

다양한 알고리즘을 활용한 지하 역사에서의 최적 대피 경로 분석

Optimal Evacuation Route Analysis in Underground Stations Using Various Algorithms

강치영 · 김현석 · 성경환 · 유민택*

Chiyeong Kang · Hyunseok Kim · Kyeonghwan Seong · Mintack Yoo

Abstract This study analyzed optimal evacuation route derivation for subway stations during disasters using Dijkstra, A*, and Greedy algorithms. A hypothetical three-level underground station was modeled, and evacuation routes were analyzed for all three algorithms. The effects of crowd density and disaster-induced movement speed reduction were incorporated. Results demonstrated that the A* algorithm exhibited the best performance, with a maximum evacuation time of 349 seconds. The Dijkstra algorithm recorded 365 seconds, while the Greedy algorithm required 1,545 seconds. Notably, the Dijkstra algorithm, which theoretically guarantees optimal solutions, yielded a longer maximum evacuation time than the A* algorithm. This counterintuitive outcome occurred because Dijkstra's path optimization performance generated identical routes for clusters located at similar nodes, paradoxically increasing crowd density. This finding suggests that excellent optimization performance can paradoxically intensify crowd density. In contrast, the A* algorithm's heuristic-based search induced route diversity, producing a natural distribution effect. These results indicate the necessity of designing algorithms that prioritize collective efficiency rather than individual optimality in evacuation route planning.

Keywords : Dijkstra algorithm, A* algorithm, Greedy algorithm, Subway station, Crowd density

초 록 본 연구는 재난 발생 시 지하 역사에 대한 최적 대피경로 도출을 Dijkstra, A*, Greedy 알고리즘을 활용하여 비교 분석하였다. 가상의 지하 3층 역사를 적용하였으며, 3가지 알고리즘에 대하여 대피경로를 분석하였다. 인원 혼잡도 및 재난에 따른 이동속도 저감 효과를 반영하였으며, 그 결과 A* 알고리즘이 최대 대피 시간 349초로 가장 우수한 성능을 보였다. Dijkstra 알고리즘은 365초, Greedy 알고리즘은 1,545초를 기록하였다. 이론적으로 최적해를 보장하는 Dijkstra 알고리즘이 A* 알고리즘보다 더 지연된 최대 대피시간을 도출하였다. 이는 Dijkstra 알고리즘의 뛰어난 최적해 선택 성능이 유사한 노드에 위치한 군집에 대해 동일한 경로를 도출하여 오히려 인원 혼잡도를 증가시켰기 때문이다. 이는 우수한 최적화 성능이 역설적으로 혼잡도를 증가시킬 수 있음을 시사한다. 반면 A* 알고리즘은 휴리스틱 기반 탐색으로 인한 경로 다양성이 자연스러운 분산 효과를 유발하였다. 이는 개별의 효율을 목적으로 하는 경로 탐색 알고리즘이 아닌 다수의 효율을 중시하는 알고리즘의 설계가 필요함을 알 수 있다.

주요어 : Dijkstra 알고리즘, A* 알고리즘, Greedy 알고리즘, 지하 역사, 인원 혼잡도

1. 서 론

현대 도시에서 지하철은 핵심적인 교통 인프라로서 시민들이 가장 많이 이용하는 교통수단 중 하나이다. 그러나 지하 역사는 폐쇄적이고 복잡한 구조적 특성상 화재, 테러, 자연재해 등의 재난 발생 시 이용객들의 대피에 어려움이 존재한다. 2022년 서울 지하철 비상 대피 시간 기준 및 초과 현황 자료에 따르면 서울 지하철 1~9호선에 해당하는 263개역 중 26.6%인 70개 역이 비상 대피 시간 기준을 초과하였다. 특히 5호선 영등포시장역의 경우에는 외부 출구까지 대피시간이 약 11분 51초로 비상 대피 시간 기준을 약 2배 이상 초과하는 것으로 나타났다[2]. 이는 신속한 대피를 위한 대피 경로 안내 시스템 도입의 필요성을 보여준다. 특히 2003년 대구 지하철 화재 참사나 2025년 서울 지하철 5호선 방화 사건과 같은 사례는 대피 경로 안내 시스템

*Corresponding author. E-mail: mintackkyoo@gachon.ac.kr.

© 2026 The Korean Society for Railway. All rights reserved.

<https://doi.org/10.7782/JKSR.2026.29.6.586>

템의 필요성을 다시 한번 보여준다. 그러나 국내에는 체계적인 대피 경로 안내 시스템이 미비한 실정이다. 국토교통부의 철도 정거장에 대한 설계 지침에 따르면, 재난 발생시 4분 이내 승강장을 벗어나고 6분 이내에 외부 탈출구로 완전히 대피를 마칠 것을 규정하고 있다[1].

국내외에서는 재난 발생 시에 대피 경로를 안내하기 위하여 다익스트라 알고리즘과 A* 알고리즘을 활용한 경로 도출 시스템을 활발히 사용하고 있다. Hwang 등[3]은 다익스트라 알고리즘을 활용하여 인원 혼잡도와 연령별 가중치를 반영한 대피 경로 도출하였고, 혼잡한 비상구에 대한 분산 유도 시스템을 제안하였다. Lee 등[4]은 화재 피해 정도를 반영한 경로를 도출하는 연구를 수행하여, 화재 확산 시뮬레이션인 fire dynamics simulator(FDS)의 결과 데이터와 연동되어 대피 경로를 탐색하는 새로운 F_A* 알고리즘을 제안하였다. Jo 등[5]은 A* 알고리즘을 활용하여 화재 상황에 의해 생기는 추가 장애물 위치, 화재 확산, 군중 밀집도 등의 데이터를 반영하여 경로를 도출하고 A* 알고리즘에 유효성을 검증하였다. Kim 등[6]은 추가 대피로 설치에 따른 대피 시간의 변화에 따른 연구를 진행하였다. Dijkstra 알고리즘을 활용한 피난 시뮬레이션 코드를 사용하여 인원 혼잡도를 반영한 최대 대피 시간을 비교 분석 하였다. Hui 등[7]은 복잡한 구조의 지하철역에서 승객 흐름률, 시설 파라미터, 군중 밀도를 고려한 대피 시간 모델을 제시하였다. 수평 보행거리, 흐름률, 계단 파라미터 등을 반영하여 게이트, 차량 출입문, 계단 등의 병목 지점을 식별하고 Pathfinder 소프트웨어를 활용한 시뮬레이션을 수행하였으며, 계산 결과와 시뮬레이션 간 5.4%의 오차로 높은 일치도를 보였다. Wang 등[8]은 지하철 화재 시 안전한 대피 전략과 대피 과정을 연구하였다. 전력 시스템 정상 시 역사 대피 모드, 전력 실패 시 터널 대피 모드를 제안하였으며, Pathfinder와 PyroSim을 이용한 시뮬레이션 결과 역사 대피 모드의 대피 시간이 526초로 터널 대피 모드 2193초의 약 1/4 수준임을 확인하였다. 총 대피 시간은 승객 수와 노약자 비율에 따라 증가하는 것으로 나타났다. Yoo 등[9]은 지하 구조물의 지진 발생 시 Dijkstra 알고리즘을 활용하여 대피자 혼잡도를 고려한 최적 대피 경로 모델을 개발하였다. 지반 조건과 지진 강도를 변화시킨 4가지 경우에 대해 분석하였으며, 혼잡도 계수를 적용한 결과 일부 구간의 대피 경로가 변경되었고 최종 대피 시간이 최대 220% 증가하였다. 혼잡도를 고려한 대체 경로 사용 시 대피 시간이 최대 45% 감소하여 실시간 혼잡도 반영의 중요성을 제시하였다. Kong 등[10]은 서울 구룡역을 대상으로 그래프 이론 기반의 대피 경로 최적화 전략을 제안하였다. 개별 이동 저항을 최소화하는 최단 경로 알고리즘과 혼잡도 감소를 위한 Greedy disjoint path 알고리즘을 비교하였으며, AnyLogic 시뮬레이션 결과 탐욕적 경로 분배 전략이 출구 사용을 균등하게 분산시켜 총 대피 시간을 5.66분에서 5.18분으로 약 10% 단축시키는 것으로 나타났다. 그러나 기존 연구들 중 다양한 알고리즘을 비교하여 대피 안내 시스템에 대한 적용성을 확인하는 연구는 미비한 실정이다.

따라서 본 연구는 다양한 최적 경로 탐색 알고리즘을 적용하여 재난 발생 시 재난 및 인원 혼잡도를 반영한 최적 대피 경로 모델에 대해 적합한 알고리즘의 기준을 제시하였다. 동일 조건에서 서로 다른 알고리즘들을 적용하고 연산 시간과 최대 대피 시간, 탐색 노드 수와 같은 알고리즘의 성능 지표를 분석하였으며, 대피 안내에 대한 알고리즘들의 적합성을 비교 검증하였다. 또한 각 알고리즘에 의해 선택된 경로 분포를 분석하여 병목 현상과 경로 집중의 메커니즘을 규명하고, 대피 시스템 설계의 기초 자료를 제공하고자 한다. 이를 바탕으로, 개별 사용자(군집)의 합리적 경로 선택을 기반으로 한 user-optimal 대피 안내가 다중 군집 환경에서는 오히려 경로 집중과 혼잡을 유발하여 시스템 전체의 대피 성능을 저하시킬 수 있음을 수치 시뮬레이션을 통해 관찰하였다. 이를 통해 본 연구는 대피 경로 최적화 문제를 알고리즘 성능 비교의 관점이 아닌, user-optimal 의사결정과 system-level 응답 간의 구조적 불일치 문제로 해석할 수 있음을 보였다.

2. 대피 경로 도출 알고리즘 종류

본 연구에서는 지하 역사 대피 경로 도출에 적합한 알고리즘을 선정하기 위해 Dijkstra, A*, Greedy 알고리즘을 비교 분석하였다. 세 알고리즘은 최적 경로 탐색 분야에서 널리 활용되며 각기 다른 특성을 갖는다. Dijkstra 알고리즘은 모든 노드를 전부 탐색하여 최적해를 보장하나, 연산 비용이 높다는 단점이 있다. A* 알고리즘은 휴리스틱 함수를 활용하여 목표 방향으로 탐색을 유도함으로써 Dijkstra 대비 효율적인 연산이 가능하며 최적해 역시 보장한다. Greedy 알고리즘은 가장 빠른 연산 속도를 갖지만 최적해를 보장하지 못한다. 이러한 특성을 고려하여 본 연구에서는 최적성과 효율성의 균형, 그리고 다중 군집 환경에서의 경로 분산 효과를 분석하기 위해 세 알고리즘을 선정하였다.

2.1 Dijkstra 알고리즘

Dijkstra 알고리즘은 1956년 Edsger W. Dijkstra가 개발한 경로 탐색 알고리즘으로, 노드와 엣지(edge)로 구성된 그래프에서 시작 노드로부터 모든 다른 노드까지의 최단 경로를 찾는 알고리즘이다. Dijkstra 알고리즘의 실행 순서는 다음과 같다.

시작노드의 거리를 0으로, 나머지 모든 노드의 거리의 한계 값을 무한대로 설정한다. 이후 방문하지 않은 노드 중 최단 거리 노드를 선택하여 방문 처리하고 해당 노드에 인접한 모든 노드를 탐색하며, 거리가 기존 경로보다 짧은 경로를 발견했을 때 경로를 갱신한다. 이 과정을 모든 노드를 방문할 때까지 반복한다.

모든 알고리즘은 특정한 목적에 대한 값을 최적화하기 위한 목적 함수를 갖는다. 본 연구에서 Dijkstra 알고리즘은 시작 노드에서 목표 노드까지의 총 비용을 최소화하는 목적 함수를 갖는다. 목적 함수는 $d(s,v) = \sum w(u,v)$ 로 정의되며, 여기서 $d(s,v)$ 는 시작 노드 s 에서 노드 v 까지의 최단 경로 비용, $w(u,v)$ 는 각 엣지의 가중치를 나타낸다. 본 연구에서 엣지 가중치는 기본 거리에 재난 가중치와 혼잡도 가중치를 곱하여 산정된다.

Dijkstra 알고리즘은 모든 노드를 탐색하는 맹목적 탐색 방식을 사용하기 때문에 이를 통해 도출된 경로는 최적해를 보장하는 경로라고 볼 수 있다[11]. 다익스트라 알고리즘의 실행 방식은 Fig. 1과 같다.

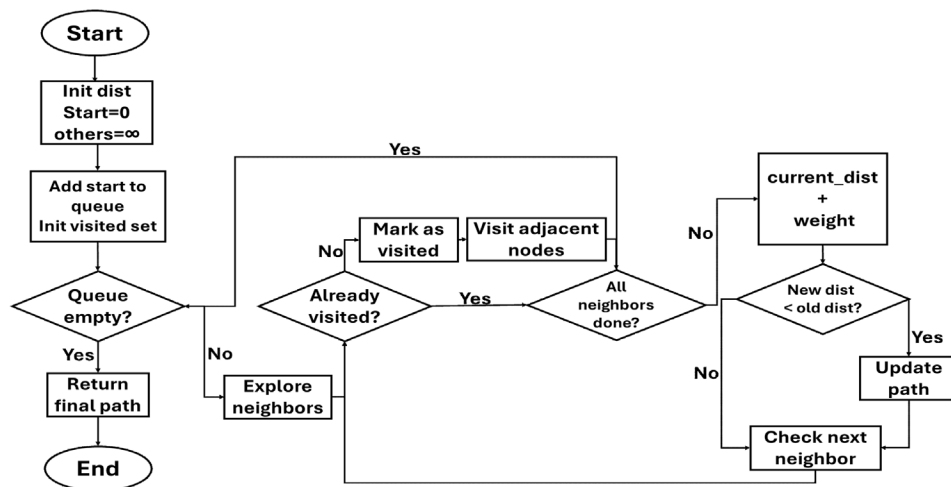


Fig. 1. Dijkstra's algorithm flow chart.

2.2 A* 알고리즘

A*알고리즘은 1968년 Peter Hart, Nils Nilsson, Bertram Raphael에 의해 개발된 휴리스틱 기반의 최적 경로 탐색 알고리즘이다. Dijkstra 알고리즘에서 확장된 알고리즘으로, 목표 지점까지의 예상 비용을 추정하는 휴리스틱을 사용하여 탐색 효율성을 증가시킨다.

A*알고리즘은 출발 노드부터 현재 노드까지의 실제 비용과 현재 노드에서 목표 노드까지의 추정 비용을 합산한 휴리스틱 함수를 기반으로 경로를 평가한다. 실제 비용에는 인원 혼잡도 등의 가중치가 반영되며, 추정 비용은 현재 노드에서 출구 노드까지의 직선거리를 사용한다. 이를 통해 출구 노드까지 최단 거리로 추정되는 경로를 우선으로 탐색하게 된다. 모든 노드를 탐색하는 것이 아닌 휴리스틱에 근거한 일부 노드만을 탐색하는 특성상 탐색 노드의 수를 감소시켜 다른 알고리즘 대비 연산 시간이 적게 소요되는 이점을 가진다.

A* 알고리즘은 Dijkstra와 동일하게 경로 비용 최소화를 목적으로 하는 최적 경로 탐색 알고리즘으로 동일한 목적 함수를 갖는다. A*알고리즘의 휴리스틱 함수는 식 (1)과 같이 정의된다[12].

$$f(n) = g(n) + h(n) \quad (1)$$

여기서 $f(n)$ 은 경로의 총 예상 비용, $g(n)$ 은 출발 노드부터 탐색 중인 노드 n 까지 경로에 인원 혼잡도 등의 가중치가 반영된 누적 비용, $h(n)$ 은 탐색 중인 노드 n 에서 출구 노드까지의 직선거리로 계산된 추정 비용이다. $h(n)$ 은 맨해튼 거리나 체비쇼프 거리 등 다양한 방안이 존재하지만, 본 연구에서는 연산 비용의 최소화과 휴리스틱의 본래 목적인 연산 속도 향상을 더 중요하 다 판단했기에 그에 부합한 직선거리를 채택하였다.

2.3 Greedy 알고리즘

Greedy 알고리즘은 각 단계에서 최적의 선택을 통해 경로를 탐색하는 알고리즘이다. Greedy 알고리즘 역시 경로 비용 최소화를 목적으로 한다. 또한, A* 알고리즘과 유사하게 휴리스틱을 사용하여 경로를 찾는 알고리즘이지만, 시작점으로부터 현재 노드까지의 실제 비용($g(n)$)을 고려하지 않는다는 점에서 차이가 있다. Greedy 알고리즘의 휴리스틱 함수는 오직 현재 노드에서 목표 지점까지의 추정 비용($h(n)$)만을 사용한다. 탐색의 모든 단계마다 누적 비용을 고려하지 않고, 도착 노드와 가장 가깝다고 예상되는 노드를 방문한다. 이러한 탐욕적 선택 방식을 사용해 목표 지점까지 매우 빠르게 도달할 수 있다. 하지만 다른 알고리즘 대비 변수에 대한 고려가 어렵다는 단점이 존재한다. 그에 대한 함수는 식 2와 같이 정의된다[11].

$$f(n) = h(n) \quad (2)$$

3. 대피 시뮬레이션 모델 개발 및 적용

3.1 대피 시뮬레이션 모델 개발

본 연구에서는 지하 역사의 대피 상황을 모사하기 위해 시뮬레이션 모델을 개발하였다. 대피 시뮬레이션 모델은 지하 역사 공간을 노드와 엣지의 그래프로 변환하고, 대피 인원을 군집 단위로 모델링하며, 각 군집이 재난과 혼잡도를 고려하여 출구까지의 최적 경로를 탐색하고 이동하는 과정을 시뮬레이션한다. 이러한 시뮬레이션 모델은 Python 언어를 기반으로 작성하였으며, 모델의 연산 효율을 위해 개별 인원이 아닌 200~500명 규모의 군집 단위로 구현하였다. 모델의 실행 방식은 초기 설정, 대피 시뮬레이션, 결과 출력의 순서로 진행된다.

대피 시뮬레이션 모델은 초기화를 진행한 다음 시뮬레이션에 필요한 초기 설정을 시작한다. 초기 설정은 다음과 같다.

먼저, 지하 역사 구조를 구현하기 위한 노드 좌표와 연결 관계, 엣지의 길이를 설정한다. 설정된 정보를 기반으로 노드와 엣지 그래프로 구현된 노드에는 인원 정보를 가진 군집이 배치되고 지정된 노드에 대해 재난을 부여한다. 재난 노드에 연결된 엣지에는 속도 저감 계수를 부여한다. 또한 정의된 이동속도, 대피 인원수에 따른 혼잡도 가중치 조건을 설정한다.

이러한 초기 설정이 완료된 후 대피 시뮬레이션을 시작한다. 대피 경로를 탐색하는 Dijkstra, A*, Greedy 알고리즘들로 시뮬레이션이 진행되며, 각 군집의 경로를 독립적으로 결정한다. 시뮬레이션 상의 1초마다 메인 루프가 작동하며 군집의 위치를 갱신해 혼잡도에 맞는 속도 저감 계수를 부여한다. 또한 혼잡도를 반영한 경로의 재탐색이 이루어지며, 더 효율적인 경로가 도출되면 경로를 갱신한다. 모든 군집이 지하 역사를 탈출하게 되면 시뮬레이션을 종료한다. 시뮬레이션 종료 후에는 각 알고리즘에서 마지막으로 대피한 군집의 대피 시간인 최대 대피 시간과 각 알고리즘의 연산시간, 탐색 노드 수 등의 결과를 출력한다.

대피 시뮬레이션 모델의 실행 방식은 Fig. 2와 같다.

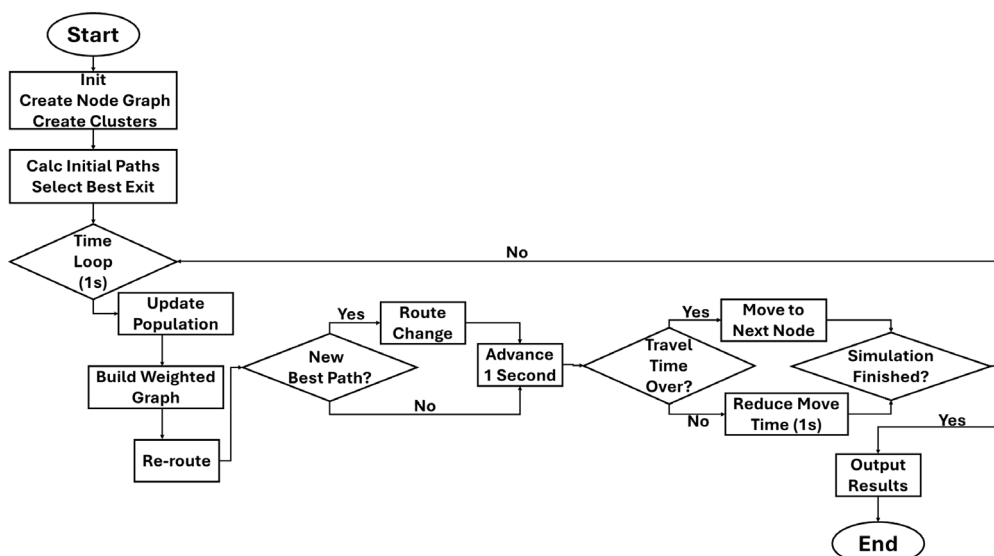


Fig. 2. Simulation flow chart.

3.2 적용 공간 모델 및 적용 모델의 입력 변수

본 연구에서는 지하 역사 구조물에서 재난이 발생한 상황을 가정하여 인원 혼잡도를 고려한 대피 시뮬레이션을 실행하였다.

본 연구에서는 공간 모델 설정을 위해 국내 주요 지하철 역사들의 구조적 특성을 분석하였다. 대규모 환승역은 지하 4층 이상의 깊이와 10개 이상의 출구를 갖추며 복잡한 환승 통로를 포함하는 반면, 중소규모 역사는 지하 2~3층 구조에 4~6개의 출구와 제한된 수의 계단을 갖는 구조이다. 본 연구에서는 이러한 분석을 바탕으로 중소규모 역사와 유사한 규모인 지하 3층의 가상 역사를 구축하였다. 중소규모 역사는 국내 지하철 전체 역사 중 상당수를 차지하며, 승강장에서 지상층까지 수직 대피가 필요하고 제한된 계단과 출구로 인해 병목 현상이 발생할 수 있는 전형적인 구조적 특성을 갖는다. 가상 역사는 특정 역사의 복잡한 세부 구조를 단순화하여 알고리즘 비교에 초점을 맞추었으며, 3층에서 2층으로 향하는 계단 6곳, 2층에서 1층으로 향하는 계단 4곳, 출구 4곳으로 구성하였다. 공간 모델은 좌표상의 노드와 엣지 그래프로 구현하였으며 노드 간의 수평 거리는 5m, 수직 거리는 10m로 설정하였다. 역사 구조를 반영한 그래프는 Fig. 3과 같다.

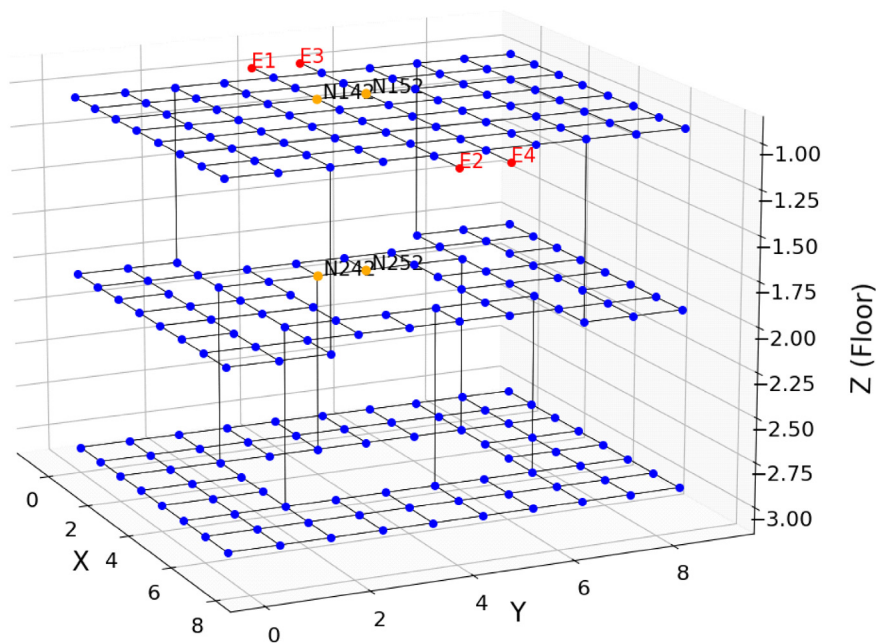


Fig. 3. Node layout.

3.3 대피 속도 및 속도 저감 계수 설정

지하 역사 내에서의 대피 경로 시뮬레이션에서 중요한 입력 변수 중 하나는 대피 인원의 이동속도이다. 대피 시간 산출을 위해서는 적절한 이동속도를 적용하고, 해당 속도에 부합하는 시간을 도출해야 합리적인 최종 대피 시간을 도출할 수 있다. 이에 지하 역사에서의 대피 속도는 국토교통부의 도시철도 정거장 및 환승 편의 시설 설계 지침[1]에서 제시된 이동속도를 사용하였다. 수평 이동의 경우 60m/분의 속도를 적용하였으며, 수직 이동에 대해서는 15m/분에 속도를 적용하였다. 본 연구에서 사용된 대피 속도 값은 Table 1과 같다.

Table 1. Evacuation speeds by evacuation factor.

EVACUATION FACTOR	EVACUATION SPEED
Horizontal transportation(platform, concourse and corridor)	60m/min
Vertical transportation(stairs, stationary escalator)	15m/min

본 연구에서는 인원 혼잡도와 재난에 따른 속도 저감 계수를 고려하여 시뮬레이션을 진행하였다. 재난 노드는 노드에 연결된 모든 엣지에 10배의 속도 저감 계수를 부여한다. 재난 노드에 대한 속도 저감 수치는 인원의 대피 속도에 영향을 줄 수 있으며, 완전 통행 불가가 아닌 통행 가능성을 고려한 임의의 값으로 설정하였다. 인원 혼잡도에 대한 속도 저감 계수는 Zhou 등 [13]에서 제시된 밀집도에 따른 대피 속도 그래프를 이용하였다. Zhou 등[13]은 실제 지진 대피 영상을 기반으로 보행자의 이동속도와 밀집도 간의 상관관계를 정량화하였으며, 밀집도가 증가할수록 대피 속도가 급격히 감소하는 특성을 혼잡도를 반영한 속도 감소 곡선 모델로 표현하였다. 본 연구에서 사용한 밀집도에 따른 대피 속도 그래프는 Fig. 4와 같다.

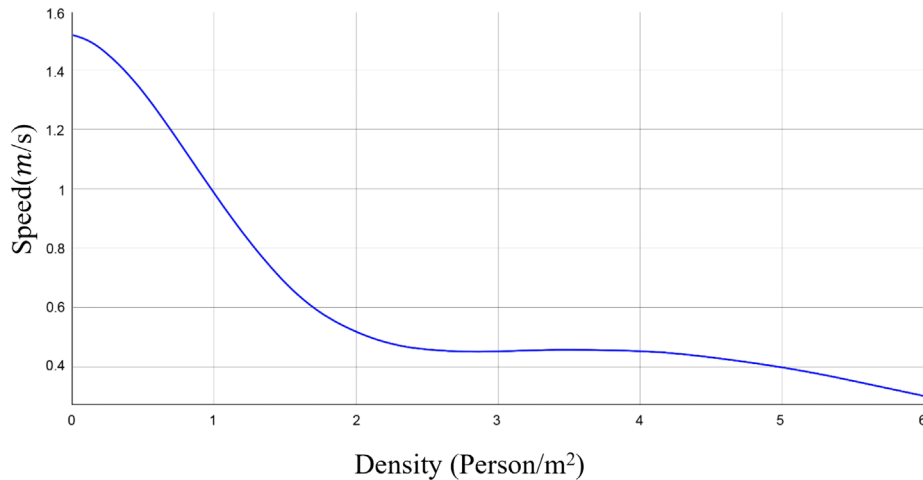


Fig. 4. Fitting curve of the evacuation speed under different density in normal state (Zhou et al., 2020).

인원 밀집도를 산정하기 위해서는 통로의 폭과 폭당 대피 수용량이 요구된다. 통로의 폭과 대피 수용량은 국토교통부의 도시철도 정거장 및 환승 편의 시설 설계 지침[1]에서 제시된 값을 사용하였다. 통로의 폭과 대피 수용량을 통해 인원 혼잡도를 발생시키지 않는 최대 인원수를 산정 가능하다. 지하 역사 통로 폭에 대해서는 국토교통부 설계 지침 최소 폭인 수평 8m, 수직 6m로 설정하였다. Table 3에 인원 혼잡도에 따른 속도 저감 계수를 3단계로 구성하였다. 대피 인원수에 대한 임계값은 통로의 폭(m)당 수용량의 곱으로 산정하였다. 이에 수평 이동 요소는 최소폭 8m와 대피 수용량 80명/m를 곱하여 640명, 수직 이동 요소는 폭 6m와 대피 수용량 60명/m를 곱하여 360명을 임계값으로 설정하였다. 단위 폭당 대피 수용량과 위 설정을 포함한 대피 인원수에 따른 대피 속도 저감 계수는 Table 2, Table 3과 같다.

Table 2. Minimum evacuation capacity per unit width (m).

Evacuation factor	Evacuation capacity
Horizontal transportation(platform, concourse and corridor)	80 persons/m·min
Vertical transportation(stairs, stationary escalator)	60 persons/m·min

Table 3. Walking speed reduction table based on crowd density.

Evacuation factor	Population(person)	Multiplier
Horizontal transportation(platform, concourse and corridor)	640	2
Horizontal transportation(platform, concourse and corridor)	1280	2.5
Horizontal transportation(platform, concourse and corridor)	1920	3
Vertical transportation(stairs, stationary escalator)	360	2
Vertical transportation(stairs, stationary escalator)	720	2.5
Vertical transportation(stairs, stationary escalator)	1080	3

위에 설명한 모든 입력 변수는 본 연구에 적용되는 파라미터로, 설계 지침 및 선행 연구에 기반하여 설정되었으며 세 알고리즘에 동일하게 적용되었다. 알고리즘별 임의의 튜닝 파라미터는 사용하지 않았다. 이를 통해 알고리즘 간 공정한 성능 비교를 수행하였다.

3.4 인원 배치 설정

본 연구에서 인원은 지하 3층에 200~500명 규모의 군집 총 9100명을 29개의 노드에 배치하였다. 인원 배치는 각 알고리즘에 대해서 시뮬레이션 시 인원 혼잡도 발생 빈도를 높이기 위해 인원 혼잡도의 발생에 적합한 인원수를 설정하여 임의로 배치하였다. 혼잡도 발생만을 높인 배치로, 특정 알고리즘의 성능에 대해 영향을 주지 않는다. 인원 배치에 대한 배치는 Fig. 5와 같다.

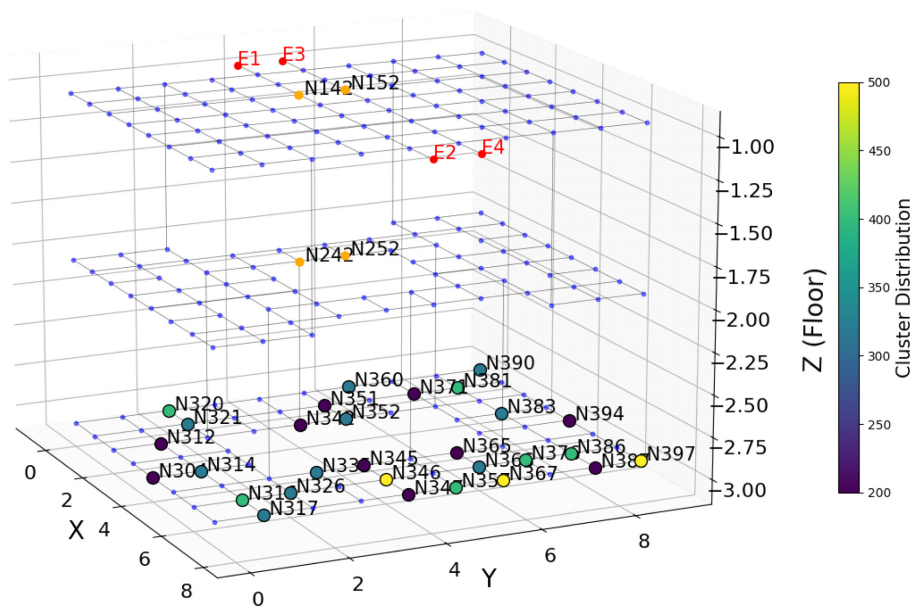


Fig. 5. Cluster layout.

4. 대피 경로 시뮬레이션 결과

4.1 혼잡도 파라미터 민감도 분석

본 연구에서 설정한 혼잡도 가중치가 시뮬레이션 결과에 미치는 영향을 확인하기 위해 혼잡도 수준별 속도 저감 계수를 변화시켜 민감도 분석을 수행하였다. 이를 통해 알고리즘 간 비교의 공정성과 결과의 일반화 가능성을 검증하고자 하였다. Table 4는 혼잡도 가중치 변화에 따른 Dijkstra 알고리즘의 최대 대피 시간을 나타낸 것이다.

Table 4. Congestion weight parameters for sensitivity analysis.

Case	Level 1	Level 2	Level 3
Baseline	2.0	2.5	3.0
Case 1	3.0	4.0	5.0
Case 2	3.0	5.0	7.0
Case 3	4.0	7.0	10

Table 5는 혼잡도 가중치 변화에 따른 각 알고리즘의 최대 대피 시간을 나타낸 것이다.

Table 5. Maximum evacuation time by congestion weight variation.

Case	Dijkstra (sec)	A* (sec)	Greedy (sec)
Baseline	365	349	1545
Case 1	576	554	2562
Case 2	752	738	3517
Case 3	1062	1042	5036

분석 결과, 혼잡도 가중치가 증가할수록 모든 알고리즘의 최대 대피 시간이 증가하였으나, 모든 케이스에서 29개 군집의 경로 선택은 동일하게 유지되었다. 이는 각 알고리즘의 경로 선택 특성이 혼잡도 파라미터 변화에도 robust하게 유지됨을 의미한다. 특히 Dijkstra 알고리즘의 경로 집중 현상, A* 알고리즘의 휴리스틱 기반 분산 효과, Greedy 알고리즘의 재난 노드 통과 등 각 알고리즘의 본질적 특성은 모든 케이스에서 일관되게 나타났다. 또한 모든 케이스에서 알고리즘 간 성능 순위가 $Dijkstra \approx A^* < Greedy$ 로 동일하게 유지되었다. 이를 통해 본 연구에서 설정한 파라미터가 특정 알고리즘에 편향되지 않았으며, 알고리즘 비교 결과가 파라미터 변화에 robust함을 확인하였다.

4.2 대피 경로 시뮬레이션 해석 결과

본 연구에서는 지하 역사의 대피 상황을 가정하여 Dijkstra 알고리즘, A* 알고리즘, 그리고 Greedy 알고리즘의 성능을 비교 분석하였다. 각 모델의 대피 경로 도출 시뮬레이션 결과는 Table 6에 요약하였다. 시뮬레이션 수행 결과 최대 대피 시간은 Dijkstra 알고리즘 365초, A*알고리즘 349초 Greedy 알고리즘 1,545초로 A*알고리즘이 가장 우수한 성능을 보였다. 연산시간과 탐색 노드 수에서는 Greedy 알고리즘이 가장 효율적인 결과를 도출하였으나, 최대 대피 시간은 A* 알고리즘의 약 4.5배인 1545초의 대피 시간을 도출하였다. 해석 결과를 살펴보면 Dijkstra 알고리즘은 연산시간이 비교적 오래 걸리고 탐색 노드 수가 많아 효율이 낮은 것으로 보였고, A* 알고리즘은 가장 빠른 최대 대피 시간과 준수한 연산 속도, 노드 탐색 수를 결과로 도출하여, 가장 효율적인 성능을 보였다. 또한 Greedy 알고리즘은 최소의 노드 탐색 수와 최소 연산시간을 기록하였으나, 가장 늦은 최대 대피 시간을 도출하여, 안내 시스템의 적용에는 한계를 보였다.

Table 6. Analysis results by algorithm.

Algorithm	Maximum evacuation time(sec)	Execution time(sec)	Number of explored nodes
Dijkstra	365	0.1327	155,923
A*	349	0.0758	43,342
Greedy	1,545	0.0250	18,546

4.3 대피 경로 시뮬레이션 결과 분석

Dijkstra 알고리즘은 이론적으로 모든 노드를 탐색해 최적해 도출이 보장되는 알고리즘임에도 불구하고 A* 알고리즘보다 16초 더 긴 대피 시간을 도출했다. 이러한 역설적인 결과가 도출된 이유는 Dijkstra 알고리즘의 이점인 경로 최적화 성능이 다중 군집 대피 상황에서는 오히려 약점으로 작용했기 때문이다. Dijkstra 알고리즘은 각 군집에 대해 출구까지의 최적의 경로를 제공하는데, 여러 군집이 동시에 대피를 시작할 때 이들이 모두 동일하거나 유사한 최적 경로로 집중되는 현상이 발생하였다. 그 결과 특정 복도나 계단에 다수의 군집이 집중되면서 병목 현상과 혼잡도 증가를 발생시켰고, 이는 전체 대피 시간을 지연시키는 주요 원인이 되었다.

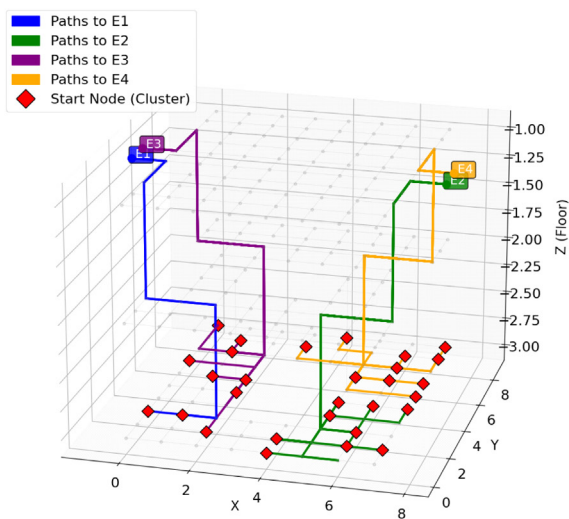
본 연구에서 각 군집의 경로 선택 과정은 교통 시스템 이론에서 정의하는 user-optimal 의사결정 구조에 해당한다. 즉, 각 군집은 자신에게 주어진 비용 함수(거리, 재난 가중치, 혼잡도 가중치)를 기준으로 출구까지의 이동 비용을 최소화하는 경로를

독립적으로 선택한다. 이는 Wardrop(1952)[14]가 정의한 user-optimal 상태와 개념적으로 동일하며, 전체 시스템의 대피 시간 최소화를 직접적으로 고려하는 system-optimal 문제 설정과는 구분된다. 본 연구에서는 system-optimal 해를 직접 산정하거나 전체 대피 시간을 목적 함수로 하는 중앙 최적화 문제를 풀지 않았으며, 대신 user-optimal 기반 경로 안내가 다중 군집 환경에서 어떠한 시스템 수준의 응답을 유발하는지를 시뮬레이션을 통해 분석하는 데 초점을 두었다.

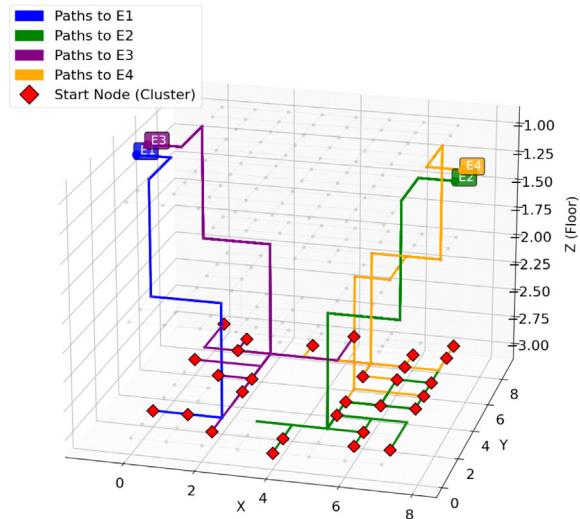
User-optimal 경로 기반 안내에 따른 집중 현상에 대해 시각적으로 확인하기 위해 알고리즘별 대피 경로에 대한 분포를 분석하였으며, 그 결과를 Fig. 6에 제시하였다. Fig. 6에서 확인할 수 있듯이, Dijkstra 알고리즘은 다른 알고리즘과 비교했을 때 계단에서 동일한 거리에 위치한 군집들이 특정 대피 경로로 집중되는 경향을 보였다.

반면 A* 알고리즘과 Greedy 알고리즘은 상대적으로 경로가 분산되어 병목 현상을 완화하는 것으로 관찰되었다. Greedy 알고리즘의 경우, 탐색 중인 노드에서 출구까지의 직선거리만을 고려한 휴리스틱 사용으로 인해 재난 상황을 고려한 경로에 왜곡을 일으켜 비효율적인 경로를 탐색하는 것으로 나타났다.

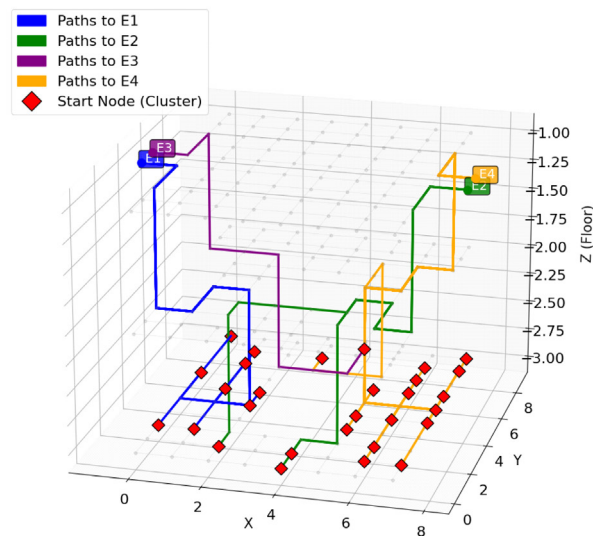
A* 알고리즘은 휴리스틱 함수를 활용하여 가장 효율적인 대피 시간을 달성하였다. 또한 가장 최적의 경로 분포를 도출하였



(a) Dijkstra's Algorithm



(b) A* Algorithm



(c) Greedy Algorithm

Fig. 6. Path distribution by algorithm.

다. 이를 통해 대피 시뮬레이션에서 Dijkstra 알고리즘 대비 저조한 최적화 성능이 다중 군집 환경에서는 오히려 장점으로 작용함을 확인하였다. 휴리스틱을 기반으로 한 경로는 각 군집이 탐색하는 경로에 미세한 차이를 발생시켰고, 이는 자연스럽게 경로 분산 효과로 이어졌다. 이는 Dijkstra 알고리즘처럼 모든 군집이 최적 경로로 몰리는 대신, 다양한 경로들로 군집을 분배하였고, 이러한 경로 다양성이 혼잡도를 낮추고 전체 대피 시간을 단축시켰다. 또한 Dijkstra 알고리즘에 비해 빠른 속도로 연산해 안내 시스템으로서 적합함을 보였다. 이는 정확한 최적화보다 혼잡 지역에 대한 회피 성능이 더 중요할 수 있다는 점을 시사한다.

또한, Greedy 알고리즘은 최소의 노드 탐색과 최소의 연산시간을 기록하였으며, 연산시간 단축의 측면에서 탁월한 성능을 보였다. 그러나 노드 탐색의 매 단계에서 출구와의 직선거리만을 고려한 휴리스틱 기반의 선택으로 지하 역사에서의 재난을 반영하지 못하는 결과를 보였다. 이는 재난으로 인해 발생하는 가중치를 고려한 경로가 교란되어 일어난 결과이다. 그로 인해 Greedy 알고리즘은 재난 노드를 통해 지나가는 경로를 도출하였고, A* 알고리즘 대비 4.5배의 최대 대피 시간이 도출되었다. 이는 재난을 고려하는 등의 여러 가지 변수가 고려되는 상황에서는 출구와의 길이만으로 효율적인 대피가 불가능함을 보여준다.

앞서 분석한 결과를 보았을 때, 이론적으로 최적해를 보장하는 Dijkstra 알고리즘이 A* 알고리즘보다 더 긴 대피 시간을 기록했다는 부분에서 알고리즘의 최적화 능력의 우수함이 실제 대피 효율성의 척도가 되지 않는다는 것을 알 수 있다.

앞서 설명한 Dijkstra 알고리즘은 개별 경로 탐색에 대해서는 최적의 경로를 제공하지만, 군집 내 모든 요소에 최적의 경로를 적용하는 경우 피난 인원이 특정 경로에 집중되는 현상을 발생시켰다. 이러한 현상을 본 논문에서는 최적화의 역설이라고 정의하고자 한다. 여기서 ‘최적화의 역설’은 교통 시스템 이론에서의 Braess 역설 [15]과 동일한 수리적 정리를 의미하는 것이 아니라, user-optimal 기반의 경로 선택이 다중 군집 환경에서 시스템 수준의 비효율로 귀결되는 현상을 설명하기 위한 개념적 정의이며 Braess 역설 [15]과 구조적으로 유사하다. 사용자들의 개별적인 합리적 선택이 전체 시스템의 비효율을 초래하고, 경로 집중으로 인한 혼잡도 증가가 핵심적인 메커니즘이며, 이러한 점에서 두 역설은 구조적으로 유사한 메커니즘을 공유한다. Braess 역설 [15]에서는 각 운전자가 자신에게 최단 경로를 선택하고, 본 연구에서는 각 군집이 Dijkstra 알고리즘으로 산정된 최적 경로를 선택한다. 두 경우 모두 이러한 개별 최적 선택이 특정 경로로의 집중을 야기하고, 이로 인한 혼잡도 증가가 전체 효율성을 저하시킨다. 다만 본 연구의 상황은 Braess 역설 [15] 과 달리 새로운 경로의 추가가 아닌 알고리즘의 최적화 성능 자체가 원인이며, 운전자의 자발적 선택이 아닌 안내 시스템에 의한 경로 할당이라는 점에서 차이가 있다. 이러한 구조적 유사성은 대피 시스템 설계에 중요한 부분이다. 본 연구에서 A* 알고리즘의 휴리스틱 함수가 각 군집에 미세하게 다른 경로를 할당하여 자연스러운 분산 효과를 유도함으로써 이 병목현상을 일부분 해결할 수 있음을 보였다.

반면 A* 알고리즘은 휴리스틱으로 인해 각 군집 간의 경로 도출 결과에 차이가 발생했고, 이로 인해 개별 최적화는 이루어지지 않았지만 자연스러운 분산 효과가 발생하여 시스템 전체적으로 보았을 때 보다 효율적인 결과를 도출하였다. 이를 바탕으로 개별 최적해의 합이 항상 전체 최적해로 연결되지 않는다는 점을 확인할 수 있다.

결과적으로 지하 역사와 같이 제한된 공간과 다수의 대피 인원, 재난과 같은 변수가 존재하는 환경에서는 뛰어난 최적화 성능보다는 적절한 수준의 경로를 다양하게 확보해 혼잡도를 분산시키는 방향이 더 효과적이었다. 이를 통해 개별의 효율을 목적으로 하는 알고리즘이 아닌 다수의 효율을 중시하는 알고리즘 설계가 필요함을 알 수 있다.

본 연구는 몇 가지 한계점을 가지고 있다. 우선 지하 역사를 가상으로 구현하면서 실제 역사의 복잡한 공간 구조를 단순화하였다. 실제 지하철역에 존재하는 다양한 장애물, 계단의 폭 차이 등의 요소들이 충분히 반영되지 못하였다. 또한 재난 상황을 특정 노드에 가중치를 부여하는 방식으로 단순하게 모델링하여 실제 화재 시 연기 확산이나 시간에 따른 위험 구역의 동적 변화 등을 충분히 반영하지 못하였다. 마지막으로 재난 상황과 인원 혼잡도에 따른 속도 저감 계수가 지하 역사에서의 실제 대피 실험이나 현장 데이터를 통해 검증되지 않았다.

향후 연구에서는 실제 지하 역사의 복잡한 구조 반영, 연기 확산 등 동적이고 현실적인 재난 모델 구현, 그리고 본 연구에서 개별 최적화로 인해 발생한 경로 집중 현상을 해결하기 위해 중앙 통제 방식을 활용한 대피 시뮬레이션 모델 개발을 진행할 계획이다.

5. 결 론

본 연구에서는 지하 역사의 재난 상황에서 효율적인 대피 경로 도출을 위해 Dijkstra, A*, Greedy 알고리즘을 비교 분석하였다. 대피 시뮬레이션 모델은 3개 층으로 구성된 가상 지하 역사에서 재난과 인원 혼잡도에 따른 속도 저감 계수를 적용하여

각 알고리즘의 성능을 평가하였다.

그 결과, A* 알고리즘이 최대 대피 시간에 대해서는 349초로 가장 우수한 성능을 보였다. 또한 알고리즘의 연산시간과 탐색 노드 수에 대해서는 0.025초, 18,546개로 Greedy 알고리즘이 가장 효율적인 결과를 보였다. 그러나 Greedy 알고리즘은 A* 알고리즘 대비 약 4.5배의 최대 대피 시간이 경과한 것으로 나타났으며, 이러한 결과는 최적 대피 경로 안내 시스템에 적합하지 않았다.

본 연구에서 Dijkstra 알고리즘은 최적화 성능이 뛰어나지만, 최대 대피 시간은 365초로 A* 알고리즘보다 느린 결과를 도출하였다. 이는 Dijkstra 알고리즘의 뛰어난 경로 최적화 성능이 다중 군집 탐색 상황에서 대피자들의 경로를 집중시켜 혼잡도를 증가시켰기 때문이다. 반면 A* 알고리즘의 휴리스틱 기반 탐색으로 유도된 군집들의 경로 다양성은 자연스러운 경로 분산으로 이어져 전체 대피 시간을 단축시켰다.

본 연구는 동일 재난 환경에서 경로 탐색 알고리즘을 체계적으로 비교하여 지하 역사 대피 안내 시스템의 알고리즘의 적합성을 비교 분석하였다. 개별 최적해의 합이 전체 최적해가 되지 않는다는 점을 확인하였다. 또한 개별에 대한 우수한 최적해보다 전체에 대한 효율적인 경로 분배가 중요함을 입증하였다.

향후 연구에서는 실시간으로 센서에서 인원 혼잡도, 재난 확산 현황을 수신하여 이를 기반으로 경로를 제공하는 대피 안내 시스템 개발과 학습이 가능한 AI 기반의 지능형 대피 안내 시스템에 대한 연구를 수행할 계획이다. 또한 본 연구는 중소규모 가상 역사를 대상으로 수행되었으므로, 대규모 환승역, 다양한 접근 시설을 포함한 복수의 실제 역사에 적용하여 알고리즘 비교 결과의 일반화 가능성을 추가 검증할 계획이다. 더불어 본 연구에서는 군집 단위 시뮬레이션을 통해 안내 시스템의 지시를 따르는 이상적 대피 상황을 가정하였으나, 향후 연구에서는 군집 내부의 속도 차이, 개별 인원의 행동 다양성, 심리적 요인 등을 반영한 개인 단위 행동 모델을 적용하여 보다 현실적인 시뮬레이션으로 확장할 계획이다.

본 연구는 다양한 알고리즘에 대한 비교 분석을 수행하였기에 향후 이러한 알고리즘들을 활용하는 대피 안내 시스템 개발의 기초 자료로 활용되어 지하 역사의 재난 대응 체계 구축에 기여할 수 있을 것으로 기대된다.

후 기

본 연구는 국토교통부/국토교통과학기술진흥원의 지원으로 수행되었음(과제번호 RS-2023-00238458).

References

- [1] Ministry of Land, Infrastructure and Transport (MOLIT) (2018) Design Guidelines for Urban Railway Stations and Transfer & Convenience Facilities, Republic of Korea.
- [2] Seoul Metro (2024) Analysis report on causes of evacuation time excess by station, unpublished internal document.
- [3] J.-S. Hwang, Y.-B. Choi (2016) Route Exploration Algorithm for Emergency Rescue Support on Urgent Disaster, *The Journal of the Korea Contents Association*, 16(9), pp. 13-21.
- [4] H.-K. Lee, J.-H. Choi, B.-M. Joo, H.-S. Cho (2025) A Study on Evacuation Route Search in Case of Fire Using the F_A* Algorithm Based on the A* Algorithm, *Journal of Fire Science and Engineering*, 39(1), pp. 22-32.
- [5] J.-W. Jo, J.-H. Bae, H.-B. You, S.-H. Bak (2025) Disaster Evacuation Route Search Using A* Algorithm, *Journal of Digital Contents Society*, 26(3), pp. 685-691.
- [6] S. Haam, H.-S. Kim, M.-T. Yoo, W.-S. Song (2025) Optimal additional evacuation route analysis in deep underground station structures using Dijkstra's algorithm, *Developments in the Built Environment*, 24, 100784, doi: 10.1016/j.dibe.2025.100784.
- [7] H. Yang, Q. Yu, H. Peng (2024) Data-driven mathematical simulation analysis of emergency evacuation time in smart station's operation management, *PLoS ONE*, 19(2), e0298622, doi: 10.1371/journal.pone.0298622.
- [8] X. Wang, B. Liu, W. Ma, Y. Feng, Q. Li, T. Sun (2024) Simulations on Evacuation Strategy and Evacuation Process of the Subway Train Under the Fire, *Fire*, 7(12), 464, doi: 10.3390/fire7120464.
- [9] M. Yoo, S. Haam, W.S. Song (2024) Congestion-Based Earthquake Emergency Evacuation Simulation Model for Underground Structure, *Buildings*, 14(10), 3217, doi: 10.3390/buildings14103217.
- [10] F. Kong, S. Ahn (2025) Graph-Theoretic Optimization Strategies for Subway Station Crowd Evacuation, *Journal of Construction Automation and Robotics*, 4(2), pp. 1-6.
- [11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (2009) *Introduction to algorithms (3rd ed.)*, MIT Press, Cambridge, MA.
- [12] Y. Yan (2023) Research on the A* Algorithm for Finding Shortest Path, *Highlights in Science, Engineering and Technology*, 46, pp. 154-160.

- [13] J.X. Zhou, S. Li, G.Z. Nie, X.W. Fan, C.X. Xia (2020) Developing a revised social force model for pedestrians' earthquake emergency evacuation, *Geomatics, Natural Hazards and Risk*, 11(1), pp. 335-356.
- [14] J.G. Wardrop (1952) Some Theoretical Aspects of Road Traffic Research, *Proceedings of the Institution of Civil Engineers*, 1(3), pp. 325-362.
- [15] D. Braess, A. Nagurney, T. Wakolbinger (2005) On a paradox of traffic planning, *Transportation Science*, 39(4), pp. 446-450.

(Received 8 December 2025; Revised 15 January 2026; Accepted 4 March 2026)

Chiyeong Kang : kangjja9219@gachon.ac.kr

Undergraduate Researcher, Department of Civil & Environmental Eng., Gachon University, 1342, Seongnam-daero, Sujeong-gu, Seongnam-si, Gyeonggi-do, Republic of Korea

Hyunseok Kim : khsss1026@gachon.ac.kr

Graduate Researcher, Department of Civil & Environmental Eng., Gachon University, 1342, Seongnam-daero, Sujeong-gu, Seongnam-si, Gyeonggi-do, Republic of Korea

Kyeonghwan Seong : khan.seong@e-umtech.com

Representative Director, e-umtech Inc., 123 Beolmal-ro, Dongan-gu, Anyang-si, Gyeonggi-do, Republic of Korea

Mintaek Yoo : mintakyoo@gachon.ac.kr

Assistant Professor, Department of Civil & Environmental Eng., Gachon University, 1342, Seongnam-daero, Sujeong-gu, Seongnam-si, Gyeonggi-do, Republic of Korea