대한조선학회논문집
Journal of the Society of Naval Architects of Korea
pISSN:1225-1143, Vol. 50, No. 5, pp. 298-306, October 2013
eISSN:2287-7355, http://dx.doi.org/10.3744/SNAK.2013.50.5.298

운송 경로 손상을 고려한 트랜스포터의 최적 블록 운송 경로 계획

허예지[1] · 차주환[2,†] · 조두연[3] · 송하철[3]
국립목포대학교 선박해양공학과 대학원[1]
국립목포대학교 해양시스템공학과[2]
국립목포대학교 조선공학과[3]

Optimal Block Transportation Path Planning of Transporters considering the Damaged Path

Ye-Ji Heo[1] · Ju-Hwan Cha[2,†] · Doo-Yeoun Cho[3] · Ha-Cheol Song[3]
Graduate School, Dept. of Naval Architecture and Marine Engineering, Mokpo National University[1]
Dept. of Ocean Engineering, Mokpo National University[2]
Dept. of Naval Architecture and Marine Engineering, Mokpo National University[3]

Nowadays, a transporter manager plans the schedule of the block transportation by considering the experience of the manager, the production process of the blocks and the priority of the block transportation in shipyard. The schedule planning of the block transportation should be rearranged for the reflection of the path blocking cases occurred by unexpected obstacles or delays in transportation. In this paper, the optimal block transportation path planning system is developed for rearranging the schedule of the block transportation by considering the damaged path. A* algorithm is applied to calculate the new shortest path between the departure and arrival of the blocks transported through the damaged path. In this algorithm, the first node of the damaged path is considered as the starting position of the new shortest path, and then the shortest path calculation is completed if the new shortest path is connected to the one of nodes in the original path. In addition, the data structure for the algorithm is designed. This optimal block transportation path planning system is applied to the Philippine Subic shipyard and the ability of the rapid path modification is verified.

Keywords : Transporter(트랜스포터), Block transportation(블록 운송), Shortest path(최단 경로), Optimal path planning(최적 경로 계획)

1. 서 론

선박 및 해양구조물 건조 시 정해진 일정 계획에 맞추어 생산 공정이 진행되어야 한다. 특히 트랜스포터를 효율적으로 운용하여 조선소 내의 대표적 물류인 블록의 운송 시간을 단축하고 운송 비용을 감소하며 물류 흐름을 원활하게 하는 것이 중요하다. 따라서 조선소에서는 트랜스포터 운용 담당자의 경험 및 블록의 공정 단계와 우선순위를 바탕으로 트랜스포터에 블록을 할당하여 운송 계획을 세우고 있다. 그러나, 현장에서 트랜스포터의 운송 상태 및 경로 등을 정확하게 파악하는 것이 어려우며, 운송 경로에 예상치 못한 장애물이 발생하거나 갑작스런 트랜스포터의 고장으로 인해 다른 트랜스포터의 경로를 막는 등 블록 운송 계획에 차질이 생기는 경우가 많아 트랜스포터를 효율적으로 운용하는 것이 어려운 실정이다. 이러한 상황에 능동적으로 대처하기 위해서는 운송 경로 손상을 고려한 트랜스포터의 최적 블록 운송 계획이 필요하다. 즉, 예상치 못한 경로의 변경에 따라 블록 운송 계획을 재수행하여, 트랜스포터의 최단 운송 및 공주행 경로를 재계산할 수 있는 시스템이 요구되고 있다. 이 때, 다양한 경로와 운송 계획에 대해 평가하여 블록을 운송하고 있는 트랜스포터 운용 기사에게 변경된 경로와 계획을 즉각적으로 전달해야 하므로, 조선소에서는 1~2초 이내의 경로 재계산 시간을 요구하고 있다.

2. 관련 연구 현황

Joo, et al. (2005)은 트랜스포터의 블록 운송 계획을 세우기 위해, 비슷한 중량을 가진 블록들을 그룹으로 묶어 각 그룹마다 한 종류의 트랜스포터로 운송하도록 가정하였다. 그리고 각 그룹에서 최단 거리에 있는 블록들을 순서대로 운송시키는 휴리스틱

접수일 : 2012년 9월 4일 | 1차 수정일 : 2013년 2월 27일 | 게재확정일 : 2013년 8월 27일
† Corresponding author : jhcha@mokpo.ac.kr, 061-450-2733

알고리즘을 적용하였다. 그러나 이러한 방법은 적재 능력이 다양한 트랜스포터를 고려하기 어려우며, 운송 블록의 순서, 트랜스포터 간의 관계, 교차 주행, 운송 경로 손상을 고려하지 않았다.

Yim, et al. (2008)과 Roh and Cha (2011)은 복수 트랜스포터의 공주행 거리 최소화를 고려한 블록 운반 계획 문제를 최적화 문제로 정식화 하였고, 개미 알고리즘과 유전 알고리즘을 적용하여 최적 해를 구하였다. 한편, 트랜스포터의 교차 주행 여부를 고려하지 않았고, 사용자 인터페이스가 부족하며, 운송 경로 손상을 반영하기 어렵다.

Cha, et al. (2012)은 사용자의 편의성을 고려한 완성도 있는 최적 블록 운송 경로 계획 시스템을 개발하였으며, 트랜스포터의 교차 주행 여부를 고려하여 계산 시간을 단축시키기 위한 특화된 알고리즘을 구현하였다. 그러나 운송 경로 손상을 고려하지 않았다.

Joo and Kim (2007)은 AGV(Automated Guided Vehicle)의 최단 경로를 계산하기 위해 각 노드의 위치와 요청 받은 작업장까지의 직선 거리를 평가 함수로 하는 A* 알고리즘을 사용하였으며, AGV의 속도를 고려하여 AGV 간의 충돌을 예측하고 회피하는 방법을 연구하였다. 그러나, AGV 운송 특성상 미리 예측할 수 없는 경로 변경은 일어나지 않으므로 이에 대한 고려는 하지 않았다.

Bock and Hoberg (2007)은 생산 공장의 생산품의 이동 경로를 고려하여 기계들을 최적 배치하는 연구를 수행하였다. 격자 눈금 위에 모든 기계들을 배치하고 각 기계의 출구에서 다음 기계로의 입구 방향으로 격자 눈금을 진행시키면서 최단 경로를 계산한다.

Hofner and Schmidt (1995)는 자동 청소 로봇의 기하학적 형상과 초기 위치, 환경 변화 등을 감지하여 이동 경로를 계산하는 연구를 수행하였다. 청소 로봇의 이동 패턴을 몇 가지로 정의하여 주어진 면적을 모두 지나도록 이동 경로를 계산하였다.

Table 1 Comparison of related works

Item	Joo, et al. (2005)	Yim, et al. (2008)	Cha, et al. (2012)	This paper
Transportation sequence	O	O	O	O
Unload distance minimization	O	O	O	O
Block weight	△	O	O	O
Multi transporter	X	O	O	O
Path search	X	O	O	O
Intersection	X	X	O	O
Damaged path	X	X	X	O

본 논문에서는 Yim, et al. (2008)과 Cha, et al. (2012)의 연구를 토대로 기존의 연구들이 가진 한계를 극복하기 위해 미리 예측하지 못한 운송 경로 손상 발생 시 이를 고려하여 경로를 재계산 할 수 있는 최적 블록 운송 경로 계획 시스템을 개발하였다. 이를 위해 출발지와 도착지를 일대일 연결하여 최단 경로를 계산할 수 있는 A* 알고리즘을 적용하였으며, 조선소의 운송 경로가 격자형에 가깝다는 점을 착안하여 격자형 거리 추정에 적합한 맨해튼 (Manhattan)방식을 A* 알고리즘의 거리 추정에 적용하였다는 점에서 Joo and Kim (2007)의 연구와 차이점이 있다. 또한, 운송 경로를 빠르게 계산하기 위해 A* 알고리즘의 시작/종료 조건을 수정하였고 이를 위한 자료구조를 제안하였다. 조선소 내 트랜스포터의 블록 운송과 관련된 연구를 비교하면 Table 1과 같다.

3. 트랜스포터의 최적 블록 운송 문제

3.1 최적 블록 운송 계획 절차

블록 운송 계획을 수립하기 위해서는 우선 트랜스포터의 적재중량, 공주행 및 주행 속력, 작업 가능 시간 등의 트랜스포터 정보와 운송해야 하는 블록의 중량 및 출발/도착지 등의 블록 정보, 그리고 트랜스포터가 이동하는 조선소의 경로(Layout)에 대한 정보가 필요하다 (Fig. 1의 ①; Yim, et al., 2008).

그 다음, 각 블록의 출발지와 도착지 간의 최단 경로, 각 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로를 미리 계산해 놓는다(Fig. 2의 ②). 이 때, 기존 연구 (Yim, et al., 2008)에서는 다익스트라(Dijkstra) 알고리즘을 사용하였고, 본 연구에서는 A*(A star) 알고리즘을 사용하였으며, 최단 경로 계산과는 별개로 추후 최적 할당이 진행되므로 최단 경로 계산 결과가 같다면 최단 경로 계산 알고리즘과 무관하게 최적 할당 결과도 같다.

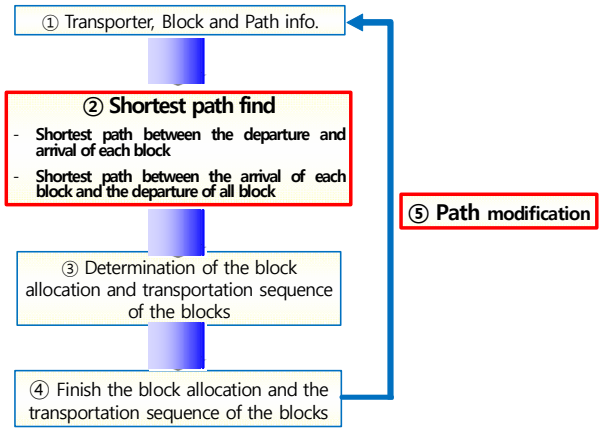

Fig. 1 The process for optimal block transportation scheduling

그 후에는 트랜스포터 별 블록 할당 및 블록 운송 순서를 결정하기 위한 최적화 문제를 정의한다. 목적함수는 트랜스포터의 공주행 시간을 최소화하면서 블록들을 운송 완료 제한 시각보다 가능한 일찍 운송하는 것이다 (Yim, et al., 2008). 제약조건으로는 운송 블록의 무게가 트랜스포터의 적재 능력보다 작아야 한다는 것, 모든 블록은 운송 완료 제한 시각 내에 운송을 완료해야 한다는 것, 모든 블록은 운반 시작 가능 시각 이후에 운송을 시작한다

는 것, 우선 순위에 따라 블록을 운송해야 한다는 것, 모든 트랜스포터는 서로 교차 주행할 수 없다는 것이 고려되어 있다. 패널티 방법을 사용하여 제약조건들을 목적함수에 반영하였다. 처음 3개의 제약 조건의 경우 블록의 중량과 적재 능력 간의 부등식, 제한 시각과 실제 운송 시작/완료 시각 간의 부등식을 비교하여 위배하는 경우 패널티를 부여하는 방식을 사용하였다 (Yim, et al., 2008). 마지막 제약 조건의 경우 운송 경로의 모든 간선(edge)마다 해당 블록이 간선에 진입한 시각과 간선을 빠져나간 시각을 자료구조로 저장하여 비교함으로써, 같은 시간대에 같은 간선을 이동한 블록이 있을 경우 패널티를 부여하는 방식을 사용하였다 (Cha, et al., 2012). 개미가 자주 다니는 길에 페로몬이라는 호르몬의 일종을 남기는 것을 착안하여 가상 개미를 주행시키는 휴리스틱 최적화 방법인 개미 알고리즘을 사용하여 초기 블록 운송 계획을 수립하고, 이를 초기해 집합으로 가정한 후, 유전 알고리즘을 수행하여 최종 블록 운송 계획을 수립한다(Fig. 1의 ③; Yim, et al., 2008). 지정된 반복회수만큼 계산하여 결과를 제공하고, 만약 결과가 만족스럽지 못하다고 판단될 경우 이어서 수행하도록 한다.

위의 과정을 통하여 결정된 블록 운송 계획에 따라 트랜스포터가 블록을 운송하게 된다(Fig. 1의 ④; Yim, et al., 2008). 이 때, 블록 운송 도중 장애물이 발생하거나, 예상치 못한 작업 지연, 트랜스포터의 고장으로 인해 운송 경로가 막히게 되는 경우에는 다른 모든 트랜스포터의 블록 운송 계획이 영향을 받게 된다. 따라서, 다시 첫 번째 단계부터 재계산을 수행해야 한다(Fig. 1의 ⑤).

3.2 블록 간의 최단 경로 계산 과정

블록 간 최단 경로 계산 과정은 두 가지로 구분된다. 첫 번째는 각 블록의 출발지와 도착지 간의 최단 경로를 계산하는 것이고, 두 번째는 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로를 계산하는 것이다. 이에 대해 예시를 통해 설명하고자 한다.

3.2.1 각 블록의 출발지와 도착지 간의 최단 경로 계산

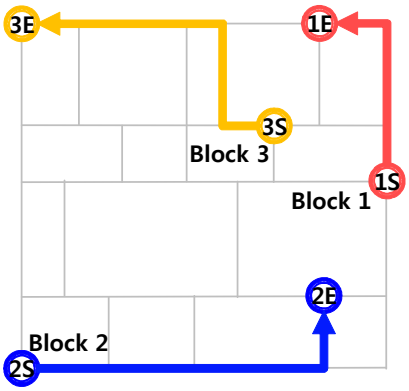

Fig. 2 An example of calculating minimum path between the start and end points of each block

Fig. 2는 3개의 블록에 대해 각 블록의 출발지와 도착지 간의 최단 경로 계산 예를 보여준다. 트랜스포터가 각 블록을 출발지에서 싣고 도착지까지 운송하는 경로를 계산하는 것이다.

각 블록의 출발지에는 “S”, 도착지에는 “E”라고 표시되어 있다. 1번 블록의 출발지(1S)에서 도착지(1E)까지, 2번 블록의 출발지(2S)에서 도착지(2E)까지, 3번 블록의 출발지(3S)에서 도착지(3E)까지, 총 3번의 최단 경로 계산을 수행해야 한다. 즉, n개 블록의 출발지와 도착지 간의 최단 경로를 계산하기 위해서는 총 n번의 계산을 수행해야 한다.

3.2.2 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로 계산(공주행 최단 경로 계산)

Fig. 3은 3개의 블록에 대해 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로 계산 예를 보여준다. 만약 Fig. 3의 (a)와 같이 1번 블록을 도착지까지 운송한 뒤에 2번 블록을 운송하려고 2번 블록의 출발지로 트랜스포터가 이동해야 하는 상황을 가정하자. 최단 경로로 이동하기 위해 1번 블록의 도착지와 2번 블록의 출발지 간의 최단 경로를 계산해야 한다. 이 때에는 트랜스포터가 블록을 싣지 않고 이동하기 때문에 이를 공주행 경로라고 한다.

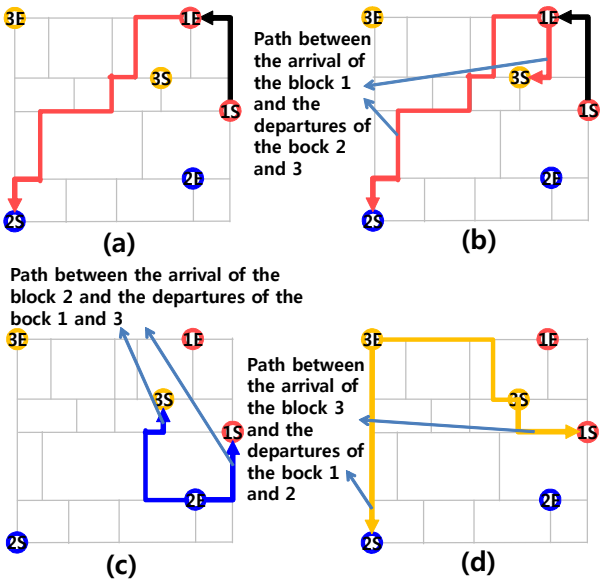

Fig. 3 An example of calculating minimum path between the end points of all blocks and the start points of other blocks

그러나 Fig. 3의 (b)와 같이 1번 블록을 운송한 뒤에 2번 블록을 운송하러 이동하는 것이 공주행 경로를 단축시킬 수 있는지, 3번 블록을 운송하러 이동하는 것이 공주행 경로를 더욱 단축시킬 수 있는지 계산을 수행해 보기 전에는 확인할 수 없다. 즉, 1번 블록의 도착지(1E)와 다른 모든 블록의 출발지(2S, 3S) 간의 최단 경로를 모두 계산해야 한다. 또한 블록 운송 순서는 세 번째 단계(Fig. 1의 ③)에서 계산되므로 어느 블록을 먼저 운송할지 아

직 결정되지 않았다. 따라서 Fig. 3의 (c), (d)와 같이 각각 모든 블록에 대해 이와 같은 계산을 수행해야 한다. 즉, 총 6번에 걸쳐 최단 경로를 계산해야 한다. 즉, n개 블록에 대해 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로를 계산하기 위해서는 총 n(n-1)번의 계산을 수행해야 한다.

4. A* 알고리즘 기반의 블록의 최단 운송 경로

조선소에서 트랜스포터가 이동하는 경로는 주로 격자형으로 되어 있다. 그럼에도 불구하고 기존 연구에서는 비격자형 경로에 유리한 다익스트라 알고리즘을 사용하였기에 본 연구에서는 격자형 경로에 유리한 A* 알고리즘을 적용하였다. 또한 기존 연구에서 사용한 다익스트라 알고리즘은 일대다 대응에 적합한 방법이기에, 각 블록의 출발지와 도착지 간의 최단 경로 계산, 손상 운송 경로의 재계산과 같은 일대일 대응에는 적합하지 않다. 이에 대해 Table 2에서 비교하였으며, 이와 별개로 본 장에서는 특정 예제를 통해 명확한 계산량의 차이를 확인하였다.

Table 2 Comparison between Dijkstra and A* algorithms

Comparison items	Dijkstra algorithm	A* algorithm
Efficient node correspondence	One-to-several	One-to-one
Search method	Search all nodes from the departure node	Search neighbor nodes between the departure node and the arrival node
Efficient path	Non-grid (straight)	Grid
Resource usage	High	Low

4.1 기존 연구의 다익스트라 알고리즘을 이용한 블록 최단 운송 경로 계산

다익스트라 알고리즘은 P개의 절점(node)과 E개의 간선(edge)으로 이루어진 그래프(graph)에서 하나의 출발지와 출발지를 제외한 다른 모든 절점 간의 최단 경로를 구하는 알고리즘이다 (Dijkstra, 1959). 출발지에서 시작하여 인접 절점부터 모든 절점으로 확장해 나가면서 출발지와 모든 절점 간의 최단 경로를 계산한다. 기존 연구 (Yim, et al., 2008)는 최단 경로 계산에 다익스트라 알고리즘을 사용하였다.

4.1.1 기존 연구의 다익스트라 알고리즘을 이용한 각 블록의 출발지와 도착지 간의 최단 경로 계산

우선, 1번 블록의 출발지(1S)와 도착지(1E)간의 최단 경로를 계산해 보자. 다익스트라 알고리즘을 사용하면 1번 블록의 출발

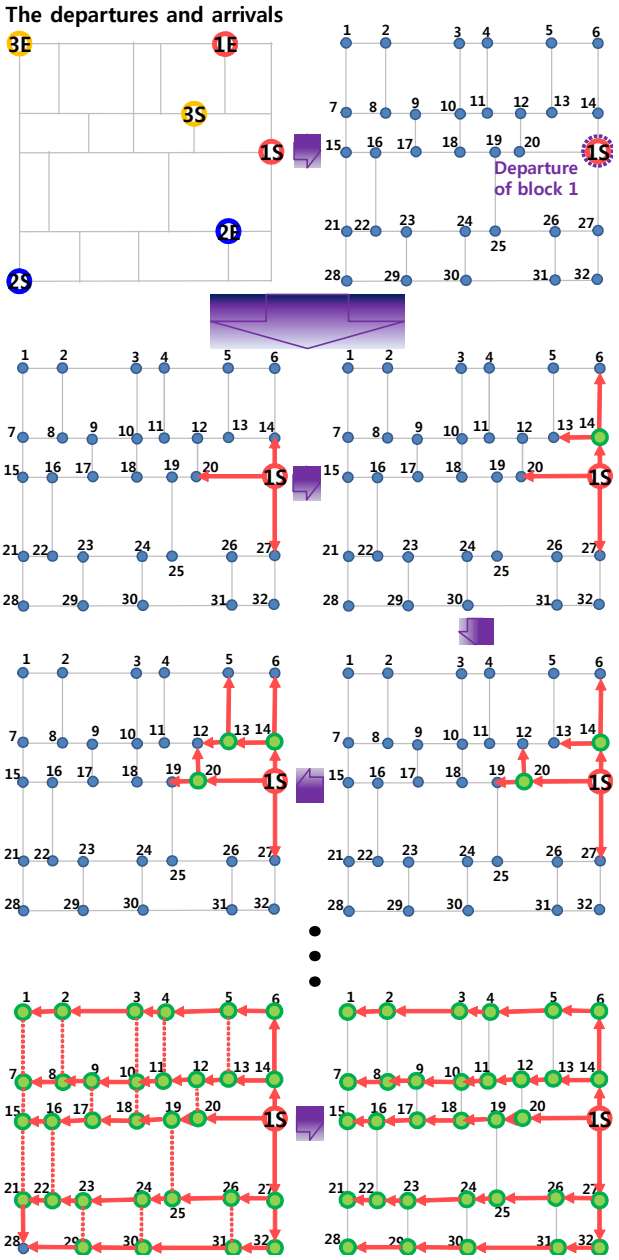

Fig. 4 The path finding between the start and end points of each block using Dijkstra algorithm

지(1S)에서 시작하여, 인근의 14, 20, 27번 절점들의 최단 거리를 업데이트한다. 그리고 최단 거리가 가장 작은 절점, 즉 14번 절점의 인근 절점인 6, 13번 절점들의 최단 거리를 업데이트한다. 이러한 방식으로 32번 절점까지 모든 절점들의 최단 거리 및 최단 경로를 계산하게 된다. 즉, 1번 블록의 도착지인 5번 절점까지의 최단 경로를 계산하기 위해 몇 개의 절점들에 대해서만 최단 경로를 계산하면 되는데, 불필요한 다른 절점들의 최단 경로까지 계산하는 것이다. 절점 및 간선의 개수가 증가할수록 불필요한 계산은 기하급수적으로 증가할 수 밖에 없다.

나머지 2, 3번 블록의 출발지인 2S, 3S에 대해서도 각각의 도착지인 2E, 3E까지의 최단 경로를 계산할 때에도 모든 절점까지의 최단 경로를 계산하므로 불필요한 계산은 블록의 개수에 비례

하여 증가한다. 이를 수식으로 정확히 표현하기는 어려우나, 다익스트라 알고리즘의 시간 복잡도가 E + P log P임을 고려하면 (Barbehenn, 1998), 약 n (E + P log P)라고 할 수 있다.

4.1.2 기존 연구의 다익스트라 알고리즘을 이용한 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로 계산

다익스트라 알고리즘을 사용하면 1번 블록의 도착지(1E)에서 다른 모든 블록의 출발지(2S, 3S)뿐만 아니라 다른 모든 절점까지의 최단 경로를 계산하게 된다. 즉, 1번 블록의 도착지에서 2, 3번 블록의 출발지까지의 몇 개 절점들에 대해서만 최단 경로를 계산하면 되는데, 불필요한 다른 절점들의 최단 경로까지 계산하는 것이다. 이러한 현상은 절점 및 간선의 개수가 증가할수록 커질 것이며, 나머지 2, 3번 블록의 도착지와 다른 모든 블록의 출발지를 계산할 때에도 반복된다.

4.2 본 논문의 A* 알고리즘을 이용한 블록 최단 운송 경로 계산

A* 알고리즘은 하나의 출발지와 하나의 도착지 간의 일대일 최단 경로를 계산하는 알고리즘이다 (Hart, et al., 1968). 출발지에서 시작하여 인접 절점으로 확장해가면서 출발지로부터의 실제 거리와 도착지까지의 추측 거리의 합을 계산하여, 그 값이 최소인 절점으로 계속 확장해 가는 방법이다.

4.2.1 본 논문의 A* 알고리즘을 이용한 각 블록의 출발지와 도착지 간의 최단 경로 계산

A* 알고리즘 이용하여 1번 블록의 출발지(1S)와 도착지(1E) 간의 최단 경로 계산 과정을 살펴보자.

i) 우선 출발지와 절점 간의 실제 거리를 G, 도착지와 절점 간의 추측 거리를 H, 실제 거리와 추측 거리의 합을 평가 거리 F라고 정의 하자. 이때, 추측 거리는 현재 절점과 도착지 간의 x좌표 거리의 차와 y좌표 거리의 차의 합을 휴리스틱 함수로 사용하는 맨해튼 (Manhattan)방식으로 계산한다.

ii) Fig. 5의 (a)는 1번 블록의 출발지(1S)의 인접 절점인 14번, 20번, 27번 절점에 대한 계산 과정을 보여준다. 먼저, 1번 블록의 출발지(1S)와 14번 절점 간의 실제 거리(G)인 2와 1번 블록의 도착지(1E)와 14번 절점 간의 추측 거리(H)인 7을 합하여 평가 거리(F)를 계산하면 9를 구할 수 있다. 20번, 27번 절점도 같은 방법으로 평가 거리(F)를 계산하면, 각각 13, 20을 구할 수 있다. 여기에서 가장 작은 평가 거리(F)를 갖는 절점인 14번 절점을 선택하여 위의 과정을 반복한다.

iii) 14번 절점의 인접 절점인 6번과 13번 절점에 대해 평가 거리(F)를 계산하여 가장 작은 평가 거리(F)를 갖는 6번 절점을 선택하게 된다(Fig. 5의 (b)).

iv) 6번 절점에서 다시 위의 과정을 반복해야 하나 인접 절점인 5번이 목표 절점인 도착지 이므로 5번 절점을 선택하면서 최단 경로 계산을 마친다. 이처럼 인접 절점 정렬 과정 중에 목표 절점인 도착지가 나오게 되면, 도착지를 선택하면서 최단 경로 계산을 마치게 된다. 만약 같은 최단 거리 값을 갖는 최단 경로가 다수일 경우 방향 전환이 최소인 최단 경로를 택하게 되며, 방향 전환 개수까지 동일한 경우 임의의 최단 경로를 택하게 된다.

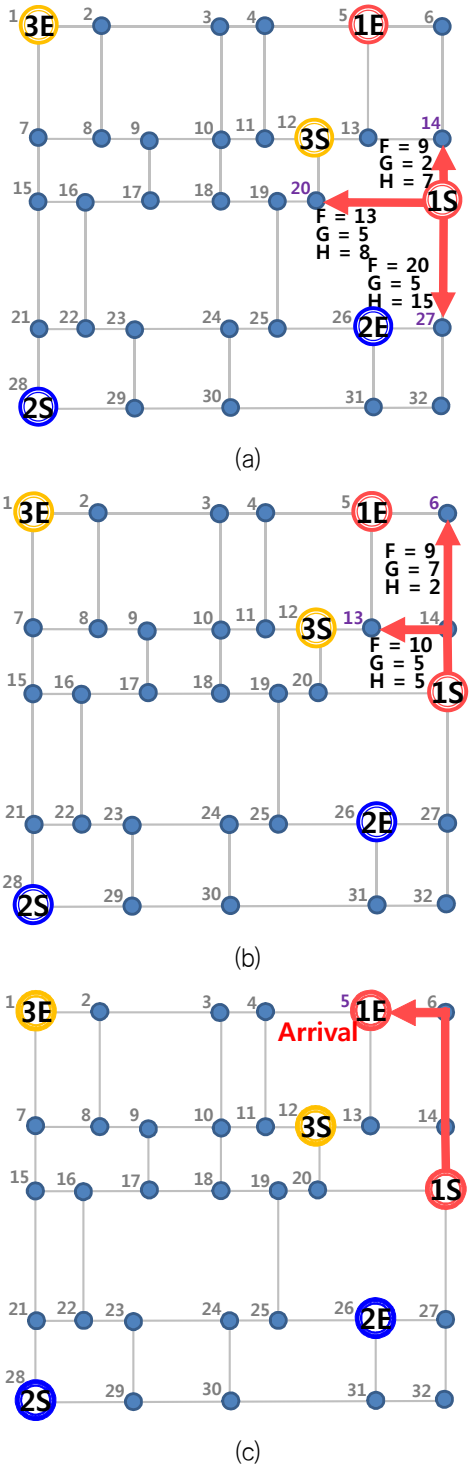

Fig. 5 The path finding between the start and end points of each block using A*algorithm in this paper

1번 블록의 출발지와 도착지 간의 최단 경로 계산 시 다익스트라 알고리즘을 사용할 경우, 1번 블록의 출발지에서 시작하여 1번 블록의 도착지뿐만 아니라 다른 모든 절점들까지의 불필요한 최단 경로를 계산하는 것에 비해, A* 알고리즘을 사용할 경우 1번 블록의 출발지에서 시작하여 1번 블록의 도착지까지 가는 인근의 절점들에 대해서만 최단 경로를 계산함을 알 수 있다. 정량적으로 정확히 비교하기는 어려우나 다익스트라 알고리즘에 비해 A* 알고리즘이 이러한 문제의 경우 더욱 효율적임을 확인할 수 있다.

4.2.2 본 논문의 A* 알고리즘을 이용한 모든 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로 계산

A* 알고리즘을 이용하여 1번 블록의 도착지(1E)와 다른 모든 블록의 출발지(2S, 3S) 간의 최단 경로 계산을 위해 1번 블록의 도착지에서 시작하여 2, 3번 블록의 도착지까지 가는 인근의 절점들에 대해서만 최단 경로를 계산하면 되므로, 다른 모든 절점까지의 불필요한 최단 경로를 계산하는 다익스트라 알고리즘에 비해 더욱 효율적임을 알 수 있다. 2, 3번 블록의 도착지와 다른 모든 블록의 출발지 간의 최단 경로를 계산할 때에도 마찬가지로 효율적이다.

다익스트라 알고리즘이 A* 알고리즘에 비해 무조건적으로 비효율적이라는 것은 아니다. 다익스트라 알고리즘은 일대다 대응에 적합한 알고리즘이므로, 한 절점에서 시작하여 매우 많은 절점까지의 최단 경로를 계산하는 문제에는 A* 알고리즘에 비해 매우 효율적이지만, 본 논문의 블록 운송 문제와 같이 일대일 대응, 또는 일대다 대응이라 할 지라도 소수 절점에 대한 최단 경로 계산에는 A* 알고리즘이 더욱 효율적이다.

5. A* 알고리즘 기반의 운송 경로 손상 알고리즘 수정

조선소에서는 장애물이 발생 또는 다른 트랜스포터의 고장 및 작업 지연으로 인해 특정 경로가 막혀있는 상황, 즉 손상된 간선(이하 손상 간선이라 함)에 의해 블록 운송 경로를 변경해야 하는 상황이 발생할 수 있다. 본 논문에서는 손상 간선에 의한 경로 변경(Fig. 1의 ⑤)을 고려하여 블록 운송 계획을 재수립 하기 위해 A* 알고리즘을 수정 및 적용하였다.

5.1 기존 연구의 다익스트라 알고리즘을 이용한 경로 변경

Fig. 6과 같이 모든 간선에 번호가 부여되어 있을 때 ⑨번 간선이 손상된 경우를 가정해 보자. 우선, 모든 운송 경로 중에서 손상 간선 ⑨번을 지나는 경로가 있는지 확인을 해야 한다. 총 n개의 블록에 대한 운송 경로는 각 블록의 출발지와 도착지 간의 최단 경로의 개수인 n개와 모든 블록의 도착지와 다른 모든 블록의 도착지 간의 최단 경로의 개수인 n(n–1)개를 합하여, 총 n + n(n–1)개가 존재하며, 각 운송 경로를 구성하는 모든 간선에 대해 확인을 해야 하기 때문에 많은 시간이 소요된다. 이렇게 손상 간선 ⑨번을 지나는 운송 경로를 찾으면, 1E → 2S, 3S → 3E, 1S → 3E 등이다. 이 중, Fig. 7과 같이 1E → 2S의 운송 경로를 자세히 살펴보자.

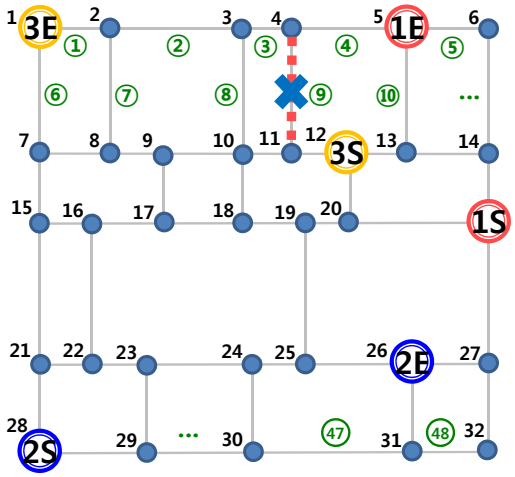

Fig. 6 The edge numbering and the damaged edge(⑨)

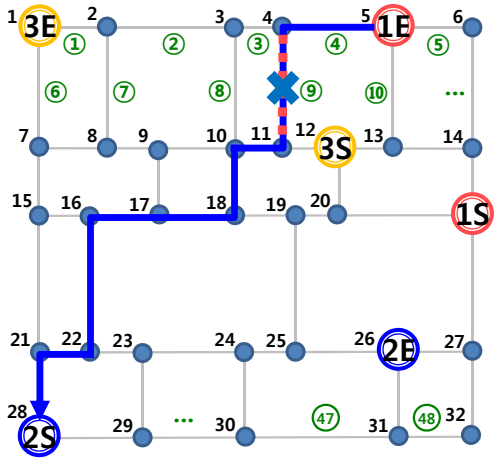

Fig. 7 The path between 1E and 2S through the damaged edge(⑨)

이제 손상 간선 ⑨번을 제거 한 후에 다시 1E와 2S 간의 최단 경로 계산을 위해 다익스트라 알고리즘을 수행해야 하며, 1E에서 시작하여 2S 뿐만 아니라 모든 절점으로의 최단 경로를 계산하게 된다. 1개의 최단 경로만 재계산하면 되는 상황에서 모든 절점으로의 최단 경로를 계산해야 하기 때문에 불필요한 많은 시간이 소요되며, 나머지 3S → 3E, 1S → 3E 에 대해서도 이와 같은 불필요한 계산을 수행해야 한다.

5.2 본 논문의 A* 알고리즘을 이용한 경로 변경

기존 연구에는 특정 간선이 어떤 운송 경로에 포함되어있는지

에 대한 정보가 없었기 때문에 특정 간선이 손상될 경우 모든 운송 경로를 다 찾아보면서 해당 간선의 경유 여부를 확인해야만 하였고 따라서 많은 계산 시간이 소요된다.

본 논문에서는 손상 간선을 지나는 운송 경로를 빠르게 확인하기 위해 Table 3과 같이 모든 간선에 해당 간선을 지나는 운송 경로를 저장하는 자료구조를 제안하였다. 5.1절과 같이 ⑨번 간선이 손상되었을 때, Table 3의 자료 구조를 통해 해당 손상 간선을 지나는 운송 경로가 1E → 2S, 3S → 3E, 1S → 3E 등이라는 것을 곧바로 확인할 수 있다.

Table 3 The data structure of the path through each edge

Edge No.	Node No.	Transportation path
1	1, 2	3S→3E, 1S→3E, 3E→1S, 3E→3S, …
2	2, 3	3S→3E, 1S→3E, 3E→1S, 3E→3S, …
3	3, 4	3S→3E, 1S→3E, 3E→1S, 3E→3S, …
4	4, 5	1E→2S, 2S→1E, …
5	5, 6	1S→1E, 1E→1S, …
6	1, 7	2S→3E, 3E→2S, …
7	2, 8	…
8	3, 10	…
9	**4, 11**	**1E→2S, 3S→3E, 1S→3E, 2S→1E, 3E→1S, 3E→3S, …**
⋮	⋮	⋮
47	30, 31	2S→2E, 2E→2S, …
48	31, 32	…

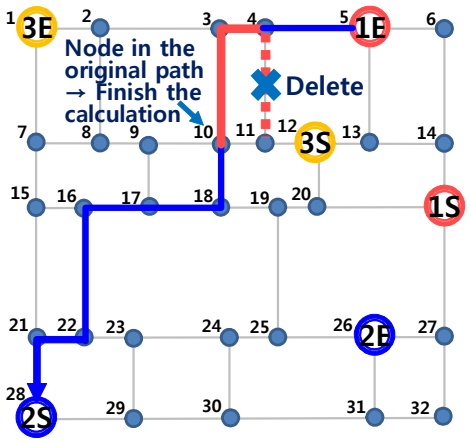

Fig. 8 The reconstruction of the shortest path between 1E and 2S using A* algorithm in this paper

Fig. 8과 같이 1E → 2S의 운송 경로를 자세히 살펴 보자. 손상 간선 ⑨번을 제거한 후에 손상 간선 직전 절점인 4번 절점을 출발지로 하고 2E를 도착지로 하여 A* 알고리즘을 수행한다. 최단 경로 계산을 재수행하면 4번에서 3번을 거쳐 10번 절점으로 최단 경로가 계산되는데, 10번 절점은 이전 최단 경로를 구성하는 절점이다. 이렇게 이전의 경로로 돌아오게 되면 계산을 완료한다. 이와 같이 모든 절점이 아닌 도착지까지의 인접 절점만을 재계산하기 때문에 다익스트라 알고리즘에 비해 계산 시간을 단축시킬 수 있다.

또한, 손상이 간선간의 교점인 절점에서 발생한 경우에는, 해당 절점을 지나갈 수 없으므로 절점과 연결 된 모든 간선이 손상되었다고 가정 후 최단 운송 경로 계산을 수행한다.

6. 적용 결과 비교

6.1 필리핀 수빅 조선소의 블록 운송 실적 정보 및 최적 블록 운송 경로 계획 결과

Fig. 9는 본 논문의 결과를 필리핀 수빅 조선소에 적용하기 위한 블록 운송 실적 정보이다. 운송 경로가 나타나 있는 수빅 조선소의 Layout과 2012년 1월 2일에 126개의 블록을 운송한 실적(Table 4, 5), 운송에 사용된 트랜스포터의 정보(Table 6) 등을

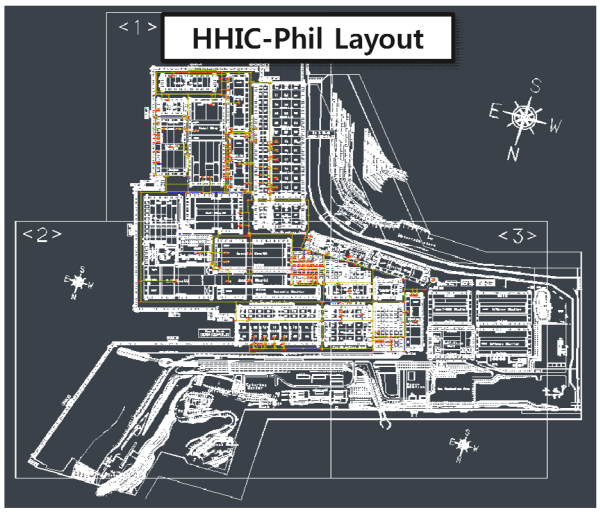

Fig. 9 The layout of the Subic shipyard in Philippines

Table 4 Block transportation data 1 of Subic shipyard in Philippines

No.	Location	
	From	To
1	BSA-C03:Block Stock-A-C-03	BPA-000:61BAY
2	BSA-C05:Block Stock-A-C-05	BPA-000:61BAY
3	PNA-320:32BAY	BPA-000:61BAY
⋮	⋮	⋮
125	BPB-C03:Blasting Painting Shop-B-C-03	BPB-0001:Blasting &Painting Shop-B
126	BPB-C03:Blasting Painting Shop-B-C-03	BPB-0001:Blasting &Painting Shop-B

수집하였다. 주어진 정보를 이용하여 수빅 조선소의 Layout에서 126개 블록의 출발지와 도착지를 지정하고, 해당 지점들을 조선소 Layout 상에서 연결하여 그래프를 만들었다. 블록의 운송 시간, 위치, 하중 등, 블록에 대한 정보를 본 논문에서 개발한 시스템에 입력하여 공주행 거리가 최소가 되는 계획을 세웠고, 실제 수빅 조선소에서 운송했던 실적에서의 공주행 시간을 분석해 두 결과를 비교하였다.

Table 5 Block transportation data 2 of Subic shipyard in Philippines

No.	EQUIPMENTS	USING TIME		Project/Block		
	Control #	Start	Finish	PROJECT	Block	Weight
1	TPX103H	1:00	4:00	NBP0073	1702	74.4
2	TPX103H	1:20	1:27	NBP0073	1701	47.1
3	TPX103H	1:56	2:10	NBP0075	3524	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
125	TPX205H	23:15	23:33	NTP0075	1209	28.2
126	TPX205H	23:33	23:48	NTP0075	1209	28.2

Table 6 Transporter data of Subic shipyard in Philippines

TP No.	TP Name	Max. ability	Unload speed	Load speed	Start time	End time
1	TPX101H	200	14	5	8:00	17:00
2	TPX101H	200	14	5	8:00	17:00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
11	TPX301H	700	11	4.5	8:00	17:00
12	TPX302H	700	11	4.5	8:00	17:00

Table 7과 같이 최적 블록 운송 경로 계획을 수행한 결과, 기존 운송 실적에 비해 공주행 시간이 52~58% 가량 감소되는 것을 확인할 수 있다.

Table 7 The result of the optimum block transportation simulation

	Num. of TP	Num. of block	Total trans. time	Unload trans. time	Ratio of Unload /total
Yard data	5(main) + 2(sub)	126	34h 39m	9h 26m	27%
Optimum data	5(main) + 2(sub)	126	29h 12m	4h 00m	14%

6.2 블록 최단 운송 경로 계산 시간 비교

기존 연구의 다익스트라 알고리즘을 사용하는 경우와 본 논문의 A* 알고리즘을 사용하는 경우 최단 운송 경로 계산 시간을 Table 8과 같이 비교하였다. Table 8의 운송 경로 손상 계산에서는 총 10개의 손상 경로가 있다고 가정하여 계산을 수행하였다. Fig.10은 10개의 손상 경로의 위치를 보여준다.

Table 8 Comparison of the calculating time between the Dijkstra and A* algorithms

Comparison items	Dijkstra algorithm (related work)	A* algorithm (thispaper)
The path between the departure and the arrival of each block	About 1 min.	Within 1 sec.
The path between the arrival of each block and the departure of all blocks	About 1 min.	52 sec.
Damage the transportation path	About 10 sec.	Within 1 sec.

계산에 사용된 컴퓨터의 사양은 i7-2세대 CPU, RAM 6MB이고, Win 7, Visual Studio 10.0 기반이다. 기존 연구에 비해 모든 블록의 도착지와 다른 모든 블록의 출발지의 계산 시간은 큰 차이가 없었다. 하지만 각 블록의 출발지와 도착지, 운송 경로 손상 시 경로 재계산 시간은 크게 단축하였음을 확인할 수 있으며, 운송 경로 손상 시 재계산을 수행하기에 충분하다고 판단된다.

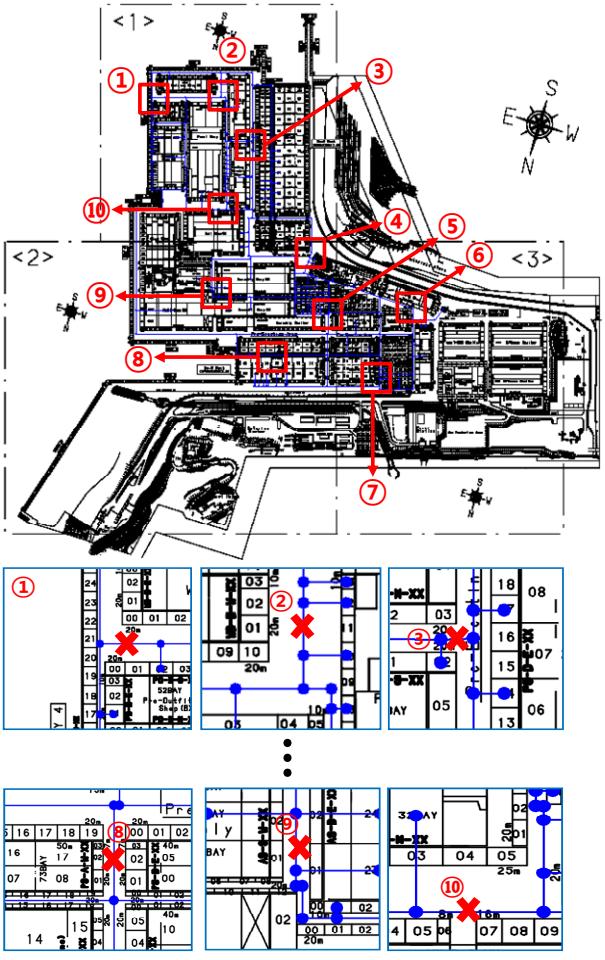

Fig. 10 Example of the damaged transportation path

7. 결론 및 향후 연구 계획

본 연구에서는 트랜스포터의 운송 경로 손상을 고려하여 블록의 최단 운송 경로 계산을 빠르게 수행하기 위해 A* 알고리즘을 수정하여 적용하였다. 필리핀 수빅 조선소의 126개 블록 운송 실적에 대해 다익스트라 알고리즘을 사용한 기존 연구와 계산 시간을 비교 분석하였다. 10개의 블록이 운송되어야 하는 경로가 손상되었을 경우, 기존 연구는 약 60초(약 1분) 가량의 최단 경로 계산 시간이 소요되었으나, 본 연구는 1초 이내에 최단 경로 계산이 완료되었다. 따라서 본 연구를 통해 운송 경로 손상을 반영한 블록의 최단 운송 경로 재계산 시 최적 블록 운송 경로 계획이 가능함을 확인하였다.

후 기

본 연구는 산업통상자원부 산업원천기술개발사업(10035331, 시뮬레이션 기반의 선박 및 해양플랜트 생산기술 개발), 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업(NIPA-2013-H0401-13-2006)의 연구결과로 수행되었음

References

Barbehenn, M., 1998. A Note on the Complexity of Dijkstra's Algorithm for Graphs with Weighted Vertices. *IEEE Transaction on computers*, 47(2), p.263.

Bock, S. & Hoberg, K., 2007. Detailed Layout Planning for Irregularly-Shaped Machines with Transportation Path Design. *European Journal of Operational Research*, 177(2), pp.693-718.

Cha, J.H. Cho, D.Y. Song, H.C. & Roh, M.I., 2012. Development and application of optimal block transportation simulation system of transporters in shipyard. *Proceedings of the Society of CAD/CAM Engineers Conference,* Pyeongchang, Korea, 1-3 February 2012.

Dijkstra, E.W., 1959. A Note on Two Problems in Connxion with Graphs. *Numerische Mathematik*, 1(1), pp.269-271.

Hart, P.E. Nilsson, N.J. & Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, 4(2), pp.100-107.

Hofner, C. & Schmidt, G.K., 1995. Path Planning and Guidance Techniques for an Autonomous Mobile Cleaning Robot. *Robotics and Autonomous Systems*, 14(2-3), pp.199-212.

Joo, C.M. Lee, W.S. & Lee, K.B., 2005. Transporter scheduling for block transportation in the shipyard. *Conference of Korean Operations Research and Management Science,* Cheongju, Korea, 13-14 May 2005.

Joo, Y.H. & Kim, J.S., 2007. Shortest Path Searching Algorithm for AGV Based on Working Environmental Model. *Journal of Fuzzy Logic and Intelligent Systems*, 17(5), pp.654-659.

Roh, M.I. & Cha, J.H., 2011. A Block Transportation Scheduling System considering a Minimisation of Travel Distance without Loading of and Interference between Multiple Transporters. *International Journal of Production Research*, 49(11), pp.3231-3250.

Yim, S.B. Roh, M.I. Cha, J.H. & Lee, K.Y., 2008. Optimal Block Transportation Scheduling considering the Minimization of the Travel Distance without Overload of a Transporter. *Journal of the Society of Naval Architects of Korea*, 45(6), pp.646-655.


허 예 지


차 주 환


조 두 연


송 하 철