[2025 전기 졸업과제 중간 보고서]

디지털트윈 활용
대형 블록 배치 최적화

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

목차

1. 과제 목표

본 과제에서는 선박마다 운임료가 상이하고, 블록마다 납기 기한이 설정되어 있으며, 운항 스케줄은 고정되어 있다고 가정한다. 이러한 현실적인 제약 조건을 고려하여, 공간 활용률을 극대화하고 적치기간과 비용을 최소화하는 블록 배치 최적화 알고리즘을 개발하는 것이 목표이다.

블록은 3차원 형상 데이터를 2.5D 복셀 형태로 변환하여 처리한다. 그 위에 휴리스틱 기반 알고리즘을 적용해 초기 배치 전략을 구성하고, 이를 강화학습으로 학습시켜 탐색 속도와 배치 효율을 향상시킨다. 알고리즘은 납기 기한을 초과하지 않도록 블록을 선별하며, 선박별 운임료를 고려해 전체 운송 비용과 적치 기간을 함께 최소화하도록 설계된다.

마지막으로, 개발된 알고리즘의 결과를 Unreal Engine 기반의 시뮬레이션으로 시각화한다. 시뮬레이션에서는 블록 배치 과정 중 충돌 여부와 하역 장비의 경로 확보 가능성을 검증함으로써 알고리즘의 실효성을 확인할 수 있다.

2. 요구 조건 및 제약 사항 분석에 대한 수정 사항

요구 조건 및 제약 사항에 기존 대비 추가되거나 구체화된 항목은 다음과 같다.

1. 블록은 크레인으로 직접 배치 가능한 블록과 트레슬 및 트랜스포터(TP)가 필요한 블록으로 구분된다. 크레인으로 이동하는 블록의 경우 이동에 제약이 없으며, 트레슬을 이용한 블록 배치 과정에서 트레슬이 차지하는 공간을 고려하여 블록 배치 알고리즘에 적용한다.

2. 납기일 이전에 도착한 블록은 최대 2~3주 동안 적치할 수 있다.

3. 모든 선박의 적재 공간은 직사각형이다.

4. 적재된 블록은 선수에서 최소 5m의 거리를 확보해야 하며, 블록 간 간격은 최소 1m 이상 유지해야 한다.

3. 설계 상세화 및 변경 내역

3.1. 알고리즘 성능 지표 설정

세 가지 요소(비용, 스케줄링, 적치 기간)를 조합한 알고리즘 평가 함수를 설계하였다. 각 항목은 서로 단위가 다르므로, 0~1 사이 값으로 정규화하여 사용한다.

1. **비용 점수 (Cost Score)**

비용은 낮을수록 좋으므로, 기준 비용 대비 절감 비율을 점수로 계산한다.

$$S_{\text{cost}} = \frac{C_{\text{baseline}} - C_{\text{actual}}}{C_{\text{baseline}}}$$

C_{baseline} = 비용을 고려하지 않은 알고리즘을 적용했을 때 운임 비용
C_{actual} = 설계한 알고리즘을 적용했을 때의 운임 비용

점수는 0~1 사이이며, 값이 클수록 비용 절감 효과가 크다는 것을 의미한다.

2. **스케줄링 점수 (Schedule Score)**

납기일 내에 출항 가능한 블록의 비율로 정의한다.

$$S_{\text{schedule}} = \frac{N_{\text{on-time}}}{N_{\text{total}}}$$

$N_{\text{on-time}}$ = 납기 내 출항 가능한 배치 블록 수
N_{total} = 전체 블록 수

3. **적치 기간 점수 (Staging Score)**

블록의 평균 적치 기간이 짧을수록 높은 점수를 부여한다. 짧은 적치 기간일수록 높은 점수를 부여받아야하므로 역정규화를 한다.

$$S_{\text{staging}} = 1 - \frac{D_{\text{avg}} - D_{\text{min}}}{D_{\text{max}} - D_{\text{min}}}$$

D_{avg} : 해당 알고리즘에서 평균 적치 주 수
D_{min} : 테스트 시나리오들 중 가장 짧은 평균 적치 기간
D_{max} : 테스트 시나리오들 중 가장 긴 평균 적치 기간

이렇게 하면

D_{avg} = D_{min} -> 점수 1 (최고)
D_{avg} = D_{max} -> 점수 0 (최악)

총 점 계산 방식 (Total Score)

세 요소의 가중치를 각각 설정하여 최종 점수를 산출한다.

$$S_{\text{total}} = v_1 \cdot S_{\text{cost}} + v_2 \cdot S_{\text{schedule}} + v_3 \cdot S_{\text{staging}}$$
$$\text{where } v_1 + v_2 + v_3 = 1$$

가중치는 프로젝트 목표에 따라 조정할 수 있다.
비용 절감이 중요하면 v_1을 높게, 납기 준수가 중요하면 v_2를 높게 설정한다.

3.2. 강화학습 상세 설계

강화학습은 배치 알고리즘의 탐색 효율과 최종 배치 품질을 향상시키기 위해 두 가지 주요 의사결정 단계에 적용된다. 단계별로 설계된 에이전트는 주어진 환경에서 보상을 최대화하도록 학습하며, 보상함수는 공간 활용도 및 비용/납기 기반 요소를 반영하여 정의된다.

1) **블록 배치 결정 에이전트 (Block Placement Agent)**
블록 배치를 최적화하기 위해 설계된 강화학습 기반의 의사결정 모듈이다. 이 에이전트는 배치 후보지점 평가에 사용되는 파라미터에 대한 가중치를 업데이트 하여 공간 활용률 등의 보상이 최대가 되도록 조절하고 최적의 블록 배치 후보 지점을 생성한다.

State (상태)	Action (행동)
- 현재 배치된 블록들의 2.5D 복셀 맵 - 후보 위치 리스트 - 블록 형상 및 크기	- 배치할 후보 지점 선택 - 배치 방향 선택 (0도 또는 180도)
Reward (보상)	**Objective (목표)**
- 배치 후 공간 활용률 증가분 - 블록 간 간섭 없는 배치 성공 시 보상 강화 - 충돌 발생 시 패널티	- 배치 후보지점에 대한 파라미터 가중치를 최적화함으로서, 공간 활용률을 극대화하는 최적의 블록 배치 전략을 학습

2) **스케줄 기반 블록-선박 매칭 에이전트 (Schedule-Aware Matching Agent)**
선박마다 운임료가 다르고, 사용하는 선박 수가 증가할수록 비용이 늘어난다. 이 에이전트는 선박의 운항 스케줄과 블록의 납기일, 적치 기간을 고려하여, 각 블록을 적절한 자항선에 매칭함으로써 선박수를 최소화하고, 납기일 초과 및 과도한 적치 기간을 방지하여 전체 운임 비용을 최소화 하는 방향으로 학습한다.

State (상태)	Action (행동)
- 각 선박의 운항 일정 및 운임료 - 블록별 납기 기한 - 현재까지 매칭된 블록-선박 리스트	- 블록을 특정 선박에 할당 - 적치 시점 조정 (가능한 경우)
Reward (보상)	Objective (목표)
- 운임 비용 절감 시 보상 - 블록이 납기 내 선박에 실릴 경우 보상 - 적치 기간이 짧을수록 높은 보상 - 납기 초과 또는 과도한 적치 시 패널티	- 전체 운임 비용과 평균 적치 기간을 최소화하는 선박 매칭

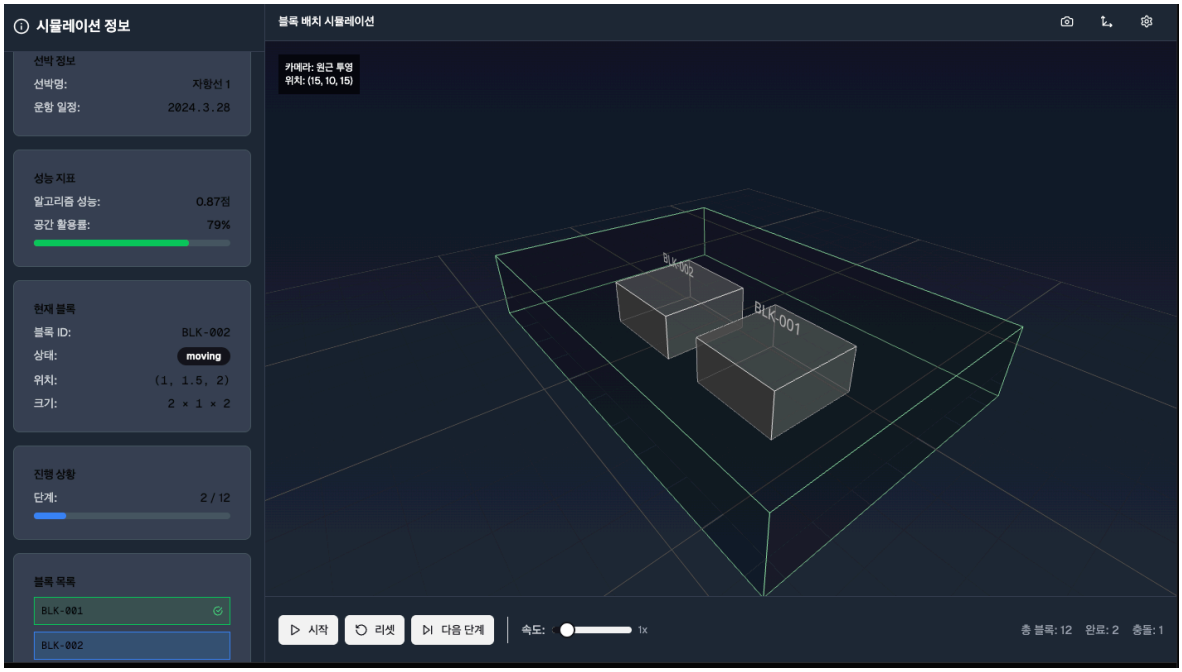
3.3. Unreal 기반 시뮬레이션 상세 설계

기존 Unity 기반으로 개발 예정이었던 시뮬레이션 프로그램을 Unreal Engine 기반으로 변경하였다.

시각화까지의 시나리오는 다음과 같다.

1 단계 : 블록 배치 알고리즘 실행	2 단계 : 최적화 결과 데이터 구조화
입력) 블록 목록 (형상, 납기일, 적재 방식) 선박 정보 (운임료, 출항 스케줄, 적재 공간) 처리) 스케줄 기반으로 블록을 선박에 할당 공간 활용률을 고려해 각 선박 내에서 블록 위치 최적화 출력) 각 블록의 배치 위치 (x, y, z) 배치된 선박, 적치 기간, 충돌 유무, 방향 데이터	알고리즘 결과를 JSON 또는 CSV로 저장
3 단계 : Unreal Engine으로 데이터 전달	4 단계 : 시뮬레이션 및 시각적 검증
Unreal Engine에서 알고리즘 결과 파일을 로드하여, 시뮬레이션 초기 상태 구성	기능) 1. 3D 공간에서 블록 배치를 애니메이션으로 확인 2. 블록 간 간격, 회전 방향, 충돌 여부를 시각적으로 확인 3. 하역 경로 확보가 가능한지 시뮬레이션

시뮬레이션 화면 설계



기능 명세

기능 번호	이름	기능
FR-1	시뮬레이션 정보 패널	시뮬레이션 상태 정보가 좌측 패널에 표시된다. 표시되는 정보) 선박 정보 : 선박명, 출항 일정 성능 지표 : 알고리즘 성능 점수, 공간 활용률 (%) 현재 블록 : 블록 ID, 상태, 위치, 크기 진행 상황 : 현재 단계 / 전체 단계 블록 목록 : 해당 선박에 실리는 모든 블록 표시
FR-2	배치 시뮬레이션 시각화 컨버스	3D 공간에서 블록 배치 상태를 실시간으로 확인할 수 있다.
FR-3	시뮬레이션 제어 UI	사용자는 시뮬레이션 상태를 제어할 수 있다. 세부 기능) 시작, 일시정지, 리셋 다음 단계(블록) 실행 속도 조절
FR-4	요약 정보 표시 (하단 바)	전체 블록 수, 완료 수, 충돌 수 들을 하단에 집계하여 표시한다.

4. 수행 내용 및 중간 결과

4.1. 배치 알고리즘 구현

선박 블록 배치 최적화의 핵심인 휴리스틱 기반 백트래킹 알고리즘을 개발하였다.

배치 알고리즘의 전체 흐름

단계	설명
1. 초기화	블록 정렬, 배치 공간 초기화, 변수 세팅
2. 백트래킹	각 블록에 대해 후보 위치 재귀 탐색
3. 후보 위치 탐색	회전 + 위치 조합 → 유효한 경우만 수집
4. 검증 및 배치	경계, 충돌, 이격 거리 검토 후 배치 수행
5. 결과 도출	최적 배치 구성 반환

핵심 알고리즘 구조

1. **의사결정 제어 모듈**

   이 모듈은 배치 알고리즘의 핵심 단계를 실제로 수행한다. 먼저, 입력된 블록을 단면적 기준으로 내림차순 정렬하여 대형 물체를 우선적으로 선택한다. 이후, 가능한 모든 배치 위치 및 회전 조합을 탐색하며, 배치 불가능 시 이전 상태로 되돌아가 다른 선택지를 시도하는 재귀적 백트래킹 메커니즘을 통해 최적해를 탐색한다. 탐색 과정에서 배치된 블록 수가 최대가 되는 구성을 지속적으로 추적하여 최종적으로 최적해를 결과로 반환한다.

2. **배치 후보 위치 탐색 모듈**

   이 모듈은 블록의 모든 유효 배치 위치 및 회전 조합을 생성한다. 블록을 두 가지 방향(0°, 180°)으로 회전시킨 후, 각 회전 상태에 대해 그리드의 모든 좌표에 대해 블록 배치 가능성을 검사한다. 이 과정에서 경계를 초과했는지 여부, 다른 블록과의 충돌 여부, 그리고 선박의 선수 이격 거리 및 블록 간 간격 유지 등 제약 조건을 검증하며, 유효한 조합을 후보 리스트로 생성하여 제어 모듈에 전달한다.

3. **가상 배치 공간 모듈**

   배치 공간은 2차원 배열로 표현되며, 각 셀은 비어있는 상태 또는 블록이 차지한 상태로 나타낸다. 이 모듈은 특정 좌표에 블록을 배치하고 그리드를 갱신하는 기능, 배치된 블록을

제거하고 상태를 복원하는 기능, 그리고 배치 유효성을 검증하는 기능을 포함하며, 재귀 탐색 과정에서 핵심적으로 활용된다.

4. 선박 특화 배치 공간 모듈

이 모듈은 일반 배치 공간의 기능을 확장하여, 실제 선박 환경에서 요구되는 제약 조건을 반영한다. 선박의 앞부분인 선수로부터 5m 이내에는 블록을 배치할 수 없도록 하고, 블록 간에는 최소 1m 간격을 유지하도록 한다. 이러한 제약 조건을 포함한 유효성 검증을 통해 현실적인 운송 환경에서도 적용 가능하도록 한다.

아래는 랜덤 생성한 블록의 배치를 시각화한 결과이다.

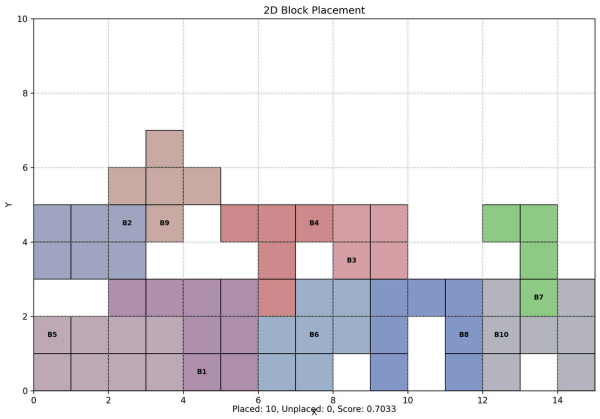


그림 1. 2D 배치도

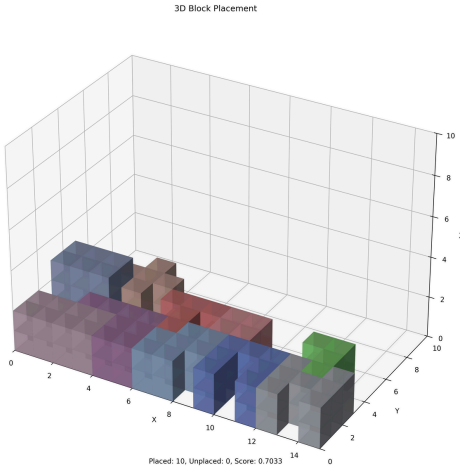


그림 2. 3D 배치도

4.2. 블록 2.5D 복셀화

배치 알고리즘에서 사용할 수 있도록 원본 3D 모델을 복셀 형태로 처리한다. 원본 FBX 형식의 3D 블록 모델을 2.5D 복셀 데이터로 변환하여 사용하였다. 2.5D 복셀은 2D 격자의 각 셀이 해당 위치의 높이 정보를 포함하는 구조로, 3D 형상의 기하학적 특성을 유지하면서도 데이터 용량과 계산 복잡도를 줄일 수 있다.

초기에는 3D 모델을 바로 2.5D 복셀로 변환하는 방식으로 접근했으나, 복잡한 형상이 뭉개지거나 손실되는 문제가 발생하였다. 해결을 위해, 먼저 3D 복셀화를 수행한 뒤 이를 2.5D로 변환하는 방식을 시도하였는데 결과적으로 원본 형상의 세밀한 특성이 더 잘 보존되었다.

복셀화 프로세스

1. 전처리 단계 : fbx to obj 변환

약 320개의 FBX 형식 3D 블록 모델을 복셀화하기 위해, 이를 OBJ 형식으로 일괄 변환하는 자동화된 전처리 스크립트를 작성했다. 파이썬 스크립트는 subprocess 모듈을 활용해 Blender를 백그라운드에서 실행시키고, 각 FBX 파일을 로드한 뒤 순수 기하학 정보만 남겨 OBJ 형식으로 저장하는 Blender 전용 코드를 실행한다. 이를 통해 대량의 모델을 사용자 개입 없이 신속하고 일관되게 변환할 수 있다.

2. 3D 복셀화 단계: 고해상도 복셀 그리드 생성

변환된 OBJ 형식을 기반으로, 1m 해상도의 3D 복셀화를 수행한다. 이 과정에서는 블록 형상의 세밀한 구조를 유지하는 것이 핵심이며, trimesh 라이브러리를 이용해 메시 내부를 빈틈없이 채운 solid 모델로 변환하였다. 복셀화에 앞서 메시의 오류(구멍, 중복 면 등)를 자동으로 수정하는 과정을 거쳐 복셀화 오류를 사전에 방지하였다. 최종적으로 생성된 3D 복셀 그리드는 각 (x, y, z) 좌표에 복셀의 존재 여부를 이진 형태로 기록한다.

3. 2.5D 변환 단계: 복셀 데이터 압축

생성된 3D 복셀 데이터는 각 (x, y) 위치에 대해 높이 방향(Z축)의 정보를 요약하여 2.5D 형태로 변환된다. 이 과정에서는 각 (x, y) 셀을 기준으로 복셀이 존재하는 최소 Z값과 최대 Z값을 탐색하며, [min_z, max_z] 형태로 압축한다. 해당 정보는 (x, y, [min_z, max_z]) 구조로 저장되며, 블록의 바닥면과 윗면 형상을 간결하고 정확하게 표현할 수 있다. 변환된 데이터는 최종적으로 VoxelBlock 객체로 포장되어 배치 알고리즘의 입력으로 사용된다.

아래는 복셀화를 수행한 원본 데이터와 복셀화 결과이다.

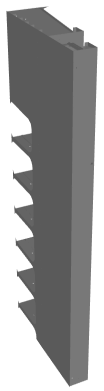


그림 3. 원본 데이터

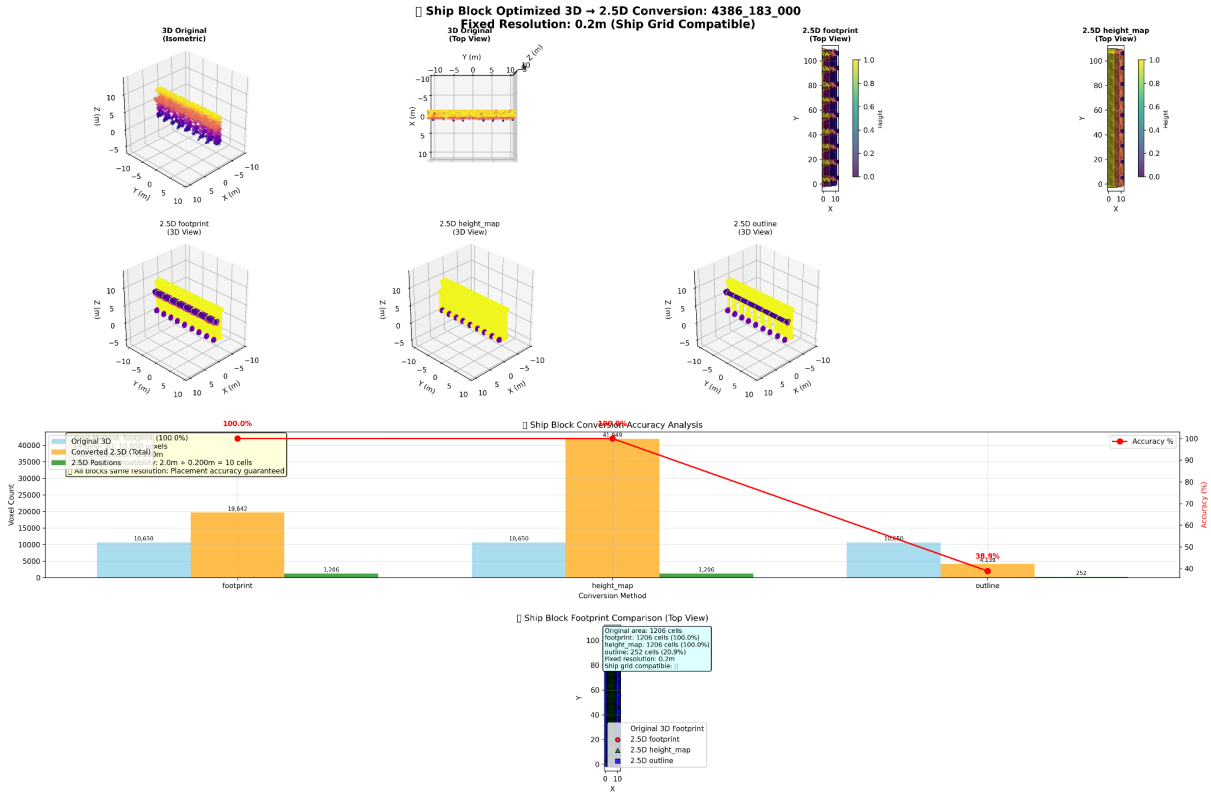


그림 4. 복셀화 결과

실험에 사용된 블록은 내부 구조와 곡면 형상을 포함한 복잡한 형태의 FBX 모델이다. 복셀 해상도는 기본적으로 1m 단위를 적용하였으며, 보다 정밀한 표현을 위해 0.2m 단위의 고해상도 실험도 병행하였다.

두 번째 줄의 이미지는 원본 3D 형상(좌측 상단)을 세 가지 방식(footprint, height_map, outline)의 2.5D 복셀화 결과를 시각적으로 비교한 자료이다. 이들 방식은 동일한 3D 복셀 데이터를 2.5D로 축소하는 서로 다른 변환 방식으로, 각 방식에 따라 복셀 셀 수, 형상 보존 정도, 계산 효율성이 달라진다.

footprint는 각 (x, y) 위치에서 블록이 존재하는 높이 구간을 기록하고, height_map은 최대 높이값만을 보존하여 외곽 형상을 정밀하게 표현한다. outline 방식은 내부 복셀을 제외하고 외곽 윤곽만 추출함으로써, 셀 수를 최소화하는 데 초점을 맞춘 방식이다.

하단의 정량 분석 결과에 따르면, footprint와 height_map 방식은 각각 약 10,000개 이상의 복셀 셀을 유지하며 원본 대비 100%에 가까운 형상 보존률을 보였다. 반면 **outline** 방식은 **내부 셀을 생략**함으로써 총 셀 수가 252개로 크게 줄어들었고, 이로 인해 보존률은 약 38.9%로 측정되었다.

그러나 이 낮은 보존률은 단점이라기보다는 오히려 outline 방식의 장점을 드러내는 지표로 해석할 수 있다. 해당 방식은 **외곽 형태만을 유지**한 채 **데이터 양을 획기적으로 줄이**기 때문에,

배치 알고리즘에 필요한 충돌 검사가 외곽 기준으로만 이루어질 경우 높은 계산 성능과 처리 속도를 기대할 수 있다. 따라서 outline 방식은 복잡한 내부 형상까지 필요하지 않은 경량 시뮬레이션 또는 배치 검증 환경에 적합한 효율적인 대안으로 고려될 수 있다.

4.3. 스케줄링을 고려한 배치

1. 블록-자항선 매칭 알고리즘

이 알고리즘은 각 블록의 납기일, 예상 적치 기간, 선박의 운항 스케줄(출항일) 등을 고려하여 블록을 적절한 선박에 배정하는 기능을 수행한다. 기본적으로 블록은 납기일 내 출항 가능한 자항선만을 후보로 가지며, 각 자항선의 남은 적재 여유 공간과 배치 가능 여부를 사전에 확인한다. 가능한 후보 선박 중, 납기일 준수 및 적치 기간 최소화를 동시에 만족할 수 있는 조합을 우선적으로 선택하며, 블록 배치 알고리즘을 적용해 운임 비용 최소화를 수행한다. 이 과정을 통해 전체 자항선-블록 매칭을 완성한다.

2. 강화 학습을 한 알고리즘 성능 향상

기초 알고리즘 구현 이후, 블록-자항선 매칭 의사결정 강화학습 프레임워크를 도입할 예정이다. 에이전트는 블록의 상태 및 자항선 정보를 바탕으로 매칭 결정을 내리고, 보상 함수는 운송 효율성과 납기 준수율을 반영하도록 설계된다. 구체적으로, 자항선 사용 대수를 줄이는 방향(=운임 비용 절감)에 보상을 부여하며, 적치 기간이 짧을수록 높은 보상이, 반대로 납기일 초과나 적치 기간 초과 시에는 패널티가 적용된다. 이를 통해 에이전트가 점진적으로 운송 효율이 높은 매칭 전략을 학습할 수 있도록 유도한다.

3. 최적화 파라미터 및 향후 계획

스케줄링-블록 배치 알고리즘의 성능을 평가하기 위한 점수 평가 기준을 설계할 것이다. 이 평가지표에는 납기일 준수, 블록 적치 기간, 자항선 운임 비용 등이 포함된다. 이후 이 점수 기준을 바탕으로 강화학습의 보상 함수를 구성하며, 학습 과정에서 각 평가 항목의 파라미터 가중치를 업데이트하여 알고리즘 평가 점수를 최대화 하기 위해 학습한다. 점수 평가 기준의 최적화를 위해 추가적으로 필요한 파라미터의 도입 여부에 대해서도 검토중이다.

5. 구성원별 진척도

이름	담당 업무 및 진척 내용
서민성	배치 알고리즘 구현 및 개선을 담당하였으며, 현재까지 핵심 구조 구현을 완료하였다. 또한 3D 모델을 기반으로 한 2.5D 복셀화 프로세스를 설계 및 구현하였으며, 현재는 납기일과 선박 스케줄을 반영한 스케줄링 기반 배치 알고리즘 개발을 진행 중이다.
김도완	배치 알고리즘 구조 설계 및 구현에 참여하였고, 강화학습 적용을 위한 관련 기법 조사 및 보상함수 설계 초안을 작성하였다. 현재는 강화학습 기반 배치 개선 전략에 대한 구체적인 설계 작업을 수행 중이다.
유주연	스케줄링 기반 배치 알고리즘에 필요한 입력 데이터를 정리하고, 해당 알고리즘에서 활용할 수 있도록 CSV 파일을 생성하여 개발을 지원하였다. Engine 기반 시뮬레이션 화면의 UI 구성과 인터페이스를 설계했다.

6. 결론 및 계획

6.1. 현재까지 성과 요약

현재까지의 주요 성과는 다음과 같다

배치 알고리즘 구현 완료
휴리스틱 기반 백트래킹 알고리즘을 중심으로 한 블록 배치 로직을 구현하였으며, 선박 특화 제약 조건(선수 이격 거리, 블록 간 간격 등)을 포함하는 배치 공간 모델을 설계하였다.

2.5D 복셀화 파이프라인 구축
FBX 모델을 기반으로 3D 복셀화를 수행하고, 이를 2.5D 데이터로 변환하는 복셀화 과정을 시도했다. Outline 방식으로 2.5D 복셀화하여 알고리즘에 적용해볼 계획이다.

시뮬레이션 설계
Unreal Engine 기반 시각화 화면의 레이아웃 및 기능 구성을 완료하였으며, 배치 알고리즘 결과를 시각적으로 검증하기 위한 전체 흐름을 정리하였다.

강화학습 및 스케줄링 구조 설계
강화학습 기반 탐색 개선 전략과 납기일·운임 스케줄을 반영한 배치 알고리즘 구현을 위한 구조 설계를 진행하였다. 이를 위해 보상 함수 구조와 주요 파라미터 정의를 포함한 상세 설계 초안을 작성하였다.

6.2. 최종 제출을 위한 향후 계획 및 개선점

향후 계획 및 개선점은 다음과 같다.

스케줄링 기반 배치 알고리즘 완성
납기일, 선박 운임료, 스케줄 제약 등을 고려하여 블록을 선박에 매칭하는 로직을 구현하고, 기존 배치 알고리즘과 통합할 예정이다.

강화학습 기반 탐색 효율 향상
현재 구조에 강화학습 기반 에이전트를 적용하여, 배치 후보 탐색 및 블록-선박 할당 과정의 성능을 개선하고 실험적 성능 비교를 진행할 계획이다.

시뮬레이션 인터랙션 및 결과 검증
Unreal Engine 기반 시뮬레이션 시스템에서 블록 배치 결과를 시각적으로 확인하고, 충돌 및 경로 확보 여부를 검증하는 기능을 구현한다.

성능 평가 및 실험 설계 정비
복수의 테스트 시나리오에 대해 배치 효율, 공간 활용률, 비용 절감 효과를 수치적으로 비교 분석할 수 있는 실험 구조를 확정하고, 알고리즘 평가 방식을 고도화한다.