

Unity ML-Agent를 활용한 기체방사선감시기 시료 이송 배관 설계

전남율* · 신수용**†

Implementing the Design of Sample Transport System of Gas Process and Effluent Radiation Monitoring System through ML-Agent

NamYul Jeon*, Shin Soo Young**†

Key Words : ANSI/HPS N13.1, ML-Agent, Clustering, Transfer Rate, RMS

ABSTRACT

Through ML-Agent, accomplish the design of GPERMS(Gas Process and Effluent Radiation Monitoring System) location and sample transfer pipe routing in nuclear power plant. And, show the proper GPERMS location and expected sample transfer ratio through the analysis of reward data collected from ML-Agent using clustering model. It can visualize whether transfer ratio requirement is satisfied or not.

1. 서 론

1.1 연구배경 및 필요성

원자력발전소는 인허가 요건이 강하게 요구되는데, 그 이유는 원자력발전소의 안전성 때문이다. 원자력발전소의 안전성은 곧 대민 보건 및 생명에 직결되기 때문이다. 방사선 감시기는 그 안전성과 관련 있는 여러 원자력발전소 내 시스템 중 하나이다.

방사선감시시스템 내에는 시스템 프로세스와 대기로 방출되는 가스 내에 방사선 누출을 감시하는 기체공정유출물방사선감시기와 소내 작업자에게 주변 지역의 방사선 정보를 제공하는 역할을 하는 지역방사선감시기가 있다. 그 중 기체공정방사선감시기는 원자력발전소 내 공기가 외부로 유출되는 지점에서 방사선 및 방사능이 유출되는지 감시한다. 방사선 및 방사능이 지속해서 생성되는 원자력발전소의 특성상 유출물을 통해서 방사능이 유출되지 않도록 계속적 감시가 중요하다.

기체공정유출물방사선감시기는 감시기 특성에 따라 프로세스 라인에 직접 붙여서 감시하는 In-Line Type과 채취한 시료를 감시하는 Off-Line Type이 있다. 이 중 Off-Line Type의 방사선감시기의 감시 형태는 감시 대상 프로세스 라인(스택 혹은 덕트) 내에 설치된 노즐을 통해 시료를 채취하고, 시료 채취 라인(튜브)을 통해 시료를 끌어와 감시기 분석기에서 방사선 수치를 측정하는 방식이다. 채취한 시료가 감시기까지의 도달하는 시료이송계통에서 성능을 보장하는 것은 감시기 노즐부터 방사선감시기까지의 전체 시스템에 있어서 방사선 감시 성능을 보장하는 중요한 역할을 하므로, 본 논문에서는 이에 대해 다루고자 한다.

1.1.1 방사선감시기 설계 요건에 따른 설계 상황

ANSI/HPS N13.1 (1999)에서는 총 이송량에 대한 성능 기준을 “자유 흐름에서 입자 및 증기 오염물질의 총 이송(이하 이송율)은 50% 이상이 되어야 한다.”[1]라고 제시하고 있다. 대한민국 내의 최신 원자력발전소에서는 이 기준을 인허가 기준으로 삼고, 성능 조건을 만족하도록 요구받고 있

* 한국전력기술(KEPCO Engineering & Construction Company, Inc.)

** 국립금오공과대학교(Kumoh National Institute of Technology)

† 교신저자, E-mail : wdragon@kumoh.ac.kr

다. 그러므로 이 기준을 적용받는 발전소에서는 성능 검증 시험을 통해서 성능 기준의 만족 여부를 보여야 한다. 이송율은 시료 채취 노즐과 방사선감시기 사이의 시료이송배관 내에서 시료가 이송되는 정도를 나타내는 것이다. 이 기준을 만족할 수 있는 가장 좋은 조건은 시료 채취 노즐 근처에 곧바로 방사선감시기가 설치되어서 시료이송배관의 길이가 매우 짧아지는 것이다.

원자력발전소 내 기기들은 IEEE 603 및 참고 요건들에 따라 ‘안전급’ 및 ‘비안전급’ 기기로 분류된다. 특히 ‘안전급’ 기기로 제시되고 있는 기기는 원자력발전소 사고 시에도 발전소 안전성을 보장하기 위해서 운전할 수 있도록 요구된다. 기기에 따라서는 사고 시에 매우 가혹한 환경 내에 놓이는 상황 속에서도 운전 가능하여야 한다. 특히 안전급 유출물방사선감시기의 시료 채취 노즐의 경우, 감시대상의 시료를 채취하여야 하므로 사고 시에는 가혹한 환경에 놓이게 된다. 하지만, 전자제품 및 반도체 부품들을 다수 포함하고 있는 방사선감시기는 가혹한 방 내에서는 기능을 유지할 수 없으므로, 가혹한 방 내에 설치할 수 없다. 역설적으로, 주요 감시 대상 프로세스들은 사고 시에 가혹한 환경에 놓임을 고려했을 때, 그 프로세스를 감시하는 감시기들이 방사선 및 온도 등의 조건이 온화한 조건에 설치되어야 하므로 감시기는 시료 채취 노즐에서 먼 곳에 설치되어야 하는 상황에 놓인다. 이에 따라 현업에서는 성능 조건을 만족하기 위해서 가급적 노즐에서 가까운 감시기 위치를 설계하고, 설치 위치가 멀더라도 이송관 설계에 따른 이송율 상승을 가져올 수 있는 위치에 설계해야 하는 까다로운 설계 상황에 봉착하는 경우가 많다. 설계상 경험적 데이터를 이용하는 것이 매우 중요하며, 이에 따라 설계자마다 설계 효율성 차이가 발생한다. 더구나 ANSI/HPS N13.1(1999)에서는 설계 및 설치 후 실질적인 성능 달성 요건만 주었을 뿐 설계 요건을 주지 않아 현업에서는 엔지니어의 설계 역량에 의존해야 하는 상황이다.

1.1.2 설계 상황에 따른 문제점

방사선감시시스템은 다른 설비 시스템들이 설치된 후 설치가 진행되어야 하는 후속 설비인데, 선행 설계가 미리 되어 있음에도 선행 설치 시스템들의 설계 변경으로 인하여, 설계가 변경하는 상황이 발생한다. 이로 인해 방사선감시기 위치 변경 사례가 발생하며, 그 시점에서 이미 설비들이 자리를 차지하고 있어서 적절한 방사선감시기 위치를 찾기는 어렵다.

이와 더불어 최근에는 ANSI N13.1(1969)가 적용되었던 구형 원자력발전소에서 방사선감시기의 성능 확보를 위해서 ANSI/HPS N13.1(1999) 적용이 요구되고 있다. 설비들이 빼곡히 자리를 차지하고 있는 상황 속에서 적절한 방사선감시기 위치를 찾고 여러 설비와 배관을 피해 시료 이송 배관 라우팅을 하는 것은 많은 어려움이 따른다.

1.2 연구 목표

본 연구에서는 Unity사의 ML-Agent를 통하여 ANSI/HPS N13.1(1999)의 이송율 요건을 만족할 수 있는 적합한 방사선감시기 위치를 제안하고, 그 위치에 대한 시료 이송 배관 라우팅과 예상 이송율을 제시하는 툴을 제안한다. 보통 강화학습 방법을 사용할 경우, 여러 위치를 평가하고 가장 높은 위치만을 제시하겠지만, 본 연구에서는 방사선감시기 설치 위치의 주변을 모두 평가하기 위해 ML-Agent를 데이터를 확보하기 위한 툴로서 이용하였고, 확보한 데이터를 평가하는 툴은 파이썬(Python)을 이용하여 작성하였다.

2. Unity ML-Agent를 통한 강화학습

2.1 강화 학습

강화학습은 머신러닝 기법 중 하나로, 주어진 환경 내에서 에이전트가 시행착오와 보상을 통해서 최대한의 보상을 받을 수 있도록 행동을 학습한다. 기존의 프로그래밍 및 머신러닝 기법 중 지도학습은 명확한 방법을 명시하여 알려주지 않아도 행동을 통한 학습을 통해 더 좋은 결과를 얻을 수 있는 학습 방법을 배운다.

강화학습의 구성요소는 보통 6가지를 제시한다. ‘에이전트(Agent)’는 환경과 상호작용하면서 행동을 학습하고 결정하는 주체이다. ‘환경(Environment)’은 에이전트가 상호작용하는 세계로 에이전트가 행동하면 행동에 대한 결과를 에이전트에게 반환한다. ‘상태(Status)’는 현재 환경의 상태를 나타내며, 다음 행동을 결정하는 원인이 되는 정보를 제공한다. ‘행동(Action)’은 에이전트가 환경 내에서 취할 수 있는 행동이다. ‘보상(Reward)’은 행동을 취한 이후 이 행동에 대해 환경으로부터 받는 정보이다. ‘정책(Policy)’은 환경 속 특정 상황에서 에이전트가 정하게 될 전략이다.

Fig. 1은 강화학습이 진행되는 과정을 나타낸 그림이다. a는 행동, s는 상태, r은 보상으로 나타내었다. 우선, 환경은 환경의 상태 정보 s를 에이전트에 전달한다. 에이전트는 받은 정보를 바탕으로 에이전트가 취할 수 있는 행동 중

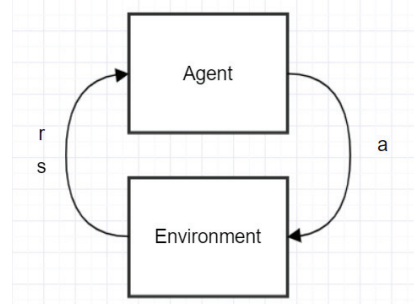


Fig. 1 강화학습 작동 원리

최적의 행동 a 를 선택한다. 행동 a 를 선택하는 기준은 정책에 의해서 결정된다. 선택한 행동 수행 후, 환경의 상태 s 는 변하게 되며, 환경은 에이전트가 결정한 행동에 대해 ‘새롭게 변경된 보상 r ’와 ‘새롭게 바뀐 환경의 상태 정보 s ’을 에이전트에게 반환한다. 이러한 과정을 반복해서 수행하면서 보상을 최대화하는 방향으로 에이전트는 학습하게 된다. [4]

학습하면서 에이전트는 최대한의 보상을 얻기 위해 과거의 보상받았던 경험을 재이용할 수 있지만, 더 효과적으로 최대한의 보상을 얻어내기 위해서 역설적으로 이용하지 않았던 방식을 탐험함으로써 더 좋은 방식이 있는지 찾아가게 된다. 에이전트는 다양한 방식의 경험을 하고 실패한 경험은 걸러내면서 더 성공적인 전략을 찾는 방식이다.

2.2 시료 이송 배관 설계로의 강화학습 도입

발전소 환경 내 다양한 설계 요소 내에서 에이전트가 설계 환경과 반복적인 상호 작용을 하면서 최적의 해결책을 찾는 강화학습(Reinforcement Learning)은 이송을 자동화 설계에 있어서 적용할 수 있는 장점이 많은 방식으로 판단된다.

직접 행동하며 학습을 해나가는 ‘에이전트(Agent)’가 장애물 등의 주변 ‘환경(Environment)’을 파악하면서 자신의 ‘상태(State)’를 이해하고 이에 따라 ‘행동(Action)’을 취하는 방식, 그리고 그 행동 안에서 자신이 이득을 취하여야 할 사유가 될 보상과 택하게 되는 ‘정책(Policy)’은 시료 채취 이송관 설계 자동화에 있어서 맞는 부분이 많다.

환경은 발전소 내 룸의 크기 조건과 내환경 요건(NPP Room Condition)을 적용한다. 발전소 내 주어진 환경에 따라서 에이전트는 상태를 이해하고 정책적(Policy)으로 이송율(Transport Ratio) 상 이득이 되는 방향으로 이동한다. 달성한 이송율의 수치에 따라 보상을 제공한다. 에이전트의 이동경로는 곧 파이프 설계 경로가 되며, 곧, Field Worker 및 파이프 설계 엔지니어의 역할을 한다. 에이전트는 보상을 이용해서 점점 이송율이 높은 지역으로 이동한다. 본 연구에서는 이동하면서 얻은 데이터들을 수집하고, 최종적으로 그 데이터를 이용한 그래프를 보여준다.

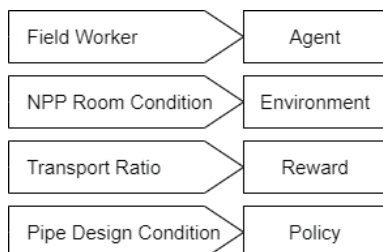


Fig. 2 이송관 설계의 강화학습 적용

* NPP : Nuclear Power Plant

2.3 Unity ML-Agent와 군집화 모델

Unity ML-Agent의 방사선감시기 시료 이송 배관 설계에 적용은 강화학습을 적용하기 위한 적합한 도구를 제공하고 있다.

Fig. 4와 같이 Unity Editor를 통해서 에이전트가 학습할 수 있는 환경을 조성할 수 있다.

Unity Editor 내 스크립트 작성을 통해 에이전트 이동, 보상, 정책과 관련한 설정을 할 수 있다. Unity Editor에서 설정한 환경을 기반으로 Fig. 5의 Unity ML-Agent Tool Kit을 통해 에피소드를 반복하여 학습을 진행한다.

이 연구에서 적용한 학습 프로그램 환경은 Table 1과 같다.

ML-Agent 내에서는 머신러닝을 위한 Agent 클래스를 제공한다. Agent 클래스에서는 머신러닝을 수행하기 위한 다양한 메서드를 제공하고 있고, 메서드 내에 새로운 내용을

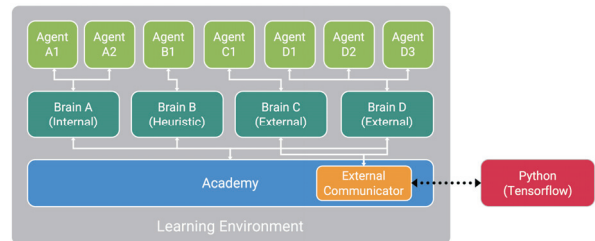


Fig. 3 ML-Agent 구조 [5]

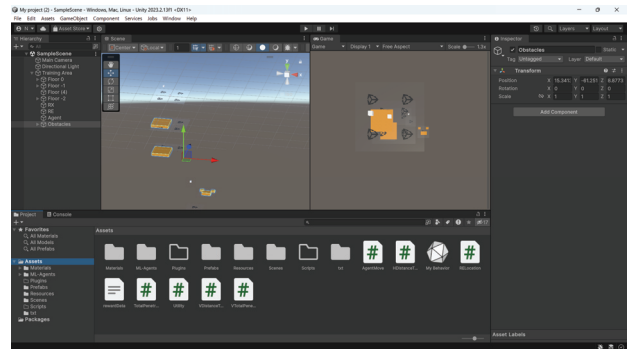


Fig. 4 Unity Editor 설계 환경



Fig. 5 Unity ML-Agent 학습

Table 1 학습 프로그램 환경

프로그램	버전
ML-Agent	0.30.0
ML-Agent-envs	0.30.0
Pytorch	2.4.1
Python	3.12.3
CUDA	11.8

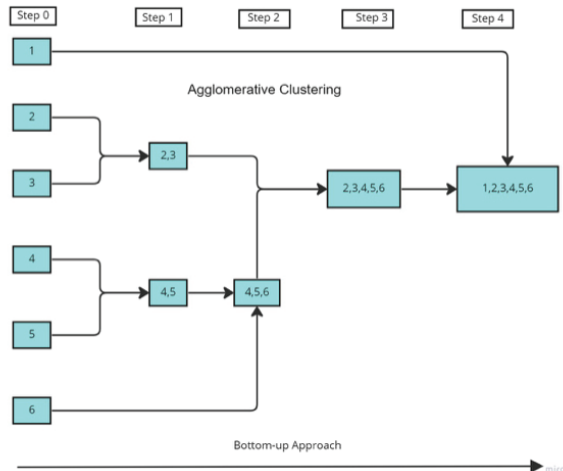


Fig. 6 병합 군집 [6]

오버라이드하여 사용할 수 있다. 예를 들어, Agent 클래스 내에는 OnEpisodeBegin(), CollectObservations() 등의 여러 메서드들이 있는데, 이 중 에이전트 움직임과 관련한 메서드는 OnActionReceived() 메서드가 있으며, 해당 메서드 내에 에이전트 이동과 관련한 내용을 오버라이드하여 작성한다. 오버라이드한 내용과 관련해서는 2.4장에서 기술하였다.

군집화 모델은 병합군집을 활용하여 보상값이 높은 군집을 찾도록 하였다.

2.3.1 Agglomerative Clustering(병합 군집) [6]

병합 군집(agglomerative clustering) 알고리즘은 계층적 군집화(hierarchical clustering)로도 불리며, 여러 개의 군집 중에서 가장 유사도가 높은 (혹은 거리가 가까운) 군집 두 개를 선택하여 하나로 합치면서 군집 개수를 줄여 나간다.

2.4 알고리즘 작성

연구 방법 및 설계 조건을 반영하여 개략적으로 표현한 알고리즘은 Fig. 7와 같다.

에이전트가 이동하기 전 방사선감시기는 환경요건이 온전한 것으로 설정된 ‘설치가능한 룸’ 위에 랜덤 좌표로 배치된다. 에이전트가 이동하여 방사선감시기와 충돌하면 에피소드

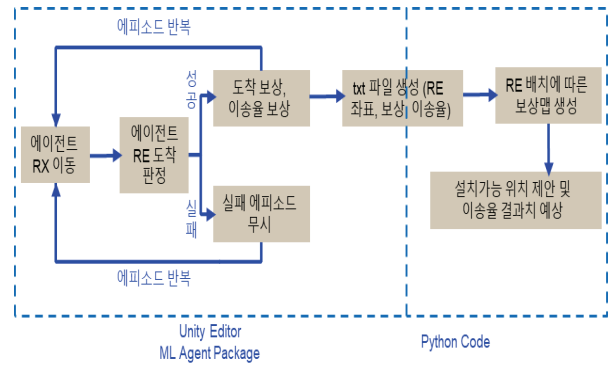


Fig. 7 학습 수행 알고리즘

* RX : 샘플 프루브 설치 위치(에이전트 이동 시작점)

** RE : 방사선감시기 설치 위치

드는 종료되고 에피소드가 진행되는 동안 집계된 보상 값은 txt 파일 내에 저장된다.

수집된 데이터는 파이썬 코드로 정리하여 보상 데이터 맵을 생성하고 가장 적절한 방사선감시기 위치를 제시한다.

2.4.1 알고리즘 및 관련 코드

알고리즘의 내용은 원자력발전소 내 기체공정 및 유출물 방사능감시기의 시료이송배관 이송율 요건 적용 및 경험 사례 [8]의 내용을 기반으로 한다. 주요 내용은 이송율 요건을 만족하기 위해서 시료 채취 노즐과 방사선감시기 사이에 설치되는 시료이송배관의 이송율(TR)이 50% 요건을 만족하여야 한다, 이를 위해서는 둘 사이가 매우 가까워야 하며, 이것이 불가능할 경우 수직 거리를 최대한 많이 사용하고 수평 거리를 최대한 적게 사용하도록 해야 한다는 것이다. 하지만, 경험 사례에서 보면, 가혹한 위치에 설치되는 감시 대상인 시료 채취 노즐(Rx)과는 달리, 방사선감시기(Re)는 환경요건이 온화한 지역에만 설치되어야 하므로 사실상 Re와 Rx는 멀어질 수밖에 없다. 이러한 내용은 Algorithm 상 다음과 같이 작성되었다. 첫째, 방사선감시기(Re)의 위치는 설치 가능한 여러 룸 중 하나를 무작위(Random)로 선정하고, 그 공간 내에 무작위(Random)로 지정한다.

둘째, 에이전트 이동 전에 방사선감시기 방향으로 방향을 바꾼다. 방향을 바꿀 시, 수직 이동 전에는 수직 방향으로 방향을 바꾸고, 수평 이동 전에는 수평 방향으로 방향을 바꾼다. 셋째, 에이전트는 장애물을 피하여 이동한다. 넷째, 수평 이동(Move_S_XY)보다는 수직 이동(Move_S_z)을 먼저 사용하도록 한다. 다섯째, 굽어지지 않는 직선 이동(S)을 하도록 요구한다. 여섯째, 수직 이동 거리 및 수평 이동 거리를 통해서 얻는 이동 거리 수치를 통해서 현재 에이전트 이동거리에 대한 이송율 값(TRNS_i)을 얻는다. 일곱째, 모든 이송율 값을 곱하여 최종적으로 총 이송율(TRNS) 값을 얻는다. 이동거리에 따른 TRNS_i은 ANSI/HPS N18.1에서 제시하고 있는

Table 2 에이전트 이동 알고리즘

Algorithm
Require : Rx, Re, TRNS _i , TR, Obxy, Obz, x, y, z, S, Move, R
Ensure : PathToTarget
1. TR = 0.5
2. P = (x, y, z)
3. D = {x, y, z}
4. FinishPoint = Re
5. FindingPath = True
7. StartPoint = Rx
8. Set_FinishPoint(R(x, y, z, Random))
9. While FindingPath do
10. if Obz = None,
11. get_D(z)
12. Move_S_z
13. get_P(z)_ (StartPoint)
14. get_TRNS _i (P(z))
15. TRNS *= TRNS _i
16. else , get_D(x, y)
17. end if
18. if Obxy = None,
19. Move_S_xy
20. get_P(x, y)
21. get_TRNS _i (P(z))
22. TRNS *= TRNS _i
23. else , get_D(z)
24. end if
25. if Obxy or Obz = Re,
26. end_Move
27. return TRNS
28. end while
29. if TRNS > TR or TRNS = TR
30. return True
31. if TRNS < TR
32. return False

* (참조 수치 및 변수) Rx : RX 위치, Re : RE 위치, TRNS : 현 총 이송율, TR : 총 이송율 요건, Obxy : x, y 평면 방향 장애물, Obz : 직각 방향 장애물, P : 이동길이, D : 이동방향, S : 직선 방향, Move : 이동, x, y, z : xyz 좌표, R : Re 설치공간

Deposition 코드를 기반하고 있다. 알고리즘에서는 자세히 표현하지 않고 있지만, TRNS_i(P(z))값은 미리 저장해둔 수직, 수평 이동거리 별 Deposition 코드 데이터와 알고리즘 내 이동 거리 데이터를 비교하여 이에 대응하는 값을 제시하도록 하는 코드이다.

Agent 클래스 내 오버라이드 코드에서는 알고리즘에서의 내용을 파이션 코드로 구현하였다.

Table 3 Agent 클래스 내 오버라이드 코드 내용

Method Override
1. Public override void OnEpisodeBegin() StartPoint = Rx TRNS = 1 RELocation()
2. Public override void CollectObservations() Sensor.Rx Sensor.Re
3. Public override void OnActionReceived() MoveAgent() HD = HDistanceTracker() VD = VDistanceTracker() HTP = CTransferRate(HD) TRNS *= HTP VTP = CTransferRate(VD) TRNS *= VTP lineRenderer
4. Public override void OnCollisionEnter() if (TRNS < 0.4), R -= 11.0f if (TRNS < 0.5), R -= 3.0f if (TRNS >= 0.5), R += 3.0f EndEpisode()

OnEpisodeBegin()은 매 에피소드마다 다시 초기화되는 내용들이다. 매 에피소드에서는 이송율 값과 에이전트 위치가 초기화되어야 하므로, StartPoint와 TRNS값을 초기화하였다. 방사선감시기의 위치는 매 에피소드마다 바뀌어야 하므로 Re의 위치를 무작위로 바꿔주는 RELocation() 메서드를 콜하도록 하였다.

CollectObservations() 메서드는 센서 관측과 관련된 메서드이다. Rx와 Re에서 센서를 통해 에이전트 이동 경로를 벡터로 측정하도록 하였다.

OnActionReceived() 메서드는 에이전트 이동과 관련된 메서드이다. MoveAgent() 메서드는 에이전트의 장애물 관측 및 이동과 관련된 코드로, 코드 작성 내용은 Table 2 내 에이전트 이동과 관련된 알고리즘 내용에 기반한다. HDistanceTracker()와 VDistanceTracker() 메서드는 수평 이동과 수평 이동을 인식하여 이동 거리를 측정하도록 하는 메서드이며, CTransferRate() 메서드는 이동 거리를 통해서 이송율을 산정하는 코드이다. 해당 코드는 Table 2의 설명에서 언급했듯이, 미리 저장해둔 Deposition 코드 데이터와 측정한 이동 거리를 비교하여 이에 대한 이송율 값을 반환한다. LineRenderer는 이동거리에 대한 흔적을 남겨서 시각적

으로 시료이송배관 라우팅을 제공한다.

OnCollisionEnter()는 에이전트 충돌과 관련된 메서드이다. 에이전트가 Re와 충돌하면 에피소드는 종료된다. 종료 전에 TRNS를 if 조건문 내의 이송율 기준 값과 비교하여 보상값을 제공한다.

Mapping과 관련된 코드는 Table 4와 같다.

Main 실행 부분은 축적 데이터를 불러올 파일 경로를 지정하고 실행을 담당한다. read_data()는 파일을 불러와서 ‘,’를 기준으로 각 방사선감시기 위치에 따른 보상 데이터를 분류하고 array에 저장하는 역할을 한다. apply_agglomerative_clustering()은 sklearn.cluster 라이브러리를 이용해서 병합군집을 적용하였다. def plot_clusters_3d()는 군집화 결과를 3D 그래프로 시각화하며, 그 내부에서는 3D 그래프를 생성하고, 군집에 색상을 지정하고, 축 레이블을 설정하고, 가장 보상이 큰 점을 찾도록 작성하였다.

2.5 시뮬레이션 결과

2.5.1 실제 설계 데이터와의 유사성 검증(설계 유효성 검증)

이송율 계산을 위해 Unity Editor를 통한 시뮬레이션을 하면서 확인하여야 할 첫 사항은 실제 설계와 같은 설계를 시뮬레이션에 반영했을 때 이송율 관점에서 비슷한 결과를 보이는지 확인하는 것이다. 시뮬레이션이 가능하더라도 그 결과치가 실제 설계와 같지 않다면 그 결과는 무의미한 것일 것이다. 비교를 위해서 실제 발전소에서 설계한 데이터와 Unity Editor에서 시뮬레이션한 결과가 같은 이송율 수치를 보이는지 확인하였다.

Fig. 8의 그림은 발전소 내 실제 도면이며, Fig. 9는 Fig. 8의 도면에 대해서 동일 조건으로 시뮬레이션하여 나온 라우팅이다. 실제 설계자료에서 이송율은 49.88%였는데, 시뮬레이션 라우팅은 Fig. 11과 같이 51.6%의 예상 이송율을 보였다. 조금의 오차는 있지만, 2% 차이 이내에서 비슷한 결과를 보이는 유의한 결과를 보였다.

2.5.2 온화한 룸 설정 및 장애물을 반영한 학습 결과

Fig. 11과 같이 에이전트가 이동할 수 있는 경로와 방사선 감시기를 설치할 수 있는 위치상에 장애물을 추가하고, 방사선감시기가 설치될 수 있는 온화한 지역을 2구역을 설정하여 학습을 진행하였다.

Table 4 Mapping Code

Mapping Python Code
1. Main 실행 부분
2. def read_data()
3. def apply_agglomerative_clustering()
4. def plot_clusters_3d()

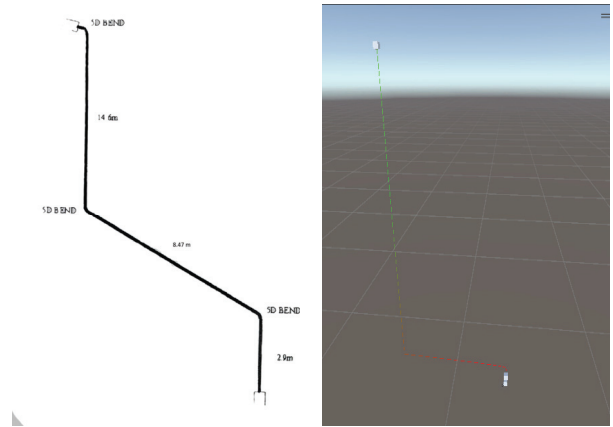


Fig. 8 실 설계도면

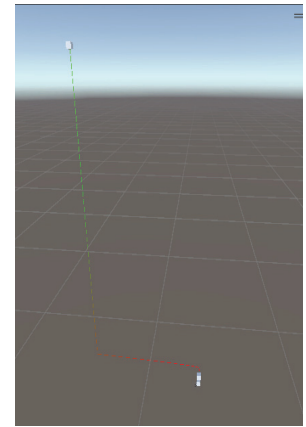


Fig. 9 시뮬레이션 라우팅

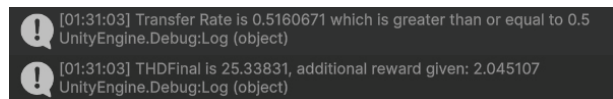


Fig. 10 시뮬레이션 라우팅에 대한 이송율

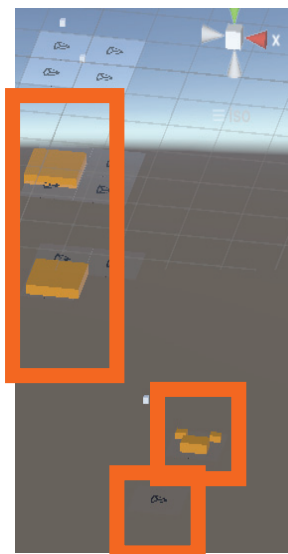


Fig. 11 구역, 장애물 추가

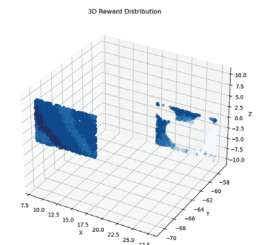


Fig. 12 3D 분포 그래프

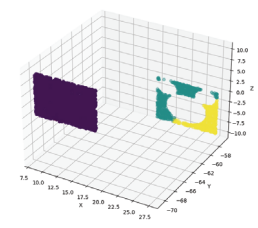


Fig. 13 군집 그래프

파일	편집	보기
rewardData.txt		
9.963511, -70.6, 1.08639, 114.437, 0.851		
15.95901, -70.6, 1.845337, 114.437, 0.59346		
22.9228, -56.91, -0.8389271, 114.437, 0.6		
11.66536, -70.6, 1.127827, 121.657, 0.666936		
15.71742, -70.6, 2.4757, 114.437, 0.59346		
8.979854, -70.6, 1.077511, 114.437, 0.81483		
9.020547, -70.6, 7.306318, 121.657, 0.666936		
27.15304, -56.91, -0.8389271, 114.437, 0.6		

Fig. 14 학습 결과 데이터

Max reward point in Cluster 0: [11.66536 -70.6 1.127827]
Total Penetration is: 0.666936
Transfer Rate Requirement is met

Fig. 15 결과 좌표 및 이송율 만족 여부

이와 관련하여 Fig. 14와 같이 결과 데이터를 얻을 수 있었고, Fig. 12과 같이 두 구역과 관련한 이송율 분포 그래프를 확인할 수 있다. Fig. 13에서는 이송율이 높은 지역을 확인할 수 있는 군집군을 얻을 수 있었고, 이를 이용해 Fig. 15에서는 보상 값을 가장 높게 달성한 군집 군에서 가장 높은 보상 값을 가진 방사선감시기의 좌표와 그 위치에서의 예상 이송율을 제시하였다. 이 좌표에서의 예상 이송율 수치인 약 66.7%의 예상 이송율을 보였고, 그 이송율은 이송율 요건을 만족함을 보였으며, 기존의 이송율보다 15.1% 더 개선된 이송율을 확보하였다.

3. 결 론

본 연구를 통해서 얻은 결론은 아래와 같다.

1. 시뮬레이션을 활용하기 위해 만든 설계 환경은 실제 원자력발전소에서 설계 시 얻은 이송율 값과 오차 범위에서 비슷한 수준의 결과치를 보인다.
2. 발전소 내 여러 기기 및 기물과 각 룸 내 환경조건을 고려하여 시행한 학습 과정을 통하여 적합한 방사선감시기 위치 및 시료 이송 배관 라우팅을 생성할 수 있다.
3. 2의 과정을 통하여 얻은 이송율 수치는 IEEE 603 등의 요건을 고려한 설계 환경 속에서 ANSI/HPS N13.1 (1999)의 시료 이송율 요건을 만족하는 방사선감시기 위치 설계 및 시료 이송 배관 라우팅 설계 구현이 가능함을 보여주었다.

이러한 설계 구현은 I. 서론 1.2항에서 언급한 설계적인 어려움 속에서 설계 업무의 효율성을 높이는 방법의 하나가 될 것으로 판단된다.

본 연구에서는 다양한 장애물을 모두 고려하지 않았기 때문에 한계가 있다. 원자력발전소 내에는 수많은 기기나 계기가 빼곡히 설치되어 있으며, 이와 함께 수많은 배관 및 전선

관 등이 설치되어 있다. 추후의 연구에서는 모든 장애물을 고려하여 연구가 진행된다면 더욱 정밀한 방사선감시기 위치 선정 및 시료 이송 배관 라우팅 설계가 이루어질 것이라 기대된다.

References

- (1) ANSI/HPS N13.1, "Sampling and Monitoring Releases of Airborne Radioactive Substances from the Stacks and Ducts of Nuclear Facilities" American National Standards Institute, Health Physics Society, 1999.
- (2) IEEE 603, "IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations", Institute of Electrical and Electronics Engineers, pp. 11, 2009
- (3) Regulatory Guide 1.97 Rev. 5, CRITERIA FOR ACCIDENT MONITORING INSTRUMENTATION FOR NUCLEAR POWER PLANTS, U.S. Nuclear Regulatory Commission (NRC), pp.8-10, 2019.
- (4) Lars Johansson, "Deep Reinforcement Learning for Multi-Agent Path Planning in 2D Cost Map Environments", Karlstad University, 2024.
- (5) Introducing-unity-machine-learning-agents, <https://unity.com/kr/blog/engine-platform/introducing-unity-machine-learning-agents>
- (6) Agglomerative Clustering, <https://builtin.com/machine-learning/agglomerative-clustering>
- (7) Birch Clustering, <https://medium.com/@noel.cs21/balanced-iterative-reducing-and-clustering-using-heirachies-birch-5680adffaa58>
- (8) Adapting total transport criteria in sampling piping design of gaseous effluent radiation monitor in nuclear power plants and its issues, NamYul Jeon · Kyuhyun Im · Hosik Ham, Journal of power engineering, 2020