Print ISSN: 1738-3110 / Online ISSN 2093-7717
http://dx.doi.org/10.15722/jds.14.5.201605.107

# Comparison of Algorithm based on the Container Remarshalling Efficiency Factor in Port Distribution

# 항만유통의 컨테이너 재정돈 성능요인에 따른 알고리즘 성능비교

Young-Kyu Park(박영규)*

Received: May 2, 2016. Revised: May 11, 2016. Accepted: May 15, 2016.

## Abstract

**Purpose** – Loading can decrease the productivity due to the possibility of carrying out with the opposite order of storage in container terminal. When the container is to be taken out, it is needed to move the container stacked upon the container to be carried out to other place temporarily. It is called as rehandling. Remarshalling, with the loading plan, is the arranging the containers before the ship arrives in order to avoid the rehandling during the carrying out. The present study tried to find out the factors affecting the efficiency when building the remarshalling plan with the utilization of neighboring storage space as a outer slot, and analyzed the efficiency of several remarshalling algorithms with the combination of those factors.

**Research design, data and methodology** – The present study used, when the remarshalling plan is prepared for utilizing the outer slot, the simulation methods in order to compare the efficiency of the remarshalling algorithms which made with the factors affecting the efficiency. The factors affecting the efficiency are the method of making the child node, method of arrangement, and possibility of application of FIX. In order to analyze the affecting factors on the efficiency, several algorithms are prepared with the combination of production of the child node and the arrangement method with the availability of FIX application. With this algorithm, the effect of the factors on the efficiency after building up of remarshalling plan with the target on the bay with 10 rows, 8 columns, and 10 indices.

**Results** – The method of rearrangement and making of a child node as the factor affecting the efficiency of remarshalling utilization of the outer bay were studied. It is efficient to combine the method of making a child node with MCS in order to reduce the number of moving the containers. For reducing the time in carrying out, it was found that all arrangement methods should be combined with RCS for the efficiency. The result of experiment shows the application of FIX with good result in case of succession ratio. In addition, when FIX was not applied, all of the most combinations resulted in short time in remarshalling. As a result, it can be concluded that the algorithm with proper combination of making the child node and the arrangement can increase the job efficiency based on the importance.

**Conclusion** – The present study suggested and analyzed the algorithms with the combination of the arrangement method, the making of child node, and FIX. It is needed to develop the algorithm to judge the possibility whether the best remarshalling plan can be built or not within the bay in order to find a better method between the two cases such as within the bay and outer the bay. As a method for extending the study on the factors affecting the efficiency, it is possible to find out the way to build the remarshalling plan within the permitted time under any storage situation.

**Keywords:** Remarshalling Plan, Performance Factor, Neighboring Bay, Outer Slot, Port Distribution.

**JEL Classifications:** C61, C88, L91, R49.

## 1. 서론

세계 물동량증가는 컨테이너 터미널수의 증가와 더불어 터미널 간의 치열한 경쟁을 초래하였으며, 터미널들은 경쟁에서 우위를 확보하기 위해 생산성을 높이는 노력을 경주하고 있다. 터미널의 생산성은 본선작업을 얼마나 효율적으로 수행하는가에 크게 영향을 받는다. 선박의 재항시간을 결정짓는 본선작업은 크게 양하작업과, 적하작업으로 이루어지는데 선박에서 내리는 순서대로 장치하는 양하작업과 달리 적하작업은 장치하는 순서와 다르게 반출할 가능성 때문에 생산성에 많은 영향을 미칠 수 있다. 적하작업을 수행하는 컨테이너들을 장치장에 장치하는 순서는 컨테이너가 장치장에 도착하는 순서에 따르게 되며, 장치장 외부로 반출하는 작업은 선박에 적재하는 순서를 따른다. 선박에 적재하기 위하여 컨테이너를 반출하는 시점에, 반출순서에 맞지 않게 장치된 컨테이너가 있다면, 아래에 놓인 컨테이너를 반출하기 위하여 위에 장치된 컨테이너를 임시로 이동시켜야 하

* First Author, Professor, Dept. of Port & Logistics, Kaya University, Korea, Tel : +82-55-330-1136, E-mail: ykpark@kaya.ac.kr

는 작업이 필요하다. 이 작업을 재취급(rehandling)이라고 하며 이로 인하여 생산성이 떨어지게 되는 것이다. 컨테이너 반입 시에 가능하다면 재취급이 발생하지 않도록 장치하는 것이 좋지만, 장치장에 컨테이너가 도착하는 순서가 무작위이므로 전혀 재취급이 발생하지 않도록 장치하는 것은 불가능하다고 볼 수 있다. 재정돈 작업(remarshalling)은 반출 작업 시에 발생할 수 있는 재취급 작업을 피하기 위하여, 적하 계획이 수립된 후 선박이 도착하기 전까지의 시간을 이용하여 컨테이너들을 미리 정리하는 작업을 말한다. 본 논문에서는 항만 유통에 있어서 컨테이너 터미널의 재정돈을 효율적으로 수행할 수 있는 계획을 수립하는 문제를 다루는데 한 베이 내의 컨테이너를 대상으로 한다. 보통 베이 들이 모두 차 있는 것이 아니므로 필요시 이웃한 베이의 빈 저장 공간을 활용하는 것이 가능하다. 본 논문에서는 이웃한 저장 공간을 외부슬롯으로 활용하여 재정돈 계획을 수립할 때, 성능에 영향을 미치는 요소들을 알아보고, 그 요소들의 조합으로 만든 재정돈 알고리즘들의 성능을 비교한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 연구를 소개하고, 3장에서는 본 논문에서 제안하는 알고리즘과 연구 방법론에 대하여, 4장에서는 실험결과와 분석에 대하여 기술한다. 마지막으로 5장에서는 연구와 관련한 시사점에 대하여 언급한다.

## 2. 선행연구 고찰

컨테이너 터미널의 장치장과 관련한 연구들 중 재취급이나 재정돈에 관한 연구들에 대하여 살펴보면 다음과 같다. 장치장의 업무 중 반입 작업에 대한 연구를 실행하는 이유는, 컨테이너를 장치장에 반입하는 방법이 반출작업을 수행할 때의 업무 효율에 영향을 미치기 때문이다. 먼저 반출작업을 할 때 발생할 수 있는 재취급을 줄이기 위한 방안에 관한 연구로 Kang(2004)은 반입 컨테이너의 무게에 대한 정보를 활용하여 장치 위치를 결정하는 방안을 제안하였다. 이 방안으로 장치할 경우, 무게에 대하여 감안하지 않고 무작위로 장치할 때 보다, 반출 작업에서 재취급이 감소하는 것으로 나타났다. Park(2011)은 선처리 기법을 장치장에 컨테이너를 반입하는 과정에 적용함으로써, 반출하는 시점에 재취급이 발생할 가능성을 낮추는 두 가지 방안을 제안하였으며, 선처리 기법의 효과를 시뮬레이션을 통하여 증명하였다.

컨테이너들의 반입이 완료된 후, 반출작업을 시작하기 전까지의 시간을 활용하여 실시하는 재정돈에 대한 연구들도 많았는데 먼저 한 베이 내의 컨테이너들에 재정돈을 실시하는 방안에 관한 연구로 Ha(2012)는 A*알고리즘을 이용하여 하나의 베이 내에서 재정돈 계획을 수립하는 방안을 제시하였으며, 베이의 장치상태에 따라서 해를 구하지 못하는 경우도 있었으나, 대체적으로 허용할 만한 시간 내에 해를 구하였다. 그리고 이적하는 횟수의 하한을 구하는 법도 함께 제안했다. Kim(2014)은 장치장을 효율적으로 운영하기 위해서, 크레인을 효과적으로 할당하여 재정돈을 수행하는 방안에 대한 연구를 하였으며, 크레인 할당에 대한 제약 없이 재정돈을 수행하는 방안이었다. 시뮬레이션 실험으로 그 방안이 효과적임을 확인하였다. Kim(2009)은 크레인 사용의 제한이 있을 때 재정돈 대상이 되는 컨테이너를 선택하는 방안에 대하여 연구를 하였다. 네 가지 재정돈 대상 컨테이너 선택 방안을 제시하였고 각각에 대하여 시뮬레이션 실험을 통해 성능을 비교하였다. Park(2010)은 Kim(2009)이 제안한 여러 가지 방안들을 비교 실험을 하였고 재정돈 계획을 실행해 가는 과정의 문제를 해결하기 위한 방안인 주기적 재선택 방안에 대하여도 효과를 확인하였다. 블록 내 이적을 실시하면서 선박에 선적하기 전까지의 시간을 활용하는 방안에 대한 연구들도 있었다. 블록내의 여러 베이에 흩어져 서 적재되어 있는 컨테이너들을 하나의 크레인으로 한 곳에 모아서 재취급이 발생하지 않도록 하는 방안을 Kang(2005)이 제안하였으며, 블록 내에 흩어져 있는 컨테이너들을 복수 크레인으로 한곳에 모으는 휴리스틱을 Oh(2005)가 제안하였다. 이 방안은 서로 교차가 불가능한 복수 크레인을 이용할 때 발생하는 크레인들 간의 간섭을 최소화함으로써 수행시간을 단축시키는 방안이었다. 다음의 <Table 1>은 이상의 관련 연구를 요약한 것이다.

**<Table 1>** Summary of related research

| No | Name | Paper Title | Main Contents |
|---|---|---|---|
| 1 | Kang (2004) | Method of Inbound Container Positioning for Minimal Rehandling Considering Weight | Proposal of container position selection method to reduce the number of rehandling using container weight information. Container weight information is obtained at container's entrance time. |
| 2 | Park (2011) | Export container preprocessing method to decrease the number of rehandling in container terminal | Proposal of preprocessing methods about incoming container to reduce the number of rehandling. Simulation experiment is used to prove effectiveness of methods. |
| 3 | Ha (2012) | A* Algorithm for Optimal Intra-bay Container Pre-marshalling Plan | Proposal of method to look up shortest incoming-container relocation sequence in container yard. Method to find lower bound of the length of the container relocation sequence is proposed. |
| 4 | Kim (2014) | Optimization of Dispatching Strategies for Stacking Cranes Including Remarshalling Jobs | Proposal of crane dispatching strategy that allows remarshalling to be mixed together with the main container yard jobs. Experimental shows that the proposed method improves terminal productivity. |
| 5 | Kim (2009) | Container Selecting Methods for Remarshalling Considering Restricted Idle Time of Crane in an Automated Container Terminal | Proposal of method that selects some containers from all export containers and then establishes the remarshalling plan with only the selected containers. The experiment shows that proposed method is better than others methods with all loading containers and remarshalls. |
| 6 | Park (2010) | Iterative Container Reselection Methods for Remarshaling in a Container Terminal | Comparison of two previous researches: genetic algorithm and heuristic algorithm based approaches and Proving the effect of iterative reselection method on decreasing the gap between plan and execution due to the unsureness of crane operation during execution. |
| 7 | Kang (2005) | Sequencing Container Moves for Intra-Block Remarshalling in a Container Terminal Yard | Presentation of an efficient search technique to find an appropriate container moving sequence during remarshalling for avoiding rehandling at the time of loading. Simulation experiments have revealed that the proposed method can make rehandling free solutions in shortl time. |
| 8 | Oh (2005) | A Heuristic Approach to Scheduling Multiple Cranes for Intra-Block Remarshalling | Presentation of an method for assigning moves of containers to each crane and to make a decision of an appropriate container moving order. |

한 베이를 대상으로 재정돈하는 기존의 연구들은 컨테이너 이동 범위를 해당 베이 내로 제한하는 방안들이 대부분이며, 베이 내로 이동범위를 제한을 하면 재정돈을 실행할 때 이동시간과 이동거리가 적게 드는 장점은 있으나, 컨테이너 장치율이 높은 베이에 대하여 재정돈 계획을 도출할 때 과도한 시간이 소요되거나, 계획을 찾지 못하는 경우가 많았다. 본 논문에서는 재정돈을 실행할 때 이동범위를 해당 베이 내로 제한하지 않고 이웃 베이로 확장할 때 성능에 영향을 미치는 요인들이 무엇이 있는지 알아보고 그 요인들을 조합하여 여러 가지 재정돈 방안들도 함께 제시한다. 그리고 시뮬레이션 실험을 통하여 그 방안들의 성능을 비교한다.

## 3. 연구방법론

먼저 관련 용어를 먼저 기술한 후 제안하는 재정돈 방안에 대하여 설명한다. 장치장에서 컨테이너들은 블록 단위로 보관되어 있으며, 각 블록은 여러 개의 베이들로 구성된다. 베이는 다시 여러 개의 스택들로, 또 각 스택은 여러 개의 단으로 이루어져 있다. 각 스택의 상단에 컨테이너를 저장할 수 있는 빈 공간을 슬롯이라 한다. 베이에 장치된 컨테이너들이 반출되는 순서는 우선순위를 사용하여 나타낼 수 있는데 높은 우선순위의 컨테이너가 낮은 우선순위보다 먼저 반출되며, 높은 우선순위에 작은 수의 인덱스 값을 부여하는 방식으로 표현한다. 외부 슬롯은 인접한 베이의 상단에 있는 빈 슬롯으로, 재정돈하려는 컨테이너를 임시로 적재할 수 있다면 대상이 되는 베이에 대한 제약은 없다. 한 스택 위에 여러 개의 슬롯을 외부슬롯으로 활용할 경우, 컨테이너 적재 순서에 제약이 있는데, 위쪽의 컨테이너 인덱스가 아래쪽 컨테이너 인덱스보다 크거나 같도록 적재해야 한다. 그 이유는 재정돈 마지막 부분에서 외부슬롯의 컨테이너를 베이 내부로 옮길 때 큰 인덱스를 가진 컨테이너를 먼저 옮겨야하기 때문이다. 본 논문에서는 너비우선 탐색을 수행하면서 최적 해를 발견할 가능성이 높다고 판단되는 노드를 먼저 탐색하는 방안을 알아보고, 이 방안에서 성능에 영향을 미치는 요소들을 설명한다. 이들 요소들의 조합으로 여러 알고리즘을 만들어서 각 알고리즘의 성능을 분석해 본다.

### 3.1. 재정돈 계획 알고리즘

제안하는 알고리즘의 기본적인 골격은 A*알고리즘(Nilsson, 1998)을 바탕으로 하고 있으며 여기에서 사용하는 용어로 노드란 베이의 상태를 의미한다. 재정돈 계획 알고리즘의 목표는 베이의 초기상태인 시작노드에서 재정돈을 완성하는 상태 즉 목표노드인 해를 찾는 것이 목적이다. 제안하는 알고리즘의 기능은 크게 3가지로 이루어져 있는데 먼저 일정 수의 베이 내 컨테이너들을 외부슬롯으로 이동하는 부분이다. 두 번째는 베이 내부에서의 이동을 수행하는 부분이며, 이 과정에서 해를 찾게 된다. 마지막 부분은 해로 결정된 노드를 대상으로 외부슬롯으로 이동했었던 컨테이너들을 내부로 이동하는 마무리 부분이다. 자세한 절차는 다음과 같다.

1. OPEN1 리스트에 시작노드를 넣고 빈 리스트 OPEN2와 CLOSED를 준비한다.
2. OPEN1 중 첫 노드 n을 선택한다. OPEN1이 비어있다면, 단계7로 넘어간다.
3. 노드 n이 외부이동을 완료한 노드이면 OPEN2에 삽입하고 단계 2로 간다. 외부이동을 완료하지 않은 노드이면 CLOSED에 삽입한다.
4. 노드 n을 가지고 자식노드 생성기준(아래 설명)으로 자식노드cn들을 생성한 후, 자식노드cn들 각각에 대하여 다음을 수행한다. ( $V(n)$ : 노드 n의 정렬기준값)
   1) CLOSE와 OPEN1에 동일한 노드e가 없으면 노드 cn을 OPEN1에 정렬기준(아래 설명)에 따라 삽입한다.
   2) CLOSED에 동일한 노드e가 있으면 자식노드 cn을 파기한다.
   3) OPEN1에 동일한 노드e가 있을 때, $V(cn) > V(e)$ 이면 노드 cn을 파기한다. 즉, 자식노드의 정렬기준값이 동일한 노드의 정렬기준값보다 크면 자식노드를 파기한다.
   4) OPEN1에 동일한 노드e가 있을 때, $V(cn) < V(e)$ 이면 노드 cn으로 대체한다. 즉, 자식노드의 정렬기준값이 동일한 노드의 정렬기준값보다 작으면 동일한 노드를 없애고 자식노드를 대체한다.
5. 단계 2로 간다.
6. OPEN2에서 제일 앞에 있는 노드n을 꺼낸 후 CLOSED에 삽입한다.
7. 노드n이 내부 부적합 컨테이너와 외부 부적합 컨테이너가 없는 노드이면 해를 발견한 것이며 단계13으로 넘어간다.
8. 노드 n에서 가능한 이적을 적용하여 자식노드cn 들을 만든다.
9. 자식노드cn 들에 대하여 다음을 수행한다.
   1) CLOSE와 OPEN2에 동일한 노드가 없으면 노드cn을 OPEN2에 정렬기준에 따라 삽입한다.
   2) CLOSED에 동일한 노드가 있으면 노드cn을 파기한다.
   3) OPEN2에 동일한 노드e가 있을 때, $V(cn) > V(e)$ 이면 노드 cn을 파기한다. 즉, 자식노드의 정렬기준값이 동일한 노드의 정렬기준값보다 크면 자식노드를 파기한다.
   4) OPEN2에 동일한 노드e가 있을 때, $V(cn) < V(e)$ 이면 노드 cn으로 대체한다. 즉, 자식노드의 정렬기준값이 동일한 노드의 정렬기준값보다 작으면 동일한 노드를 없애고 자식노드를 대체한다.
10. OPEN2가 비게 되면 알고리즘은 실패로 종료한다.
11. OPEN2의 노드들은 정렬기준(아래 설명)에 따라 정렬한다.
12. 단계7로 간다.
13. 노드n에서 외부슬롯에 있는 컨테이너들 중 우선순위가 가장 낮은 컨테이너를 찾는다.
14. 그 컨테이너를 적재할 수 있는 스택을 찾아 그 스택으로 이동한다.
15. 외부슬롯의 노드가 모두 이동이 되면 알고리즘은 정상 종료한다.
16. 단계13으로 간다.

단계 4에서는 여러 개의 컨테이너를 한꺼번에 이동함으로써 자식노드를 생성하고, 단계 8에서는 단순히 하나의 컨테이너를 이동해서 자식노드를 생성한다. 단계 13이후에서는 자식 노드를 생성하지 않고 단계 7에서 찾은 해에 해당하는 노드에서 외부 슬롯에 있는 컨테이너를 내부로 옮기는 마지막 정리 작업만 수행하게 된다. 단계4에서 여러 컨테이너를 한꺼번에 이적하여 자식노드를 생성하는 이유는 알고리즘 수행에 너무 많은 시간을 소비하는 것을 피하기 위함이다.

## 3.2. 성능관련 요인

여기서는 관련용어와 알고리즘의 성능에 영향을 주는 요인들을 언급하고 그 특성을 설명한다. 먼저 부적합 컨테이너는 베이 내의 각 스택에서 자신이 놓여있는 위치의 아래에 자신보다 작은 수의 인덱스를 가지는(높은 우선순위의) 컨테이너가 놓여있는 컨테이너들을 말하는데, 이런 부적합 컨테이너들은 재정돈을 할 때 반드시 재배치되어야 하는 컨테이너들이다. 부적합 컨테이너의 수가 성능에 영향을 미치는 중요한 요인인데, 이는 부적합 컨테이너 수가 많아지면 재정돈을 위하여 이동해야하는 컨테이너 수가 많아지기 때문이다. 부적합 컨테이너는 다시 내부 부적합 컨테이너와 외부 부적합 컨테이너로 나누어지는데, 베이 내부에 있는 부적합 컨테이너를 내부 부적합 컨테이너라고 하고, 베이 외부에 있는 부적합 컨테이너를 외부 부적합 컨테이너라고 한다. 외부 부적합컨테이너는 이웃 베이에 이미 이동한 컨테이너들 중 그 시점에 베이 내로 이동한다면, 어떤 스택으로 이동하여도 내부 부적합 컨테이너가 되는 컨테이너를 말한다. 내부 부적합 컨테이너를 필요에 의해서 외부로 이동하게 되는데 내부부적합 컨테이너를 외부로 이동하면 반드시 외부 부적합 컨테이너가 되는 것은 아니다. 어떤 컨테이너들과 함께 이동하느냐에 따라 장치상태가 달라지기 때문이다. 그리고 선택 재배치 컨테이너의 수가 성능에 영향을 미치게 되는데 베이 내의 컨테이너들 중 내부 부적합 컨테이너는 아니지만 재정돈을 위하여 필요에 의해 재배치되어야 하는 컨테이너를 선택 재배치 컨테이너라고 한다. 부적합 컨테이너 수와 마찬가지로 부적합 컨테이너들의 인덱스 합도 성능에 영향을 미치게 된다. 같은 수의 부적합 컨테이너라도 인덱스가 낮을수록 적합컨테이너가 되기 쉽기 때문이다. 지금까지 언급한 성능에 영향을 주는 요인들은 베이의 장치상태에 띠라 이미 결정된 요인들이다. 재정돈을 수행하는 과정에서 성능에 영향을 미치는 요인들이 있는데, 이런 요인들에 대한 처리방식에 따라 재정돈 알고리즘의 성능이 달라진다. 다음은 이런 요인들에 대하여 알아본다.

### 3.2.1. 자식노드 생성방식

재정돈 알고리즘의 단계4에서는 컨테이너를 외부로 이동시키는 것으로 자식노드를 생성하게 되는데 이 자식노드 생성방식에 따라 알고리즘의 성능이 달라진다. 단계4에 명시한 '자식노드 생성기준'은 외부로 이동하는 컨테이너를 선정하는 기준이며 2가지가 있다. 이 방식들은 하나의 컨테이너만 이동해서 자식노드 생성하는 것이 아니고 여러 개를 한꺼번에 이동하는데 그 이유는 하나씩 이동할 경우, 생성하는 자식노드들의 수가 너무 많아지기 때문에 외부이동으로 인한 성능개선효과가 없기 때문이다. 두 가지 방식 모두 선정된 컨테이너의 수는 비어있는 외부슬롯의 수를 초과할 수 없으며 어떤 방식을 사용하는가에 따라 성능이 달라지므로 실험으로 성능을 분석한다.

#### 3.2.1.1. 부적합 컨테이너 이동 방식(Misplaced Container Selection : MCS)

재정돈 알고리즘 단계4에서 하나의 노드에 대하여 부적합 컨테이너를 보유한 스택의 수만큼 자식노드를 생성하게 되는데, 각 스택에 대하여 부적합 컨테이너들을 모두 이동함으로써 자식노드를 생성하는 방식이다. 각 스택마다 부적합 컨테이너 수는 고정이므로 MCS 방식의 경우 자식노드를 생성하기 위하여 이동하는 컨테이너 수는 각 스택마다 고정되어 있다.

#### 3.2.1.2. 재배치 컨테이너 이동 방식(Rearranged Container Selection : RCS)

이 방식은 하나의 스택에 대하여 부적합 컨테이너들 뿐만 아니라 재취급 컨테이너들까지 이동함으로써 자식노드를 생성하는 방식이다. 재취급 컨테이너들을 포함하는 이유는 재취급 컨테이너까지 외부로 이동하고 나면 스택에 저장할 수 있는 컨테이너의 인덱스의 우선순위가 낮아지므로 그 스택에 저장할 수 있는 컨테이너의 인덱스 범위가 커지기 때문이다. 재취급 컨테이너를 선정하는 기준에 따라 이동하는 컨테이너 수는 변할 수 있는데 작업의 단순성과 효율을 감안하여 적합컨테이너들 중 우선순위가 가장 낮은 컨테이너를 적재할 수 있도록 선정기준을 만들었다. $n_e$을 외부 슬롯 중 비어있는 슬롯의 수, $m_j$을 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 작은 수만큼 이적을 해서 자식노드를 생성한다.

### 3.2.2. 정렬방식

위에 언급한 재정돈 알고리즘에서 OPEN 리스트들에 자식노드를 삽입할 때 정렬기준에 따라 삽입을 하게 되는데 이 기준이 성능에 큰 영향을 미치게 된다. 이 알고리즘이 최적 해를 발견할 가능성이 높다고 판단되는 노드를 먼저 탐색하면서 빠르게 해를 찾는 방법을 사용하는데, 탐색하는 순서는 결국 삽입순서인 정렬기준이 되기 때문이다. 정렬방식에 따라 4가지 방식을 기준으로 선택하였다.

#### 3.2.2.1. A*알고리즘의 기준을 따르는 방식 (ASTAR 방식)

이 방식은 자식노드를 OPEN 리스트에 삽입하는 기준으로 부적합 컨테이너들의 수에 그 노드까지 오기 위하여 이동한 컨테이너 수를 합한 값을 사용한다. 이 방식은 A* 알고리즘(Nilsson. 1998)을 바탕으로 해서 (Ha, 2012)에서 사용한 방식을 응용하였다. A* 알고리즘에서는 노드들을 평가하기 위하여 평가함수(f)를 사용하는데 평가함수는 깊이 추정값(g)과 휴리스틱 평가값(h)의 합으로 정의한다. 깊이 추정값은 시작 노드에서 현재 노드까지의 이동하는데 소요되는 거리 추정값이고, 휴리스틱 평가값은 현재 노드에서 목표 노드까지의 이동하는데 소요되는 거리 추정값을 의미한다. 그래서 평가값은 베이의 시작 상태에서 해를 구하는 데까지 이동하는 컨테이너의 이동 개수를 나타내며, 깊이 추정값은 시작 노드에서 현재 노드까지의 컨테이너 이동 개수를 의미한다. 휴리스틱 추정값은 현재 노드에서 해노드까지의 컨테이너 이동개수를 나타내며, 휴리스틱 추정값은 내부와 외부의 부적합 컨테이너 수에 선택 재배치 컨테이너 수의 합으로 구한다.

#### 3.2.2.2. 부적합 컨테이너의 수 기준 방식 (FAST 방식)

최적의 해를 찾기 위해서는 동일한 부적합 컨테이너들이 남아있을 때, 그 노드까지 오기 위하여 이동한 컨테이너 수가 적은 노드를 선택하여 탐색을 하는 것이 유리하다. 그래서 부적합 컨테이너 수에 이동한 컨테이너 수를 합한 것을 기준으로 정렬하는 것이 최적의 해에 가까운 해가 된다. 그러나 그 정렬기준은 탐색해야하는 노드의 수가 많아지게 하고 일정한 시간 내에 해를 찾는 것을 어렵게 만들기도 한다. 이 방식(FAST 방식)은 정렬기준을 부적합 컨테이너 수만 가지고 하고, 부적합 컨테이너 수가 같을 경우 이동한 컨테이너 수로 정렬하는 방식을 사용하여, 최적의 해를 찾는

것보다 탐색시간을 줄이는데 중점을 둔 방식이다. 부적합 컨테이너 수란 결국 앞으로 이동해야할 컨테이너 수가 되므로 정렬기준은 목표노드까지 남은 최소 이동수가 기준이 된다.

3.2.2.3. 부적합 컨테이너의 인덱스합 기준 방식(Sum of Misplaced Container Index : ISUM 방식)

이 방식에서는 OPEN 리스트들에 노드를 삽입할 때 정렬기준으로 부적합 컨테이너들의 인덱스들의 합을 사용한다. 같은 수의 부적합 컨테이너가 정돈되지 않고 남아있는 여러 노드들 중, 낮은 인덱스의 컨테이너가 남아 있는 노드를 선택하는 것 해를 찾을 가능성이 높다는 개념을 바탕으로 한 방식이다.

3.2.2.4. 고정 스택 수 기준 방식(FIX 방식)

이 알고리즘은 자식노드를 생성하기 단계 8에서 컨테이너를 이동할 때, 조건을 만족하는 특정 스택에 있는 컨테이너를 이동할 컨테이너 선택 대상에서 제외하는 방식이다. 이는 이동하는 컨테이너 수를 줄여서 자식노드 생성의 수를 줄임으로써 알고리즘의 복잡도를 낮추려는 의도이다. 2가지가 제한이 있는데 첫째가 가득 찬 스택이면서, 부적합 컨테이너가 없는 스택으로, 해당 스택의 컨테이너를 다른 스택으로 이동 것을 제외한다. 물론 가득 찬 스택이니 해당스택으로 이동하는 것도 불가능하다. 두 번째는 하나가 비어 있으면서, 부적합 컨테이너가 없는 스택으로, 해당 스택으로 이동은 허용하나, 해당스택에서 다른 스택으로 이동을 제한한다. 성능을 비교하기 위하여 이 방식은 위의 3가지 방식과 병행해서 사용한다.

## 4. 연구 결과

자식노드 생성방식과 정렬방식이 성능에 미치는 영향을 분석하기 위하여 이들의 방식을 조합한 여러 가지 알고리즘들을 대상으로 그 성능을 비교하는 실험을 하였다. 알고리즘들은 C++을 사용하여 구현하였으며, 장치율 80%의 10열, 8단, 인덱스 10의 베이 40개를 대상으로 실험을 하였다. 실험환경으로 외부슬롯수를 35개로 설정하였고, 방문노드 수를 2,000개로 제한함으로써 해를 얻는데 너무 많은 시간이 소모되는 것을 방지하였다. 성공률, 실행 시간, 이동 컨테이너 수들의 평균값으로 결과를 나타내었으며 그 결과 값들의 비교를 통해서 각 알고리즘들의 성능을 비교해 보았다. 다음은 실험을 위하여 성능의 요소들의 조합으로 만든 알고리즘들이다.

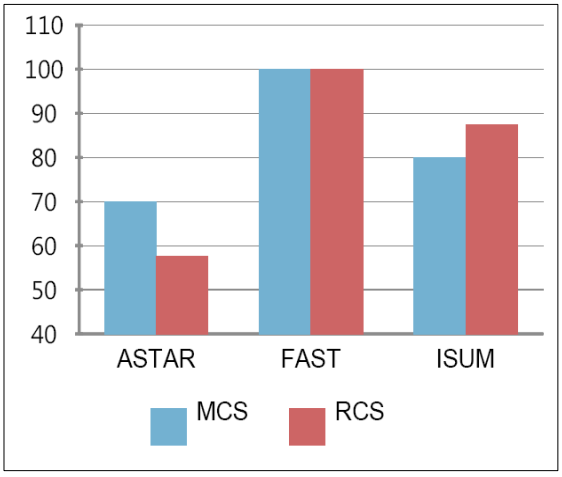
### 4.1. 실험 알고리즘

실험알고리즘은 다음과 같은 조합으로 구성하였다. 자식노드 생성방식 + 정렬방식 + FIX방식의 조합인데 각자를 언급하면 다음과 같다. 이들의 실험결과들을 조합하여 아래에 결과 분석을 하였다.

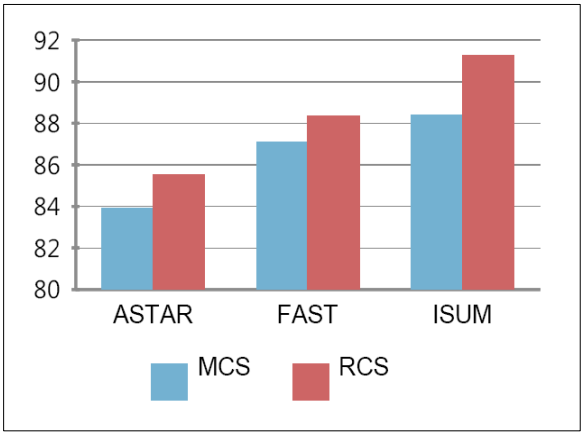
1) MCS 방식 + ASTAR 방식
2) RCS 방식 + ASTAR 방식
3) MCS 방식 + FAST 방식 + FIX 방식
4) MCS 방식 + FAST 방식
5) RCS 방식 + FAST 방식 + FIX 방식
6) RCS 방식 + FAST 방식
7) MCS 방식 + ISUM 방식 + FIX 방식
8) MCS 방식 + ISUM 방식
9) RCS 방식 + ISUM 방식 + FIX 방식
10) RCS 방식 + ISUM 방식

### 4.2. 실험 결과 분석

자식노드의 생성방식과 정렬방식의 성능을 알아보기 위하여, 각 조합의 성공률에 대하여 실험하였으며 실험 알고리즘 1), 2), 4), 6), 8), 10)에 대하여 실험한 결과를 <Figure 1>에서 <Figure 3>에 나타내었다. <Figure 1>을 보면 정렬 방식에 따른 성공률은 자식노드 생성방식과 상관없이 FAST방식이 100%로 가장 높고, 그 다음이 ISUM, ASTAR임을 알 수 있다. 자식노드 생성방식인 MCS와 RCS방식에 따른 성공률은, <Figure 7>의 FAST 방식에 대한 추가 실험에서 MCS 방식이 좋은 것으로 나온 결과와 함께 비교하면 ASTAR와 FAST 방식은 MCS 방식과 조합하는 것이 좋고 ISUM 방식은 RCS 방식과 조합하는 것이 좋은 결과를 낸다고 볼 수 있다.



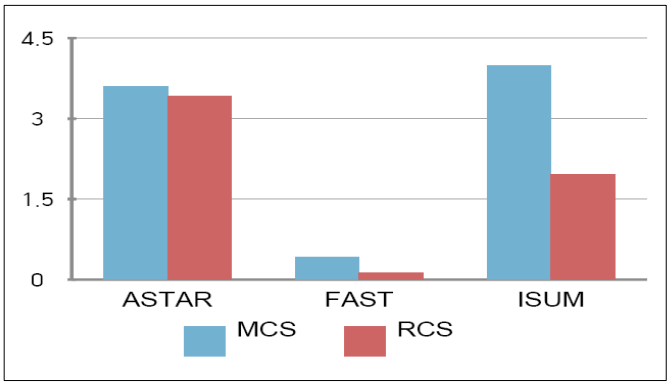

**<Figure 1>** Comparison of succession ratio of algorithm (Unit:%)




**<Figure 2>** Comparison of the numbers of container transportation of algorithm

<Figure 2>에는 재정돈에 필요한 컨테이너 이동횟수를 파악하는 실험 결과가 나타나 있는데 정렬방식에 따른 성능비교에서는

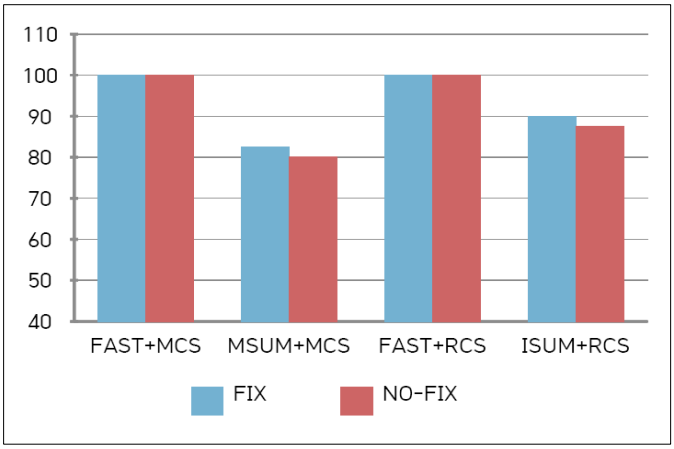
ASTAR가 가장 좋고 그 다음이 FAST, ISUM 순으로 나타나 있다. 자식노드 생성방법에 따른 비교로는 정렬방식과 상관없이 MCS방식이 RCS방식보다 좋은 것을 알 수 있다. 이는 재정돈에 필요한 컨테이너 이동횟수를 기준으로 볼 때 어떤 정렬방식이던 자식노드 생성방식은 MCS 방식과 조합하는 것이 효과적이라고 볼 수 있다.




**<Figure 3>** Comparison of the program execution time of algorithm (Unit:Sec)

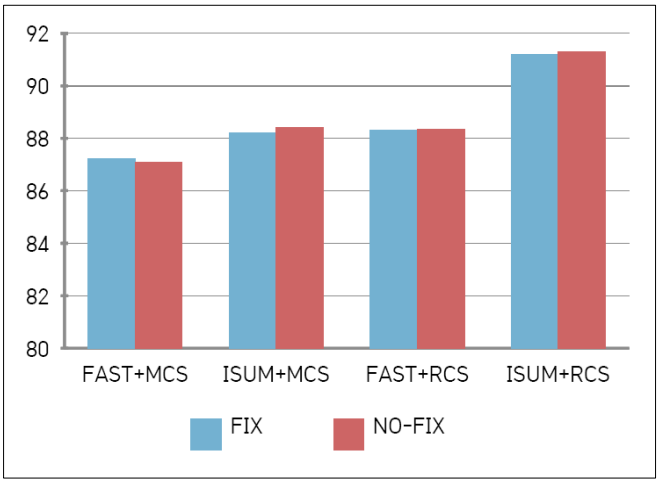
<Figure 3>은 수행시간에 대한 비교실험 결과인데 FAST방식이 자식노드 생성방식에 상관없이 가장 좋은 것으로 나타나 있으며 그리고 ASTAR, ISUM의 순으로 나타났다. 자식노드 생성 방식에 따른 비교는 RCS방식이 MCS방식보다 좋은 것으로 보인다. 이 실험결과 수행시간을 줄이려면 모든 정렬방식은 자식노드 생성방식으로 RCS 방식과 조합하는 것이 효율적이라는 것을 알 수 있다.

다음은 FIX 방식의 효과를 분석하기 위한 실험으로 실험 알고리즘 3), 5), 7), 9)에 대하여 실험한 결과를 <Figure 4>부터 <Figure 6>까지 나타내었다. 자식노드 생성방식과 정렬방식을 조합한 알고리즘들에 대하여 FIX 방식을 적용한 경우와 적용하지 않은 경우의 성능을 비교하였다. 실험을 단순하게 하기 위하여 3가지 정렬방식에서 ASTAR방식은 제외하였다. 먼저 <Figure 4>에는 성공률을 볼 수 있는데 FAST방식에서는 둘 다 100%이므로 비교가 불가하나, 아래 <Figure 7>에 FAST 방식에 대한 추가 실험의 결과에 FIX 방식이 좋은 것을 알 수 있으며, ISUM방식에서도 FIX 방식이 다소 좋은 것을 알 수 있다.



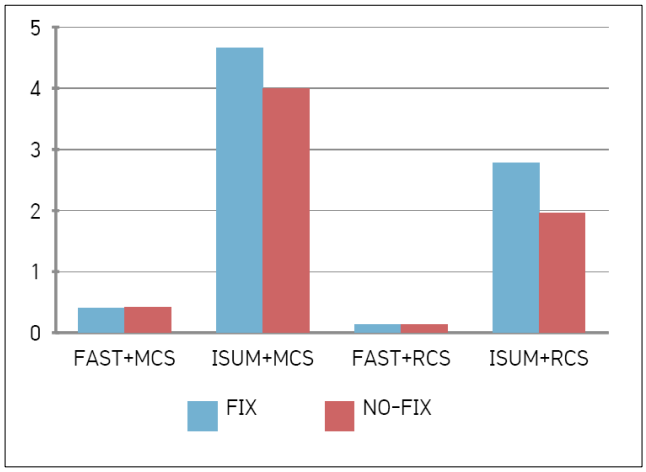

**<Figure 4>** Comparison of succession ratio of algorithm (Unit:%)

<Figure 5>에서 재정돈을 실행할 경우 이동해야하는 컨테이너의 수에 대한 실험결과를 볼 수 있는데 FIX방식을 적용하는 것이 좋은 경우와 하지 않는 것의 성능 좋은 경우가 섞여 있어 두 방식에 따라 달라지는 경향은 발견할 수 없었다.



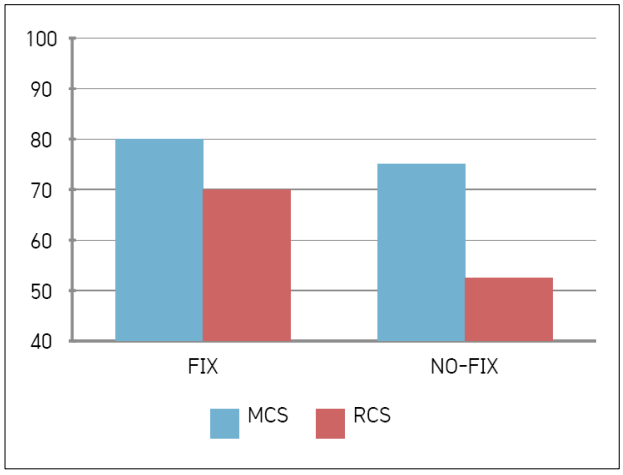

**<Figure 5>** Comparison of the numbers of container transportation of algorithm

<Figure 6>에서는 알고리즘 실행시간에 대한 실험결과를 나타내고 있는데 거의 모든 조합에서 FIX를 적용하지 않는 것이 수행하는데 적은 시간이 필요한 것을 알 수 있다.




**<Figure 6>** Comparison of the program execution time of algorithm (Unit:Sec)

앞의 실험결과에서는 여러 알고리즘들을 비교하기 위하여 외부슬롯을 35개로 설정하고 실험을 실시하였는데, 그 결과 FAST방식은 100%의 성공률 때문에 비교가 불가능한 항목들이 발생했다. 그래서 자식노드 생성방식인 MCS 방식과 RCS 방식과 조합한 FAST 방식에 대하여, 외부슬롯 수를 5개로 줄여 성공률을 낮춘 별도의 실험을 추가로 실시하고 그 결과를 아래 그림<Figure 7>에 나타내었다. 실험 알고리즘을 기준으로 보면 3), 4), 5), 6)에 대하여 실험한 결과 중 성공률만 나타낸 것이다.




**<Figure 7>** Comparison of succession ratio of algorithm (Unit:%)

<Figure 7>은 FAST 방식에 대하여, FIX 방식을 적용한 알고리즘과 적용하지 않은 알고리즘의 성공률을 비교한 결과이다. 자식노드 생성 방식을 기준으로 보면 MCS 방식이 조금 좋은 것을 알 수 있는데 위의 <Figure 1>의 설명에 반영하였으며, FIX방식의 적용여부에 대한 기준으로는 FIX방식을 적용하는 것이 다소 좋은 것을 알 수 있다. 이 결과 역시 위의 <Figure 4> 결과설명에 반영하였다.

이상의 실험을 종합하면 자식노드 생성방식인 MCS와 RCS, 정렬방식인 ASTAR, FAST, ISUM의 조합을 할 때에 어떤 요소를 중시하느냐에 따라 더 나은 조합의 선택은 달라진다는 것을 알 수 있다. 재정돈에 필요한 컨테이너 이동횟수를 줄이려면 어떤 정렬방식이던 자식노드 생성방식은 MCS 방식과 조합하는 것이 효과적이라고 볼 수 있고 수행시간을 줄이려면 모든 정렬방식은 자식노드 생성방식으로 RCS 방식과 조합하는 것이 효율적이라는 것을 알 수 있었다. 성공률의 경우는 정렬방식에 따라 달라지는데 ASTAR와 FAST 방식은 MCS 방식과 조합하는 것이 좋고 ISUM 방식은 RCS 방식과 조합하는 것이 좋은 결과를 낸다. FIX 방식을 적용할 것인가에 대한 결과는, 두 가지 정렬방식으로만 실험을 하였으나 성공률을 기준으로 보면 두 방식 모두 FIX 방식을 적용하는 것이 좋은 것을 알 수 있었으며, 재정돈을 실행할 경우에는 어떤 것이 좋은지 알 수 없었다. 그리고 거의 모든 조합에서 FIX 방식을 적용하지 않는 것이 수행하는데 적은 시간이 필요한 것을 알 수 있었다. 결국 어떤 것이 중요한가에 따라 자식노드 생성방식과 정렬방식을 적당히 조합한 알고리즘을 사용하는 것이 작업효율을 높일 수 있다고 결론내릴 수 있겠다.

## 5. 결론

장치장의 성능을 저해하는 요소로 적하작업 시에 발생할 수 있는 재취급을 피하는 것은 장치장 운영에서 중요한 요소이다. 재취급을 최소화하기 위하여 장치장에 컨테이너를 반입하는 시점에 장치위치를 결정하는 방안에 대한 연구도 있으나, 반입 시 장치하는 방안으로 재취급을 완전히 해결하기는 어려운 것이 현실이다. 반입이 완료하여 반출계획이 결정된 후, 선박이 도착하기 전의 유휴시간을 활용하여 재정돈을 실시하는 여러 방안으로 블록내 이적 방안들과 베이내 이적 방안들에 대한 연구가 있어 왔다. 본 논문도 한 베이의 이적을 다루고 있으며, 이적하는 방법에 있어 기존의 방법과 달리 인접한 이웃 베이를 활용하는 방안에 대하여 연구를 실시하였다. 이웃 베이의 저장 공간을 외부슬롯으로 활용하여 재정돈 계획을 수립하는 경우, 어떤 요소들이 성능에 미치는지 그 요소들의 영향을 알아보기 위하여 여러 가지 조합의 알고리즘을 수립하였고, 성능을 비교, 분석하였다.

### 5.1. 연구결과

시뮬레이션 기법을 통하여 여러 베이에 대하여 실험을 수행한 결과는 다음과 같다. 정렬방식인 ASTAR, FAST, ISUM을 기준으로 볼 때, 자식노드 생성방식과의 조합은 컨테이너 이동횟수를 줄이려면 MCS 방식과 조합하는 것이, 수행시간을 줄이려면 RCS 방식과 조합하는 것이 효율적이다. 높은 성공률을 위해서는 ASTAR와 FAST 방식은 MCS 방식, ISUM 방식은 RCS 방식과 조합하는 것이 좋다. FIX 방식에 대하여는 성공률을 높이려면 FIX 방식을 적용하는 것이 좋고, 수행시간을 줄이려면 FIX 방식을 적용하지 않는 것이 좋다. 결국 재정돈 계획 수립을 할 때 베이의 상황과 필요에 따라 최적의 조합을 선택하여야 할 것이다.

### 5.2. 한계와 시사점

본 연구는 외부슬롯을 이용하는 방안에 대한 연구로, 외부슬롯으로의 이동은, 베이내 이동에 비교하여 시간과 비용이 많이 드는 단점이 있다. 그래서 가능하면 베이내 이동으로 해를 찾는 것이 바람직하나 베이의 장치상태에 따라 베이내 이동으로 해를 찾을 수 없는 경우가 발생하며 그럴 때는 이웃한 베이를 활용하여 해를 찾는 것이 필요하다. 그리고 외부슬롯을 활용하여 재정돈을 하면 장치율이 높아도 해를 찾는데 문제가 없으므로 장치율을 높여서 장치장의 효율을 높일 수 있다. 그래서 향후 추가 연구로는, 먼저 베이내 재정돈으로 최적의 재정돈 계획을 수립할 수 있는지 없는지를 빠르게 판단할 수 있는 알고리즘 개발에 관한 연구를 들 수 있을 것이다. 빠른 시간 내에 베이내 재정돈을 실시할지 이웃 베이를 활용한 재정돈을 실시할지 판단하기 위해 필요하며 현장에 바로 적용할 수 있는 연구라 생각한다. 그리고 성능에 영향을 미치는 요소들에 대한 연구를 확대하면 아무리 어려운 장치형태라도 허용할만한 시간 내에 최적의 베이내 재정돈 계획을 수립하는 방안을 찾는 것도 가능하리라 생각된다. 또한 외부슬롯을 활용하여 최적의 재정돈 계획을 찾는 방안에 대한 연구를 계속하는 것도 필요할 것으로 생각된다.

## References

Ahmadinia, H., Muhaimin, K., & Ofori, E. (2015). Primary Analysis of Information Distribution at Walkbase Company: Developing an Information Strategy. International *Journal of Industrial Distribution & Business,* 6(4), 5-16.

Bae, J. W., Park, Y. M., & 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,* 30(6), 457-464.

Ha, B. H., & Kim, S. S. (2012). A* Algorithm for Optimal Intra-bay Container Pre-marshalling Plan. *Journal of the Korean Institute of Industrial Engineers,* 38(2), 157-172.

Kang, J. H., Oh, M. S., Ru, K. R., & Kim, K. H. (2004). Method

of Inbound Container Positioning for Minimal Rehandling Considering Weight. *Proceedings of The 2004 KIISS Fall Conference,* 271-278.

Kang, J. H., Oh, M. S., Ru, K. R., & Kim, K. H. (2005). Sequencing Container Moves for Intra-Block Remarshalling in a Container Terminal Yard. *Journal of the Korean Institute of Industrial Engineers,* 29(1), 83-90.

Kim, J. E., Park, K. Y., Park, T. J., & Ryu, K. R. (2009). Container Selecting Methods for Remarshalling Considering Restricted Idle Time of Crane in an Automated Container Terminal. *Journal of Korean Navigation and Port Research,* 33(10), 715-722.

Kim, J. H. (2016). Port Co-operations between Public and Private Sector. *East Asian Journal of Business Management,* 6(1), 13-17.

Kim T. K., Yang. Y. J., Bae. A. K., & Ryu. K. R. (2014). Optimization of Dispatching Strategies for Stacking Cranes Including Remarshalling Jobs. *Journal of Korean Navigation and Port Research*, 38(2), 155-162.

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

Oh, M. S., Kwang, J. H., Yu, K. R., & Kim, K. H. (2005). A Heuristic Approach to Scheduling Multiple Cranes for Intra-Block Remarshalling. *Journal of Korean Navigation and Port Research,* 29(5), 447-455.

Park, K. Y., Park, T. J., & Ryu, K. R. (2010). Iterative Container Reselection Methods for Remarshaling in a Container Terminal. *Journal of Korean Navigation and Port Research,* 34(6), 503-509.

Park, Y. K., & Kwak, K. S. (2011). Export container preprocessing method to decrease the number of rehandling in container terminal. *Journal of Korean Navigation and Port Research,* 35(1), 77-82.

Su, S. (2013). A study of Chinese distribution policies and challenges. *Journal of Industrial Distribution & Business,* 4(1), 11-14.