한국항해항만학회지 제40권 제3호
J. Navig. Port Res. Vol. 40, No. 3 : 113-120, June 2016 (ISSN:1598-5725(Print)/ISSN:2093-8470(Online))
DOI : http://dx.doi.org/10.5394/KINPR.2016.40.3.113

컨테이너 터미널에서 이웃 베이를 활용한 컨테이너 재정돈 계획

† 박영규

† 가야대학교 항만물류학과 교수

Remarshalling Plan Using Neighboring Bay in Container Terminal

† *Young-Kyu Park*

† *Department of Port & Logistics, Kaya University, Gimhae 621-748, Korea*

요 약 : 선박에 싣기 위하여 컨테이너 장치장에서 반출하는 컨테이너 상단에 다른 컨테이너들이 장치되어 있다면 위의 컨테이너들을 임시로 옮기는 작업이 필요한데 이런 작업을 재취급이라 하며, 이로 인하여 장치장의 생산성은 떨어지게 된다. 장치장에서 반출 작업을 수행할 때 재취급 작업을 하지 않도록 하기 위하여, 적하 계획이 수립된 후 선박이 도착하기 전까지의 유휴 시간을 이용하여 컨테이너들을 미리 정리하는 작업을 재정돈 작업이라고 한다. 보통의 경우 베이내의 이동만으로 재정돈 작업 계획을 수립하는데, 컨테이너가 장치된 상태에 따라 너무 많은 시간을 사용한 후 결국 해를 구하지 못하는 경우가 발생한다. 본 논문에서는 이웃한 베이의 상단을 임시 저장 공간으로 활용하여 재정돈 작업 계획을 수립할 수 있는 방안을 제안한다. 이 방안이 복잡한 장치상태에서도 허용된 시간 내에 작업 계획을 수립할 수 있다는 것을 보이기 위하여, 수치실험을 통해 제안한 방안의 성능을 평가하였다. 다양한 베이를 대상으로 실험한 결과 복잡한 베이에 대하여 허용된 시간 내에 유효하고 효율적인 계획을 수립할 수 있음을 확인하였다.

핵심용어 : 재정돈 계획, 베이, 재취급, 컨테이너

Abstract : *If there are containers stacked upon the container to be fetched out of a container yard to vessel, rehandling which moves those containers to other places temporarily is needed. In order to avoid such rehandling, remarshalling which rearranges containers should be done before the vessel arrives. The remarshalling plan is commonly generated within a bay. It happens, however, that the generation of the intra-bay remarshalling plan within the permitted time is not possible because of bad stacking conditions. This paper presents the remarshalling algorithm which uses the empty slots of the neighboring bay as a temporary storage space. Simulation experiments have shown that the presented algorithm can generate the remarshalling plan within the permitted time under any staking conditions.*

Key words : *Remarshalling plan, Bay, Rehandling, Container*

1. 서 론

컨테이너 장치장에서 실시하는 작업은 크게 컨테이너를 장치장내부로 반입하는 작업과, 장치장외부로 반출하는 작업으로 이루어져 있다. 컨테이너 터미널의 생산성 관점에서 보면 두 가지의 작업에 소요되는 시간을 최소화 하는 것이 중요하다. 수출 컨테이너의 경우 장치장내부로 반입하는 작업의 순서는 컨테이너가 장치장에 도착하는 순서에 따르게 되며, 장치장외부로 반출하는 작업은 보통 선박에 싣는 순서를 따른다. 선박에 컨테이너를 싣기 위하여 컨테이너를 장치장 외부로 반출하는 시점에, 만약 장치장에서 반출하는 순서에 맞지 않게 적재된 컨테이너가 있다면, 아래에 놓인 컨테이너를 반출하기 위하여 위의 컨테이너 -나중에 반출할- 를 임시로 이동시켜야 하는 작업이 필요하며 이를 재취급(rehandling)이라고 한다. 이런 재취급은 컨테이너 터미널 생산성을 저해하므로 장치장 내부로 컨테이너를 반입 할 때 가능하다면 재취급이 발생하지 않도록 적재하는 것이 좋다. 그러나 컨테이너의 장치장 반입은 컨테이너가 장치장에 무작위로 도착하는 순서에 따르므로 전혀 재취급이 발생하지 않도록 배치하는 것은 불가능하다고 볼 수 있다. 장치장 반출시 발생할 수 있는 재취급을 피하기 위하여 선박이 도착하기 전에 컨테이너들을 미리 정리하는 작업을 재정돈이라고 하며 본 논문에서 다루고자 하는 주제이다. 컨테이너를 저장할 수 있는 공간을 슬롯이라 하는데 보통 베이들이 모두 차 있는 것이 아니므로 필요시 이웃한 베이의 빈 슬롯을 활용하는 것이 가능하다. 본 논문에서는 이웃한 베이의 빈 슬롯을 활용하여, 어떠한 장치상태에서도 허용할 만한 시간 내에 재정돈 계획을 수립할 수 있는 방안을 제시한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 연구를 소개하고, 3장에서는 본 논문에서 제안하는 알고리즘에 대하여, 4장에서는 실험결과와 분석에 대하여 기술한다. 마지막으로 5장에서는 결론과 향후 과제를 언급한다.

† Corresponding author : 종신회원, ykpark@kaya.ac.kr 055)330-1136

2. 관련 연구

장치장과 관련하여 수행되는 작업으로 수출컨테이너의 경우 크게 장치장 내부로의 반입작업, 이적작업 그리고 적하작업으로 이루어져 있으며 이런 작업들을 효율적으로 수행하기 위한 연구가 진행되어 왔다. 먼저 장치장에 컨테이너를 반입하는 과정에서 재취급을 줄이는 방안에 관한 연구로 Kang(2004b)은 반입 컨테이너의 무게에 대한 정보를 활용하여 재취급을 줄이기 위한 장치 위치를 결정하는 방안을 제안하였다. 이 방안으로 장치할 경우, 무게에 대하여 감안하지 않고 무작위로 장치할 때 보다 효과적인 것으로 나타났다. Park(2011)은 장치장에 컨테이너를 반입하는 과정에서 선처리를 수행함으로써 장치장에서 반출하는 시점에 재취급이 발생할 가능성을 낮추는 두 가지 방안을 제안하였으며 시뮬레이션을 통하여 그 효과를 증명하였다. 컨테이너들의 장치장 반입이 완료된 후부터 선박에 선적하기 전까지의 시간을 활용하여 재취급을 방지하는 방안에 대한 연구들도 많았는데 먼저 한 베이 내의 컨테이너들에 대하여 실시하는 재정돈에 대한 연구로 Kang(2004a)은 분지한계법을 이용하여 컨테이너 이동 계획을 수립하는 방안을 제시하였다. 이 논문에서 최소 컨테이너의 이동횟수인 하한 값을 효과적으로 계산하는 방안도 함께 소개하였다. Ha(2012)는 최소 크기의 재정돈 계획인 최단 길이를 가지는 이적 순서를 도출하는 방안을 제안하였다. 이 연구에서는 A*알고리즘을 사용하여 상대적으로 짧은 시간에 베이 내의 이동으로 최적해를 도출하는 방법과 최소이적횟수의 하한을 구하는 방법도 함께 제안하였다. 본 논문에서 제안하는 방안도 A*알고리즘을 바탕으로 재정돈 계획을 수립하지만 Ha(2012)와 다른 점은 다음과 같다. 첫째 베이 내의 이동만이 아닌 이웃한 베이로의 이동을 포함한다는 것이고, 둘째는 베이 내의 이동으로 구하지 못한 해를 구한다는 점이다. 그 외 차이점은 해당하는 곳에서 언급한다. 그리고 선박에 선적하기 전의 시간에 블록내 이적을 통하여 재취급을 방지하는 방안에 대한 연구로 Kang(2005)는 하나의 크레인으로 블록내에 흩어져 있는 컨테이너들을 재취급이 발생하지 않게 한 곳에 모으는 방안을 제안하였으며 Oh(2005)은 복수 크레인으로 블록 내에서 컨테이너들을 이적할 때 효율적으로 수행할 수 있는 휴리스틱을 제안하였다. 서로 교차가 불가능한 복수 크레인을 이용하여 블록내의 컨테이너들을 이적할 때 크레인들 간의 간섭에 의해 발생할 수 있는 지연을 최소화하는 이적 계획을 수립하는 방안이었다. 크레인의 유휴시간을 이용하는 제약 없이 재정돈을 실시하는 방안에 대한 연구도 있었다. Kim(2014)은 효율적인 장치장 운영을 위하여, 본 작업 중이라도 기회가 있을 때마다 재정돈을 수행하도록 크레인 작업을 할당하는 방안을 제안하였다. 시뮬레이션을 통하여 제안하는 방안이 터미널 생산성에 효과가 있음을 보였다. 그 외에도 Bae(2006)는 수직형 블록을 대상으로 이적작업을 계획하는 방안을 장치위치 할당문제와 장비 작업순서 문제로 분할하여 모형화하였으며 이를 위하여 혼합정수계획법과 동적계획법을 사용하였다.

재정돈에 대한 기존의 연구에서 제안하는 방법들은 컨테이너 이동 범위를 해당 베이 내로 제한하는 방안들이며 이 방안들은 단순한 베이에 대하여 이동시간과 이동비용이 적게 드는 장점은 있으나, 복잡한 문제에 대해서는 해를 찾기 위하여 너무 많은 시간을 소요한 후 해결하지 못하는 경우가 많았다. 그래서 본 연구에서는 이웃한 베이를 이용함으로써 복잡한 문제에 대한 해를 찾는 알고리즘을 제안한다.

3. 제안 알고리즘

본 장에서는 먼저 재정돈 문제와 관련된 용어를 먼저 소개한 후 제안하는 재정돈 방안에 대하여 설명한다. 장치장은 앞의 설명처럼 블록이 여럿 모인 구조로 되어 있으며 각 블록은 다시 여러 베이들로 이루어져 있다. 베이란 Transfer Crane이 움직이지 않고 작업할 수 있는 컨테이너들의 묶음으로 아래 Fig. 1에 나타내었다. Fig. 1에는 4단 5열의 구조를 가진 베이를 표현하였는데 흰색 베이 1개와 회색 베이 2개를 나타내었다. 흰색 베이에서 재정돈 작업을 수행할 경우, 흰색 베이 주위의 베이들 (Fig. 2의 회색 베이들)이 이웃한 베이들이 된다. 이웃한 베이들이 같은 규격의 컨테이너들을 적재하고 있다면 재정돈 작업을 할 때, 이웃한 베이들 상단의 빈 슬롯에 컨테이너를 임시로 이동할 수 있다.

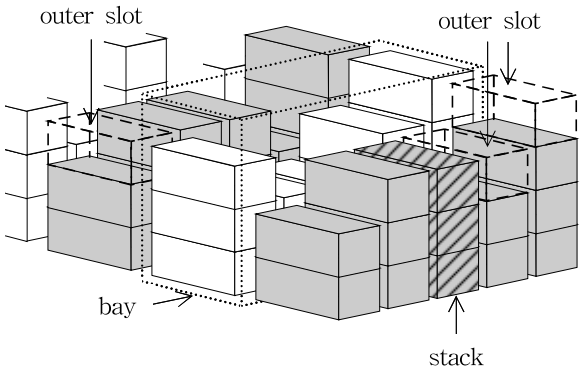

Fig. 1 Example of bay, stack and outer slot

베이 내부의 빈 슬롯으로 이동하는 것 보다 베이 외부의 빈 슬롯(이하 외부 슬롯으로 기술함)으로 이동할 경우, (Transfer Crane의 베이간 이동이 발생하므로) 이동 시간과 비용이 더 소요되나, 외부 슬롯을 이용함으로써 베이 내부의 빈 슬롯만 이용하여 해결할 수 없는 재정돈 문제들을 해결할 수 있게 된다. 외부 슬롯은 같은 사양의 컨테이너가 적재된 이웃 베이 상단의 빈 슬롯 중 임시 저장을 위하여 선택한 것이기 때문에, 여러 개의 외부 슬롯이 있을 수 있다. 예를 들면 Fig. 1에서 이웃한 베이들이 같은 사양의 컨테이너를 적재한 베이들이라

면, 3개의 외부슬롯을 선택해야 할 경우, 앞쪽 회색 베이의 빈 슬롯 7개 중 2개, 뒤쪽 회색 베이의 빈 슬롯 6개 중 1개를 외부 슬롯(점선상자로 표현)으로 사용할 수 있을 것이다. Fig. 2에는 Fig. 1을 위에서 본 모습을 나타내었다. Fig. 1의 흰색 베이에 Transfer Crane이 위치하고 있다고 가정한 그림이다. 회색 베이들이 앞에서 말한 것처럼 이웃한 베이들을 나타낸다.

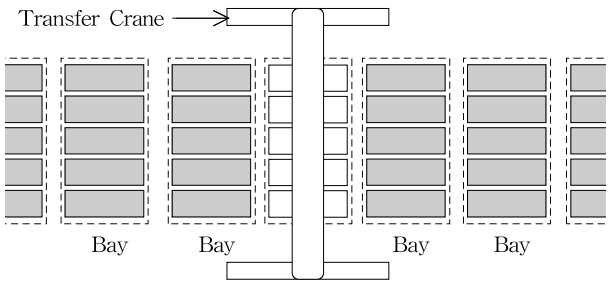

Fig. 2 Example of block

베이에 장치된 컨테이너들이 장지장에서 반출되는 순서는 우선순위를 사용하여 나타낸다. 높은 우선순위의 컨테이너가 낮은 우선순위보다 먼저 반출되며, 높은 우선순위에 작은 값의 인덱스를 부여하는 방식으로 표현한다. 여러 개의 컨테이너를 한 스택 위에 있는 여러 외부 슬롯들에게 적재할 경우, 아래쪽 인덱스가 위쪽보다 같거나 작은 순서로 적재해야 한다. 이는 다음에 설명하는 알고리즘의 세 번째 파트에서 인덱스를 기준으로 외부슬롯에 있는 컨테이너를 내부로 이동하기 때문이다. 내부 부적합 컨테이너는 베이 내의 각 스택에서 자신의 아래에 자신보다 높은 우선순위(작은 값의 인덱스)의 컨테이너가 하나라도 있는 컨테이너들로 재정돈을 위해서 반드시 재배치되어야 하는 컨테이너들이다. 외부 부적합컨테이너는 외부 슬롯으로 이동한 컨테이너들 중 베이 내로 이동할 때 어떤 스택으로 이동하여도 내부 부적합 컨테이너가 되는 컨테이너를 말한다. 선택 재배치 컨테이너는 베이 내의 컨테이너들 중 내부 부적합 컨테이너는 아니지만 재정돈을 위하여 필요에 의해 재배치되는 컨테이너를 의미한다. 아래 Fig. 3에서와 같이 베이 내에 7개의 컨테이너가 외부에 2개의 컨테이너가 놓여 있을 경우, 베이 내부에 빗금 친 컨테이너가 내부 부적합 컨테이너를 외부슬롯에 빗금 친 컨테이너가 외부 부적합 컨테이너를 나타낸다. 그림에서 작은 사각형은 컨테이너를, 내부 숫자는 인덱스를 나타내는데 외부슬롯의 3을 베이내로 이동을 한다면, 어떤 스택으로 이동을 하더라도 부적합 컨테이너가 된다. 반면에 외부슬롯의 1은 내부의 2번째나 4번째 스택에 적재할 경우 부적합 컨테이너가 되지 않는다. 내부 부적합컨테이너를 모두 들어내더라도 재정돈을 위해서는 추가로 컨테이너를 (스택4의 1이나 스택 2의 2 컨테이너들 중 선택하여) 더 들어내어야 하는데 이런 컨테이너를 선택 재배치 컨테이너라고 한다.

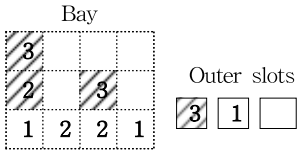

Fig. 3 Bay and outer slots

본 논문에서 제안하는 방법은 A*알고리즘(Nilsson, 1998)을 바탕으로 재정돈 계획을 수립하는(해를 찾는) 방안으로, 너비우선탐색과 비슷하지만 문제의 특성에 대한 정보를 바탕으로 가장 좋은 경로 상에 있다고 판단되는 노드를 우선 방문하는 탐색방법이다. 여기서 노드는 베이를 나타낸다. A*알고리즘에서는 평가값을 사용하여 노드들을 평가하는데 평가값은 평가함수 f(n) = g(n) + h(n)로 구한다. g(n)는 깊이 추정값, 즉 시작 노드에서 노드 n까지의 최단 거리에 대한 추정값이고, h(n)은 노드 n에 대한 휴리스틱 추정값으로 노드 n에서부터 목표노드까지의 최단 거리에 대한 추정값을 나타낸다. f(n)은 베이의 초기상태인 시작노드에서 재정돈을 완성하는 상태 즉 목표노드까지 탐색하기 위해 이동하는 컨테이너의 개수를 의미하며, g(n)은 시작 노드에서 노드 n까지 이동한 컨테이너의 개수를 의미한다. h(n)은 노드 n에서 목표노드까지 앞으로 이동해야할 컨테이너의 개수를 의미한다. 본 논문에서 h(n)은 내부 부적합 컨테이너의 수, 외부 부적합 컨테이너 수 그리고 최소 선택 재배치 컨테이너의 수의 합으로 구한다.

알고리즘의 각 파트에 대한 구체적인 절차는 다음과 같다. 이 절차들은 Nilsson(1998)과 Ha(2012)에 기술된 알고리즘을 참조하여 기술한다. 본 제안알고리즘과 Ha(2012)의 알고리즘의 차이는 첫째, 아래 첫 번째 파트와 세 번째 파트가 Ha(2012)에는 없으며, 둘째 OPEN 리스트들을 정렬하는 방식이 다르다는 점이다. Ha(2012)의 알고리즘에서의 정렬기준은 평가함수 f(n)의 값이지만 제안 알고리즘은 h(n)이다. 외부슬롯과 관련되는 부분들도 Ha(2012)의 알고리즘과 다른 부분들이다.

[첫 번째 파트]

1. 시작노드만 들어 있는 OPEN1 리스트와 비어있는 OPEN2 리스트(OPEN2 리스트는 두 번째 파트에서 사용된다.)와 CLOSED 리스트를 준비한다.
2. OPEN1이 비게 되면 두 번째 파트로 간다.
3. OPEN1의 제일 앞에 있는 노드 n을 꺼낸 후 다음을 수행한다. 노드 n에 빈 외부슬롯이 없으면 그 노드를 정렬기준에 따라 OPEN2에 삽입하고 단계2를 다시 한다. 빈 외부 슬롯이 있으면 CLOSED에 삽입한다.
4. 노드 n에서 '자식노드 생성방법(아래 설명)'으로 자식노드의 집합 M을 만든다.
5. M에 포함된 각각의 노드 m에 대하여 다음을 수행한다. m과 동일한 노드가 OPEN1과 CLOSED에 없

으면 m을 OPEN1에 삽입한다. m과 동일한 노드m'가 OPEN1에 있고 m의 평가값이 m'의 평가값보다 작으면, m'를 m으로 대체한다.

6. OPEN1와 OPEN2에 노드들은 휴리스틱 추정값(h(n))에 따라 오름차순으로 정렬한다. 이때 동일한 휴리스틱 추정값을 가지는 노드들은 깊이 추정값에 따라 내림차순으로, 동일한 깊이 추정값을 가질 때는 부적합컨테이너의 인덱스 합에 따라 오름차순으로 정렬한다.
7. 단계2로 간다.

[두 번째 파트]

1. OPEN2에서 제일 앞에 있는 노드n을 꺼낸 후 CLOSED에 삽입한다.
2. 노드n이 내부 부적합 컨테이너와 외부 부적합 컨테이너가 없는 노드이면 해를 발견한 것이며 세 번째 파트로 넘어간다.
3. 노드 n에서 가능한 이적을 적용하여 자식노드의 집합 M을 만든다.
4. M에 포함된 각각의 노드 m에 대하여 다음을 수행한다. m과 동일한 노드가 OPEN2와 CLOSED에 없으면, m을 OPEN2에 삽입한다. m과 동일한 노드 m'가 OPEN2에 있고 m에서의 평가값이 m'의 평가값보다 작으면, m'를 m으로 대체한다.
5. OPEN2가 비게 되면 세 번째 파트를 실행할 필요 없이 실패로 끝난다.
6. OPEN2에 노드들은 휴리스틱 추정값에 따라 오름차순으로 정렬한다. 동일한 휴리스틱 추정값을 가질 때는 깊이 추정값에 따라 오름차순으로 정렬한다.
7. 단계2로 간다.

[세 번째 파트]

두 번째 파트가 정상종료가 되었을 경우 실질적인 해를 찾은 것이 되며, 세 번째 파트에서는 마지막 정리 작업만 수행한다.

1. 노드n에서 외부슬롯에 있는 컨테이너들 중 우선순위가 가장 낮은 컨테이너를 찾는다.
2. 그 컨테이너를 적재할 수 있는 스택을 찾아 그 스택으로 이동한다.
3. 외부슬롯의 노드가 모두 이동이 되면 알고리즘은 정상 종료한다.
4. 단계1로 간다.

첫 번째 파트 4단계에 명시한 '자식노드 생성방법'에 대하여 설명한다. 여기서는 외부슬롯으로 컨테이너를 이동시켜서 자식노드를 생성하는데 이때 옮길 컨테이너를 선택하는 기준은 다음과 같다.

1. 내부 부적합 컨테이너
2. 외부 부적합 컨테이너 수를 줄일 수 있는 컨테이너

1번 기준으로 내부 부적합 컨테이너를 외부로 옮기면 Fig. 6의 4)에서처럼 적합 컨테이너로 바뀔 가능성이 있다. 2번 기준은 Fig. 6의 3)의 예처럼 외부 슬롯에 있는 부적합 컨테이너의 수를 줄일 수 있으며, 외부로 이동하는 적합 컨테이너의 우선순위가 낮을수록 효과가 크다. n_e을 비어있는 외부 슬롯의 개수, m_j을 스택 j에 적재된 부적합 컨테이너 수 그리고 w_{ji}을 스택 j에 적재된 적합 컨테이너들 중 인덱스 i보다 낮은 값의 인덱스(높은 우선순위)의 컨테이너 수라고 하면 i값을 최댓값의 인덱스(가장 낮은 우선순위)로 설정한 후 m_j+w_{ji}와 n_e 중 낮은 수만큼 외부슬롯으로 옮겨서 자식노드를 생성한다. 이것이 첫 번째 파트에서 자식노드 생성하는 방법이다. 두 번째 파트에서는 단순히 하나의 노드에서 하나의 컨테이너 이적으로 가능한 모든 자식노드를 생성한다. 세 번째 파트에서는 자식 노드를 생성하는 과정이 없이, 두 번째 파트에서 찾은 해에 해당하는 노드에서 외부 슬롯에 있는 컨테이너를 내부로 옮기는 마지막 정리 작업만 수행하게 된다. 첫 번째 파트에서 한꺼번에 여러 컨테이너를 한꺼번에 이적하여 자식노드를 생성하는 방법과 두 번째 파트의 정렬기준을 첫 번째 파트의 정렬기준과 다르게 적용하는 것 모두 해를 구하기 위해 너무 많은 시간을 소비하는 것을 피하기 위함이다.

알고리즘을 실행할 때 하나의 노드로 자식노드를 많이 생성하게 되므로 기존의 노드와 동일한 노드 생성을 막는 것이 중요하다. 장치형태가 달라도 동일한 노드인지 판단하기 위하여 노드를 인덱스에 따라 사전순으로 정렬한 형태로 만드는 것을 정규화라하는데 이 방법은 Ha(2012)에서 사용되었다. 예를 들면 아래의 첫 베이를 2-0-0, 1-2-0, 2-3-0로 표현하고 사전순으로 정렬하여 1-2-0, 2-0-0, 2-3-0으로 바꾸어 마지막 노드처럼 표현하는 것을 정규화라한다. (빈슬롯은 0), 정규화로 표현하면 좌측 4개의 노드는 모두 오른쪽 노드가 된다.

Fig. 4 Example of layout normalization

다음은 3단 4열의 베이를 대상으로 알고리즘을 수행하는 과정을 나타내었다. 각각의 표는 베이를 나타내며, 표안의 숫자는 컨테이너의 인덱스를 나타내고, 표 위에 숫자는 노드 번호, g=깊이추정값, h=휴리스틱추정값을 나타내는데 노드번호는 노드가 생성되는 순서를 의미한다. g값은 시작노드에서 그 노드가 될 때까지 이동한 컨테이너 수가 되고, h값은 앞으로 이동할 컨테이너 수로, 부적합 컨테이너수라고 보면 되는데 빗금친 칸으로 부적합 컨테이너를 나타내었다. 각각의 표아래

의 상자 두개는 외부슬롯을 나타내며 노드간의 화살표는 자식 노드 생성을 의미한다. 자식노드 생성은 컨테이너이동으로 생성하게 되는데 화살표 중 굵은 선은 해를 구하는 과정을 의미한다. 가로선(파선)들은 첫 번째 파트외 두 번째 파트의 경계선과 두 번째 파트와 세 번째 파트의 경계선을 나타낸다.

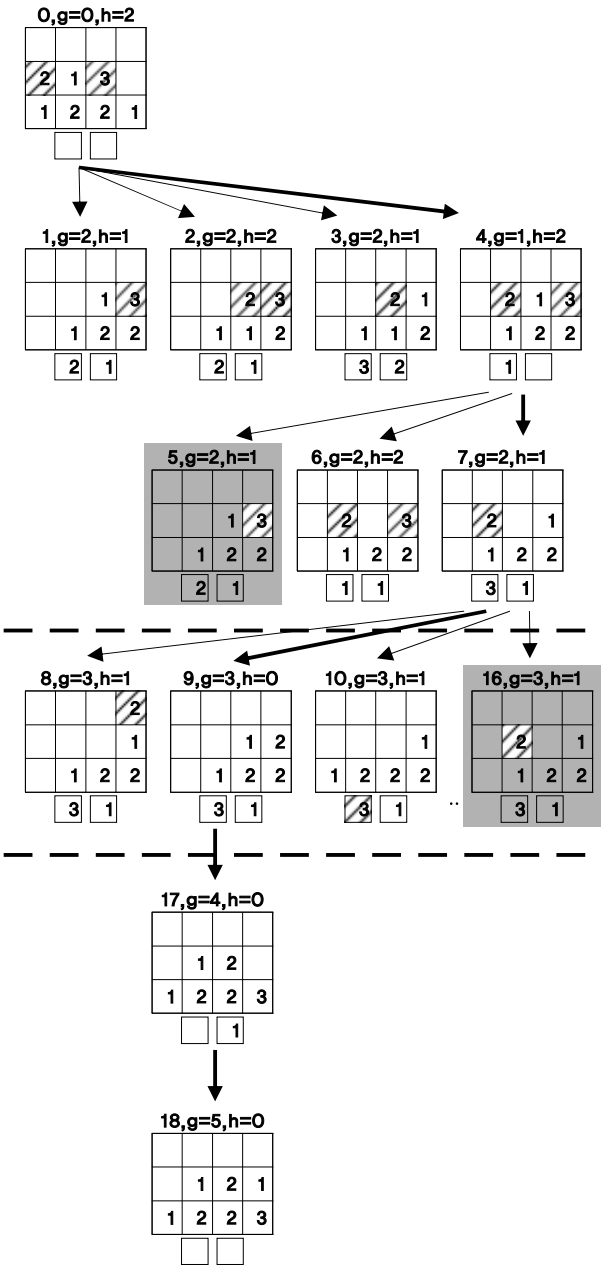

Fig. 5 Example of algorithm execution

OPEN2 리스트와 CLOSED 리스트는 비어있고, 노드0만 OPEN1 리스트에 있는 상태로 알고리즘을 시작한다. 먼저 노드0를 꺼내서 CLOSED 리스트에 삽입한다. 노드 0는 컨테이너 이동이 없었으므로 g=0이고 부적합컨테이너가 2개이므로 h=2가 된다. 노드0를 가지고 첫 번째 파트에서 언급한 '자식노드 생성방법'을 따라 자식노드를 만들게 되는데, 노드1은 노드0의 스택1에서 컨테이너2개를 외부슬롯으로 이동시켜서 만든 후 정규화한 자식노드이다. 빈 외부슬롯이 없으므로 노드1을 OPEN2에 삽입한다.(위의 첫번 째, 파트 단계3참조)

노드를 삽입할 때 CLOSED와 OPEN1에 동일한 노드가 있는지 확인하고, CLOSED에 동일한 노드가 있으면 이미 자식노드를 생성한 적이 있는 노드와 동일한 노드이므로 삽입을 하지 않고 생성된 자식노드를 삭제를 한다. OPEN1에 동일한 노드가 있으면 두 노드의 평가값(g값과 h값의 합)을 비교하여 낮은 값을 가지는 노드를 리스트에 남겨둔다. 노드2는 노드0의 스택2에서 컨테이너2개를 외부슬롯으로 이동시켜서 만든 후 정규화한 자식노드이다. 노드3도 같은 과정을 거친다. 노드4는 노드0의 스택4의 컨테이너가 1개 뿐이므로 컨테이너 1개를 외부슬롯으로 이동시킨 후 정규화한 노드이다. 빈 외부슬롯이 있으므로 노드4는 OPEN1에 삽입된다.

노드0를 가지고 자식노드를 생성하는 과정을 마쳤으므로 다시 OPEN1에서 다음 노드인 노드4를 꺼내서 CLOSED에 삽입한 후 외부슬롯을 하나 더 채우면서 자식노드들 5,6,7을 생성하게 된다. 노드 5,6,7도 외부슬롯이 채워졌으므로 모두 OPEN2에 삽입된다. 노드 5를 삽입할 때 동일한 노드1이 이미 OPEN2에 있고 두 노드의 평가값(g=2, h=1)이 같으므로 노드5는 삭제된다. OPEN1이 비었으므로 첫 번째 파트는 종료된다. 이 시점에 OPEN2에는 노드 1,2,3,6,7이, CLOSED에는 노드0,4가 있게 된다. OPEN2에서는 h값을 기준으로 오름차순으로 정렬되어 있는데, 노드1,3,7의 h=1이므로 처음에, 노드2,6의 h=2이므로 그 다음에 정렬되어 있다. 모든 노드들이 2개의 컨테이너를 옮겼으므로 g=2로 같다. 두 번째 파트에서는 OPEN2에서 노드를 선택한 후 컨테이너 하나를 옮기면서 만들 수 있는 자식노드를 모두 생성하는 과정을 거치게 된다. 노드 1,3,7중 하나가 임의로 선택되는데 여기서는 노드7을 가지고 자식노드를 생성하는 과정을 그림에 나타내었다. 컨테이너 하나를 이동시켜 만들 수 있는 자식노드를 모두 만든 후 h값을 기준으로 OPEN2에 정렬하면서 삽입하게 된다. 노드11에서 노드15의 자식노드 생성은 생략하였는데, 자식16의 노드가 노드 7과 같고 평가값이 크므로 OPEN2 에 삽입하지 않고 삭제한다. 노드7의 자식노드생성이 끝난 후 OPEN2에서 다음 노드를 선택하면 (9의 h값이 0이므로 OPEN2의 가장 앞에 정렬되어 있다) 노드 9가 선택된다. 앞으로 이동해야할 컨테이너 개수인 h값이 0이므로 노드9가 찾는 해가 되고 두 번째 파트가 종료된다. 마무리 작업을 수행하는 세 번째 파트를 시작하여 외부슬롯에 있는 컨테이너들을 높은 값의 인덱스 3부터 베이 내부로 옮기게 되는데 옮길 수 있는 스택을 찾아서 모두 이동을 하면 세 번째 파트도 종료하게 된다.

다음은 3단 4열의 베이를 대상으로 제안 알고리즘을 수행하여 해를 찾은 후, 해와 관련된 노드만 정규화를 하지 않은 형태로 나타내었다. 부적합 컨테이너들은 사선으로 표시하였으며, 각각의 스텝에서 변하는 내부 부적합 컨테이너 수와 외부부적합 컨테이너 수를 기록하였다.

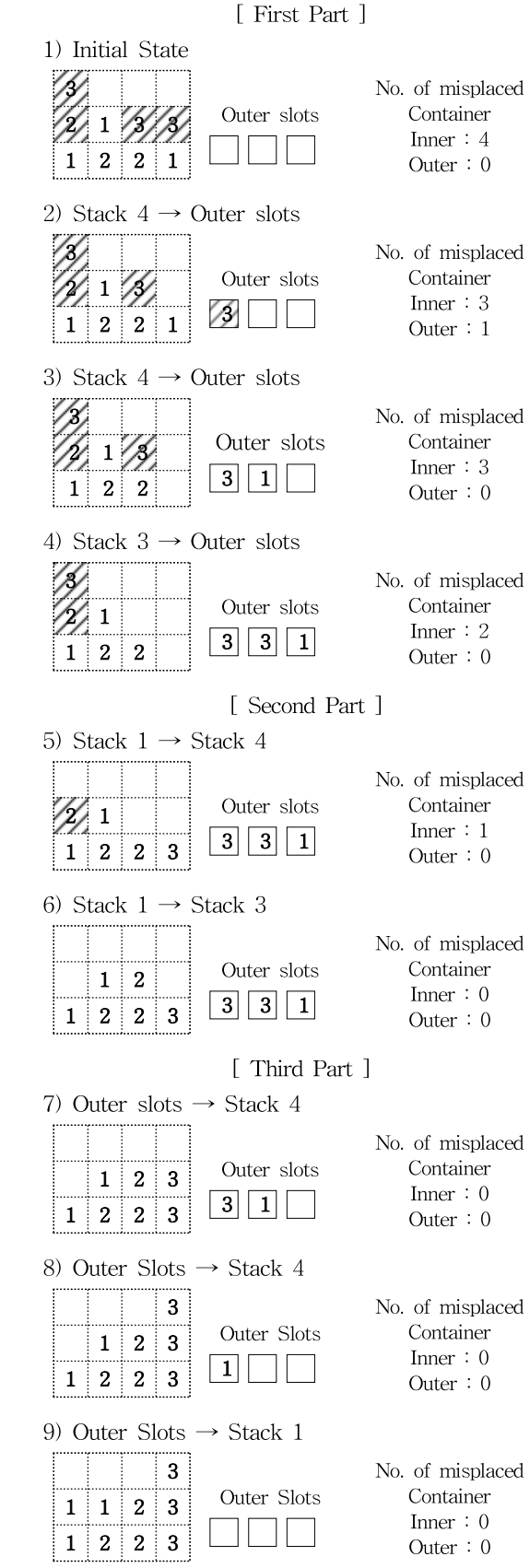

Fig. 6 Example of finding solution

위의 예의 3)에서 외부 부적합 컨테이너 3이 적합 컨테이너로 변한 이유는 컨테이너1을 외부로 옮긴 결과, 외부 컨테이너3을 스택4에 옮긴다면 부적합 컨테이너가 되지 않기 때문이다. 4)에서는 컨테이너 3이 베이 내부에서는 부적합 컨테이너였으나 외부로 옮김으로 해서 적합 컨테이너가 되는 것을 알 수 있다. 3)이 수행되고 난 후 외부 부적합 컨테이너 수가 0이 되었으므로 외부 슬롯에 있는 컨테이너를 내부로 옮겨도 문제가 없는 상태가 되었으며 6)에서 내부 부적합 컨테이너의 수도 0개가 되었기 때문에 두 번째 파트가 종료될 수 있었다.

4. 실험 결과 및 분석

제안한 방안의 성능을 분석하기 위하여 Microsoft Visual C++ 2010 Express 환경에서 알고리즘을 구현하였으며, Intel Core i5-2400 CPU 3.10GHz, 4GB RAM 하드웨어와 Window7(32bit)운영체제상에서 실험을 하였다. 먼저 베이 내 재정돈으로 해결할 수 있는 문제에 대하여 기존의 알고리즘과 성능을 비교한다. 다음으로, 베이내 이동으로 해를 구할 수 없는 복잡한 문제를, 허용할 만한 시간에 해결하는 것을 입증하기 위한 실험 결과를 보인다.

4.1 기존의 연구와 비교

제안하는 알고리즘의 성능을 기존의 연구결과와 비교하기 위하여 기존의 연구에서 성능 비교가 의미 있다고 파악된 세 개의 문제를 대상으로 실험을 하였다. Table1에서 Table3까지 세 개의 문제를 나타내었으며 이 베이들은 베이내 재정돈으로 해를 구할 수 있는 문제들이므로 베이내 재정돈 방안이 더 효과적이다. 제안알고리즘은 외부슬롯 수를 1로 두고 실험을 하였으며, 이동수는 재정돈을 수행하기 위한 컨테이너의 이동수를 의미한다.

Table 1 Bay P1

			3	1	
3	2	1	1	3	2
1	1	3	2	1	1

Table 2 Bay P2

1								4	
3		7			0	6		6	5
3	0	6		5	5	7		9	5
3	6	5	2	8	0	5		6	1
5	8	1	5	8	2	8	2	1	4

Table 3 Bay P3

		4				5		6			
2		2		3		1	1	5			3
2	5	6		4	1	5	2	3	6	6	5
4	1	3	5	6	6	2	6	1	3	4	4
4	2	6	3	2	1	6	1	5	2	1	6

Table 4의 결과를 보면 제안알고리즘으로 구한 해의 이동수(Size of Plan)가 Ha(2012)의 것보다 약간 많은 것은 베이내 이동으로 해를 구할 수 있는 베이를 대상으로 실시한 실험이라서 베이내 재정돈 방안이 효율적일 수 있기 때문이다. 본 논문에서 제안하는 알고리즘은 베이내 재정돈으로 구할 수 없는

베이에 대하여 해를 구하는 것이 목적이다. 그리고 실험결과에서 이동수나 실행시간(CPU time)에 큰 차이가 나지 않으면서, P3의 경우 방문한 노드 수(Visits)가 현격하게 줄어든 것을 보면 제안한 알고리즘도 효율적인 알고리즘이라는 것을 알 수 있다.

Table 4 Comparison against previous study

	(Ha, 2012)			Our Results(Outer Slot = 1)		
	Size of Plan	Visits	CPU Time(Sec)	Size of Plan	Visits	CPU Time(Sec)
P1	9	10	< 0.1	9	10	< 0.1
P2	15	24	< 0.1	17	17	0.10
P3	30	1,369	< 1.9	34	50	1.96

Table 5 Solutions of the problem in Table 4

	Solution
P1	4→outer slot, 5→4, 3→4, 1→3, 5→3, 1→5, 2→1, 6→1, outer slot→3
P2	10→outer slot, 10→5, 2→5, 7→2, 6→10, 8→4, 7→8, 6→2, 9→2, 3→8, 9→8, 3→8, 3→8, 3→7, 9→3, 9→3, outer slot→3
P3	8→outer slot, 8→1, 4→2, 12→4, 5→4, 6→4, 12→2, 3→12, 5→12,3→12, 8→6, 8→4, 3→8, 5→8, 7→8, 7→5, 7→8, 2→8, 10→2, 10→3, 7→10, 9→7, 9→7, 9→3, 5→3, 5→10, 2→5, 2→7, 2→7, 6→5, 6→5, 11→5, 11→5, outer slot→6

Table 5에는 각 문제에 대한 실험인 이적순서를 나타내었다. 숫자는 스택의 좌측을 1로 부여한 번호를 의미한다.

4.2 복잡한 문제에 대한 실험 결과

기존의 베이내 정돈으로 해결할 수 없는 복잡한 문제들에 대하여 제안알고리즘이 재정돈 계획을 수립할 수 있음을 보이기 위하여 최대로 복잡한 베이를 대상으로 실험을 하였다. 장치율이 높을수록, 같은 장치율일 경우에는 인덱스가 커질수록 문제가 복잡해지므로 장치율 80%인 8단 6열, 8단 8열, 8단 10열, 8단 12열 구조의 베이들을 대상으로 최대 인덱스 값을 갖는 경우도 포함하여 실험을 한다. 이 베이들은 Ha(2012)에서 해를 전혀 구하지 못한 복잡한 문제 베이들이다. 8단을 대상으로 실험한 것은 가장 복잡도가 높은 베이를 선택하기 위해서이며 다양한 실험을 위하여 열을 6열에서 12열로 변화시켰다. 실험을 위하여 베이 자료 생성은 컨테이너 도착 순서를 반영하여 인덱스를 무작위로 생성하였으며, 인덱스 값이 컨테이너 수와 같은 경우에는 모든 컨테이너의 인덱스를 다르게 만들었다. 실험은 아래 실험방법으로 실시하였다.

[실험 방법]

1. 베이마다 정해진 수의 컨테이너를 생성한다.
2. 각 스택마다 할당하는 컨테이너 수를 무작위로 설정한다.
3. 각 컨테이너의 인덱스를 정해진 범위내의 값으로 무작위로 설정한다.
4. 이런 베이를 100개 생성하여, 실험을 수행한 후 그 평균값을 구한다.

각 실험에서 방문하는 노드수를 제한하고 그 수를 초과하는 경우 실패로 간주함으로써, 허용할 수 있는 시간 내에 그 결과가 나오도록 하였으며 실험결과를 설명하는 곳에 미 해결 시 최대 실행시간을 명시하였다. 아래의 Table들의 수치는 성공한 실험만을 대상으로 기록하였으며, Average CPU Time은 실행시간으로 단위는 초이며, Average Visits는 해를 구하기 위하여 방문한 노드의 평균이다. 먼저 Table 6은 8단 6열의 베이로 외부슬롯을 6으로 설정한 후, 인덱스가 성능에 미치는 영향을 보기 위하여 실시한 실험이다. 마지막의 인덱스 38은 컨테이너 수와 같은 조건, 즉 모든 컨테이너의 인덱스가 다른 베이이다. 방문노드 수를 10,000개로 제한하였으며, 이 실험에서 실패한 경우를 보면, 최대실행시간이 660.8초였다. 그리고 해를 구하는데 10초를 초과하는 경우가 거의 없다는 것을 알 수 있다. 인덱스가 커지면 성공률도 떨어지고 방문노드수가 높아지는 것을 보아 인덱스가 성능에 미치는 영향을 알 수 있다.

Table 6 Experimental results according to change of index

No. of Indices	10	15	20	25	38
No. of Success	93	85	73	72	53
Average CPU Time	2.02	10.11	7.26	8.83	9.64
Average Visits	737.3	1510.6	1419.9	1479	1608.1

다음은 외부슬롯이 성능에 미치는 영향을 알아보기 위하여 실시한 실험이다. 위의 실험과 같은 8단 6열의 베이를 대상으로 인덱스 수를 10으로 설정한 상태에서 외부슬롯 수를 3에서 15까지 변화하면서 실시하였다.

Table 7 Experimental Results according to change of outer slots

No. of outer slots	3	6	9	12	15
No. of Success	92	93	98	98	100
Average CPU Time	0.5	2.01	1.05	1.07	2.32
Average Visits	492.2	737.3	297.2	297.2	199.6

이 실험에서는 실패한 경우의 실행시간은 최대 511.9초였으며, 실험 결과를 보면 외부슬롯의 수를 증가하면 성공률이 높아지다가 결국 100%에 도달하게 되는 것을 알 수 있다. Table 6에서의 실험처럼 인덱스 수가 커지거나, (실험데이터를 제시하지는 않았지만)컨테이너가 많아지게 되면 성공률이 떨어지게 되지만, 외부슬롯 수를 증가하면서 해를 구하면, 성공률 100%에 도달하는데 문제가 없다는 것을 알 수 있는 실험이다.

Table 8은 가장 어려운 문제 베이들 3개에 대하여 외부슬롯수를 증가하는 방법으로 해를 구한 실험 결과이다. 장치율 80%, 모든 컨테이너의 인덱스가 서로 다른 조건(컨테이너 수와 같은 인덱스)의 베이들을 대상으로 최종 해를 구한 결과이다. 외부 슬롯을 증가하여 실험하는 과정을 나타내지는 않았

지만, 각 베이에 대하여 3, 4번 슬롯을 증가하는 실험으로 성공률 100%인 Table 8의 결과를 얻을 수 있었다.

Table 8 Experimental Results of difficult problem instances

No. of stacks	8	10	12
No. of tiers	8	8	8
No. of Container	51	64	77
No. of Indices	51	64	77
No. of outer slots	24	21	24
No. of Success	100	100	100
Average CPU Time	0.09	4.51	16.6
Average Visits	120.1	320	398

실험결과를 보면 100% 성공하는 해를 찾는 데에, 8단 12열의 베이의 경우는 16.6초가 결렸으나, 8열 8단의 베이의 경우 1초미만의 시간 내에 해를 찾을 수 있음을 알 수 있다. 8단의 여러 가지 열의 베이에 대하여 80%의 장치율과 모든 컨테이너의 인덱스가 다른, 복잡한 조건에 대하여 해를 100% 구할 수 있었으므로 이 베이들보다 복잡하지 않은 어떠한 베이도 해를 구할 수 있다는 것은 당연할 것이다. 그리고 이 실험보다 복잡한 베이라 하더라도 슬롯의 수를 증가시키는 방법으로 해를 구할 수 있을 것임은 충분히 짐작할 수 있다. 그래서 본 논문에서 제안하는 재정돈 방안을 사용하면 아무리 복잡한 구조의 베이라도 해를 구할 수 있음을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 이웃한 베이의 저장 공간을 외부슬롯으로 활용함으로써, 베이내의 이적만으로 재정돈 계획을 수립할 수 없는 복잡한 베이에 대하여 재정돈 계획을 수립하는 방안을 제안하였다. 그 효과를 입증하기 위하여 최대한 복잡한 베이에 대하여 실험을 실시하였고, 어떤 조건의 베이에 대해서도, 외부 슬롯수를 증가하면서 해를 찾는 방법으로 허용할만한 시간 내에 재정돈 계획을 수립할 수 있다는 것을 확인하였다. 제안한 재정돈 방안이 높은 장치율의 베이에 대하여 해를 구할 수 있으므로 현재보다 장치율을 높여서 장치장을 운영하는 부대효과가 발생할 수 있지 않을까 기대해 본다. 제안한 방안이 베이 내 재정돈 방법으로 해결할 수 없는 경우에 해를 구하는 것이 목적이지만, 이웃한 베이로의 이동이 베이간 Transfer Crane 이동을 필요로 하므로, 베이 내 이동보다 컨테이너 이동에 시간과 비용이 많이 든다는 것은 사실이다. 그리고 제안한 방법으로 구한 해가 효과적인 해이면서 짧은 시간에 구할 수 있기는 하지만 최적의 해라는 보장이 없다는 점도 지적해야할 사항이다.

향후 외부 슬롯을 활용한 재정돈 방안을 이용하여 최적의 해를 구하는 연구가 필요할 것으로 생각한다. 그리고 베이 내부의 이동만으로 재정돈 계획을 수립할 수 없는 경우, 그 판단을 빨리 할 수 있는 방안을 찾는 연구도 필요하리라 생각한다.

References

[1] Bae, J. W., Park, Y. M. and Kim, K. H.(2006), “Assignment and Operation Sequencing for Remarshalling of a Vertical Yard Block in Automated Container Terminals”, Journal of Korean Navigation and Port Research, Vol. 30, No. 6, pp. 457-464.

[2] Ha, B. H and Kim, S. S.(2012), “A* Algorithm for Optimal Intra-bay Container Pre-marshalling Plan”, Journal of the Korean Institute of Industrial Engineers, Vol. 38 No. 2, pp. 157-172.

[3] Kang, J. H., Ru, K. R. and Kim, K. H.(2004a), “Effective Method of Intra-bay Remarshalling in Container Yard”, Proceedings of The 2004 KIISS Fall Conference, pp. 287-295,.

[4] Kang, J. H., Oh, M. S., Ru, K. R. and Kim, K. H.(2004b), “Method of Inbound Container Positioning for Minimal Rehandling Considering Weight”, Proceedings of The 2004 KIISS Fall Conference, pp. 271-278.

[5] Kang, J. H., Oh, M. S., Ru, K. R. and Kim, K. H.(2005), “Sequencing Container Moves for Intra-Block Remarshalling in a Container Terminal Yard”, Journal of the Korean Institute of Industrial Engineers, Vol. 29, No. 1, pp. 83-90.

[6] Kim T. K., Yang. Y. J., Bae. A. K. and Ryu. K. R.(2014), “Optimization of Dispatching Strategies for Stacking Cranes Including Remarshalling Jobs”, Journal of Korean Navigation and Port Research, Vol. 38, No. 2, pp. 155-162.

[7] Nilsson, N. J.(1998), Artificial Intelligence : A New Synthesis, Morgan Publishers, Inc., San Francisco, California.

[8] Oh, M. S., Kwang, J. H., Yu, K. R. and Kim, K. H. (2005), “A Heuristic Approach to Scheduling Multiple Cranes for Intra-Block Remarshalling”, Journal of Korean Navigation and Port Research, Vol. 29, No. 5, pp. 447-455.

[9] Park, Y. K. and Kwak, K. S.(2011), “Export container preprocessing method to decrease the number of rehandling in container terminal”, Journal of Korean Navigation and Port Research, Vol. 35, No. 1, pp. 77-82.

Received 22 April 2016
Revised 9 June 2016
Accepted 9 June 2016