Print ISSN: 2233-4165 / Online ISSN 2233-5382
JIDB website: http://www.jidb.or.kr
doi:http://dx.doi.org/10.13106/jidb.2020.vol11.no10.75

Performance Analysis of a Combination of Carry-in and Remarshalling Algorithms

Young-Kyu PARK[1], Kyung-Ho UM[2]

Received: September 16, 2020. Revised: September 29, 2020. Accepted: October 05, 2020.

Abstract

Purpose: The container terminal is an area that plays an important role in the country's import and export. As the volume of containers increased worldwide, competition between terminals became fiercer, and increasing the productivity of terminals became more important. Re-handling is a serious obstacle that lowers the productivity of terminal. There are two ways to reduce re-handling in the terminal yard. The first method is to load containers in terminal yards using effective carry-in algorithms that reduce re-handling. The second method is to carry out effective remarshalling. In this paper, the performance of various carry-in algorithms and various remarshalling algorithms are reviewed. Next, we try to find the most effective combination of carry-in algorithm and remarshalling algorithm. **Research design, data and methodology:** In this paper, we analyze the performance of the four carry-in algorithms, AP, MDF, LVF, RP and the four remarshalling algorithms, ASI, ASI+, ASO, ASO+. And after making all the combinations of carry-in algorithms and remarshalling algorithms, we compare their performance to find the best combination. To that end, many experiments are conducted with eight types of 100 bays through simulation. **Results:** The results of experiments showed that AP was effective among the carry-in algorithms and ASO+ was effective among remarshalling algorithms. In the case of the LVF algorithm, the effect of carrying in was bad, but it was found to be effective in finding remarshalling solution. And we could see that ASI+ and ASO+, algorithms that carry out remarshalling even if they fail to find remarshalling solution, are also more effective than ASI and ASO. And among the combinations of carry-in algorithms and remarshalling algorithms, we could see that the combination of AP algorithm and ASO+ algorithm was the most effective combination. **Conclusion:** We compared the performance of the carry-in algorithms and the remarshalling algorithms and the performance of their combination. Since the performance of the container yard has a significant effect on the performance of the entire container terminal, it is believed that the results of this experiment will be effective in improving the performance of the container terminal when carrying-in or when remarshalling.

Key words: Carry-in, Preprocessing, Rehandling, Remarshalling Algorithm, Terminal Yard

JEL Classification Code: C61, L91, C88, R49.

1. Introduction

컨테이너 터미널은 국가의 수출입에 있어서 중요한 역할을 담당하고 있는 물류의 중요 거점이다. 트럭을 통해서 터미널에 들어온 컨테이너들이 선박을 통하여 터미널에서 나가거나, 선박을 통하여 들어온 컨테이너들이 트럭을 통해서 터미널을 나가는 등 해상운송과 해상운송의 연결접점으로 역할을 수행하고 있는 것이다. 세계적으로 컨테이너의 운송량 즉, 물동량이 증가하게 되면서 터미널 간의 경쟁이 치열해지게 되었고, 터미널의 생산성을 높이는 것이 더욱 중요하게 되었다. 컨테이너 터미널의 생산성을

1 First Author & Corresponding Author, Professor, Department of Port & Logistics, Kaya University, Korea. Tel: +82-55-330-1136, E-mail: ykpark@kaya.ac.kr
2 Professor, Department of Social Welfare, Kaya University, Korea. Tel: +82-55-330-1104, E-mail: ukyungho@hanmail.net

높이기 위해서는 터미널에 입항하는 선박들의 재항 시간을 줄이는 것이 관건이 된다. 선박에서 컨테이너를 내리는 작업을 양하 작업이라 하고 선박에 컨테이너를 싣는 작업을 적하작업이라 하는데(Kim & Park, 1996) 이들의 하역과 관련한 작업시간을 줄이는 것이 재항 시간을 줄이는 것이 된다. 양하 작업에 비교해 볼 때 적하작업이 재항 시간에 상대적으로 더 큰 영향을 미치는데 적하작업은 장치장에 장치되어 있는 컨테이너들의 상태에 따라 영향을 많이 받게 된다. 제일 상단에 놓여 있는 컨테이너를 반출할 때는 문제가 없으나, 아래쪽에 놓여있는 컨테이너를 반출할 때는 위에 놓여있는 다른 컨테이너들을 먼저 치워야 하는 추가 작업이 필요하게 된다. 이런 작업들을 재취급(Rehandling)이라 하며(Kang, Oh, Ru & Kim, 2004) 장치장의 성능을 떨어뜨리는 주요한 요인이 된다. 재취급은 컨테이너를 장치장에 적재하는 순서와 컨테이너를 장치에서 반출하는 순서가 서로 다르기 때문에 발생하게 된다. 컨테이너들을 장치장에 장치하는 순서는 장치장에 무작위로 도착하는 컨테이너의 도착순서에 따라 정해진다. 그리고 선박에 컨테이너를 싣는 순서에 따라 컨테이너의 반출순서가 정해진다. 선박에 컨테이너를 싣는 순서는 선박의 안정성을 위하여 원칙적으로 무거운 컨테이너들을 아래쪽에 싣는 기준에 따라 정해진다.(Kang, Oh, Ru & Kim, 2004) 장치장에 컨테이너들이 모두 적재된 후 선박에 컨테이너를 싣기 시작할 때까지 며칠 간의 시간이 있는데 그 시간을 이용하여 재취급을 줄이기 위한 재정돈(remarshalling) 작업을 할 수도 있다. 재정돈 작업이란 반입 작업이 끝난 후 선박이 도착할 때까지 시간적 여유가 있을 경우 수행하는 작업이다(Kim, Yang, Bae, & Ryu, 2014). 즉 반출시에 재취급이 발생하지 않도록 하기 위하여 컨테이너들을 재배치하는 작업을 말한다. 그래서 컨테이너 장치장에서 재취급을 줄이기 위한 방법으로 수행할 수 있는 작업이 2 가지가 있다. 첫째가 장치장에 컨테이너가 도착할 때 재취급 발생을 줄이는 방법으로 적재를 하는 반입 방법이고 두 번째가 재정돈 해를 찾은 후 재정돈을 수행하는 방법이다. 본 논문에서는 이런 두 가지 방법 –반입과 재정돈– 들을 조합하여, 그들의 성능을 비교해 보고자 한다. 이를 위하여 반입 알고리즘 4 가지(기존의 효과적인 반입 알고리즘 2 가지와, 본 논문을 위하여 제안한 2 가지 알고리즘)와 재정돈 알고리즘 4 가지(기존의 효과적인 재정돈 알고리즘 2 가지와, 본 논문을 위하여 제안한 2 가지 알고리즘)를 대상으로 조합을 구성하고자 한다. 시뮬레이션을 통하여 먼저 반입 알고리즘과 재정돈 알고리즘들의 성능을 비교해보고, 그들 알고리즘으로 구성된 조합들의 성능을 비교하여 최적의 조합을 찾아보고자 한다. 다음은 논문의 구성에 대한 부분이다. 2 장은 관련 연구를 다루고 3 장은 연구방법론을 다룬다. 4 장은 연구결과를 다루는데 실험결과와 결과 분석에 대하여 기술하며, 5 장에서는 연구요약과 시사점 및 토론에 대하여 기술한다.

2. 관련 연구 및 용어

2.1. 관련 연구

컨테이너 터미널과 관련하여 다양한 연구들이 있다. 먼저 장치장에서 컨테이너를 적재하고, 컨테이너를 반출하는데 사용되는 대표적인 장비인 트랜스퍼 크레인과 관련된 연구들을 살펴본다. Kim, Park, Park and Ryu (2009)은 제한된 시간내에 효율적으로 재정돈 작업을 수행하기 위하여 사용가능한 트랜스퍼 크레인 중 가장 효과적인 크레인을 선택하는 방안에 관한 연구를 수행하였다. 이 연구에서 작업할 컨테이너를 선택하는데 네 가지 방안을 사용하였으며 시뮬레이션을 통해서 이들 방안들에 대한 성능을 비교 분석하였다. 그리고 Oh, Kwang, Yu and Kim (2005)는 복수 트랜스퍼 크레인을 활용하여 여러 블록들에 흩어져 있는 컨테이너들을 한 곳에 모으는 방안을 연구하였다. 이 연구에서 서로 교차가 불가능한 복수 트랜스퍼 크레인들이 서로의 간섭을 최소화하면서도 작업하는 시간을 줄일 수 있는 방안을 제시하였다. 그리고 컨테이너가 터미널에 도착하면 효율적으로 장치장의 영역을 할당해주는 연구와(Jeong, Kim, Woo & Seo, 2012; Mohammad, Nima & Nikbakhsh, 2009; Zhang, Liu, Wan, Murty, & Linn, 2003). 장치장 영역이 할당된 뒤, 구체적인 컨테이너 장치순서와 장치위치를 효율적으로 정해 주는 연구가 있었으며(Kang, Ryu & Kim, 2006; Kim, Park & Ryu, 2000), 컨테이너를 선박에 싣는 순서에 대한 연구들이 있었다.(Imai, Sasaki, Nishimura & Papadimitriou, 2006; Lee, 2011). 그 외에도 터미널 전반적인 운영에 관한 연구들도 많이 있었다. 시뮬레이션 기법을 이용하여 컨테이너 터미널의 하역 작업에 대한 생산성을 분석하는 논문도 있었는데 (Ha & Choi, 2004) 단일 장비에 의한 단순 작업을 대상으로 생산성을 분석한 것이 아니라, 안벽장비, 이송장비 그리고 야드장비 들이 연계작업을 수행할 때 얻어지는 결합 생산성에 대한 연구였으며 최적의 장비조합을 산출하였다. 그리고 컨테이너 터미널의 레이아웃에 대한 연구로 수평형 배치안과 수직형 배치안에 대하여 최적의 레인수와 레인 운영 방식 그리고 적정 장비 대수 등을 결정하기 위한 연구들이 있었다(Kim, Won, Yang, Kim, & Bae, 2001) 최근 들어서 자동화 컨테이너 터미널에 대한 연구도 활발해지고 있는데, 본선 작업 생산성 증대를 위한 야드 운영에 대한 연구도 있다(Kim, Kim, Fibrianto & Hong, 2018). 이 연구에서는 재정돈 버퍼를 구성하여 작업생산성을 개선하는

모델을 제시하여 생산성 개선을 증명하였다. 그리고 Seo, Yi and Kim (2018)는 컨테이너 시스템에 대한 모델을 개발하여 수송구간에서 발생하는 교착 상태를 해소하는 교착 회피 규칙에 대한 연구를 하였으며, 시뮬레이션 기법으로 성능을 비교하는 실험을 하였다. 그리고, 컨테이너 터미널의 생상성과 관련한 연구들도 많이 있었다. Cha, S. H. and Noh, C. K. (2014)는 컨테이너 터미널의 야드 트랙터와 선석크레인 등 이송장비에 대하여 적용할 수 있는 Pooling System 을 제안하여 터미널 생산성을 향상시키기 위한 연구를 수행하였다. 그리고 Cha, S. H. and Noh, C. K (2018)는 DGPS 를 컨테이너터미널의 자동화 시스템장비에 적용하여 터미널 장비들의 효율성과 터미널 생산성을 높이는 연구를 수행하였다. Chung, C. Y. and Shin, J. Y. (2011)는 적하작업과 양하작업을 수행하는 두개의 안벽 크레인에 대하여 하나의 야드 트랙터가 공동 작업하는 듀얼 사이클에 대하여 연구를 하였다. 이 연구에서 야드의 배치와 할당, 야드의 운영 등에 대하여 연구를 수행하였다. 마지막으로 본 논문과 관련 있는 분야인 반입과 재정돈에 관한 연구를 알아보면, 먼저 베이 내의 재정돈 방안에 대한 연구로서, Hirashima, Ishikawa and Takeda (2006)은 컨테이너의 재정돈을 위하여 강화학습을 활용하는 방안을 연구하였고, Kang et al. (2004)은 컨테이너 터미널에 컨테이너를 반입할 때 컨테이너 무게를 활용하는 방법으로 재취급을 줄일 수 있는 적재 위치를 찾는 방안을 제안하였다. 컨테이너 무게에 대한 고려없이 임의의 위치에 장치하는 경우와 비교해 볼 때, 재취급이 감소하는 효과가 있다는 것을 증명하였다. Park and Kwak (2011)은 컨테이너를 반입할 때 선처리를 수행하여 재취급을 줄이는 방안에 대한 연구를 하였다. 이 연구에서 시뮬레이션 기법으로 선처리의 재취급 감소효과를 증명하였다. 본 논문은 이 연구에서 제안했던 선처리 알고리즘을 포함하여, 4 가지 반입알고리즘을 사용한다. 재정돈과 관련한 연구로 Ha and Kim (2012)은 A* 알고리즘 (Nilsson, 1998)을 사용하여 베이 내에서 효율적으로 재정돈 해를 구하는 방안을 연구하였는데 베이의 복잡도가 상당히 높을 경우 재정돈 해를 찾지 못하는 경우가 있기는 했었지만, 일반적인 복잡도에서는 재정돈 해를 효과적으로 찾을 수 있다는 것을 시뮬레이션 기법으로 증명하였다. Park (2016)은 재정돈 해를 찾는데 있어서 컨테이너의 이동을 대상 베이 내로 제한하지 않고 이웃한 베이들의 상단에 있는 비어 있는 슬롯들을 사용하는 방안을 제시하였으며, 이 방안으로 복잡도가 높은 베이들을 대상으로 재정돈을 수행할 경우, 재정돈의 해를 찾는 효과가 있음을 시뮬레이션 기법으로 증명하였다. 본 논문은 Ha and Kim (2012)이 제안한 재정돈 알고리즘과 Park (2016)이 제안한 알고리즘과 본 논문을 위하여 제안한 2 개의 알고리즘을 포함하여 4 가지 재정돈 알고리즘을 사용한다.

2.2. 관련용어와 장치장 관련 작업

2.2.1. 관련용어

장치장과 관련된 용어를 먼저 알아본다. 장치장은 여러 컨테이너들을 장치하기 위한 공간으로, 여러 블록들로 구성되어 있다. 아래 <Figure 1>이 하나의 블록 중 일부를 나타낸 그림이다. 하나의 블록은 여러 개의 베이들로 구성이 되어 있으며, 아래 그림에는 흰색으로 표현한 베이 하나와 회색으로 표현한 베이들이 3 개 묘사되어 있다. 하나의 베이는 다시 여러 개의 스택으로 구성이 되어 있는데(Zhang et al., 2003) 제일 앞쪽에 있는 베이를 보면 사선으로 표현되어 있는 3 개의 컨테이너가 보이는데 이것이 스택을 나타내는 하나의 예이다. 그러니까 스택은 여러 컨테이너를 세로로 적재하고 있는 것을 말하며 하나의 스택에서 컨테이너를 적재할 수 있는 공간을 슬롯이라고 한다. 예를 들고 있는 사선의 스택은 3 개의 슬롯에 컨테이너가 적재되어 있으며, 최대 4 개의 컨테이너를 적재할 수 있는 스택을 묘사하고 있다. 베이의 구성을 표현할 때 슬롯을 '단'으로 스택을 '열'로 표현하기도 하는데 이 그림은 4 단 5 열의 베이를 묘사하고 있는 것이다. 지금 흰색 베이에서 작업하고 있다고 가정할 경우 회색 베이들은 이웃한 베이들이 되고 그 중 (점선으로 묘사한) 비어 있는 슬롯들이 본 논문에서 다루는 재정돈 알고리즘 중에서 언급한 '이웃한 베이의 빈 슬롯'이 된다. 그림에서는 예로 나타내기 위하여 이런 이웃한 베이의 여러 개 빈 슬롯들 중 3 개를 묘사하고 있다.

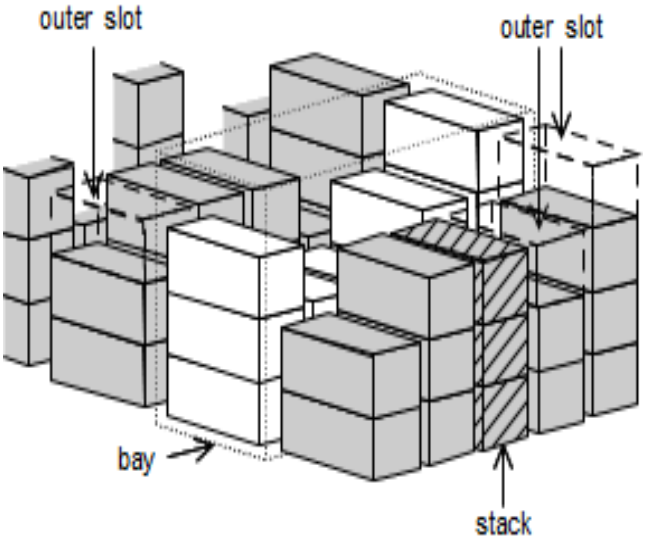

Figure 1: Example of A Container Block

우선순위는 컨테이너들을 반출하는 순서를 나타내는 수로 인덱스라고도 하며 컨테이너의 무게를 기준으로 정해진다. 컨테이너의 무게가 무거울수록 높은 우선순위를 부여하며 낮은 우선순위의 컨테이너보다 먼저 반출한다. 이는 먼저 반출된 무거운 컨테이너를 선박의 하단에 장치함으로써 선박의 안정성을 높이기 위한 것이다. 우선순위는 모든 컨테이너들에서 서로 다르게 부여할 수도 있고, 컨테이너들을 무게를 기준으로 몇 개의 그룹으로 나누어서 부여할 수도 있다

2.2.2. 장치장 관련 작업

컨테이너들이 장치장에 반입이 될 때부터 반출이 될 때까지 수행되는 작업과 그 작업과 관련된 알고리즘을 순서대로 나타내면 다음과 같다. 수출 컨테이너의 경우 선박이 도착하기 일정시간 전까지 장치장에 반입이 되어야 하는데 가능하면 반출시 재취급이 발생하지 않도록 적재하기 위하여 반입알고리즘을 적용을 하게 된다. 그리고 선박이 도착하기 전까지 유휴시간이 있을 경우를 대비하여 재정돈의 해를 구하는 알고리즘 즉 재정돈 알고리즘을 수행하게 된다. 해를 구한 경우, 유휴시간이 생기면 필요할 경우 재정돈 작업을 수행하게 된다. 재정돈의 해를 구하지 못하거나, 해를 구했더라도 유휴시간이 생기지 않으면 재정돈 작업을 수행하지 못하게 된다. 그리고 선박이 도착하면 반출을 수행하게 되고, 이 과정에서 컨테이너의 적재 상황에 따라 재취급이 발생할 수도 있다.

3. 연구방법론

먼저 이 논문에서 다루는 반입 알고리즘들과 재정돈 알고리즘들에 대하여 설명한다. 실험을 통하여 이들 반입 알고리즘들과 재정돈 알고리즘들의 성능을 비교하고, 그리고 이들 알고리즘들을 조합하여 그들의 성능을 비교하는 방식으로 최적의 조합을 찾고자 한다. 다음은 설명을 위하여 앞으로 사용하게 될 기호들이다.

C_0 : 현재 장치하려고 하는 컨테이너
n_0 : 베이 내의 비어 있는 스택 수
S : 베이 내의 전체 스택 수
P_0 : 현재 장치하려고 하는 컨테이너의 우선순위
mP_i : i 스택의 대표 우선순위 값
P_i^k : i 열의 위에서 k 번째 컨테이너의 우선순위, 스택에 오직 하나의 컨테이너만 존재할 경우 P_i^2의 값은 0 이 된다.
D_{ij} : i 열의 상단 컨테이너를 j 열로 이동할 때 발생하게 되는 두 열의 상단 컨테이너들의 우선순위 합의 차이를 말하며 구하는 식은 $D_{ij} = P_i^2 - P_j^1$과 같다.

3.1. 반입 알고리즘

효과적인 반입 알고리즘을 수행하면 다음 단계인 재정돈과 반출을 효율적으로 수행할 수 있고 결과적으로 장치장의 성능을 높일 수 있다. 반입 알고리즘은 크게 선처리를 수행하는 알고리즘과 선처리가 없는 알고리즘으로 나눌 수 있다. 선처리는 반입하는 과정에 컨테이너를 미리 옮기는 작업이므로 선처리 수행 여부에 따라서 반입과정에 별도의 컨테이너 이동이 발생한다. 다음에 소개하는 4 개의 반입알고리즘 중 AP 알고리즘만 선처리를 수행하는 알고리즘이며, 나머지 3 개는 선처리를 수행하지 않는 알고리즘이다. 선처리를 수행함으로써 재정돈의 해를 찾을 때의 성능과 재정돈을 수행할 때의 성능에 영향을 미칠 수 있으며, 재정돈 수행없이 반출만을 수행할 때에도 성능에 영향을 미칠 수 있다.

3.1.1. AP 알고리즘

이 알고리즘은 Park and Kwak (2011)이 개발한 알고리즘으로, 장치장에 컨테이너를 반입할 때 수행하는 알고리즘이다. AP (Anticipatory Preprocess) 알고리즘은 반입을 수행하기 전에 선처리를 수행하는 알고리즘으로, 컨테이너 C_0을 반입할 때 특별한 조건들을 검토하여, 그 조건들이 만족될 때, 기존에 적재된 다른 컨테이너들을 이동한 후(선처리를 수행한 후), 컨테이너 C_0 을 장치하는 알고리즘이다. 선처리는 반출시 재취급을 줄이기 위하여 수행을 한다. 다음은 선처리의 조건에 대한 설명이다.

$$n_0 = 0 \text{ or } P_0 < min_{i=1,j=1}^{S,S}\{D_{ij}\} \quad (1)$$

$$D_{ij} < 0 \quad (2)$$

선처리 조건은 위의 2 가지인데, 위의 조건 (1)는 선처리를 수행할 필요가 있는 시기를 나타내는 조건이다. 이 조건은 컨테이너 C_0의 반입으로 인해서 빈 스택이 없어지게 되거나 재취급이 발생할 수 있을 때를 나타낸다. 조건 (2)는 선처리의 이동 가능 조건을 나타내는데 컨테이너의 이동으로 스택 상단의 컨테이너 우선순위의 합이 줄어들게 되는 때를 의미한다. 선처리는 위의 조건 (1)를 만족하는 시기에 조건 (2)를 만족해야 선처리를 실행하게 된다.

$$min_{i=1,j=1}^{S,S}\{D_{ij}\} \quad (3)$$

여러 가지 이동이 이동가능 조건을 만족할 경우, 가장 효과적인 이동을 찾게 된다. 상단 컨테이너들의 우선순위 합의 차이가 최소가 되는 이동이 가장 효과적인 이동이 된다. 위의 조건 (3)에 해당하는 컨테이너를 찾아서 i 열에서 j 열로 이동하면 가장 효과적인 이동이 된다. 선처리를 수행한 후 장치할 스택을 정할 때는 MDF 알고리즘의 방식을 활용한다.

3.1.2. MDF 알고리즘

이 알고리즘은 Kang et al. (2004)이 제시한 알고리즘으로 컨테이너를 반입하는 단계에서 장치할 위치를 컨테이너 무게를 기준으로 선정함으로써 재취급을 줄이는 방안이다. MDF(Minimum Difference First)방식으로 반입을 실시할 경우, 컨테이너 무게를 고려하지 않고 반입하는 경우와 비교할 때, 재취급이 감소 효과가 있는 것이 증명되었다. 컨테이너를 장치할 위치를 선정하는 기준으로 위의 논문에서는 컨테이너 무게를 사용하였는데, 본 논문에서는 우선순위 개념으로 대체 사용하였다. 이 알고리즘은 새로운 컨테이너를 적재할 때 그 컨테이너의 우선순위와 이미 장치되어 있는 컨테이너들의 우선순위들을 비교하여 반출을 수행할 때 재취급이 발생하지 않을 위치를 선정하여 컨테이너를 장치하는 방법이다. 다음은 장치할 스택을 선정하는 방안에 대한 설명이다. 먼저 스택마다 가장 높은 우선순위 값을 구하여 그 스택의 대표 우선순위 값 mP_i로 설정한다. 빈 스택의 경우 그 대표 우선순위 값을 0 으로 둔다. 우선순위 값 P_0 을 가진 새로운 컨테이너 C_0 이 반입되면 다음의 조건에 해당하는 스택에 장치를 한다.

$$mP_i < P_0 \text{ or } max_{i=1}^{S}\{mP_i\} \quad (4)$$

$$min_{i=1}^{S} < \{mP_i\} \quad (5)$$

위의 조건 (4)는 재취급이 발생하지 않는 스택을 선택하는 조건으로, 반입 컨테이너의 우선순위보다 작은 대표 우선순위를 가진 스택을 선정한다는 것을 나타낸다. 이런 스택이 여러 개가 있을 경우, 이들 중에서 가장 큰 대표 우선순위 값을 갖는 (차이가 가장 적은, Minimum Difference) 스택을 선택하는 조건이다. 이런 스택이 없으면, 재취급 발생을 최소화하는 스택을 골라야 하는데 위의 조건 (5)처럼 가장 낮은 대표우선순위 값을 갖는 스택을 선정한다. 이 스택이 반입 컨테이너의 우선순위와 차이가 가장 적은 스택이 된다.

3.1.3. LVF 알고리즘

이 알고리즘은 본 논문을 위하여 만들게 된 반입 알고리즘이다. 2.3.2 의 MDF 알고리즘과 반대되는 개념의 알고리즘으로 반입하는 단계에서 장치할 위치를 재취급이 가장 많이 발생할 위치에 선정하는 알고리즘이다. 이 방식으로 장치하는 이유는 이 방식으로 반입위치를 선정할 경우 반출시 재취급이 많이 발생할 수는 있으나, 재정돈 과정에서 해를 찾는데 더욱 도움이 될지도 모르기 때문에 개발한 알고리즘이다. LVF(Least Value First)알고리즘의 개념을 간단하게 설명한다. 먼저 스택마다 가장 낮은 우선순위 값을 그 스택의 대표 우선순위 값 mP_i로 설정한다. 빈 스택의 경우 그 대표 우선순위 값을 무한대로 둔다. 우선순위 값 P_0 을 가진 새로운 컨테이너 C_0 이 반입이 되면 다음의 조건에 해당하는 스택에 장치를 한다.

$$mP_i > P_0 \text{ or } min_{i=1}^{S}\{mP_i\} \quad (6)$$

$$max_{i=1}^{S} < \{mP_i\} \quad (7)$$

위의 조건 (6)은 재취급이 발생하는 스택을 선택하는 조건으로 반입 컨테이너의 우선순위보다 큰 대표 우선순위를 가진 스택을 선택하는 것을 의미이다. 이런 스택이 여러 개가 있을 경우, 이들 중에서 가장 작은 값을 갖는 (Least Value) 스택을 선택하게 된다. 이런 스택이 없으면, 재취급 발생을 최대화하는 스택을 골라서 장치를 해야 하는데 위의 조건 (7)처럼 가장 큰 대표 우선순위 값을 갖는 스택을 선정한다. 이 조건들은 위의 MDF 알고리즘의 조건과 완전 반대되는 조건임을 알 수 있다.

3.1.4. RP 알고리즘

이 알고리즘도 본 논문을 위하여 만들게 된 반입 알고리즘이다. RP(Random Position) 알고리즘의 기본 개념은 컨테이너의 장치 위치를 무게를 기준으로 선정하지 않고 반입 순서에 따라서 순서대로 차곡차곡 장치하는 알고리즘이다. 이는 특별한 기준이 없이 장치를 할 경우 쉽게 생각할 수 있는 방식으로 아무런 기준도 없이 장치하는 것을 구현한 알고리즘이며, 다른 알고리즘과의 비교를 위한 알고리즘이다.

3.2. 재정돈 알고리즘

재정돈이란 반출시에 재취급이 발생하지 않도록 하기 위하여, 반입이 끝난 후의 유휴시간을 이용하여 베이 내의 컨테이너들을 옮기는 작업이다. 구분을 하자면, 한 베이의 컨테이너를 비어 있는 다른 베이로 옮기면서 수행하는 방법이 있고, 다른 베이로 옮기지 않고 그 베이 내에서 수행하는 방법이 있다. 장치장에 여유가 없는 경우 후자의 방법으로 재정돈을 수행할 수 밖에 없다. 이 논문에서 다루는 것도 베이 내에서 수행하는 재정돈 알고리즘에 관한

것이다. 그리고 용어의 혼돈을 막기 위하여 '재정돈 알고리즘을 수행한다'라는 말과 '재정돈을 수행한다'라는 용어를 먼저 설명한다. 먼저 '재정돈 알고리즘'을 수행한다는 말은 '재취급이 발생하지 않는 컨테이너 이동순서를 찾는다'라는 의미이고, '재정돈'을 수행한다라는 말은 '재정돈 알고리즘이 찾은 이동순서에 따라 컨테이너를 이동한다'라는 뜻이다. '재취급이 발생하지 않는 이동순서'를 '해'라는 용어로 나타낸다. 즉, 재취급이 발생하지 않는

그리 이동하는 순서를 찾으면 해를 찾은 것이 되는 것이다. 그래서 '재정돈 알고리즘'을 수행해서, 재정돈을 수행할 방법을 찾아야(해를 구해야) '재정돈'을 수행할 수가 있게 된다. 앞으로 '재정돈 알고리즘'과 '재정돈'을 앞의 설명처럼 구분해서 사용을 할 것이다. 이 논문에서 다루는 재정돈 알고리즘 4 가지는 모두, '하나의 베이 내에서 재취급이 발생하지 않는 컨테이너 이동순서를 찾는' 알고리즘으로, Ha and Kim (2012)가 제안한 알고리즘을 기반으로 하는 알고리즘들이다. 이 알고리즘은 A*알고리즘을 사용하여 짧은 시간에 최적의 해를 구하는 방안이다. 베이의 장치 상태에 따라서 재정돈 방안 즉 해를 찾지 못하는 경우도 있었으나, 대부분의 경우 허용할 만한 시간 안에 재정돈 방안을 구할 수 있었다. 해를 구할 경우, 그 해를 바탕으로 재정돈을 수행하면, 반출시 재취급이 발생하지 않게 된다. 다음은 이 알고리즘 개념에 대한 간단한 설명이다. 이 알고리즘은 문제 특성에 관한 정보를 이용하여 가장 짧은 이동 경로에 있다고 생각되는 노드를 우선 방문하는, 너비우선탐색과 비슷한 탐색방법이다. 여기서 노드는 베이를 의미한다. 이 알고리즘에서는 평가함수 $f(n) = g(n) + h(n)$ 로 구한 평가 값을 사용하여 노드들을 평가한다. 여기서 $g(n)$ 은 깊이 추정값으로, 초기상태인 베이의 시작 노드에서부터 평가대상인 노드 n까지의 최단 거리를 나타내고, $h(n)$ 은 휴리스틱 추정값으로 평가대상인 노드 n에서부터 목표 노드까지의 최단 거리를 나타낸다. $f(n)$은 탐색을 위해서 시작 노드에서 목표 노드까지 이동하는 컨테이너 개수를 의미하며, $g(n)$은 시작 노드에서 노드 n 까지 이동한 컨테이너의 개수를 의미한다. $h(n)$ 은 노드 n 에서 목표 노드까지 앞으로 이동해야 할 컨테이너의 개수를 의미한다. 시작 노드에서 시작하여, 평가함수 $f(n)$의 값을 바탕으로 필요시 컨테이너를 이동하여 자식 노드를 생성해 나가면서, $h(n)$이 0 이 되는 방안 즉 해를 구하게 된다. 만약 제한된 시간 내에 해를 구하지 못하거나, 모든 자식 노드들에 대하여 탐색을 했는데도 해를 구하지 못하면 해 찾기를 실패하게 된다.

어떤 재정돈 알고리즘을 실행하느냐에 따라 해를 찾느냐 못 찾느냐가 결정될 수 있으므로, 재정돈 알고리즘이 장치장 전체 성능에 영향을 미치게 된다. 이 논문에서 다루는 4 가지 재정돈 알고리즘들은 2 가지 방식으로 분류를 할 수 있는데, 재정돈을 베이 내에서 수행하는가, 인접 베이의 빈 슬롯을 활용하여 수행하는가로 분류를 할 수 있다. 베이 내에서 수행하는 알고리즘은 ASI(A Star In), ASI+(A Star In+) 알고리즘 들이고, 인접 베이의 빈 슬롯을 활용하는 알고리즘은 ASO(A Star Out), ASO+(A Star Out+) 알고리즘 들이다. 또 다른 분류는 재정돈 해를 찾지 못한 경우, 재정돈을 수행하지 않느냐, 수행하느냐에 따라서 다시 2 가지로 분류할 수 있다. 재정돈을 수행하지 않는 알고리즘은 ASI, ASO 알고리즘들이고, 해가 없어도 재정돈을 수행하는 알고리즘은 ASI+, ASO+ 알고리즘들이다. 다음은 이들 알고리즘에 대하여 간략하게 설명한다.

3.2.1. ASI 알고리즘

ASI (A Star In) 알고리즘은 위에서 언급한 Ha and Kim (2012)가 제안한 알고리즘을 실험을 위하여 직접 구현한 것이다. 이 알고리즘으로 재정돈 방안, 즉 해를 찾지 못할 경우에는 재정돈을 수행할 수가 없다.

3.2.2. ASI+ 알고리즘

ASI+(A Star In +)이 알고리즘은 위의 ASI 알고리즘으로 재정돈 방안을 찾지 못할 경우에도 재정돈을 수행하기 위한 알고리즘이다. ASI 알고리즘과 같은 방식으로 재정돈 해를 찾다가, 해를 못 찾는 경우, 재취급 발생이 최소가 되는 방안을 찾는 알고리즘이다. 이 방안으로 재정돈을 수행할 경우 재취급이 발생을 하는 것은 어쩔 수가 없으나, 재정돈을 수행하지 않았을 때보다 재취급이 줄어드는 효과를 기대하고 만든 알고리즘이며, 이 논문을 위하여 개발한 알고리즘이다. 해를 구하지 못하더라도 최소의 재취급이 발생하는 방안을 구하는 개념에 대하여 간단하게 소개한다. ASI 알고리즘을 수행하면서 각 노드에서 구한 $h(n)$을 관리하는 방식이다. 해를 구한다는 것은 $h(n)$이 0 되는 노드를 구하고 그때까지의 컨테이너 이동과정이 결국 재정돈의 순서, 즉 해가 되는 것이므로, 알고리즘이 끝날 때까지 $h(n)$를 관리하여 최소의 $h(n)$을 갖는 노드와 그때까지의 컨테이너 이동과정을 구하면 재취급이 최소가 되는 재정돈 방안을 구한 것이 된다.

3.2.3. ASO 알고리즘

ASO(A Star Out) 알고리즘은 위의 알고리즘과 같이 Ha and Kim (2012)가 제안한 알고리즘을 기반으로 하여 Park (2016)이 제안한 알고리즘이다. 기존의 연구들은 컨테이너 베이 내에서만 재정돈을 수행하는 제한을 가지고 있었는데 이 논문에서는 이동에 있어서 그런 제한을 없앴다. 이 알고리즘은 재정돈을 수행할 베이의 이웃한 베이 상단에 있는 빈 슬롯을 활용하여 재정돈을 실시하는

알고리즘으로 시뮬레이션 기법으로 그 효과가 입증되었다. 다양한 실험을 위해서는 사용하는 빈 슬롯의 수를 여러 가지로 변화시키면서 해보는 것도 의미가 있겠으나, 너무 경우의 수가 많아지는 점때문에 ASO 알고리즘과 ASO+ 알고리즘 모두 외부슬롯 사용 숫자를 1 개로 제한을 하였다. 고 이 알고리즘도 ASI 알고리즘과 마찬가지로 해를 찾지 못하면 재정돈을 실행하지 못하는 방안이다.

3.2.4. ASO+ 알고리즘

ASO+(A Star Out +) 알고리즘은 위의 ASO 알고리즘이 재정돈 방안을 찾지 못할 때, 즉 해를 찾지 못할 때도 재취급 발생이 최소가 되는 방안을 찾는 알고리즘이다. ASI+알고리즘과 개념이 같으나 이웃한 베이를 활용한 ASO 알고리즘에 적용하였다는 것이 다르다. 이 알고리즘은 ASO 알고리즘과 같은 방식으로 재정돈의 해를 찾는 작업을 수행하고, 해를 못 찾는 경우, 최소 재취급이 발생하는 방안을 찾아서 재정돈을 수행하는 알고리즘이다. 이 알고리즘은 ASO 알고리즘을 확장한 방식으로 이 논문을 위하여 개발한 알고리즘이며 해를 구하지 못하더라도 최소의 재취급이 발생하는 방안을 구하는 개념은 ASI+알고리즘에서 설명한 내용과 동일한 방안이다.

4. 연구 결과

수출 컨테이너가 장치장에 반입되어 반출될 때까지의 단계를 보면 장치장에 반입한 후에는 시간적 여유가 있고, 필요할 때 재정돈을 수행하고, 그렇지 않을 때는 수행을 하지 않는다. 그후 선박이 도착하면, 반출작업을 하게 된다. 이 논문에서 다루는 반입 알고리즘은 AP, MDF, LVF, RP 로 4 가지를 다루게 된다. 그리고 재정돈 알고리즘은, ASI, ASI+, ASO, ASO+ 로 역시 4 가지를 다루게 된다. 그리고 앞에 언급한 것처럼 재정돈은 시간적 여유가 있고 필요할 때 수행하게 된다. 이들 두 종류의 알고리즘의 조합들을 보면 반입 4 가지 알고리즘과 재정돈 4 가지 알고리즘, 그리고 재정돈을 수행하지 않는 경우까지 포함하여 5 가지의 경우에 대하여 여러 조합이 발생한다. 즉 반입 알고리즘 4 가지, 재정돈 알고리즘 4 가지에 수행하지 않는 경우 1 가지, 그래서 총 20 가지의 경우가 발생하게 된다. 이 논문에서는 반입알고리즘의 성능과 재정돈 알고리즘의 성능을 각각 알아보고, 이들 알고리즘의 조합의 성능을 알아본다. 그래서 어떤 조합으로 반입과 재정돈을 수행하는 것이 효율적인지 알아본다.

4.1. 실험

4.1.1. 실험 환경

이 실험을 위하여 여러 가지 베이를 대상으로 시뮬레이션으로 실시하였다. 각 알고리즘들의 구현을 위하여 Microsoft Visual C++ 2010 을 사용하였고, 빠른 실험을 위하여 두 시스템에 나누어 실험하였으며, 그 실험을 위한 시스템들의 사양은 다음과 같다.

[1] Windows 10, Intel(R) Core(TM) i5-9400, 8.GB

[2] Windows 10, Intel(R) Core(TM) i3-8100, 16.GB

실험대상이 되는 여러 가지 베이들은 열과 단을 기준으로 2 종류인 8 열 6 단, 10 열 8 단 베이들을 가지고 컨테이너 수와 인덱스 수를 변경하여 총 8 가지의 베이들을 선정하였다. 이들은 아래 표<Table 1>로 나타냈다. 이 표에 있는 각 베이들에 대하여 100 개씩 무작위로 생성하여 실험을 실시하였다.

Table 1: Specification of Bay to be Tested

Test Bay	Column	Row	No. of Container	No. of Index
TBS1	8	6	28	10
TBS2	8	6	28	28
TBS3	8	6	38	10
TBS4	8	6	38	38
TBS5	10	8	48	10
TBS6	10	8	48	48
TBS7	10	8	64	10
TBS8	10	8	64	64

4.1.2. 실험 내용

성능을 측정할 내용과 항목, 측정방법 등 실험 내용에 대하여 자세히 살펴본다. 먼저 각 반입알고리즘의 경우, 그 성능을 측정하는 것은 반입 알고리즘을 시행한 후 1) 재정돈이 필요 없는 베이 즉, 재취급이 발생하지 않게 된 베이가 얼마나 많은 가로 측정하는 방법과 2) 재정돈이 필요한 베이들을 반출할 때 재취급이 얼마나 많이 발생하는 가로 측정하는 2 가지 방법을 병행하여 시행한다. 재정돈 알고리즘의 경우도 반입 알고리즘처럼, 그 성능을 측정하는 것은 재정돈 알고리즘을 실행한 후 1)재정돈 해를

찾은 베이 즉, 재취급이 발생하지 않게 된 베이가 얼마나 많은 가로 측정하는 방법과 2) 해찾기를 실패한 베이들을 재정돈을 할 수 있으면 재정돈을 실행한 후, 반출할 때 재취급이 얼마나 많이 발생하는 가로 측정하는 2 가지 방법으로 시행한다. 재정돈 알고리즘을 실험하려면 먼저 반입과정을 거친 후에 수행해야 하고 이 반입 알고리즘으로 인한 영향을 없도록 하기 위하여 반입 알고리즘을 거친 모든 베이들을 대상으로 실험을 하였다. 이 두 알고리즘들의 조합의 경우, 먼저 반입알고리즘을 수행하고, 재정돈 알고리즘을 수행한 후 1)재정돈 해를 찾은 베이 즉, 재취급이 발생하지 않게 된 베이가 얼마나 많은 가로 측정하는 방법과 2) 해찾기를 실패한 베이들을 재정돈을 할 수 있으면 재정돈을 실행한 후, 반출할 때 재취급이 얼마나 많이 발생하는 가로 측정한다. 그래서 측정할 수치는 위의 각 1) 실험에서는 재취급이 발생하지 않게 된 베이의 수가 되고, 2) 실험에 대해서는 재취급 수가 된다. 이를 표로 요약하여 나타내면 다음 <Table 2>과 같다. 여기서 실험대상은 각 알고리즘에 대하여 측정하게 되는 베이가 어떤 것인지를 나타내는데, '실험 베이 전체'일 경우 각 알고리즘에 대하여 8 종류 베이 100 개씩을 의미한다.

Table 2: Test Content

Test Content	NO	Test Item	Bays to be Tested	Measured Value	Test Phase			
					Carry-in	Solution	Remarshalling	Carry-out
Performance of Carry-in	1	Effect on remarshalling	All the bays to be tested	No. of bays without rehandling	Test			
	2	Effect on rehandling	All the bays to be tested	No. of container movements due to rehandling	O	X	X	Test
Performance of Remarshalling	3	Ability to find solutions	Bays in need of remarshalling	No. of bays without rehandling	O	Test		
	4	Effect on rehandling	Bays that failed to find a solution	No. of container movements due to rehandling	O	O	O	Test
Performance of Combination	5	Ability to find good bays	All the bays to be tested	No. of bays without rehandling	O	Test		
	6	Effect on rehandling	Bays that failed to find a solution	No. of container movements due to rehandling	O	O	O	Test

이 표에서 사용한 기호나 설명의 의미는 다음과 같다. 'Test Phase' 란의 'carry-in'는 반입알고리즘을 이용하여 반입 작업을 거친다는 의미이고, 'solution'은 해를 찾는 과정 즉, 재정돈 알고리즘을 거친다는 의미이다. 'remarshalling'은 재정돈 작업을 수행한다는 의미이며, 'carry-out'는 반출작업을 수행한다는 의미이다. 'O'는 실험과정으로 그 작업을 수행한다는 의미이며, 'X'는 그 실험을 수행하지 않았다는 의미이다. 'Test'는 'O'과 동일하게 그 작업을 수행하였고 추가적으로 그 작업의 결과로 나온 측정수치(Measured Value)를 구하였다는 의미이다. 예를 들어 설명하면 NO 2 의 경우, 'carry-in'는 'O'이므로 반입과정을 거쳤으며, 'solution'과 'remarshalling'은 'X'이고, 'carry-out'이 'test' 이니까, 재정돈 알고리즘과 재정돈은 수행하지 않고 반출만 수행하여 측정수치를 구하였는데, 'Measured Value'가 'no. of container movements due to rehandling'이니까 측정수치를 '재취급 때문에 이동한 컨테이너 갯수'라는 의미가 된다.

4.2. 실험 결과 분석

4.2.1. 반입 알고리즘들의 성능 분석

이 실험은 위의 성능실험표 <Table 2>의 1 번에 해당하는 실험으로 반입알고리즘의 성능을 실험하는 2 가지 실험 중 첫번째 실험이다. 시험대상인 8 가지 베이 각 100 씩에 대하여 4 가지 반입 알고리즘을 수행한 후 재취급이 발생하지 않은 베이수를 세어, 반입알고리즘의 성능을 측정한다. 그 실험 결과를 아래 <Table

3>에 나타냈다. 이 <Table 3>의 값들은 반입알고리즘을 수행한 후, 반출단계에서 재취급이 발생하지 않는 베이의 수를 의미하며, 이 수치가 높을수록 반입 알고리즘의 효과가 큰 것을 의미하게 된다. 상단의 TBS1 부터 TBS8 까지 모든 베이들에 대하여 실험을 하였으며, 다른 베이들보다 베이 TBS1 이 상대적으로 효과가 높은 것은 베이 당 컨테이너 수와 인덱스가 다른 베이들에 비하여 가장 적기 때문으로 생각된다. AP 알고리즘의 경우, TBS2 의 베이에 대한 결과 값이 92 이라는 것은 실험대상 베이 100 개 중 92 개가 재취급이 발생하지 않게 되었다는 뜻이다. LVF 와 RP 의 경우 그 값이 0 이다. 이 결과들을 보면 AP 알고리즘이 실험대상의 반입 알고리즘들 중 효과가 가장 큰 알고리즘이라는 것을 알 수 있다. 이는 AP 알고리즘이 반입과정에서 선처리를 한 효과라고 볼 수 있다. 즉 AP 알고리즘은 반입과정에서 다른 알고리즘과 달리 미리 컨테이너를 이동시켜 두기 때문에 재취급이 발생하지 않는 베이의 수가 상대적으로 많다는 것이다.

다음 실험은 위의 성능실험표 <Table 2>의 2 번에 해당하는 실험으로 반입알고리즘의 성능을 실험하는 2 가지 실험 중 두번째 실험이다. 위의 1 번 실험 결과 재취급이 발생하지 않는 베이는 재정돈을 할 필요가 없지만, 재정돈이 필요한 베이들은 여유시간이 없어서 재정돈을 실행할 수 없는 경우, 반출과정에 재취급이 발생하게 된다. 이때 발생하는 재취급으로 인한 컨테이너의 이동횟수 역시 반입알고리즘의 성능에 따라 달라지므로 이 수를 이용하여 반입알고리즘의 성능을 비교할 수 있다. 즉 반입알고리즘의 성능이 좋을수록, 재취급으로 발생하는 컨테이너 이동횟수가 적을 것이기 때문에 재취급으로 인한 컨테이너의 이동횟수를 가지고 반입알고리즘의 효과를 알 수가 있다. 실험 결과는 아래 표 <Table 4>에 나타나 있다. 실험 결과를 보면 AP 와 MDF 의 경우에 비하여 LVF 와 RP 의 성능이 상대적으로 떨어지는 것을 알 수 있다. 이는 앞의 LVF 와 RP 의 알고리즘 설명에 언급한 것처럼 두 알고리즘의 반입 정책에 기인한 것으로 보인다.

Table 3: Number of Bays without Rehandling

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	100	92	86	50	100	75	83	20	606
MDF	98	67	73	9	100	43	74	1	465
LVF	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0

Table 4: Number of Container Movements moved Due to Rehandling

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	0	9	18	83	0	47	22	226	405
MDF	2	61	37	392	0	198	36	850	1576
LVF	3575	5290	4979	7724	7072	12287	10111	17267	68305
RP	1316	1428	2316	2725	2769	3205	4616	5702	24077

4.2.2. 재정돈 알고리즘의 성능 분석

다음 실험은 위의 성능실험표 <Table 2>의 3 번에 해당하는 실험으로 재정돈 알고리즘의 성능을 알아보기 위한 두 실험 중 첫번째 실험이다. 실험과정을 설명하면, 먼저 반입 알고리즘을 수행한다. 반입 단계가 끝난 후의 베이들 중 재정돈이 필요한 베이들만 대상으로, 재정돈 알고리즘을 수행한다. 재정돈 알고리즘 수행결과 재정돈의 해를 찾은 베이의 수로서 재정돈 알고리즘의 효과를 측정을 하였다. 실험결과는 아래 표 <Table 5>에 나타냈다. 이 표를 보면 알 수 있지만, 재정돈이 필요한 베이의 수(N bay)는 4 가지 알고리즘에 대하여 모두 같다. 이는 반입단계까지는 같은 알고리즘을 거쳤기 때문에 재정돈이 필요한 베이수가 같기 때문이다. 그리고 ASI 과 ASI+의 경우, 두 실험결과의 값이 같고, ASO 과 ASO+의 경우도 두 실험결과 값이 같은 것을 알 수 있다.

이는 당연한 결과로써 ASI+ 알고리즘은 ASI 알고리즘과 해를 찾는 단계까지는 내용이 같기 때문이다. 해를 못 찾았을 경우 ASI 알고리즘은 재정돈을 수행하지 못 하지만, ASI+알고리즘은 재정돈을 수행하며 이것이 두 알고리즘의 차이이다. 이 차이는 ASO 과 ASO+에 대해서도 같으며 그래서 ASO 과 ASO+ 역시 값이 같은 것이다. 그래서 비교를 ASI, ASI+ 알고리즘과 ASO, ASO+ 알고리즘으로 해야 되며, ASO, ASO+ 알고리즘의 성능이 더 좋은 것을 알 수 있다. 그리고 ASO 알고리즘과 ASO+ 알고리즘의 성능을 비교해 보면 4 가지 중 가장 성능이 좋은 재정돈 알고리즘을 알 수 있게 되며 그 실험의 결과를 바로 다음 실험인 4 번 실험에 나타냈다.

Table 5: Number of Bays that Succeeded in Finding a Solution

Test Bay	TBS1		TBS2		TBS3		TBS4		TBS5		TBS6		TBS7		TBS8		SUM	
remarshalling	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*	N bay*	S bay*
ASI	202	188	241	189	241	123	341	115	200	103	282	119	243	80	379	42	2129	959
ASI+	202	188	241	189	241	123	341	115	200	103	282	119	243	80	379	42	2129	959
ASO	202	202	241	241	241	241	341	329	200	197	282	259	243	181	379	163	2129	1813
ASO+	202	202	241	241	241	241	341	329	200	197	282	259	243	181	379	163	2129	1813
Total	808	780	964	860	964	728	1364	888	800	600	1128	756	972	522	1516	410	8516	5544

*N bay : Number of bays that need to be rearranged,
*S bay : Number of bays found for rearranging solutions

다음의 실험결과들은 위의 성능실험표 <Table 2>의 4 번에 해당하는 실험으로 재정돈 알고리즘의 성능을 알아보기 위한 두 실험 중 두번째 실험이다. 실험과정을 설명하면, 먼저 반입 알고리즘을 수행한다. 반입 단계가 끝난 후의 베이들 중 재정돈이 필요한 베이들만 대상으로, 재정돈 알고리즘을 수행한다. 재정돈 알고리즘 수행결과 재정돈의 해를 못 찾은 베이들을 대상으로 ASO 알고리즘은 재정돈을 수행할 수 없으므로, ASO+ 알고리즘만 재정돈을 수행한다. 재정돈 수행이 끝난 뒤 반출을 수행할 경우의 재취급 수를 측정하여 알고리즘의 효과를 측정을 하였다. 이 실험결과를 통하여 바로 앞의 실험결과에서 언급한 것처럼 ASI 알고리즘과 ASI+알고리즘의 성능을 비교할 수 있다. 이 실험은 위의 3 번 실험의 다음단계에 해당하는 실험으로서, 3 번 실험이 반입 단계가 끝난 후의 베이들 중 재정돈이 필요한 베이들만 대상으로 재정돈 해를 구하는 실험이었는데, 이번 실험은 재정돈 해를 못 구한 베이에 대하여 반출을 실행할 경우 발생하는 컨테이너 이동횟수 즉, 재취급 수를 구하는 실험이기 때문이다. 수행하는 재정돈 알고리즘이 <Table 6>의 경우 ASI 알고리즘, <Table 7>의 경우 ASI+ 알고리즘이다.

Table 6: Number of Container Movements Occurred during Carry-out after ASI Algorithm

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	0	0	0	20	0	19	5	167	211
MDF	0	5	0	278	0	141	5	815	1244
LVF	0	42	516	2224	31	1150	2833	5925	12721
RP	208	709	2298	2725	2660	3178	4616	5702	22096

Table 7: Number of Container Movements Occurred during Carry-out after ASI+ Algorithm

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	0	0	0	14	0	10	5	146	175
MDF	0	2	0	137	0	93	4	572	808
LVF	0	9	206	907	15	792	1421	4399	7749
RP	81	380	1638	2059	2075	2612	3957	4983	17785

재정돈의 해를 구하지 못한 상황에서는 재정돈을 수행하지 못하는 ASI 알고리즘과 달리, ASI+알고리즘은 재정돈을 수행할 수 있으며, 그로 인해서 ASI+알고리즘이 재취급수가 ASI 알고리즘보다 적은 것을 알 수 있다. 즉 ASI+알고리즘의 성능이 ASI 알고리즘의 성능보다 더 좋다는 것을 알 수 있다. ASI 알고리즘과 ASI+알고리즘의 성능을 비교할 때, 반출시 재취급 수만 비교한다면 ASI+의 재취급수가 ASI 알고리즘의 재취급수보다 적기 때문에 성능이 더 좋다고 할 수 있다. 그러나 ASO+ 알고리즘의 경우, 재정돈을 수행하기 때문에 재취급수가 줄어든 것이고 재정돈을 수행하기 위한 컨테이너 이동횟수가 반입알고리즘에 따라 아주 큰 수이기도 하므로 성능을 고려할 때 이점도 같이 고려해야 할 것이다. 물론 재취급의 수가 장치장의 성능에 미치는 영향이 큰 점을 고려한다면, 재정돈의 수가 더 낮은 ASI+ 알고리즘이 더 좋은 알고리즘이라고 할 수 있다. 다음은 ASO 알고리즘과 ASO+알고리즘에 대한 실험결과 표들이다. 결과는 ASI 알고리즘과 ASI+알고리즘과 비슷하지만 ASO+의 경우가 더 좋은 성능을 보이는 것을 알 수 있다.

이상의 실험결과를 보면 4 개의 알고리즘 중 ASI 알고리즘보다 ASI+알고리즘이 성능이 좋고, ASO 알고리즘보다는 ASO+알고리즘이 더 좋으며, 4 개 알고리즘 전체의 수치를 비교해 보면 ASO+ 알고리즘이 가장 성능이 좋다는 것을 알 수 있다. 그리고 위의 성능실험표 <Table 2>의 3 번에 해당하는 실험결과와 성능실험표 <Table 2>의 4 번에 해당하는 두 실험결과를 같이 고려해서 재정돈 알고리즘들의 성능을 비교해도 같은 결론을 내릴 수 있다. 위의 3 번 실험결과에 따르면, ASI, ASI+알고리즘보다 ASO, ASO+알고리즘이 더 좋은 성능을 보이고 있고, 위의 7 번 실험결과에 따르면, ASO 알고리즘보다 ASO+알고리즘의 성능이 더 좋다는 것을 알 수 있으므로 앞서 언급한 것처럼 4 개 알고리즘 전체 중 ASO+ 알고리즘이 가장 좋은 성능을 가진 알고리즘이라고 할 수 있다.

Table 8: Number of Container Movements Occurred during Carry-out after ASO Algorithm

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	0	0	0	0	0	4	0	91	95
MDF	0	0	0	2	0	36	1	426	465
LVF	0	0	0	65	0	0	193	3748	4006
RP	0	0	0	282	86	521	2708	5106	8703

Table 9: Number of Container Movements Occurred during Carry-out after ASO+ Algorithm

Carry-in Algorithm	Test Bay								SUM
	TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	0	0	0	0	0	2	0	69	71
MDF	0	0	0	1	0	14	1	226	242
LVF	0	0	0	52	0	0	160	2780	2992
RP	0	0	0	188	72	418	2147	4178	7003

4.2.3. 반입 알고리즘과 재정돈 알고리즘의 조합에 대한 성능 분석

이 실험은 위의 성능실험표 <Table 2>의 5 번에 해당하는 실험으로 두 알고리즘의 조합의 성능을 비교하기 위한 두 실험 중 첫번째 실험이다. 실험과정을 설명하면, 먼저 반입 알고리즘을 수행한다. 그리고 재취급이 발생하지 않는 베이수(α)를 구한다. 재취급이 발생하는 베이에 대하여, 재정돈 알고리즘을 수행하여 재정돈 해를 구한 후, 재정돈을 수행할 수 있는 경우에 재정돈을 수행하여 재취급이 발생하지 않는 베이수(β)를 구해서, 그 수를 더한다(α+β), 이렇게 구한 재취급이 발생하지 않는 수(α+β)가 반입알고리즘과 재정돈 알고리즘, 이 두 알고리즘의 조합으로 재취급이 발생하지 않는 베이수가 되며, 그 수가 클수록 성능이 좋다고 판단한다. 아래의 표<Table 10>은 반입알고리즘과 재정돈 알고리즘을 차례대로 수행한 후 재취급이 발생하지 않는 베이수, 즉 (α+β)에 해당하는 값을 나타낸 표이다. ASI 알고리즘과 ASI+ 알고리즘이 재정돈 해를 구하는 데까지는 같고, ASO 알고리즘과 ASO+ 알고리즘이 재정돈 해를 구하는 데까지는 같기 때문에, 이 표에 ASI+ 알고리즘의 결과와 ASO+알고리즘의 결과는 따로 나타내지 않았다.

Table 10: Number of Bays without Re-handling after Carry-in and Remarshalling Algorithm

Carry-in Algorithm	Remarshalling Algorithm	Test Bay								SUM
		TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	ASI	100	100	100	91	100	95	98	52	736
MDF	ASI	100	98	100	54	100	71	98	11	632
LVF	ASI	100	98	81	29	99	70	41	0	518
RP	ASI	86	52	1	0	4	1	0	0	144
AP	ASO	100	100	100	100	100	98	100	78	776
MDF	ASO	100	100	100	99	100	95	99	57	750
LVF	ASO	100	100	100	98	100	100	96	38	732
RP	ASO	100	100	100	91	97	84	43	11	626

위의 표 <Table 10>을 보면, AP 알고리즘과 ASO 알고리즘의 조합(또는, AP 알고리즘과 ASO+ 알고리즘의 조합)의 합계값이 776 으로 가장 크므로, 이 조합이 성능이 가장 좋은 것을 알 수 있다. 재정돈 알고리즘이 ASO 알고리즘인 경우를 가지고 반입알고리즘들을 비교하면, AP 와 MDF 의 경우, MDF 가 다소 성능은 낮으나 비슷한 성능이라는 것을 볼 수 있다. MDF 와 LVF 의 경우, MDF 와 LVF 가 성능을 비교해 보면, 반입과정에서의 성능은 <Table 3>에서 보듯이 LVF 가 MDF 보다 상대적으로 떨어지지만, 재정돈 과정까지 거치면 비슷해지는 것을 알 수 있다. 이는 다르게 얘기하면, LVF 로 반입을 할 경우, 반입에서는 효과가 떨어지지만 재정돈과정에서는 효과는 더 높다고 볼 수 있다. 그리고 AP 와 MDF 의 경우 AP 알고리즘이 반입과정의 선처리에 부담이 있다면, AP 보다는 MDF 가 유리할 수도 있다고 볼 수 있으며, 그렇지 않다면 AP 가 성능이 가장 좋다는 것을 알 수 있다. 다음 실험은 위의 성능실험표 <Table 2>의 6 번에 해당하는 실험으로 두 알고리즘의 조합의 성능을 비교하기 위한 두 실험 중 두번째 실험이다. 실험과정을 설명하면, 먼저 반입 알고리즘을 수행한다. 그리고 재정돈 알고리즘을 수행하여 재정돈 해를 구한 후, 재정돈을 수행할 수 있는 경우에 재정돈을 수행하고, 재정돈을 수행할 수 없는 경우 재정돈을 생략한 후, 반출과정을 수행한다. 이때 발생하는 재취급 수로 두 알고리즘의 성능을 측정한다. 각 표에 순차적으로 수행하는 두 알고리즘을 명시하였으며, 두 알고리즘의 조합의 성능이 우수할수록 반출과정에 재취급이 발생하지 않기 때문에, 이 실험의 수치가 낮을수록 성능이 높다고 볼 수 있다.

이 실험결과들의 수치를 비교해 보면 RP 알고리즘과 ASI 알고리즘의 성능이 가장 나쁘고, AP 알고리즘과 ASO+알고리즘의 조합이 성능이 가장 좋다는 것을 알 수 있다. 이는 반입알고리즘 중 RP 알고리즘의 성능이 가장 나빴고, AP 알고리즘의 성능이 가장 좋았으며, 재정돈 알고리즘 중에는 ASI 알고리즘의 성능이 가장 나빴고, ASO+알고리즘의 성능이 가장 좋게 나왔으므로, 그 둘의 조합의 성능 역시 가장 나쁘게, 그리고 가장 좋게 나타나는 것은 당연한 결과라는 생각이 든다.

Table 11: Number of Container Movements by Re-handling after Carry-in and Remarshalling Algorithm

Carry-in Algorithm	Remarshalling Algorithm	Test Bay								SUM
		TBS1	TBS2	TBS3	TBS4	TBS5	TBS6	TBS7	TBS8	
AP	ASI	0	0	0	20	0	19	5	167	211
MDF	ASI	0	5	0	278	0	141	5	815	1244
LVF	ASI	0	42	516	2224	31	1150	2833	5925	12721
RP	ASI	208	709	2298	2725	2660	3178	4616	5702	22096
AP	ASO	0	0	0	0	0	4	0	91	95
MDF	ASO	0	0	0	2	0	36	1	426	465
LVF	ASO	0	0	0	65	0	0	193	3748	4006
RP	ASO	0	0	0	282	86	521	2708	5106	8703
AP	ASI+	0	0	0	14	0	10	5	146	175
MDF	ASI+	0	2	0	137	0	93	4	572	808
LVF	ASI+	0	9	206	907	15	792	1421	4399	7749
RP	ASI+	81	380	1638	2059	2075	2612	3957	4983	17785
AP	ASO+	0	0	0	0	0	2	0	69	71
MDF	ASO+	0	0	0	1	0	14	1	226	242
LVF	ASO+	0	0	0	52	0	0	160	2780	2992
RP	ASO+	0	0	0	188	72	418	2147	4178	7003

5. 연구요약과 시사점 및 토론

5.1. 연구요약

컨테이너 터미널의 성능에 영향을 미치는 장치장과 관련한 작업으로 반입과 재정돈, 그리고 반출작업이 있다. 이 논문에서는 반입 알고리즘들의 성능과 재정돈 알고리즘의 성능 그리고 반입알고리즘과 재정돈알고리즘들을 다양하게 조합하여 그 성능을 비교해 보았다. 먼저 반입 과정의 알고리즘으로 효과가 입증된 MDF 와 AP 알고리즘과 이 논문을 위하여 새로 제안한 LVF 와 RP 알고리즘의 성능을 알아보았으며, 4.2.1.에서 알 수 있듯이 반입 알고리즘은 AP 의 알고리즘이 효과적이었으며, 이는 선처리라는 다른 알고리즘에는 없는 작업을 먼저 수행한 결과로 재취급을 줄이는 효과를 얻을 수 있었다. 만약 이 선처리 작업을 하지 못할 경우는 MDF 알고리즘이 효과적이라는 것을 알 수 있었다. 이 논문에서 제안한 RP 알고리즘의 경우, 그 효과가 상대적으로 아주 낮은 것을 알 수 있었는데 이는 컨테이너를 아무 기준도 없이 장치하는 것이 얼마나 장치장의 성능을 떨어뜨리는지 극단적으로 보여주는 결과라는 생각이 든다. LVF 알고리즘의 경우는 다음에 오는 재정돈 알고리즘에서 효과가 있기를 기대하고 제안한 알고리즘이었으며, 반입 과정에서의 성능은 나쁜 편이었으나, 재정돈의 해를 찾는 데는 효과가 있었던 점은 주목할 만한 결과라고 생각된다.

재정돈 알고리즘으로 효과가 입증된 A*알고리즘을 기반으로 한 ASI 알고리즘, ASI+알고리즘, ASO 알고리즘 그리고 ASO+알고리즘들의 성능도 비교해 보았다. 4.2.2 에서 보았듯이 반입 알고리즘과의 관계를 보지 않고 재정돈 알고리즘만 고려할 경우, ASO+ 알고리즘이 가장 효과적이라는 것을 알 수 있었다. 실험한 재정돈 알고리즘들을 같이 비교해 보면, 재정돈 해를 구하지 못했을 때, 기존의 알고리즘인 ASI 알고리즘과 ASO 알고리즘은, 재정돈을 실시할 수 없었는데, 이 논문에서 새로 제안한 ASI+알고리즘과 ASO+알고리즘은 해를 구하지 못하더라도, 최소의 재취급이 발생하는 방안으로 재정돈을 수행할 수 있었다. 그리고 재정돈을 수행하지 않는 경우보다 재취급 발생이 줄어든 것을 파악할 수 있었다. 이는 재정돈을 수행할 여유시간이 있음에도 불구하고 해를 구하지 못한 경우 재정돈을 수행하지

못하여 반출시 많은 재취급이 발생하여 장치장의 성능이 떨어지는 것을 피할 수 있는 방안을 찾은 것이라 의미가 있다고 볼 수 있다. 물론, 감소하기는 하지만 재취급이 발생하지 않는 것은 아니니까, 재정돈을 수행하고 재취급을 줄일 것인지, 재정돈을 수행하지 않고 더 많은 재취급을 감수할 것인지는 상황에 따라서 결정할 문제라고 생각한다. 그리고 반입 알고리즘과 재정돈 알고리즘들의 조합에 대한 성능을 실험한 결과, 3.2.3.에서 보았듯이 AP 와 ASO+ 알고리즘의 조합이 가장 효과적이라는 것을 알 수 있었다. 반입알고리즘 중 가장 효과적인 AP 알고리즘과 재정돈 알고리즘 중 가장 효과적인 ASO+ 알고리즘의 조합이 가장 효과적이라는 것은 어찌 보면 당연한 결과이기도 하다. 반입 알고리즘만 봤을 때도 AP 알고리즘이 효과적이었으니까, 반입은 AP 알고리즘으로 수행하고, 수행결과 재정돈을 수행하지 않을 경우 재취급이 발생할 수 있는 상황이고, 재정돈을 수행할 시간적 여유가 있고 재정돈이 필요할 때는 ASO+알고리즘을 수행하는 것이 가장 효과적인 선택이 될 것으로 판단된다.

5.2. 시사점 및 토론

본 논문의 시사점을 알아본다. 본 논문은 반입 알고리즘과 재정돈 알고리즘 그리고 그들 알고리즘들의 조합에 대하여 성능을 비교해 보고 최적의 조합을 찾았다는 점에서 의미가 있다고 생각한다. 이 논문에서 찾은 조합은 터미널에서 컨테이너를 반입할 때와 재정돈을 시행할 때 적용할 수 있으리라 기대한다. 다음은 몇 가지 아쉬운 점에 대하여 언급한다. 먼저 실험할 경우의 수가 많고 베이의 종류가 많아서 더 다양한 실험을 하지 못한 것이 아쉬운 점이다. ASO+알고리즘의 경우, 외부슬롯의 수를 1 로 제한하여 실험을 하였는데 그 수를 변화하면서 더 다양한 결과를 얻지 못한 점 역시 아쉬운 점이다. 그리고 재정돈 알고리즘을 베이내 알고리즘만을 대상으로 실험하였다는 점도 아쉬움으로 남는다. 그래서 향후 본 논문과 관련한 연구로, 베이들 간 재정돈 알고리즘에 대한 연구를 수행하고자 한다. 그리고 반입 알고리즘들과 베이들 간 재정돈 알고리즘의 조합에 대한 성능 비교에 대하여도 연구해 보고자 한다. 그때는 더 많은 베이와 외부슬롯의 수를 변화시키면서 실험을 하기를 기대한다.

References

Cha, S. H., & Noh, C. K. (2014). A Study on the Application of Transfer Equipment Pooling Systems for Enhancing Productivity at Container Terminals, *Journal of Navigation and Port Research*, 38 (4), 399-407.

Cha, S. H., & Noh, C. K (2018). A Study on Application of Yard Transportation Equipment Automation System in the Container Terminal, *Journal of Navigation and Port Research*, 42 (3), 217-226.

Chung, C. Y., & Shin, J. Y. (2011). Efficient Yard Operation for the Dual Cycling in Container Terminal, *Journal of Navigation and Port Research*, 35 (1), 71~76.

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.

Ha, T. Y., & Choi, T. S. (2004). Analysis of Combined Productivity at Automated Container Terminal Using Simulation, *The Korean Operations Research and Management Science Society*, 643-646.

Imai, A., Sasaki, K., Nishimura, E., & Papadimitriou, S. (2006). Multi objective Simultaneous Stowage and Load Planning for a Container Ship with Container Rehandle in Yard Stacks, *European Journal of Operational Research*, 171(2), 373-389.

Jeong, Y. H., Kim, K. H., Woo, Y. J., & Seo, B. H. (2012). A Simulation Study on a Workload based Operation Planning Method in Container Terminal, *Industrial Engineering and Management Systems*, 11(1), 103-113.

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., Ryu, K. R., & Kim, K. H. (2006). Deriving Stacking Strategies for Export Containers with Uncertain Weight Information, *Journal of Intelligent Manufacturing*, 17, 399-410.

Kim, B. S., Kim, J. H., Fibrianto, H. Y., & Hong, S. D. (2018). A Remarshalling Buffer Location Model in a Rail-based Container Terminal, *Korean Journal of Logistics*, 26 (3), 1-16.

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, K. H., & Park, Y. M. (1996). A slot assignment method in the container yard for export containers considering their weights, *Journal of the Korean Institute of Industrial Engineers*, 22 (4), 753-770.

Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving Decision Rules to Locate Export Containers in Container Yards, *European Journal of Operational Research*, 124, 89-101.

Kim, K. H., Won, S. H., Yang, C. H., Kim, Y. H., & Bae, J. U. (2001). Evaluation of Yard Layouts of Automated Container Terminals by Using the Simulation, *International Journal of Management Science* 1(1), 418-421.

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

Lee, S. W. (2011), A Genetic Algorithm for the Container Pick-Up Problem, *IE Interface*, 24(4), 362-372.

Mohammad Bazzazi, Nima Safaei & Nikbakhsh Javadian(2009). A genetic algorithm to solve the storage space allocation problem in a container terminal, *Computers & Industrial Engineering* 56(1), 44–52.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis*, San Francisco, USA: 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, Y. K. (2016). Remarshalling Plan Using Neighboring Bay in Container Terminal, *Journal of Korean Navigation and Port Research,* 40(3), 113-120.

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.

Seo, J. H., Yi, S. H. & Kim, K. H., (2018). A Simulation Study on the Deadlock of a Rail-Based Container Transport System, *J. Navig. Port Res*. 42(1), 47-56.

Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage Space-Allocation in Container Terminals, *Transportation Research Part B*: *Methodological*, 37(10), 883-903.