한국 CAD/CAM 학회 논문집
제 16권 제 3호 2011년 6월 pp. 207-215

유전 알고리즘을 기반으로 한 조선소 블록 적치장의 재배치 최소화

노명일*, 임병석**

Minimization of the Rearrangement of a Block Stockyard Based on the Genetic Algorithm

Myung-Il Roh* and Byeong-Seog Im**

ABSTRACT

Due to its large size, a ship is first divided into scores of blocks and then each block is constructed through various shops, such as the assembly shop, the painting shop, and the outfitting shop. However, each block may not be directly moved to the next shop and may be temporarily laid on a block stockyard because the working time in each shop is different. If blocks are laid on the block stockyard without any planning, the rearrangement of the blocks by a transporter are required because the blocks have the different in and out time. In this study, an optimal layout method based on the genetic algorithm was proposed in order to minimize the rearrangement of the blocks in the block stockyard. To evaluate the applicability of the proposed method, it was applied to a simple layout problem of the block stockyard.

Key words : Block stock yard, Genetic algorithm, Optimization, Shipbuilding

1. 서 론

선박 및 해양 구조물의 선체는 크기가 크고 복잡하기 때문에 다수의 블록으로 분할하여 조립한 후에 각각의 블록들을 조립하여 완성한다. 이때, 각 블록은 완성되기까지 조립, 도장, 의장 등 다양한 공정을 거치게 된다. 그런데 각 공정에서의 작업 시간이 서로 다르기 때문에 공정을 마친 어떤 블록은 다음 공정으로 바로 가지 못하고 조선소 내 블록 적치장에 임시로 놓이게 된다. 블록 적치장에 블록들을 임시로 놓을 때 계획 없이 놓게 되면, 각 블록은 입고 시간과 출고 시간이 서로 다르기 때문에 적치장 내 블록들의 재배치 작업이 발생하게 된다. 예컨대, 지금 어떤 블록을 트랜스포터(transporter, 조선소 내에서 블록을 운반하기 위한 운송 수단)를 이용하여 적치장에서 출고하려고 하는데 방해가 되는 블록이 있을 경우 이를 먼저 다른 곳으로 옮겨 놓은 후 해당 블록을 출고해야 한다. 만약 이러한 방해 블록들이 많은 경우 해당 블록의 출고 시간은 점점 증가하게 되고 트랜스포터에 의한 불필요한 이동이 증가해 결국 선박 건조 공기를 증가시켜 조선소의 생산성에 큰 영향을 미치게 된다. 그러나 블록 적치장 내의 블록들을 입출고 시간을 고려하여 효과적으로 배치한다면 입출고 시 블록의 재배치 작업을 최소화할 수 있을 것이다.

조선소 내의 효율적인 블록 적치장 배치를 통해 생산성 향상과 생산 비용을 절감하려는 연구를 최근 현대중공업에서 수행한 바 있다. 박창규 등[1]은 블록 적치장에 놓인 각 블록에 지번을 할당하는 알고리즘과 적치장의 상황을 실시간으로 모니터링 할 수 있는 시각적인 기능을 개발하였다. 이외에 블록 적치장의 배치 문제와 관련된 연구로서, 컨테이너 터미널에서의 컨테이너 배치 문제에 대한 기존의 여러 가지 연구가 수행된 바 있다. 이준욱 등[2]은 마순차적으로 반입되는 수출 컨테이너의 배치 장소를 결정할 때 트랜스퍼 크레인(transfer crane, 컨테이너를 운반하기 위한 운송 수단)의 사용량을 최소화하고 컨테이너 적하 시 재배치를 최소화하기 위해 유전 알고리즘(genetic

*교신저자, 정회원, 울산대학교 조선해양공학부
**학생회원, 울산대학교 대학원 조선및해양공학과
- 논문투고일: 2010. 09. 11
- 논문수정일: 2011. 03. 04
- 심사완료일: 2011. 03. 09

algorithm)을 이용한 방법을 제안하였다. 이와 유사하게 최영진 등[3], 강재호 등[4]은 컨테이너 터미널에서 컨테이너의 재배치를 최소화하기 위한 휴리스틱 방법을 제안하였다. 이외에도 조선 해양 분야에서 배치 문제에 관련된 연구로서 한성남 등[5]은 유전 알고리즘을 기반으로 최적 공간 배치 알고리즘을 제안하였다.

본 연구에서는 블록 적치장 내 블록들을 효과적으로 배치하여 블록 입출고 시 불필요한 재배치 작업을 최소화하기 위해, 적치장 내 블록들의 최적 배치 문제를 수학적으로 정식화하였고, 이를 풀 수 있는 최적 배치 방법을 유전 알고리즘을 기반으로 제안하고 구현하였다. 마지막으로 제안된 방법의 효용성을 검토하기 위해 간단한 예제에 적용하여 그 결과를 평가하였다.

2. 블록 적치장의 최적 배치 문제

2.1 블록 적치장의 최적 배치 문제

본 연구에서 해결하고자 하는 문제는 블록 적치장 내 블록들의 최적 배치를 찾는 것이다. 이때 최적 배치의 목적은 블록 적치장 내에서 불필요하게 이동되는 블록의 수와 이동 거리를 최소화하는 것이다. 즉, 해당 블록의 이동 시 이동을 방해하는 블록의 수와 이동 거리를 최소화 하는 것이다.

블록 적치장은 작업 중인 블록들을 임시로 놓기 위한 공간으로서 조선소 야드에서 넓은 공간을 차지하고 있다. 일반적으로 블록 적치장은 내부를 몇 개의 구역으로 나누어 관리하게 된다. 본 연구에서는 Fig. 1과 같이 블록 적치장의 형상은 직사각형이고 내부는 동일한 크기의 여러 구역으로 나뉘어져 있다고 가정하였다. 또한 블록 이동을 위해 트랜스포터가 이용되며 이는 적치장 내부에서 Fig. 2와 같이 직선으로만 움직인다고 가정하였다. 실제로 적치장 내부에 이미 존재하고 있는 기존 블록들과 트랜스포터의 길이, 회전 반경 등을 고려하면 트랜스포터는 적치장 내부에서 회전이 어렵기 때문에 직선으로의 출입이 가능할 것이다.

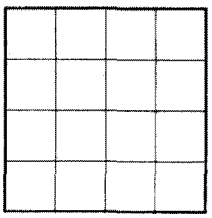


Fig. 1. Example of the block stockyard having 4×4 regions.

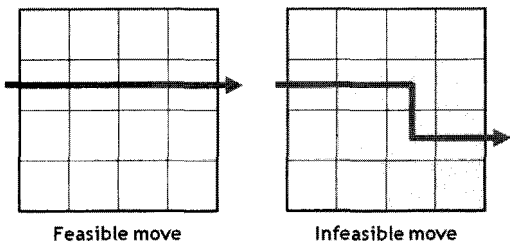


Fig. 2. Moving pattern of a transporter in the block stockyard in this study.

본 연구에서 블록 적치장의 최적 배치를 위해 설계자로부터 주어지는 입력 정보는 다음과 같다.

- 블록 적치장 내 구역의 개수
- 블록 적치장 내 배치되는 블록의 총 개수
- 최적 배치 적용 기간(배치 기준 시점과 종점)
- 블록 적치장의 입출고 블록 정보

여기서, 블록 적치장의 입출고 블록에 대한 상세 정보는 다음과 같다.

- 블록의 이름: 5개의 문자(XXXXX)로 구성된다고 가정함. 예, S0101, D0102, E0203
- 블록의 입출고 시간: 12자리의 숫자(YYYYMMDDHHMM)로 표현된다고 가정함. 예, 200902201030는 2009년 2월 20일 10시 30분임을 나타냄

한편, 본 연구에서는 블록 적치장의 최적 배치와 관련하여 몇 가지 용어를 정의하였다. 적치장은 초기에 어떠한 블록도 존재하지 않을 수도 있고 또는 몇 개의 블록('기존 블록')이 이미 놓여 있을 수도 있다. 이 후 시간이 지남에 따라 새로운 블록들이 입고되는데 이 블록들을 '입고 블록'이라고 한다. 이후 출고되는 블록들을 '출고 블록'이라고 한다. 또한, 블록이 입고 및 출고될 때 임시로 이동되어야 하는 기존 블

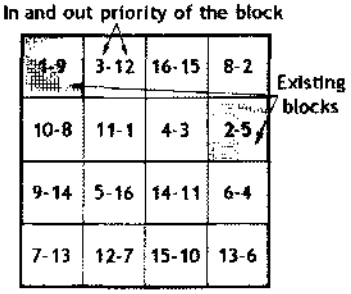


Fig. 3. Example of the block stockyard layout.

록들을 '방해 블록'이라고 한다. 본 연구에서 '기존 블록'은 최적화에 의해 위치가 변경되지 않는다고 가정하였다.

Fig. 3은 블록 적치장 내 블록들의 배치 예를 나타낸 것이다. 그림에 나타난 블록 적치장은 16개의 구역을 가졌음을 알 수 있다. 각 구역에 적힌 숫자(예, '1-9', '3-12', '16-25' 등)는 각 구역의 지번이 아닌 해당 구역에 놓인 블록의 속성을 의미하며, 앞의 숫자는 블록의 입고 시간을 고려한 우선 순위(입고 순서)를, 뒤의 숫자는 블록의 출고 시간을 고려한 우선 순위(출고 순서)를 나타낸다. 즉, 작은 숫자가 먼저 입고되어지고, 먼저 출고 되어지는 블록이라는 것이다. 예컨대, '1-9'라 적힌 구역에 놓여 있는 블록은 첫 번째로 적치장에 입고된 후 9번째로 출고된다는 것을 의미한다. 물론, 입고 순서와 출고 순서가 같은 블록이 있을 수 있다. 이 블록은 입출고 순서만 동일할 뿐 각각의 입고 시간과 출고 시간은 다를 수 있으므로, 같은 시간에 입고되었다 출고 된다고 볼 수 없다. 즉, 본 연구에서는 설명의 편의상 각 구역에 배치되는 블록을 블록의 ID가 아니라 해당 블록의 입고 시간과 출고 시간을 활용하여 입출고 우선 순위로 표현(「블록의 입고 순서-블록의 출고 순서」)하였다.

Fig. 3에 나타난 블록 적치장 내 블록들의 배치를 다시 살펴보면, 『11-1』 블록은 11번째로 적치장으로 입고되고, 첫 번째로 출고된다. 먼저, 이 블록이 입고될 때는 『3-12』 블록이나 『10-8』 블록이 먼저 놓여 있어 이들이 방해 블록이 되며, 역시 이 블록을 출고할 때에도 이들이 방해 블록으로서 작용한다. 이처럼 『11-1』 블록의 측면에서 보면 이의 입출고 시 방해 블록이 존재하여 이 블록을 입출고하려면 방해 블록들을 임시로 이동시켜야 하는 재배치 비용이 발생하므로 이는 좋은 배치라고 볼 수 없다.

2.2 최적 배치 문제의 수학적 정식화

본 연구에서는 블록 적치장의 최적 배치 문제를 수학적으로 정식화하기 위해 다음과 같은 변수와 기호를 사용하였다.

- M : 블록 적치장 내에 있는 방해 블록들을 운반하기 위해 필요한 비용의 합(재배치 비용)
- I_{ij} : i번째 행과 j번째 열에 존재하는 블록을 입고할 때 존재하는 방해 블록들의 이동 거리의 합
- s_{ij} : i번째 행과 j번째 열에 존재하는 블록의 입고 시간
- O_{ij} : i번째 행과 j번째 열에 존재하는 블록을 출고할 때 존재하는 방해 블록들의 이동 거리의 합
- e_{ij} : i번째 행과 j번째 열에 존재하는 블록의 출고 시간
- m, n : 적치장 내 구역의 행과 열의 번호
- c : 단위 이동 거리당 운반 비용
- S : 블록 적치장 배치의 기준이 되는 시작 시점
- E : 블록 적치장 배치의 기준이 되는 종료 시점

블록 적치장의 최적 배치 문제를 수학적으로 정식화하면 다음과 같다.

$$\text{Minimize} \quad F = M = \sum_{i=1}^{m}\sum_{j=1}^{n} c(I_{ij} - O_{ij}) \tag{1}$$

$$\text{Subject to} \quad S < s_{ij} \tag{2}$$

$$e_{ij} < E \tag{3}$$

여기서, 식 (1)은 목적 함수로서 블록 적치장 내 방해 블록들의 재배치 비용을 최소화하는 것을 나타낸다. 식 (2), (3)은 제약 조건으로서, 식 (2)는 각 블록의 입고 시간이 최적 배치의 시작 시점 보다는 이후여야 함을, 식 (3)은 각 블록의 출고 시간이 최적 배치의 종료 시점 보다는 이전이어야 함을 나타낸다. 즉, 제약 조건은 블록 적치장의 배치 기간 내에 존재하는 블록들만을 배치의 대상으로 한다는 의미이다. Fig. 4는 배치 기간 내에 존재하는 다양한 블록들의 입고 및 출고 시간을 보여준다. 본 연구에서는 서로 다른 입출고 시간을 가지는 적치장 내 블록들의 재배치 비용을 최소로 하는 최적 배치를 찾는 것이 목적이다.

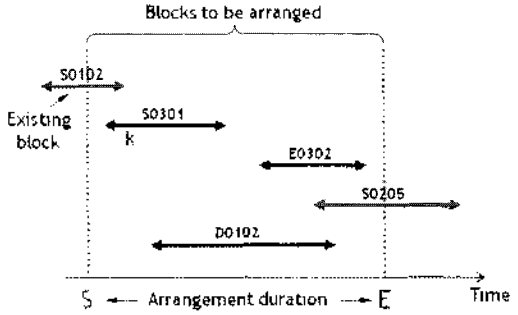


Fig. 4. Blocks in the block stockyard having the different in and out time.

3. 블록 적치장의 최적 배치 방법

3.1 제안된 최적 배치 방법의 개요

본 연구에서는 다양한 분야에서 활발히 이용되고 있는 유전 알고리즘을 이용하여 블록 적치장의 최적

배치 방법을 제안하였다. 유전 알고리즘은 다윈이 주장한 자연 진화의 법칙인 적자 생존과 자연 도태의 원리를 기반으로 하여 정립된 최적화 알고리즘이다[6]. 유전 알고리즘은 초기 모집단(population)에서 출발하여 유전 법칙을 이용하여 새로운 세대로 집단을 계속 진화시켜 나가는데 세대 변천은 선택(selection), 교배(crossover), 돌연 변이(mutation)의 3가지 유전자 연산에 의해서 이루어진다.

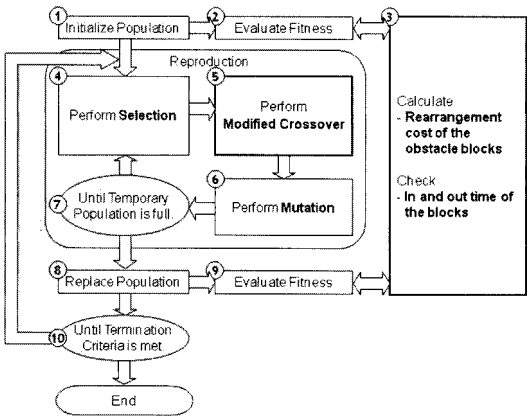


Fig. 5. Scheme of the proposed method for the block stockyard layout.

선택은 모집단 내의 각 해를 평가하여 우수하다고 평가 받는 해를 자기 복제를 허용함으로써 다음 세대에 그 세력을 확장하게 하는 것이다. 교배는 선택 이후 변화된 세력에서 해들로 하여금 무작위로 모두 짝을 짓게 한 후 짝들이 서로 일부를 교환하게 함으로써 새로운 해를 만들어낸다. 돌연 변이는 염색체(chromosome)의 일부를 임의로 바꾸어 주는 것이다. 이렇게 3가지의 기본 유전자 연산을 이용하여 진화를 계속하게 되면 초기해 보다 더 나은 개선된 해를 도출할 수 있다. 최적의 해를 찾고자 한다. 유전 알고리즘에 대한 자세한 내용은 많은 참고 문헌 등에 나타나 있으므로 여기서는 생략하기로 한다. Fig. 5는 유전 알고리즘을 기반으로 하여 본 연구에서 제안된 블록 적치장의 최적 배치 방법을 나타낸다.

3.2 블록 적치장 배치의 표현 방법

블록 적치장 내 블록들의 배치는 유전 알고리즘의 인코딩(encoding) 과정에 의해 염색체로 표현될 수 있다. 그리고 그 염색체는 디코딩(decoding) 과정에 의해 다시 블록 적치장 내 블록들의 배치로 표현될 수 있다. 본 연구에서는 블록 적치장 내 블록들의 배치를 표현하기 위해 2차원 행렬 형태의 염색체를 이용하였다. Fig. 6은 4×4 크기의 블록 적치장 내 블록들의 배치를 2차원 행렬 형태의 염색체로 표현한 예를 나타낸 것이다.

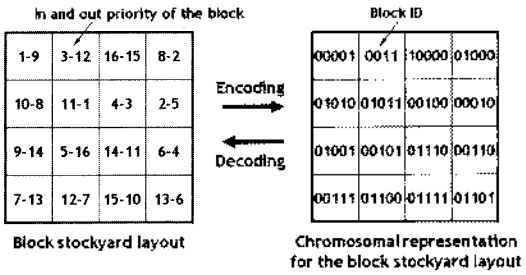


Fig. 6. Example of the block stockyard layout and the corresponding representation of the chromosome.

3.3 블록 적치장의 배치에 따른 재배치 비용의 계산 방법

블록 적치장의 최적 배치 문제의 목적 함수는 블록의 입출고 시 발생하는 방해 블록들의 재배치 비용을 최소화하는 것이다. 아래에서는 블록 적치장의 배치에 따른 재배치 비용의 계산 방법을 설명하기로 한다.

Fig. 7을 이용하여 「11-1」 블록을 입고할 때의 재배치 비용을 계산해 보자. 단, 설명의 편의상 여기서는 모든 블록이 모두 입고된 후에 출고가 시작된다고 가정한다.

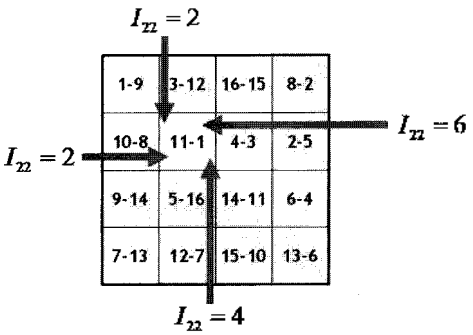


Fig. 7. Example of calculating the rearrangement cost for inserting 「11-1」 block.

「11-1」 블록을 적치장으로 입고할 때 트랜스포터는 동서남북의 4 방향으로 진입할 수 있다. 북쪽으로부터 진입하여 「11-1」 블록을 입고할 경우, 「3-12」 블록(방해 블록)이 이미 적치장 내에 입고되어 있으므로 이 블록을 한 구역 바깥쪽으로 이동시킨 후 「11-

1』 블록을 해당 위치에 넣고, 다시 『3-12』 블록을 원래의 위치에 놓아야 한다. 따라서 『11-1』 블록을 입고할 때 『3-12』 블록의 재배치를 위한 단위 이동 거리는 넣고 빼는데 총 2 구역이 된다. 마찬가지로, 서쪽으로부터 진입하여 『11-1』 블록을 입고할 경우, 『10-8』 블록의 재배치를 위해 총 2 구역의 이동이 필요하고, 남쪽으로부터 진입할 경우, 『5-16』 블록의 재배치를 위해 총 4 구역의 이동이 필요하며, 동쪽으로부터 진입할 경우, 『4-3』 블록의 재배치를 위해 4 구역, 『2-5』 블록의 재배치를 위해 2 구역 등 총 6 구역의 이동이 필요하다. 따라서 『11-1』 블록의 입고를 위해 북쪽 또는 서쪽으로의 진입을 통해 최소 총 2 구역의 재배치 이동이 필요함을 알 수 있다. 따라서 전체 재배치 비용(식 (1))의 목적 함수) 중 『11-1』 블록(2행 2열에 존재)의 입고를 위해 필요한 방해 블록의 재배치 비용 $I_{22} = 2$임을 알 수 있다.

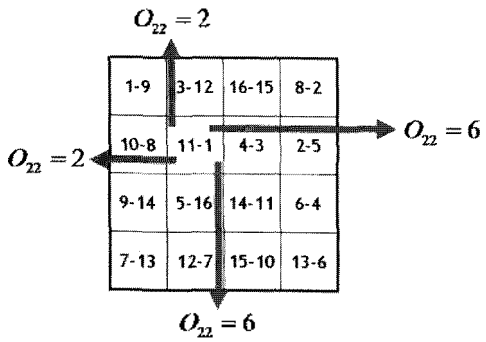


Fig. 8. Example of calculating the rearrangement cost for taking out 『11-1』 block.

이제 Fig. 8을 이용하여 『11-1』 블록을 출고할 경우를 살펴보자.

북쪽으로부터 진입하여 『11-1』 블록을 출고할 경우, 『3-12』 블록(방해 블록)이 이미 적치장 내에 입고되어 있으므로 이 블록을 한 구역 바깥쪽으로 이동시킨 후 『11-1』 블록을 빼내고, 다시 『3-12』 블록을 원래의 위치에 놓아야 한다. 따라서 『11-1』 블록을 출고할 때 『3-12』 블록의 재배치를 위한 단위 이동 거리는 넣고 빼는데 총 2 구역이 된다. 마찬가지로, 서쪽으로부터 진입하여 『11-1』 블록을 출고할 경우, 『10-8』 블록의 재배치를 위해 총 2 구역의 이동이 필요하고, 남쪽으로부터 진입할 경우, 『5-16』 블록의 재배치를 위해 4구역, 『12-7』 블록의 재배치를 위해 2 구역 등 총 6 구역의 이동이 필요하며, 동쪽으로부터 진입할 경우, 『4-3』 블록의 재배치를 위해 4 구역, 『2-5』 블록의 재배치를 위해 2 구역 등 총 6 구역의 이동이 필요하다. 따라서 『11-1』 블록의 출고를 위해 북쪽 또는 서쪽으로의 진입을 통해 최소 총 2 구역의 재배치 이동이 필요함을 알 수 있다. 따라서 전체 재배치 비용(식 (1))의 목적 함수) 중 『11-1』 블록(2행 2열에 존재)의 출고를 위해 필요한 방해 블록의 재배치 비용 $I_{22} = 2$임을 알 수 있다.

이상의 과정을 블록 적치장 내 모든 위치의 블록에 대해 반복적으로 수행하면 해당 블록 적치장의 배치에 대한 재배치 비용을 계산할 수 있다.

다음은 재배치 비용의 계산 과정을 의사 코드(pseudo code)로 표현한 것이다.

Set M equal to 0

For all locus of an individual Calculate obstacle blocks' distance from this block to the right hand side (mcR).

Calculate obstacle blocks' distance from this block to the left hand side (mcL).

Calculate obstacle blocks' distance from this block to the topmost (mcT).

Calculate obstacle blocks' distance from this block to the bottommost (mcB).

Return temporary moving cost equal to min (mcR, mcL, mcT, mcB).

Calculate moving cost M = I + O.

End For

3.4 유전자 연산

본 연구에서 제안된 방법 내에서 사용된 유전자 연산을 보다 상세히 설명하면 아래와 같다.

3.4.1 모집단의 초기화

유전 알고리즘을 기반으로 하는 제안된 방법에서는 다수의 개체(individual)로 구성되는 초기의 모집단을 만든 후 이들을 유전자 연산을 통해 점점 발전시키면서 최적해에 근접한 개선된 모집단을 만들게 된다. 모집단 내의 각 개체는 하나의 적치장 배치를 나타내며, 본 연구에서는 초기 모집단 생성 시 각 개체를 랜덤으로 생성하였다. 아래에서는 초기의 모집단 생성 시 하나의 개체(적치장 배치)를 만드는 과정을 설명한 것이다. Fig. 9에 나타나 있듯이, 블록 적치장은 4×4의 크기이고, 현재 (1, 1) 위치(1행 1열)에 『1-9』 블록이, (2, 4) 위치에 『2-5』 블록이 기존 블록으로서 놓여 있다고 가정한다. 따라서 이후 배치 가능한 블록의 총 개수는 최대 14가 된다. 물론 블록이 이미 놓

여 있는 위치에는 새로운 블록이 놓일 수 없다. 각 구역에 새로운 블록이 놓일 수 있는지를 표시하기 위해 각 구역은 'true' 또는 'false'의 속성 값을 가지게 된다. 예컨대, 이미 블록이 놓여 있는 (1, 1), (2, 4) 위치의 구역들은 모두 'true'의 속성 값을 가지고 있다. 그리고 블록이 놓여 있지 않은 나머지 구역들은 초기에 'false'의 값을 가지고 있다가 해당 구역에 블록이 놓일 때 'true'의 값으로 바뀌게 된다.

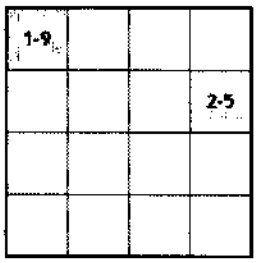


Fig. 9. Initial block stockyard layout having two existing blocks.

이후 14개의 모든 블록들이 입고 순서에 따라 현재 'false'의 값을 가지고 있는 구역 중 하나에 임의로 배치되면 하나의 적치장 배치 즉, 하나의 모집단 내 개체가 완성된 것이다. Fig. 10은 이와 같은 과정을 거쳐 생성된 초기 모집단 내 하나의 적치장 배치의 예를 나타낸다.

1-9	3-12	16-15	8-2
10-8	11-1	4-3	2-5
9-14	5-16	14-11	6-4
7-13	12-7	15-10	13-6

Fig. 10. Block stockyard layout having 16 blocks.

3.4.2 선택 연산

선택 연산은 이후의 유전자 연산 단계를 위해 모집단으로부터 두 개의 부모 개체를 선택하는 과정으로 본 연구에서는 'roulette wheel selection' 방법[6,7]을 이용하였다.

다음은 선택 연산을 의사 코드로 표현한 것이다.

Sum of all chromosome finesses in population (sum S).

Generate random number from interval[0, S] (r).

[Loop] Go through the population and sum finesses from 0 (sum s). When the sum s is greater than r, stop and return the chromosome where you are.

3.4.3 교배 연산

교배 연산은 선택 연산을 통해 선택된 두 개의 부모 개체로부터 새로운 자식 개체를 생성하는 과정으로서 본 연구에서는 'uniform order based crossover'와 'modified crossover'를 이용하였다.

(1) Uniform order based crossover

Uniform order based crossover는 2차원 행렬 형태의 염색체에 적용할 수 있는 교배 연산으로서 Jackobs[7]가 제안하였다. Fig. 11을 이용하여 이 방법을 설명하면 다음과 같다. 여기서, 유전자 연산은 실제로 염색체에 대해 적용되기 때문에 Fig. 11의 각 개체의 염색체는 각 블록의 ID를 나타내는 이진수로 표현되어야 하나 설명의 편의상 블록의 입출고 순서로 표현하였다. 먼저 부모 개체의 염색체와 동일한 크기를 같은 2차원 행렬 형태의 염색체를 생성하고 염색체 내 각 유전 인자의 값은 0 또는 1로 임의로 부여한다. 유전 인자의 값이 1인 위치에 해당하는 첫 번째 부모 개체('Parent 1')의 블록을 첫 번째 자식 개체('Child 1')의 해당 위치에 복사한다. 그리고 'Child 1'의 나머지 위치에는 'Parent 2'의 배치 순서를 참조하여 아직 배치되지 않은 블록들을 하나씩 채우게 된다. 이를 마치면 첫 번째 자식 개체가 완성된다. 이와 마찬가지로, 유전 인자의 값이 0인 위치에 해당하는 두 번째 부모 개체('Parent 2')의 블록을 두 번째 자식 개체('Child 2')의 해당 위치에 복사한다. 그리고 'Child 2'의 나머지 위치에는 'Parent 1'의 배치 순서를 참조하여 아직 배치되지 않은 블록들을 하나씩 채우게 된다. 이를 마치면 두 번째 자식 개체를 얻을 수 있다.

다음은 uniform order based crossover 연산을 이용하여 첫 번째 자식 개체를 생성하는 과정을 의사 코드로 표현한 것이다.

Generate randomly a bit array having the same as parent.

Copy contents from 'Parent 1' to 'Child 1' wherever element contains '1'.

Push elements we get from previous step so that they appear in the same order they appear in 'Parent 2'.

Copy these permuted elements to gaps associated with a '0' on 'Child 1'.

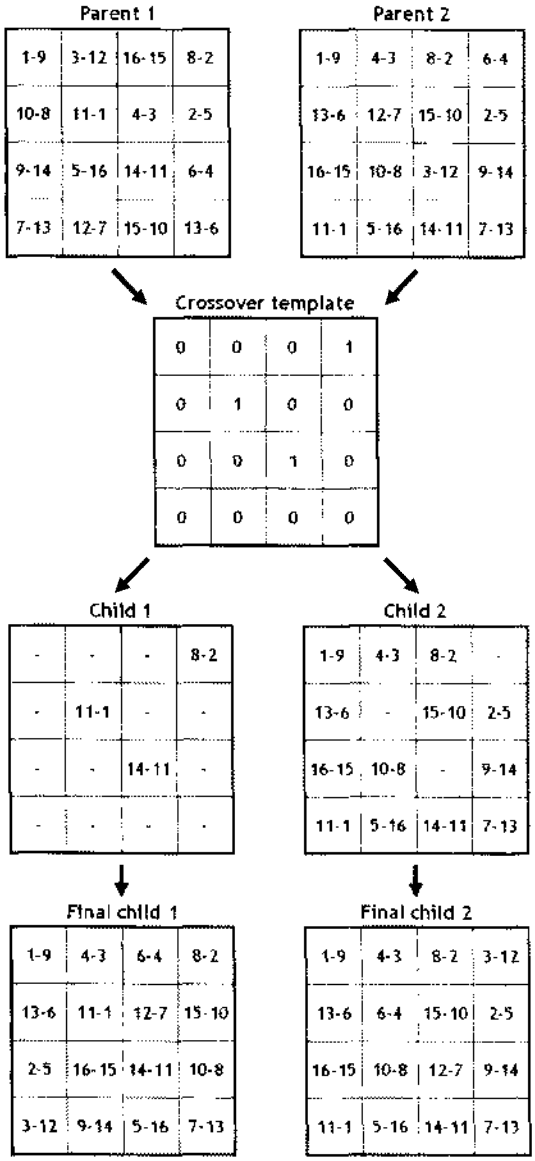


Fig. 11. Example of the uniform order based crossover operation.

(2) Modified crossover

블록 적치장의 배치 기간 이전에 놓여 있는 '기존 블록'의 위치는 최적화 과정 중 변경되지 않는다고 본 연구에서는 가정하였다. 그런데 앞의 uniform order based crossover를 수행하게 되면 기존 블록의 위치가 변경될 수 있다. 이를 조정하기 위해 본 연구에서는 modified crossover를 추가적으로 사용하였다. 물론 기존 블록이 없을 경우 본 연산은 실행할 필요가 없다. Fig. 12는 modified crossover의 예를 나타낸다. 「11-1」 블록이 기존 블록이라고 가정했을 때 uniform order based crossover를 수행한 후 이 블록의 위치가 변경되었고 modified crossover에 의해 이 블록의 위치가 조정(해당 위치에 놓여 있는 블록과의 위치 교환)되었음을 알 수 있다.

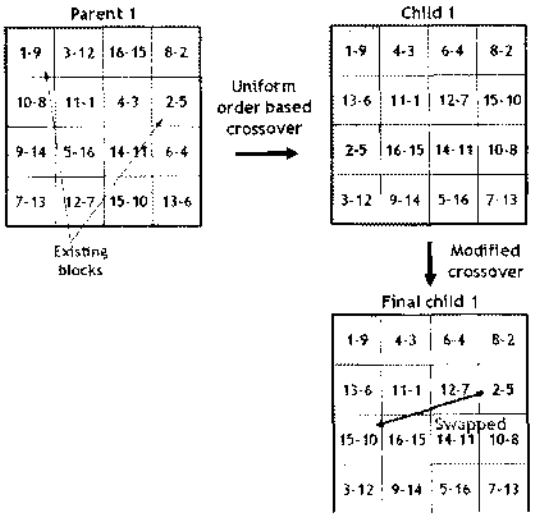


Fig. 12. Example of the modified crossover operation.

다음은 uniform order based crossover 연산에 의해 생성된 첫 번째 자식 개체에 대해 modified crossover 연산을 적용하는 과정을 의사 코드로 표현한 것이다.

[Loop] Scan all genes of chromosome of 'Parent 1'.

If (Does this locus be getting fixed with an allele?)

Return locus

If (Is 'Parent 1'-allele not equal to 'Child 1'-allele at this locus?)

Lookup and **return** locus on 'Child 1' that keep the same allele on 'Parent 1'.

Call swap function

End If

End If

End Loop

3.4.4 돌연 변이 연산

돌연 변이 연산은 교배 연산을 통해 얻어진 각 자

식 개체에 대해 염색체(블록 적치장의 배치) 내 두 개의 유전 인자(블록)을 임의로 선택하여 Fig. 13과 같이 상호 교환함으로써 새로운 자식 개체를 생성하는 것이다. 물론 돌연 변이 연산 시 기존 블록의 위치를 이동되면 안 된다는 조건을 추가적으로 고려해야 한다. 즉, 기존 블록은 돌연 변이 연산을 위해 선택이 되지 않도록 해야 한다.

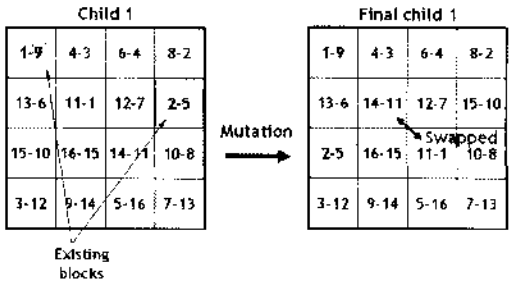


Fig. 13. Example of the mutation operation.

다음은 돌연 변이 연산을 의사 코드로 표현한 것이다.

Do {
  Call random function to get the first locus.
} **While** (locus is fixed.)
Get first locus.
Do {
  Call random function to get the second locus
} **While** (locus is fixed.)
Get second locus.
Swap (first locus, second locus).

4. 제안된 최적 배치 방법의 적용 예

본 연구에서 제안한 최적 배치 방법의 효용성을 평가하기 위해, 5×5 크기를 가지는 블록 적치장에 22개의 블록들을 배치하는 문제에 적용해 보았다. 22개의 블록 중 3개의 블록(「1-1」, 「2-2」, 「3-3」 블록)은 최적화에 의해 위치가 변경되지 않는 기존 블록이라고 가정하였다. Fig. 14는 블록 적치장에 놓일 블록들의 입출고 정보를 나타낸다.

Fig. 15(a)는 최적화 전의 블록 적치장의 초기 배치를 나타낸 것이다. 이 배치에 따라 재배치 비용을 계산하면 8이 된다(단, 단위 이동 거리당 유반 비용(c)은 1이라고 가정). 하지만 Fig. 15(b)와 같이 최적화 후에는 재배치 비용이 0이 됨을 알 수 있다.

Block No.	출고 시간	Block No.	입고 시간
		1-1	2009-11/09-08:00
		2-2	2009-11/09-09:00
		3-3	2009-11/09-10:00
		15-4	2009-11/09-11:00
		17-5	2009-11/09-15:00
		6-6	2009-11/09-16:00
		12-7	2009-11/10-11:00
		4-8	2009-11/10-17:00
		5-9	2009-11/11-10:00
		16-10	2009-11/11-11:00
		7-11	2009-11/11-17:00
		11-12	2009-11/14-18:00
		8-13	2009-11/13-12:00
		9-14	2009-11/13-10:00
1-1	2009-11/13-12:00		
		14-15	2009-11/13-13:00
2-2	2009-11/13-14:00		
3-3	2009-11/13-16:00		
4-8	2009-11/14-13:00		
5-9	2009-11/14-14:00		
		13-16	2009-11/14-15:00
		19-17	2009-11/15-10:00
6-6	2009-11/15-12:00		
7-11	2009-11/15-14:00	18-18	2009-11/15-14:00
		21-19	2009-11/15-15:00
8-13	2009-11/15-15:00		
9-14	2009-11/15-17:00		
10-15	2009-11/16-11:00		
		12-20	2009-11/16-12:00
11-12	2009-11/16-13:00		
12-7	2009-11/16-14:00	22-21	2009-11/16-14:00
		20-22	2009-11/16-15:00
13-16	2009-11/17-10:00		
14-10	2009-11/17-12:00		
15-4	2009-11/17-14:00		
16-20	2009-11/17-17:00		
17-5	2009-11/18-11:00		
18-18	2009-11/19-10:00		
19-17	2009-11/19-13:00		
20-22	2009-11/19-16:00		
21-19	2009-11/20-14:00		
22-21	2009-11/20-15:00		

Fig. 14. Block information for the block stockyard layout.

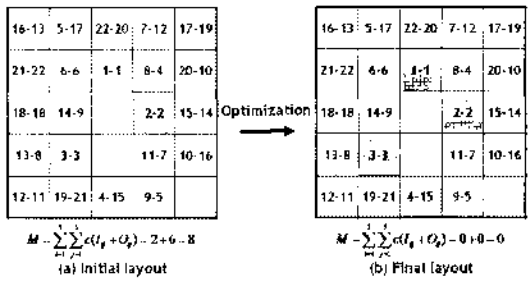


Fig. 15. Block stockyard layout before and after optimization using the proposed method.

5. 결론 및 향후 연구 계획

본 연구에서는 조선소의 블록 적치장 내에서 발생할 수 있는 블록들의 재배치 비용을 최소화함으로써 선박 건조 생산성을 향상시키기 위한 블록 적치장의 최적 배치 방법을 제안하였다. 이를 위해, 블록의 입

출고 시간을 고려한 블록 적치장의 최적 배치 문제를 수학적으로 정식화 하였고, 이를 풀 수 있는 최적 배치 방법을 유전 알고리즘을 기반으로 제안하고 구현하였다. 마지막으로 제안된 방법의 효용성을 검토하기 위해 간단한 예제에 적용하였다. 그 결과, 초기 배치에 비해 재배치 비용을 크게 줄임으로써 실제 조선소의 적치장 배치 문제에 적용 가능함을 확인하였다.

본 연구에서는 적치장 내 하나의 구역에 하나의 블록만 들어갈 수 있다고 가정하였으나, 실제 선박의 블록은 그 크기가 다르므로 하나의 블록이 여러 구역을 차지할 수도 있다. 향후 이를 해결하기 위해 제안된 방법을 개선할 것이며, 또한 보다 복잡한 문제에 대해 본 방법을 적용하여 제안된 방법의 효용성과 효율성을 제고할 예정이다.

감사의 글

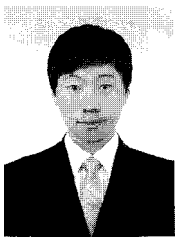
본 연구는 지식경제부 산업원천기술개발사업(10035331, 시뮬레이션 기반의 선박 및 해양플랜트 생산기술 개발)과 한국연구재단(KRF-2008-314-D00494, KRF-2009-0086033, R33-2008-000-10150-0)의 지원을 받아 연구되었음을 밝히며, 이에 감사드립니다.

참고문헌

1. 박창규, 서순용, "조선 소립 블록 운영 관리에 관한 사례 연구", 한국경영과학회논문집, 제23권, 제2호, pp. 175-185, 2006.
2. 이준욱, 홍동희, 정태충, "유전자 알고리즘을 이용한 수출 컨테이너 장치 위치 설정에 관한 연구", 2003년 한국정보과학회 춘계학술발표회, pp. 443-445, 제주, 2003. 4. 25-26.
3. 최영진, 오명섭, 강재호, 신수민, 류광렬, 김갑환, "컨테이너 재취급 최소화를 위한 이적 위치 결정 휴리스틱의 성능 비교", 2004년 한국지능정보시스템학회 춘계학술발표회, pp.382-391, 서울, 2004. 6. 11.
4. 강재호, 류광렬, 김갑환, "장치장에서 베이 내 컨테이너의 효율적인 재정돈 방안", 2004년 한국지능정보시스템학회 추계학술발표회, pp. 287-295, 서울, 2004. 11. 19.
5. 한성남, 이규열, 노명일, "개선된 유전자 알고리즘을 이용한 최적 공간 배치 설계에 관한 연구", 한국CAD/CAM학회논문집, 제6권, 제3호, pp. 174-183, 2001.
6. Goldberg, D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, Reading, MA, 1989.
7. Jacobs, S., "On Genetic Algorithms for the Packing of Polygons", *European Journal of Operational Research*, Vol. 88, No. 1, pp. 165-181, 1996.

노 명 일

1998년 서울대학교 조선해양공학과 학사
2000년 서울대학교 조선해양공학과 석사
2005년 서울대학교 조선해양공학과 박사
2005년~2007년 서울대학교 공학연구소 /해양시스템공학연구소 선임연구원
2007년~현재 울산대학교 조선해양공학부 조교수
관심분야: Computer-Aided Ship Design and Manufacturing, Simulation-Based Design/Manufacturing, 최적 설계, CAD/CAM/CAE, CAGD



임 병 석

2009 울산대학교 조선해양공학부 학사
2011년 울산대학교 조선및해양공학과 석사
2011년~현재 STX 조선해양 기술연구소 생산기술연구실 공정기술팀 주임
관심분야: 모델링 및 시뮬레이션, 최적 설계