy() 프로시저는 재배치를 완료하고 다음 단계의 추천을 요구하는 전체 과정을 반복한다.

```
Look_Ahead(layout L_curr, advice adv)
{
    Tree_DFS(L_curr, 0, ∅, adv);
}
Tree_DFS(layout L^d, int depth, moves sol^d,
                                advice adv)
{
    adv.cost = ∞;
    if (|sol^d| + depth+ low_bound_of(L^d) ≧ |S_best|)
        return;
    if (|L^d| = 0 or depth = MAXDEPTH) {
        Get_Initial_Sol(L^d, sol_e);
        if (|S_best| ≧ |sol_e|) S_best = sol_e;
        adv.cost = depth+ |sol_e|; ····· ②
    }
    else {
        get_possible_6type_relocations(L^d, mvs);
        for each mv ∈ mvs {
            L^{d+1} = L^d ⊕ mv, sol^{d+1} = sol^d + mv;
            Tree_DFS(L^{d+1}, depth+ 1, sol^{d+1}, childadv);
            if (childadv.cost < adv.cost)
                adv.mv = mv, adv.cost = childadv.cost;
        }
    }
}
```

Fig. 14. Look-a head(DFS) Procedure of GLAH Algorithm

```
Greedy(layout L_init,)
{
    Get_Initial_Sol(L_init,, S_best);
    S_curr = ∅;
    while (|L_init| ≠ 0) {
        if (|S_curr| + low_bound_of(L_init) ≧ |S_best|)
            break;
        Look_Ahead(L_init, advice);
        L_nit = L_nit, ⊕ mv; ····· ③
        S_curr = S_curr + advice.mv;
    }
}
```

Fig. 15. Greedy Procedure of GLAH Algorithm

# IV. Performance Evaluation & Improvement on GLAH Algorithm

## 1. Performance characteristics of GLAH

단계별 DFS 탐색 루트노드를 평가된 최선의 경로로 진행하지 않고 전체루트 노드(초기 배치)에서의 탐색을 반복하는 IDA*와 전체 루트노드부터의 모든 경로를 전체 단말노드(축약 완료 배치)까지 탐색하는 TSP 알고리즘과의 비교를 통해 GLAH 알고리즘의 특성을 조명한다.

### 1.1 Experiment environment

실험이 실시된 플랫폼은 인텔 i7-4770 4코어 프로세서, 32GB 메모리, 우분트 리눅스 14.04 LTS 운영체제 시스템이다. 데이터는 sshh(nn) 모형의 실험 군으로 분류하여 각 집단별로 난수를 이용해 생성한 1,000개의 개체에 대한 최적 해의 길이와 실행시간의 평균을 비교한다. 예를 들면 '0307(15)' 실험 군은 3개의 스택, 각 스택의 최대 높이 7, 쌓여있는 컨테이너 전체 수가 15개임을 의미한다. 컨테이너들이 스택에 놓이는 배치와 동일 반출 우선순위의 개수 등은 모두 랜덤으로 결정된다. 최적 해의 길이와 수행시간 모두 짧을수록 우수하다.

### 1.2 Quality of the optimal solution

CRP에서 성능평가를 위해 사용하는 전형적인 데이터 규격인 0307(15), 0409(28), 0510(40), 0612(60), 1012(100)의 다섯 군에 대한 각 알고리즘별 최적 해 결과를 그림 16에 보였다. GLAH는 규모가 작은 경우에는 약간 열세이고 규모가 커짐에 따라 그 우수성이 뚜렷한 것으로 나타났다.

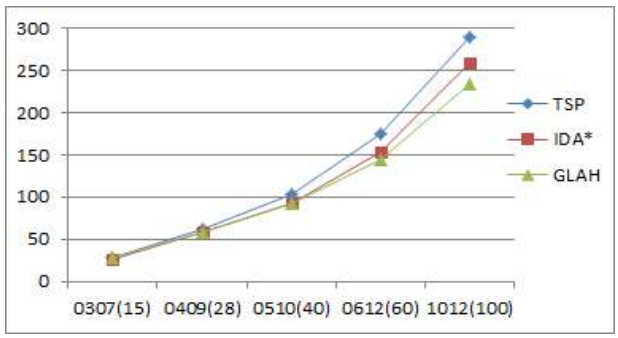


Fig. 16. Comparison of Optimal Solution Quality(unit: count)

### 1.3 Computing time

그림 16의 실험을 위해 각 알고리즘에 최대 소모시간 10초를 설정하였는데 그 결과가 그림 17과 같다. 이 그림에서 TSP와 IDA*는 제한시간 내 종료 비율이 매우 낮지만, GLAH는 거의 다항 알고리즘이라 할 수 있을 정도로 어떤 경우에도 3초를 초과하지 않음을 볼 수 있다. 즉, 컨테이너 재배치 작업이라는 실시간적 측면을 고려하면 TSP와 IDA*는 실용성이 낮은 것으로 판단된다.

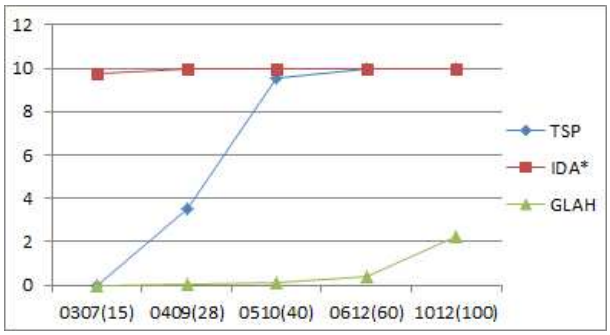


Fig. 17. Comparison of Execution Time(unit: sec)

## 2. Performance improvement of GLAH

GLAH에 대한 성능 개선은 DFS의 비용 예측의 정확도와 초기 해의 우수성 측면에서 접근한다.

### 2.1 Cost estimation process of DFS

GLAH의 시간적 우수성은 우선 그림 15 Greedy() 프로시저의 ③에서 선택된 경로 중 최선이라 판단되는 경로를 오직 하나만 선택하여 전진하는 결정론적 알고리즘을 추구했다는 점에 있다. 그럼에도 최종 해의 결과 또한 우수한 이유는 그 선택근거가 되는 그림 14 Tree_DFS() 프로시저 ②에서의 경로비용 예측 정확성이 높기 때문인데, 이 비용은 초기 해를 구하는 Get_Initial_Sol() 프로시저의 결과에 기반하고 있다. 따라서 이 초기 해 방법의 결과보다 해당 노드(배치)의 비용 즉, 복잡성을 좀 더 잘 대변할 수 있는 지표를 발견할 수 있다면 최종 결과는 더 개선될 것이다. 이에 따라 가능성이 있는 후보 지표를 다음과 같이 정의하고 결과를 측정한다.

• LWB(low bound) : 식 (7)로 정의되는 재배치 하한선으로 오치 컨테이너 수가 지배

• NOB(sum of the numbers of containers prohibited by a badly placed container) : 오치 컨테이너에 의해 반출 방해를 받고 있는 컨테이너들의 수. 즉 오치 컨테이너 아래의 컨테이너들 중, 반출 순위가 더 높은 것들의 개수의 합

• POB(sum of the priority differences between a badly placed container and the containers prohibited by the badly placed container) : 오치 컨테이너와 오치 컨테이너에 의해 방해받고 있는 컨테이너들 사이의 반출 우선순위 차의 합

• HOB(sum of the height differences between a badly placed container and the containers prohibited by the badly placed container) : 오치 컨테이너와 오치 컨테이너에 의해 방해받고 있는 컨테이너들 사이의 스택 거리(높이 차)의 합


Fig. 18. A Sample Stack for New Cost Metrics

예를 들어 오치 컨테이너가 8과 12인 그림 18의 스택 예에서 각 지표 값은 아래와 같다.

◦ NOB = 1(12가 방해하는 컨테이너 7)+ 2(8이 방해하는 컨테이너 5와 7) = 3.

◦ POB = (12 - 7) + (8 - 5) + (8 - 7) = 9.

◦ HOB = (2 - 1) + (4 - 3) + (4 - 1) = 5.

위 지표들을 적용한 최적 해의 비교결과는 그림 19와 같은데, 지표 HOB가 유일하게 모든 실험 군에서 우수함을 보이고 있다.

| data \ metric | GLAH | LWB | NOB | POB | HOB | Trend |
|---|---|---|---|---|---|---|
| 0307(15) | 27.32 | 27.35 | 27.48 | 27.44 | 27.3 | |
| 0409(28) | 59.08 | 59.61 | 60.13 | 59.84 | 58.67 | |
| 0510(40) | 92.4 | 93.28 | 93.61 | 93.11 | 91.21 | |
| 0612(60) | 144.2 | 145.5 | 145.8 | 145 | 142.2 | |
| 1012(100) | 233.2 | 234.3 | 234.4 | 233.4 | 230.2 | |

Fig. 19. Comparison of DFS Cost Estimation Metrics(unit: count)

### 2.2 Exploration process for the initial solution

GLAH의 초기 해 탐색은 그림 12 Get_Initial_Sol() 프로시저의 ①과 같이 주어진 배치에서 긴급 컨테이너 선정으로 시작한다. 그런데 그림 9의 선정 프로시저는 후보 스택들의 1차 기준이 동일할 때 2차 기준에는 주목하지 않는다. 이를 보완하기 위해 2차 기준으로 스택별 NOB, POB, HOB와 단순 오치 컨테이너 수인 NBC(number of badly-placed containers in a stack)를 적용한 결과를 그림 20에 보였다. 이 그림에서 NBC가 유일하게 모든 실험 군에서 우수하게 나타난다.

| data \ metric | GLAH | NBC | NOB | POB | HOB | Trend |
|---|---|---|---|---|---|---|
| 0307(15) | 29.83 | 29.79 | 29.8 | 29.84 | 29.84 | |
| 0409(28) | 66.74 | 66.61 | 66.63 | 66.67 | 66.69 | |
| 0510(40) | 104 | 103.8 | 103.9 | 103.9 | 103.88 | |
| 0612(60) | 163 | 162.8 | 162.8 | 162.8 | 162.89 | |
| 1012(100) | 256.9 | 256.9 | 256.9 | 257 | 257.05 | |

Fig. 20. Evaluation on Urgent Stack Selection Criteria in Determining Initial Solution Process(unit: count)

GLAH 초기 해 탐색의 또 다른 특징은 그림 10의 b에 보인 보조스택에 의한 간접정렬과 그림 11의 끼워 넣기에 의한 사전정렬이다. 그림 21은 이들이 초기 해 결과에 미치는 영향을 검증한 결과인데, 끼워 넣기는 오히려 악영향을 미치는 것으로 나타나 제거 필요성이 있다. 이 그림에서 GAP은 끼워 넣기만을, ASS는 보조스택만을, NON은 어느 쪽도 적용하지 않음을 각각 의미한다.

| data \ metric | GLAH | GAP | ASS | NON | Trend |
|---|---|---|---|---|---|
| 0307(15) | 29.83 | 29.9 | 29.53 | 29.59 | |
| 0409(28) | 66.74 | 67.09 | 65.4 | 65.77 | |
| 0510(40) | 103.97 | 104.62 | 101.39 | 102.09 | |
| 0612(60) | 162.96 | 164.33 | 157.53 | 159.38 | |
| 1012(100) | 256.94 | 258.82 | 245.05 | 247.5 | |

Fig. 21. Side Effects of Assistant Stack and Gap Utilization in Determining Initial Solution Process(unit: count)

### 2.3 Comprehensive application of improvements, and analysis on the optimal solution

주어진 배치에 대한 새로운 복잡도 지표인 HOB의 적용과 보조스택에 의한 간접정렬 프로세스 제거 등 DFS 비용 예측 및 초기 해 탐색 프로세스에 대한 개선점들을 모두 적용한 알고리즘 GLAH'의 성능평가 결과를 그림 22, 23에 보였다. 그림 22는 각 실험군에 대한 GLAH와 GLAH' 두 알고리즘의 최소 컨테이너 이동 횟수 비교이고, 그림 23은 각 실험군 별 이동 횟수의 차이를 보였다. 이 두 그림에서 보는 바와 같이 모든 실험군에서 GLAH' 가 우수하고, 배치 규모가 클수록 개선의 폭이 증가하는 특성이 있으며, 이 연구에서 최대 규모인 1012(100) 즉, 10개의 스택, 최대 높이 12, 전체 컨테이너 수 100개인 실험군에서 3.6%의 컨테이너 이동횟수 감소 효과가 있으며 이는 곧 시간적 비용 절감을 의미한다.

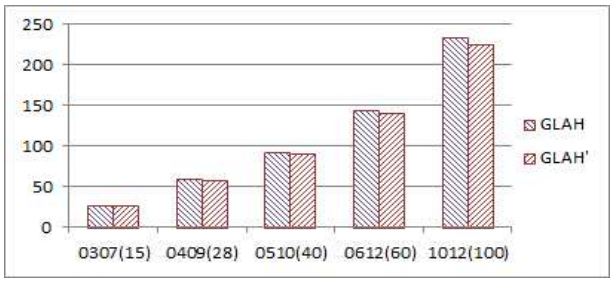


Fig. 22. Performance of the Improved GLAH Algorithm(unit: count)

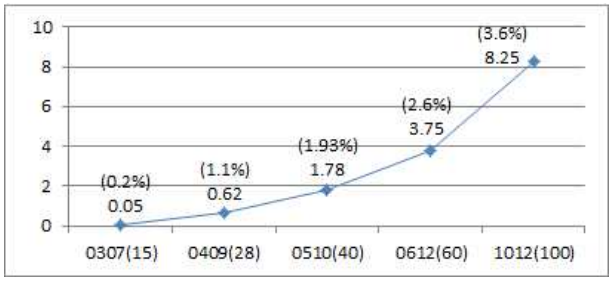


Fig. 23. Trend of the Performance Increment of the Improved GLAH Algorithm(unit: count)

## V. Conclusions

부두 컨테이너 터미널의 효율적 운영을 위한 컨테이너 재배치 작업용 CRP 알고리즘은 그 속성상 결정론적 알고리즘이 불가능하여 시행착오적 직관과 실험에 의한 휴리스틱 기법을 도입하지 않을 수 없다. 이 때 도입된 휴리스틱은 모든 개별 경우에서 모두 적합하지 않으므로 평균적으로 우수하게 나타나는 기준을 발견해야 한다. 이 연구에서는 가장 최근 연구 중의 하나인 GLAH 알고리즘을 심층 분석하고, 이를 개선하는 사례를 보이고 검증하였다. 15, 28, 40, 60, 100 등 다섯 개 유형의 컨테이너로 구성된 실험 군 각각에 1000개의 랜덤 배치를 두고, 모든 실험 군에서 최적 해 도출을 개선할 수 있는 비용 예측 지표를 발견하여 적용한 결과 최대 3.6%의 개선효과를 보였다. 여기서 3.6%는 알고리즘 자체의 시간적 관점이 아닌 도출 결과에 대한 것으로 실제 현장에서의 비용 절감을 의미한다.

이 연구의 의의는 개선효과 못지않게 모든 실험 군에 적용 가능한 휴리스틱 지표를 발견하는 사례를 보임으로써 향후 파급효과가 더 높은 새로운 지표도출 가능성을 높였다는 점에서도 크다고 판단된다.

## REFERENCES

[1] B. Jin, W. Zhu, and A. Lim, "Solving the container relocation problem by an improved greedy look-ahead heuristic," European Journal of Operational Research, Vol. 240, No. 3, pp. 837-847, February 2015.

[2] W. Zhu, H. Qin, and A. Lim, "Iterative Deepening A* Algorithms for the Container Relocation Problem," IEEE Transactions on Automation Science and Engineering, Vol. 9, No. 4, pp. 710-722, October 2012.

[3] F. Forster, A. Bortfeldt, "A tree search procedure for the container relocation problem", European Journal of Operational Research, Vol. 39, No. 2, pp. 299-309. February 2012.

[4] K. H. Kim, "Evaluation of the number of rehandles in container yards," Computers & Industrial Engineering, Vol. 32, No. 4, pp. 701-711, September 1997.

[5] K. H. Kim, Y. M. Park, and K. R. Ryu, "Deriving decision rules to locate export containers in container," European Journal of Operational Research, Vol. 124, No. 1, pp. 89-101. July 2000.

[6] J. Kang, K. R. Ryu, K. H. Kim, "Deriving stacking strategies for export containers with uncertain weight information," Journal of Intelligent Manufacturing, Vol. 17, No. 4, pp. 399-410, August 2006.

[7] Y. Lee. N. Y. Hsu, "An optimization model for the container pre-marshalling problem," Computers & Operations

Research, Vol. 34, No. 11, pp. 3295-3313, November 2007.
[8] M.S. Gheith, A. B. Eltawil, N. A. Harraz, and S. Mizuno, "An Integer Programming Formulation and Solution for the Container Pre-marshalling Problem ," Proceedings of CIE44 & IMSS'14, pp. 2047-2056. October 2014.
[9] Y. Lee, S. L. Chao, "A neighborhood search heuristic for pre-marshalling export containers," European Journal of Operational Research, Vol. 196, No. 2, pp. 468-487. July 2009.
[10] Wikipedia(ILP: Integer Linear Programming), https://en.wikipedia.org/wiki/Integer_programming
[11] K. H. Kim, G. P. Hong, "A heuristic rule for relocating blocks," Computers & Operations Research, Vol. 33, No. 4, pp. 940-954, April 2006.
[12] Wikipedia(A* Algorithm), https://en.wikipedia.org/wiki/A*_search_algorithm
[13] D. Ku, T. S. Arthanari, "Container relocation problem with time windows for container departure," European Journal of Operational Research, Vol. 252, No. 3, pp. 1031-1039. August 2016.

## Authors

Hyung-Bong Lee received the B.S. and M.S. degrees in Computer Science and Statistics from Seoul National University, Korea, in 1984 and 1986 respectively and Ph.D. degree in Computer Science from Kangwon National University, Korea, in 2002. Dr. Lee joined the faculty of the Department of Computer Engineering at Gangneung-Wonju National University, Gangneung, Korea, in 2004. He is currently a Professor in the Department of Computer Engineering at Gangneung-Wonju National University. He is interested in embedded systems and sensor networks.

Ki-Hyeon Kwon received the B.S., M.S. and Ph.D. degrees in Computer Science from Kangwon National University, Korea, in 1993, 1995 and 2002, respectively. Dr. Kwon joined the faculty of the Dept. of Electronics, Information & Communication Engineering at Kangwon National University, Samcheok, Korea, in 2002. He is currently a Professor in Dept. of Electronics, Information & Communication Engineering at Kangwon National University. He is interested in pattern recognition, image processing and embedded software.