[2025 전기 졸업과제 착수 보고서]

디지털트윈 활용
대형 블록 배치 최적화

지도 교수 : 김원석
팀 명 : 블록버스터
팀 원: 201524420 김도완
202055552 서민성
202155649 유주연

목차

1. **과제 개요** [3]

1.1. 연구 배경 [3]
1.2. 연구 목표 [4]
1.3. 문제 정의 [5]

2. **기존 연구 및 관련 기술** [6]

2.1. 공간 배치 알고리즘 [6]
2.2. 메타휴리스틱 기법 [7]

3. **시스템 구조 및 사용 기술** [9]

3.1. 전체 구성도 및 주요 기술 스택 [9]

4. **연구 내용 및 수행 방법** [10]

4.1. 3D 형상 기반 2D 공간 배치 최적화 [10]
4.2. 운송 용량-운임 비용을 고려한 최적화 [10]
4.3. 심층 강화학습 기반 성능 향상 [10]
4.4. 디지털 트윈 기반 3D 시뮬레이션 및 시각화 [11]

5. **개발 일정 및 역할 분담** [12]

5.1. 개발 일정 [12]
5.2. 역할 분담 [13]

1. 과제 개요

1.1. 연구 배경

현재 선박에 블록을 배치하는 작업은 주로 블록을 위에서 본 2차원 형상(2D)을 기준으로 이루어진다. 그러나 실제 블록은 높이를 포함한 입체적 구조를 가지므로, 이러한 2D 기반 배치 방식은 공간 활용에 한계가 있다. 그 결과, 선박에 실을 수 있는 블록의 수가 줄어들어 운송 효율이 저하되고, 전체 운송 비용이 증가하는 문제가 발생한다.

또한 블록의 상하차 과정에는 **트랜스포터(T/P)**와 **트레슬(Trestle)** 같은 보조 장비가 필수적으로 사용된다. 이 장비들을 고려하지 않은 배치는 장비 간 간섭이나 공간 낭비를 유발할 수 있으며, 이는 추가적인 비용 증가로 이어질 수 있다.

더불어, 블록이 적재되는 선박은 종류에 따라 내부 구조가 상이하다. 선박 구조가 달라지면 배치 가능한 공간의 형태도 달라지고, 이에 따라 탑재할 수 있는 블록의 종류나 수량 역시 달라진다. 따라서 다양한 형태의 선박에 대응 가능한 **범용적 블록 배치 알고리즘**을 개발한다면, 새로운 선박을 도입할 때도 추가적인 설계 없이 효율적이고 유연한 배치가 가능해질 것이다.

그림 1. 트레슬과 트랜스포터

트랜스포터는 블록과 트레슬을 아래에서 들어 올려 운반하는 장비이다. 선박 내부까지 직접 진입해 블록을 이동시킬 수 있다.

트레슬은 블록을 안정적으로 지지하는 받침대 역할을 한다. 하단이 비어 있어 트랜스포터가 아래로 진입할 수 있도록 설계된다.

1.2. 연구 목표

본 연구의 목적은 블록, 트랜스포터, 트레슬 등 실제 운송에 사용되는 구조물을 고려하여, 선박 내 블록 적재를 자동화하고 최적화할 수 있는 시스템을 구축하는 것이다. 다음은 네 가지 세부 목표이다.

1. 2.5D 복셀 기반 형상 데이터 구축

- 블록, 트랜스포터, 트레슬의 입체적인 구조를 반영하기 위해 **3D 모델을 2.5D 복셀 형태 (x,y, 3) 배열**로 변환한다.
- 복셀은 각 (x, y) 위치에 대해 지면 기준으로 **빈 공간 - 채워진 공간 - 이후 빈 공간** 의 상태를 표현한다. 이 방법은 Full 3D 복셀보다 빠르면서도 입체 정보 활용이 가능하다.
- 이러한 2.5D 복셀 표현 방법을 사용하여 블록 간 간섭, 경로 확보 여부, 구조물 회피 등을 판단한다.

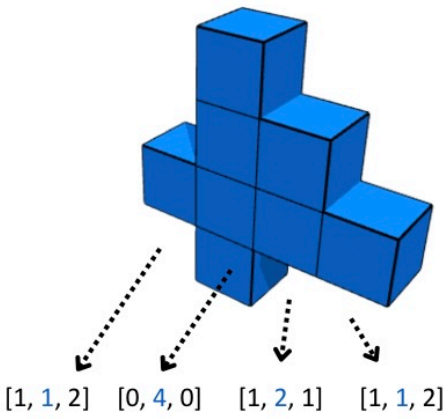

그림 2. 블록 형상에 대한 2.5D 복셀 표현 예시

2. 휴리스틱 기반 최적 배치 알고리즘 개발

- **백트래킹을 활용한 배치 알고리즘**으로 블록을 가장 효율적으로 적재할 수 있는 방법을 탐색한다.
- 배치 후보 위치는 **Bin Packing Problem을 응용한 후보 생성기**를 통해 도출하며, 블록 간 간섭이나 하역 장비의 진입 가능성 등을 고려하여 필터링한다.
- 배치 탐색 시 각 블록의 배치 방향은 0도 또는 180도 이다.

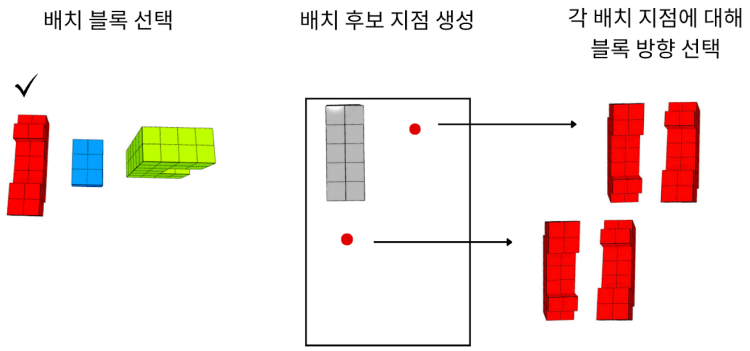

그림 3. 블록 배치 흐름 시각화

3. 강화학습 기반 배치 개선 연구

- 휴리스틱 기반 알고리즘으로부터 얻은 탐색 데이터를 바탕으로, **강화학습 기반의 의사결정 에이전트**를 구성한다.
- 블록 선택, 후보 지점 선택, 배치 방향 선택 등을 담당하는 **에이전트**를 설계하여, 탐색 속도를 향상시키고 유사 최적해를 빠르게 도출한다.
- 또한 학습 데이터 부족 문제가 예상되기 때문에 합성 데이터를 생성하고 이를 학습에 활용한다.

4. 디지털트윈 기반 시각화 및 시뮬레이션 환경 구축

- 개발된 배치 알고리즘의 결과를 **디지털트윈으로 시각화**하여 실제 구조물과의 간섭 여부, 경로 확보 문제 등을 검증한다.
- 블록, 트랜스포터, 트레슬의 설치 및 배치 과정을 3D 공간에서 재현함으로써, 실제 작업에 가까운 환경에서 알고리즘 적용 가능성을 평가한다.

1.3. 문제 정의

■ 목적

- 선박의 배치 공간 내에서 블록 간 간섭이나 진입 경로 문제 없이 가능한 많은 블록을 배치하여, 전체 **면적 활용률을 극대화**하는 것이 목표

■ 제약 조건

- 블록은 임의의 3D 형상이며, **비정형** 블록 존재
- 블록은 **단층**으로 배치

- 선박 **높이에 제약 없음**
- 블록 간 겹침 없이 배치
- 트랜스포터의 진입 경로가 확보되어야 함
- 블록은 트레슬 위에 배치되어야 하며, 배치 방향은 0도 혹은 180도 중 선택

입력 정보	**출력 정보**
- **선박의 3D 배치 공간 형상** 및 높이 - **블록의 3D 형상 정보** - **트레슬** 및 **트랜스포터의 크기** 및 **간섭 영역**	- 각 블록의 **배치 좌표** (x, y, z) 및 **방향 (0 or 180)** - **배치 성공 여부** 및 **전체 활용률**

2. 기존 연구 및 관련 기술

2.1. 공간 배치 알고리즘

공간 배치 알고리즘은 제한된 공간 내에 다양한 크기와 형태의 객체를 효율적으로 배치하는 것을 목적으로 하며, 물류, 조립, 포장 등 다양한 산업 분야에서 활용된다. 본 연구에서 다루는 블록 배치는 대형 구조물을 실공간에 배치해야 하는 복잡한 문제로, 일반적인 배치 알고리즘보다 현실적인 제약을 고려해야 한다.

2.1.1 2D Bin Packing 및 Strip Packing

2D Bin Packing은 정해진 크기의 2차원 영역(예: 직사각형 컨테이너)에 여러 개의 사각형 아이템을 겹치지 않도록 배치하여 **공간 낭비를 최소화하거나 사용된 bin의 개수를 최소화하는 문제**이다. Strip Packing은 bin의 폭은 고정되어 있고 높이는 무한한 상태에서, **최소 높이로 모든 아이템을 배치**하는 문제이다.

이러한 2D 기반 방식은 계산 속도 및 구현이 간단하다는 장점이 있지만, 블록의 실제 높이 정보나 입체적 간섭을 반영하지 못하므로, **현실적인 3D 배치 문제에는 한계가 존재**한다.

2.1.2 3D Bin Packing

3D Bin Packing은 3차원 공간 내에서 아이템의 부피, 위치, 회전을 고려하여 배치하는 문제이다. 이는 실제 컨테이너 로딩, 항공 화물 적재 등에서 사용되며, **입체적 충돌 방지와 안전성 확보**를 동시에 고려해야 한다.

3D Bin Packing에서는 보통 **완전한 3차원 격자(voxel)**를 사용하여 공간을 표현하거나, 각 아이템의 AABB(Axis-Aligned Bounding Box)를 기준으로 충돌 판정을 수행한다. 하지만 전체 공간을 3D 격자로 표현할 경우, 격자의 수가 기하급수적으로 증가하여 **연산 속도와 메모리 사용량에 큰 부담**이 된다.

2.1.3 2.5D Bin Packing (본 연구의 접근 방식)

본 연구에서는 기존 2D 또는 3D 표현 방식의 한계를 보완하고자, **2.5D 복셀 기반 공간 표현 방식**을 도입한다. 이 방식은 각 (x, y) 좌표에 대해 **z축 방향의 상태를 3개의 값으로 요약**하여 표현한다.

구체적으로는 다음과 같은 구조로 공간을 표현한다:

- V(x, y) = [빈 공간 높이, 채워진 높이, 위쪽 빈 공간]

이러한 2.5D 표현 방식은 다음과 같은 장점이 있다.

- **3D 입체 정보는 유지하면서, 연산량은 2D 수준으로 경량화**할 수 있다.
- 블록의 높이, 간섭, 트레슬/트랜스포터 진입 조건 등을 반영 가능하다.
- 완전한 3D 격자보다 훨씬 빠른 연산이 가능하며, 백트래킹 기반 탐색에도 적합하다.

2.2. 메타휴리스틱 기법

2.2.1 휴리스틱과 메타휴리스틱

휴리스틱(Heuristic)과 메타휴리스틱(Metaheuristic)은 모두 복잡한 최적화 문제를 효율적으로 해결하기 위한 근사적 접근 방식이지만, 그 적용 방식과 범용성 측면에서 차이를 보인다. 휴리스틱은 특정 문제에 특화된 경험적 규칙이나 알고리즘으로, 예를 들어 “큰 블록부터 우선 배치한다”거나 “왼쪽 아래부터 빈 공간을 채운다”는 식의 **정형화된 전략**을 의미한다. 이 방식은 구현이 간단하고 연산 속도가 빠르다는 장점이 있으나, 문제 구조가 바뀔 경우 성능이 급격히 저하되며, **전역 최적해를 놓칠 가능성**이 크다.

반면, 메타휴리스틱은 다양한 문제에 적용 가능한 일반화된 최적화 프레임워크로, 단일 해가 아닌 해 집단을 기반으로 반복적인 탐색을 수행하며, 경우에 따라 의도적으로 열등한 해도 수용함으로써 해 공간을 더욱 **폭넓게 탐색**할 수 있도록 한다. 이러한 특성 덕분에 메타휴리스틱은 복잡도나 불확실성이

높은 문제에서도 보다 안정적으로 **전역 최적해에 근접**할 수 있다. 다만, 계산 자원이 많이 소요되고 알고리즘 설계 시 하이퍼파라미터 조정이 필요하다는 점은 고려해야 할 요소이다.

결과적으로, 휴리스틱은 특정 문제에 대해 빠르고 실용적인 해법을 제공하는 반면, 메타휴리스틱은 더 복잡하고 다양한 문제에 **유연하게 적용**할 수 있는 고차원적 탐색 기법이라 할 수 있다.

2.2.2 유전 알고리즘 (Genetic Algorithm)

유전 알고리즘(Genetic Algorithm, GA)은 생물의 진화 과정을 모방해 더 나은 해를 점차 찾아가는 방식의 최적화 알고리즘이다. 이 알고리즘은 문제에 대한 여러 개의 후보 해를 먼저 무작위로 만든 다음, 그중 성능이 좋은 해들을 선택하고 서로 정보를 **섞거나 일부를 바꾸는 과정을 반복**하면서 점점 더 좋은 해를 만들어 낸다.

각 해는 특정한 형식(예: 숫자 목록이나 이진 코드)으로 표현되며, 알고리즘은 세대를 거듭하면서 이 해들을 평가하고 선택한다. 선택된 해는 **교차(crossover)** 과정을 통해 서로 조합되고, **돌연변이(mutation)**를 통해 일부 값이 무작위로 바뀌기도 한다. 이 과정을 반복하다 보면, 해의 집단은 점점 더 좋은 방향으로 진화하게 된다.

유전 알고리즘은 해를 하나씩 바꿔보는 방식보다 **더 넓은 범위를 탐색**할 수 있어, 복잡하거나 규칙이 명확하지 않은 문제에도 잘 작동한다. 특히 블록의 배치 순서를 결정하거나, 제한된 공간에 여러 요소를 효율적으로 배치해야 하는 문제처럼 해의 조합이 중요한 상황에서 효과적이다.

다만, 계산 시간이 길어질 수 있고, 알고리즘의 성능은 교차나 돌연변이 확률 같은 설정값에 따라 달라질 수 있기 때문에, 문제에 맞게 잘 조정하는 것이 필요하다. 그럼에도 불구하고, 유전 알고리즘은 단순한 규칙만으로는 풀기 어려운 다양한 문제에서 유용하게 쓰일 수 있는 방법이다.

2.2.3 활용 가능성

유전 알고리즘은 본 과제의 최적화 구조에서 다음의 두 가지 방식으로 활용될 수 있을 것으로 기대된다.

첫째, 유전 알고리즘은 백트래킹 방식의 **보완 도구**로 작용할 수 있다. 가능한 모든 블록 배치 조합을 완전 탐색하는 백트래킹 방식은 해의 수가 많아질수록 계산량이 급격히 증가하고, 이에 따라 탐색 속도가 느려지는 한계가 있다. 이와 달리, 유전 알고리즘은 해 공간 전체를 모두 탐색하지 않고도 높은 품질의 배치 조합을 빠르게 도출할 수 있는 장점이 있다. 이렇게 생성된 양질의 해는 강화학습의 초기 전략(policy) 설정이나 보상 함수 설계 단계에서 유용한 기반이 될 수 있다.

둘째, 강화학습과 유전 알고리즘을 결합한 **하이브리드 구조**로 활용하는 방식이다. 예를 들어, 강화학습이 제안한 배치 후보들을 염색체로 간주하고, 이를 유전 알고리즘으로 조합하거나 변형함으로써 더 나은 배치 결과를 생성할 수 있다. 이후, 이렇게 개선된 결과를 다시 강화학습의 학습 과정에 반영하면, 정책의 수렴 속도를 높이는 동시에 해의 품질도 함께 향상시킬 수 있다.

3. 시스템 구조 및 사용 기술

3.1. 전체 구성도 및 주요 기술 스택

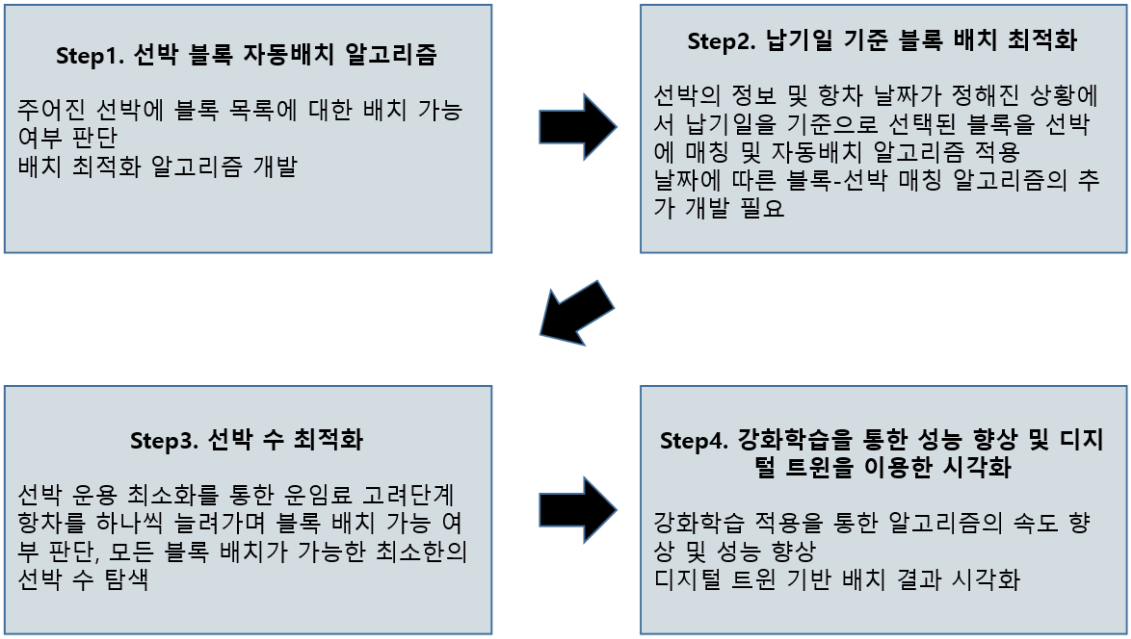

그림 4. 전체 구성도

단계	내용	주요 기술스택
Step 1	선박당 블록 자동배치 알고리즘	Python
Step 2	납기일 기준 블록 선별 및 배치 최적화	NumPy, Pandas
Step 3	블록 배치 기반 선박 수 최적화	Python
Step 4	강화학습 기반 성능 최적화	PyTorch 또는 TensorFlow
Step 5	디지털 트윈 기반 시각화	Unity

4. 연구 내용 및 수행 방법

4.1. 3D 형상 기반 2D 공간 배치 최적화

각 선박에 대해 주어진 블록 목록의 배치 가능 여부를 판단하고, 적절한 블록 자동 배치 알고리즘을 개발한다. 블록과 선박의 공간은 2.5D 복셀 형태로 추상화되며 Python을 기반으로 Bin Packing 알고리즘을 활용한 초기 배치 알고리즘을 구현한다. 유전 알고리즘이나 메타 휴리스틱 기법을 적용하여 배치 효율을 향상시키기 위해 확장할 예정이다.

기술 스택: Python
핵심 작업: 복셀 데이터 추상화, 블록 배치 초기 알고리즘 구현

4.2. 운송 용량-운임 비용을 고려한 최적화

운송 가능한 선박 리스트와 납기일에 대한 정보를 바탕으로, 시점별 선박 운항에 맞추어 블록을 선별하고 배치하는 알고리즘을 개발한다.
이 단계에서는 출항 일정이 정해진 선박을 기준으로, 납기일을 초과하지 않도록 블록을 선발하는 알고리즘을 개발한다. 이후에에 블록과 선박을 매칭하여 Step1의 배치 알고리즘을 적용한다.
또한, 납기일 조건을 만족하면서 가능한 운용 선박수를 최소화하는것을 목표로 한다. 배치 최적화를 통해 선박을 순차적으로 증가시키며 최소한의 선박 수를 만족하는 알고리즘을 개발한다.

기술 스택:Numpy
핵심 작업: 납기일 기반 블록 선별 알고리즘 개발, 선박 수 최소화 로직 구현

4.3. 심층 강화학습 기반 성능 향상

기존 알고리즘의 탐색 한계를 보완하기 위해 강화학습을 적용한다. PyTorch 및 TensorFlow 기반의 심층 강화학습 모델을 활용하여 에이전트는 블록을 어떤 위치에, 어떤 순서로 배치할지를 결정한다. 각 배치 결과에 대해 보상이 주어지고 에이전트는 보상을 최대화하는 효율적인 배치를 학습한다. 예를 들어 배치 최적화에 따른 운송-운임 비용의 최소화에 따라 높은 보상을 부여한다. 반대로 납기일을 초과하는 등의 경우에는 낮은 보상을 부여한다.

기술 스택: PyTorch or TensorFlow
핵심 작업: 보상 함수 설계 심층 강화 학습

4.4. 디지털트윈 기반 3D 시뮬레이션 및 시각화

최종 단계에서는 Unity 엔진을 활용한 디지털 트윈 시뮬레이션 환경을 구축하여, 배치 알고리즘의 적용 갈과를 3D로 시각화한다.
Python에서 생성된 배치 데이터를 JSON 등 인터페이스 포맷으로 Unity에 전달하여 Unity에서 블록 배치 결과를 시각화하고 배치 검증을 위한 가상 시뮬레이션 환경을 구축한다.

기술스택: Unity
핵심 작업: 시각화 인터페이스 개발

5. 개발 일정 및 역할 분담

5.1. 개발 일정

단계	작업	5				6				7				8				9			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
기획	자료조사																				
알고리즘 개발	복셀화																				
	Level1																				
	Level2																				
	Level3																				
디지털트윈 시각화	구조물 모델링																				
	합성 데이터																				
	시각화																				
알고리즘 개선	강화학습																				
	유전																				
실험 및 비교 분석	성능비교																				
	알고리즘 도출																				
마무리	테스트																				
	보고서 작성																				
	발표 준비																				

5.2. 역할 분담

이름	공동 작업	개별 담당 작업
김도완	블록 배치 알고리즘 (Level 1~ Level 3) 설계 및 구현 보고서 작성	강화학습 기반 배치 알고리즘 구현
서민성		디지털트윈 시뮬레이션 개발
유주연		블록 복셀화 구현, 합성 데이터 생성