Distributed AI Classroom Evacuation Model

Implementing Coordination Mechanisms and Game Theory in NetLogo

Presenter: Blai Gené Mora
Affiliation: Università degli Studi di Modena e Reggio Emilia (Unimore)
Departament: Dipartimento di Ingegneria 'Enzo Ferrari'

The Problem: Classroom Evacuation Safety

- **Initial observation:** High-density classrooms with narrow aisles and fixed seating.
- **Concerns:** Restricted movement and potential for severe bottlenecks during emergencies.
- **The Question:** How does evacuation strategy affect survival rates?
- **Goal:** Develop a program to simulate and compare ‘intelligent’ versus ‘non-intelligent’ evacuation scenarios.

Introduction & Project Goals

Objective

Simulate a realistic emergency evacuation using Autonomous Agents.

Core Research Question

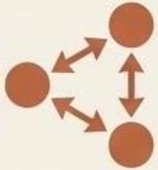
How do distributed coordination mechanisms affect the efficiency (total time) of evacuation compared to uncoordinated movement?

The Problem

- Congestion
- Deadlocks (collisions)
- The “faster-is-slower” effect in bottlenecks.

DAI Principles Implemented

Distributed Coordination



Agents don't have a central leader. They use a Reservation System to negotiate space.

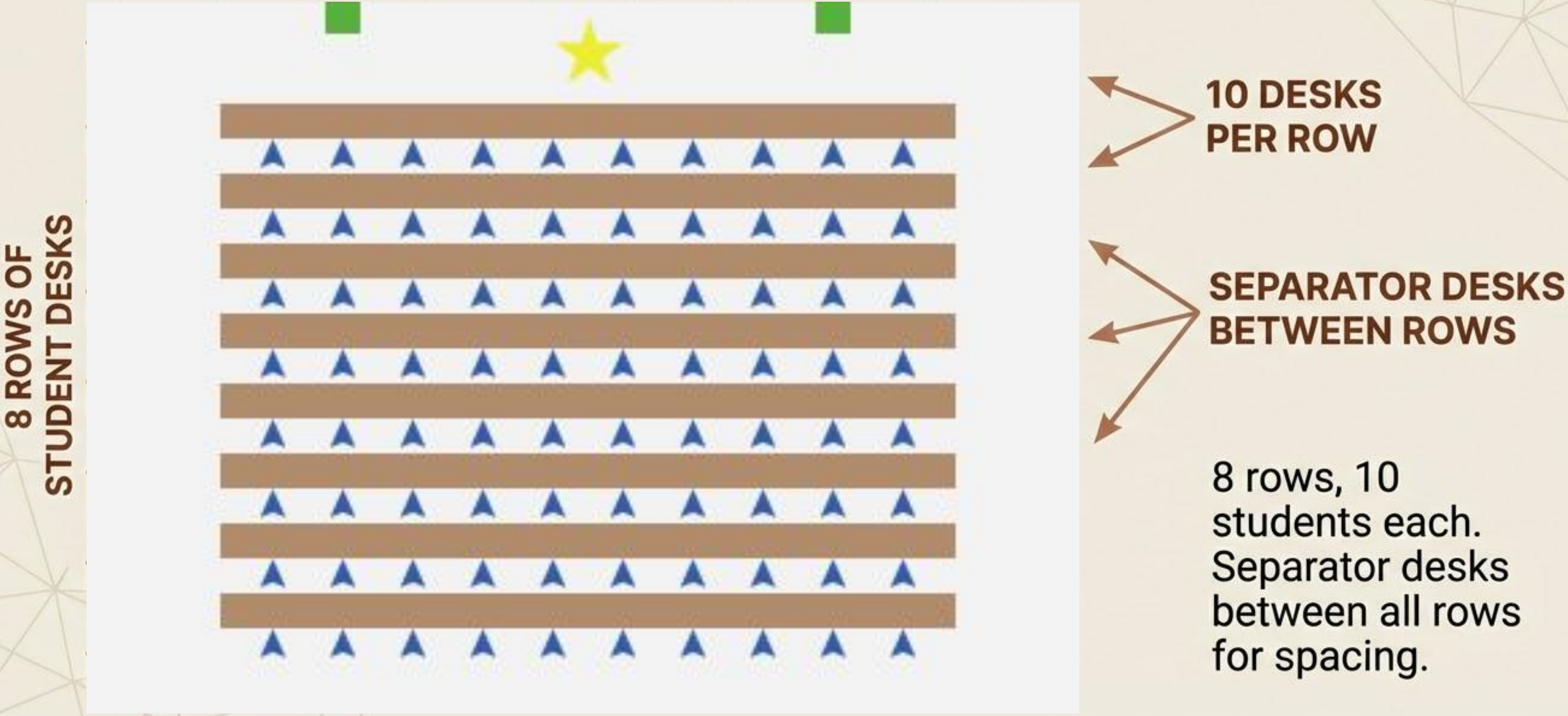
Game Theory

Conflict resolution using Hierarchical Priority (Professor vs. Student) and Greedy Tie-breaking.

Collective Intelligence

Emergent behavior where global order (queuing) arises from local interactions and Flood Fill Pathfinding.

THE WORLD: CLASSROOM LAYOUT



8 rows, 10 students each. Separator desks between all rows for spacing.

The General Algorithm (Overview)

Goal: A collision-free, efficient evacuation.

Environment Awareness

Static pathfinding (Dijkstra/Flood Fill) to find the shortest path.

Iterative Negotiation

Agents “talk” to each other through the patches to reserve space before moving.

Conflict Resolution

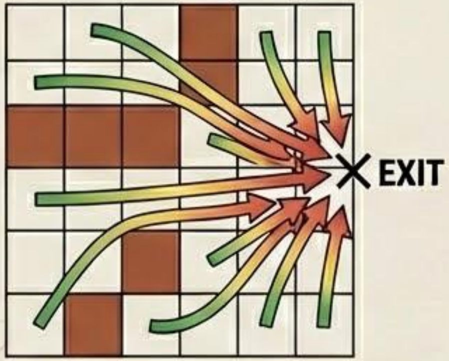
Using priority rules to decide who moves first when two agents want the same spot.

The “Tick” Cycle Prepare → Plan/Negotiate → Move → Record.

Phase 1: Global Pathfinding (The Map)

Logic: Environment Scoring

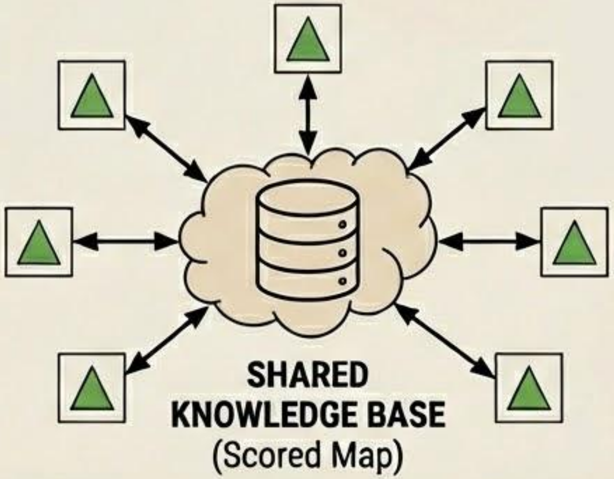
- Before any agent moves, the environment is “scored.” Every patch calculates its distance to the nearest exit, accounting for walls.



Scored Environment Map (Abstract) - Proximity Flow

DAI Concept: Shared Knowledge Base

- This provides the Shared Knowledge Base that all agents use for local decision-making.



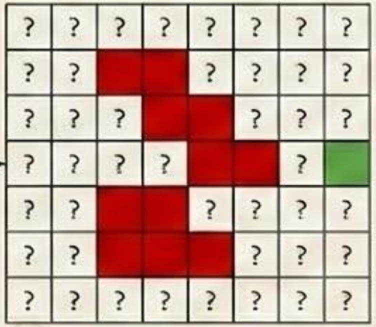
DAI: Pathfinding (Dijkstra/Flood Fill)

Foundation for Coordination & Collective Intelligence: Calculating Shortest Paths

1. Initialization

Sets infinite cost for non-exit patches and identifies obstacles.

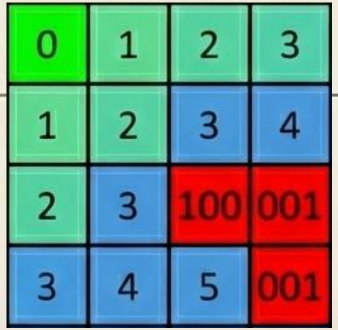
```
ask patches with [pcolor != green] [
  set path-cost 100000
  if obstacle? [ set path-cost 100001 ] ; Blocked patch
]
```



2. Propagation Loop (Flood Fill)

Iteratively updates path costs from neighbors to find minimum distance.

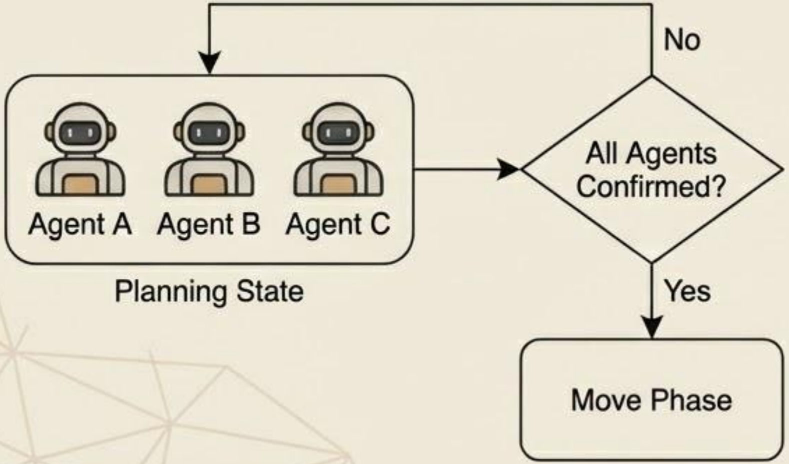
```
let cost-changed true
while [cost-changed] [
  set cost-changed false
  ask patches with [not obstacle? and path-cost > 0] [
    let current-cost path-cost
    let min-neighbor-cost min [ path-cost ] of neighbors4
    if current-cost > min-neighbor-cost + 1 [
      set path-cost min-neighbor-cost + 1
      set cost-changed true
    ]
  ]
]
```



Phase 2: The Planning Loop (Negotiation)

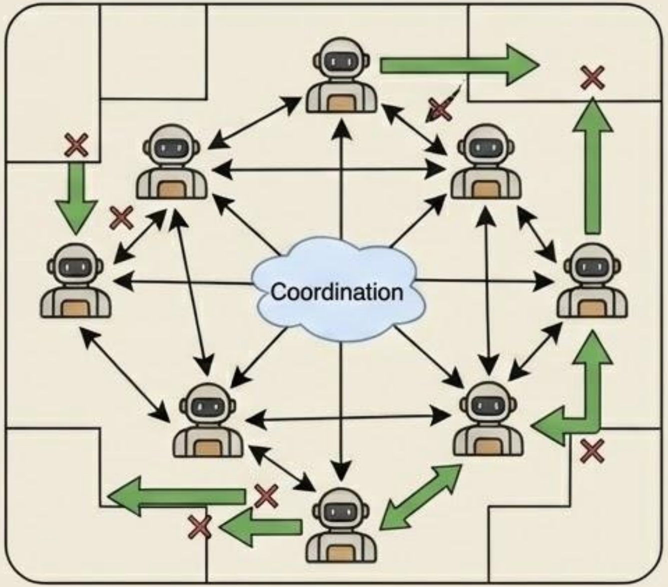
Logic: Agents don't move immediately.

- They enter a "planning state."
- This loop repeats until every agent has a confirmed destination or knows it must stay still.



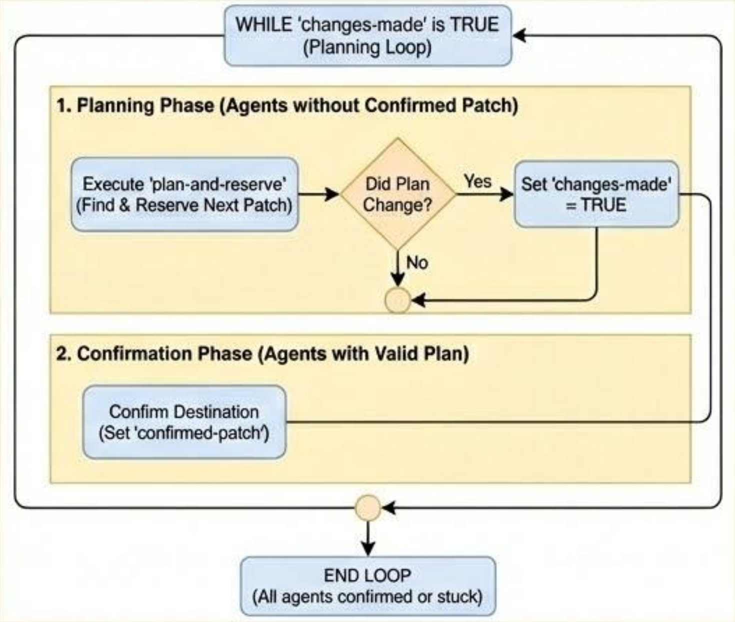
DAI Concept: Distributed Coordination.

- It ensures that movement is synchronized and collision-free.



Code Implementation: The Planning & Negotiation Loop

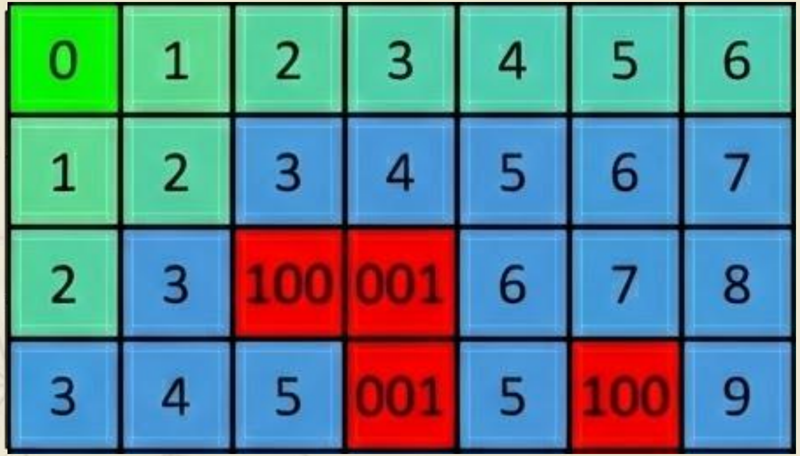
```
while [changes-made] [
  ask turtles with [confirmed-patch = nobody] [
    plan-and-reserve
    if planned-patch != current-planned-patch [
      set changes-made true ]
  ]
  ask turtles with [planned-patch != nobody] [
    set confirmed-patch planned-patch ]
]
```



Logic: Agents enter a 'planning state' where they negotiate and reserve patches. This loop continues until every agent has a confirmed destination or knows it must stay still, ensuring synchronized movement.

Phase 3: Selection & Scoring

Logic: Each agent looks at its neighbors and filters out obstacles and patches that are already occupied by agents who aren't moving. It then picks the one with the lowest path-cost.



Code Implementation:

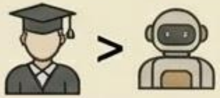
```
let candidates neighbors with [
  not obstacle? and
    (not any? turtles-here or
     all? turtles-here
     [confirmed-patch != nobody])
    and path-cost < my-path-cost
]

let best-candidate min-one-of
  candidates-filtered [ path-cost ]
```

Phase 4: Conflict Resolution (Game Theory)

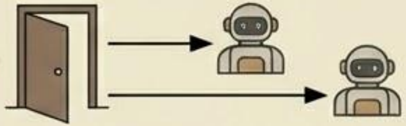
Logic: If two agents pick the same best-candidate, they compare attributes.

Priority Rules:

Agent Type: Professor > Student.



Distance: Closer to exit = Higher priority.



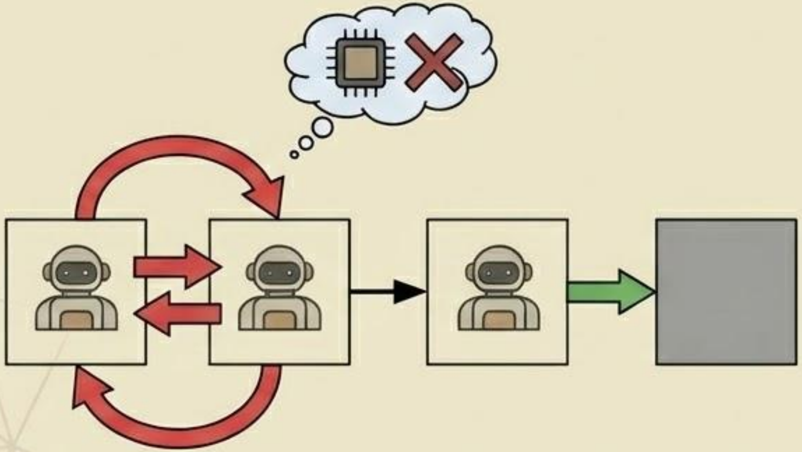
ID: Final tie-breaker.

Code Implementation:

```
if rival-type = "professor" and
  agent-type = "student" [
    set i-win? false
]
if rival-type = agent-type [
  if my-d < rival-d [
     set i-win? true
]
```

Phase 5: Anti-Deadlock & Memory

Logic: To prevent agents from getting stuck in an infinite loop (moving back and forth), the agent “remembers” failed attempts.



Code Implementation:

```
Fragment de codi

set reservation-blacklist lput
best-candidate reserva-blacklist
set obstacle-avoid-history lput
patch-here obstacle-avoid-history
```

DAI Concept: Agent Robustness. Local memory prevents systemic failure (deadlocks).

Experimental Strategy & Scenarios

Benchmarking Distributed Coordination vs. Greedy Behavior

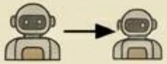
The Comparative Method

Goal: To quantify the efficiency of the Distributed AI mechanisms.

Model A: Proposed DAI (My Code)

- Uses Reservation Protocol & Game Theory.
- Agents negotiate space before moving.

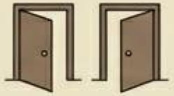
Model B: Naive Greedy (Baseline)



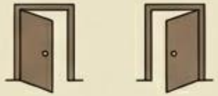
- Standard "Best-First" movement.
- Agents move to the best neighbor without coordination (causes collisions).

The 5 Test Scenarios

- Central Symmetric Exits (Baseline Flow)



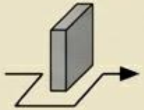
- External Symmetric Exits (Long Distance)



- Single Exit (Extreme Bottleneck)

- Asymmetric Capacity (Intelligent Choice)

- Blocking Wall (Pathfinding Resilience)



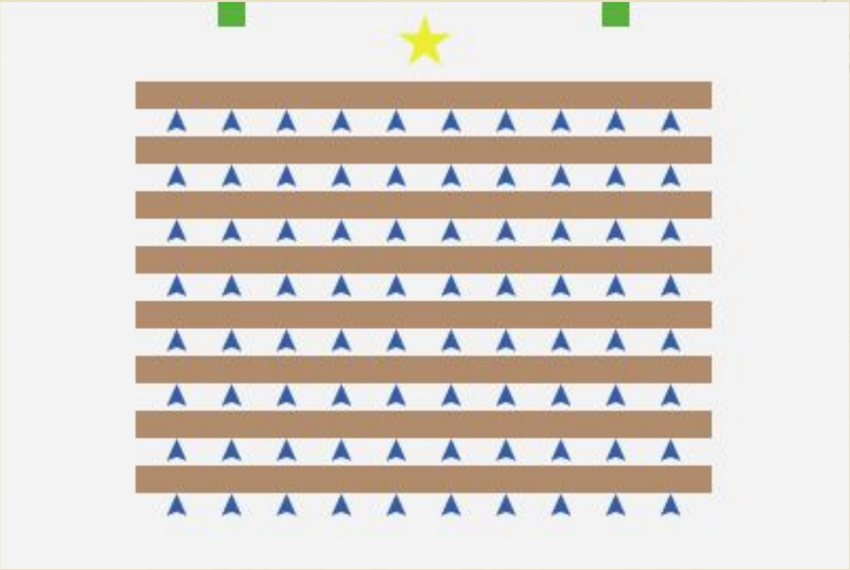
Scenario 1: Central Symmetric Exits

The Setup: Two identical exits placed centrally (Coordinates: x=-7 and x=7).

The Goal (Baseline): To establish a performance baseline.

What we observe:

- Ideal load balancing: Agents should naturally split 50/50 between the two exits.
- Ordered abandon of the rows.
- Verifies that the basic pathfinding (Flood Fill) works correctly without obstacles.



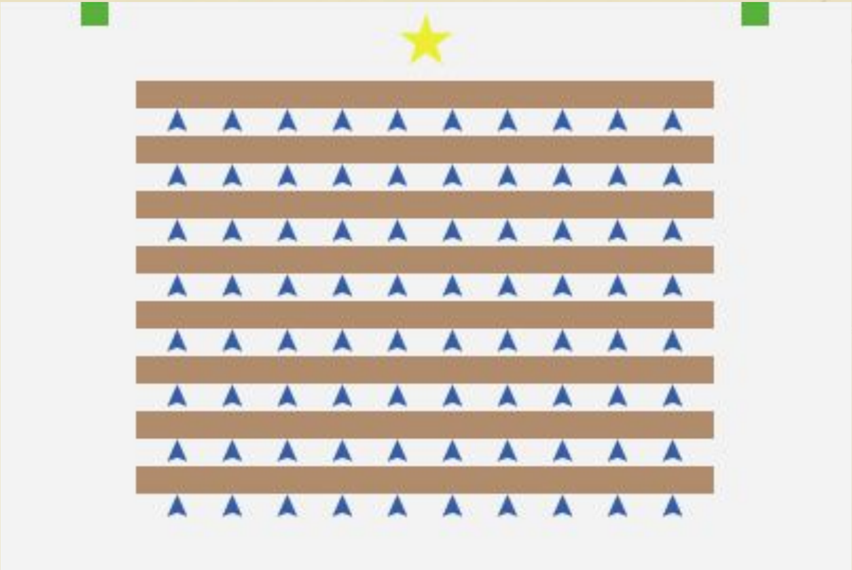
Scenario 2: External Symmetric Exits

The Setup: Two identical exits placed at the far corners of the room (Coordinates: x=-12 and x=12).

The Goal (Distance Impact): To test the impact of travel distance on coordination.

What we observe:

- Agents have to travel longer distances to exit.
- We analyze if the ‘reservation system’ holds up over longer paths or if ‘traffic waves’ occur as agents cross from the center to the edges.



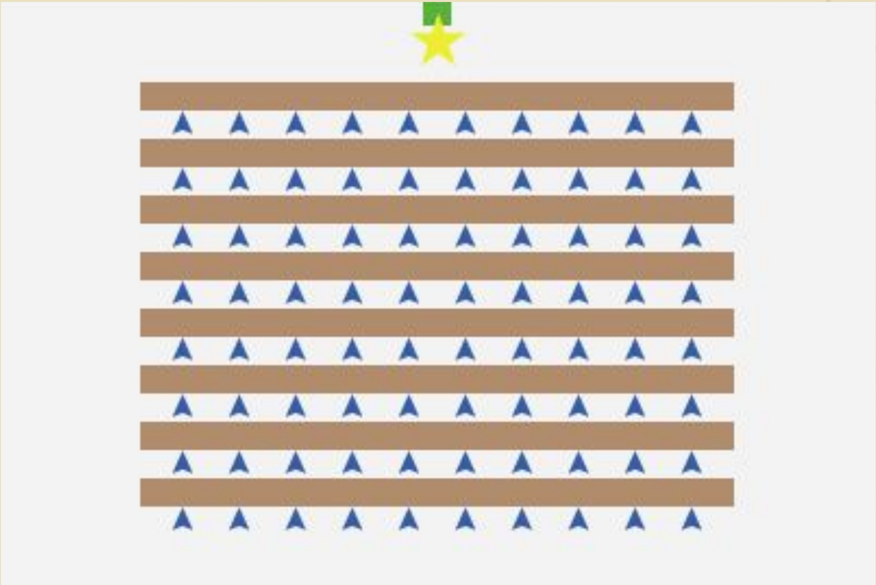
Scenario 3: Single Exit (The Bottleneck)

The Setup: Only one central exit is open.

The Goal (Stress Test): To test Emergent Queuing and Deadlock Resolution Resolution.

What we observe:

- This is the hardest scenario for coordination.
- **Without DAI (Bad Code):** Agents should jam the door (clogging).
- **With DAI (Good Code):** We expect to see an orderly "arch" formation or single-file line as agents respect the reserved-by variable.



Scenario 4: Asymmetric Capacity

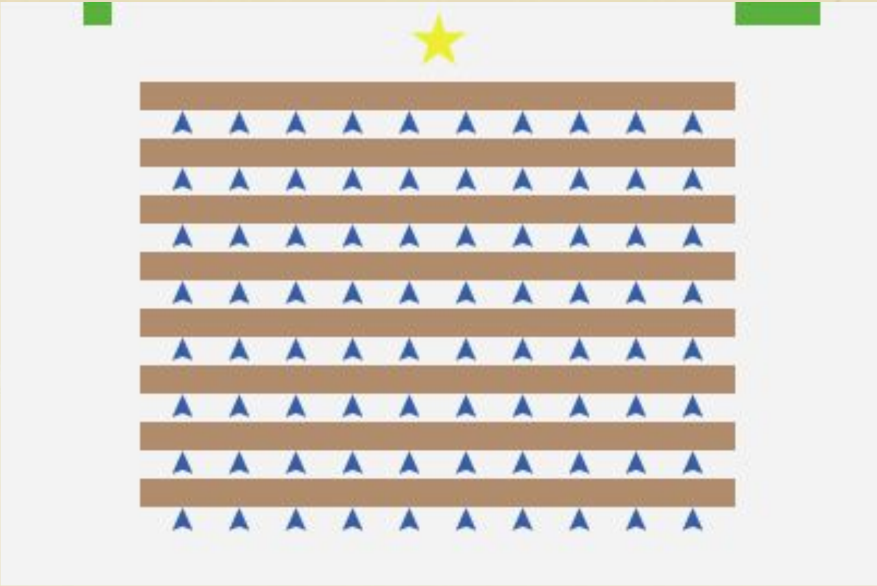
The Setup:

- **Left Exit:** Narrow (1 patch) with high "friction" cost (Cost +6).
- **Right Exit:** Wide (3 patches) with low cost.

The Goal (Collective Intelligence): To test Decision Making.

What we observe:

- Will agents choose the 'wider' exit even if the narrow one is closer?
- Demonstrates how the path-cost variable influences agent distribution, simulating the preference for lower-density exits.



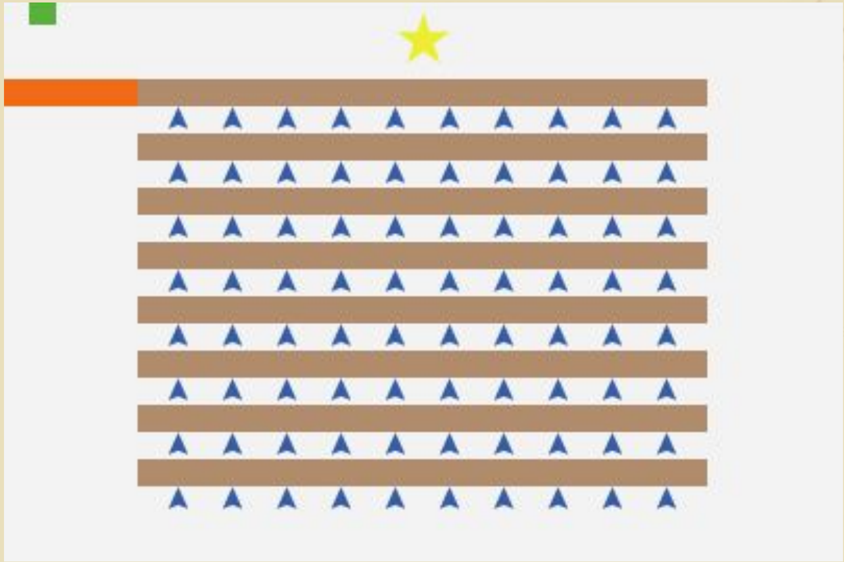
Scenario 5: Blocking Wall

The Setup: A large horizontal wall blocks the direct path to the exit.

The Goal (Resilience): To test Non-Linear Pathfinding.

What we observe:

- Agents cannot move in a straight line (Greedy approach fails here).
- Proves that the Flood Fill algorithm correctly propagates cost around obstacles.
- Tests if the ‘Reservation System’ works in narrow corridors formed by the wall.

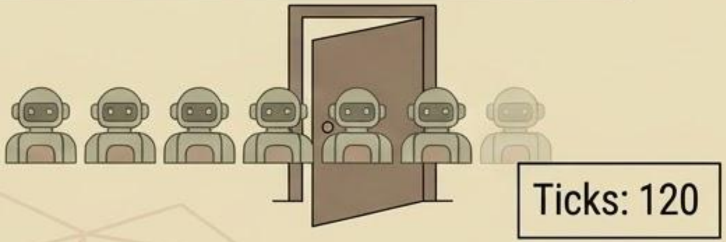


Comparative Results (Good vs. Bad Code)

Visual Comparison:

Bad Code: Agents overlapping, getting stuck, high “ticks” count.

Good Code: Smooth flow, no overlaps.



Quantitative Data:

Total Evacuation Time:

Deadlocks (Conflicts):

Conclusion & Results

Key Findings: Coordination reduces 'panic' behavior and physical blocking.

Emergent Behavior: Orderly lines (queuing) emerge naturally from the reservation protocol.

Final Thought: Distributed AI can make physical spaces safer through better flow management.