Congestion Avoidance in Multi-Agent-based Egress Simulation

Bikramjit Banerjee, Matthew Bennett, Mike Johnson, Adel Ali
University of Southern Mississippi
Hattiesburg, MS 39406
Keywords: Intelligent agents, Applications: Game AI, agent-based simulation
Contact: Bikramjit.Banerjee@usm.edu

Abstract

We focus on multi-agent-based egress simulation from large sports stadiums. We present the efficient Portal Clearinghouse *algorithm for locating bottlenecks that are temporarily congested by agents. The purpose of this step is to allow agents who intend to use the congested portals to re-rout and avoid adding to existing congestions. We present simple heuristic techniques for other sub-problems in congestion avoidance, and demonstrate our approach on a small hand-crafted domain. Interestingly, we observe that a sophisticated vision based avoidance behavior can be emulated with our simple and efficient technique, on some maps.*

1 Introduction

Crowd behavior simulation [1, 2, 3, 4, 5] has been an active field of research because of the utility of simulation in several applications such as emergency planning and evacuations, designing and planning pedestrian areas, subway or rail-road stations, besides in education, training and entertainment. In this paper, we focus on egress behavior from a large sports stadium, using a distributed multi-agent-based simulation framework. Although most of the relevant issues of domain modeling and agent pathing can be solved using traditional techniques from visual simulation and gaming, there is one issue that has not been addressed adequately, to the best of our knowledge. When agents form a congestion at a bottleneck, other agents who intend to pass through the same bottleneck should not keep accumulating and expanding the congestion. Instead, these agents should find an alternate route and avoid existing congestions. We call this problem *congestion avoidance*. We are not aware of any existing technique that solves this problem in time linear with the number of agents or bottlenecks. This is precisely our contribution in this paper.

In order to keep the computational complexity limited to linear in the number of agents or bottlenecks, we do not implement any costly vision/perception mechanism. Instead, we simulate the effect of vision through indirect means. We introduce the technique of *Portal Clearinghouse* (PCH) which identifies the set of bottlenecks that are temporarily blocked by agents. We use AI blind search technique for identifying agents who can "see" a congestion and then use the popular A* [6] algorithm for re-routing groups of agents with common destinations. The PCH is the main novel contribution of this paper, which solves the major sub-problem of *congestion location* in a scalable manner ($O(n+p)$ where n is the number of agents and p is the number of portals). We demonstrate the working of our congestion avoidance mechanism on a small map with 3 corridors between the start and the target regions, showing agents successfully using the longer routes in an attempt to avoid congestion along the shortest route.

2 Related work

In order to face the complex challenges that crowd simulation poses, primarily three approaches[1] have been traditionally used. One approach assumes that the individuals are passive entities that drift in the presence of forces – the so-called "social forces" model [1] and the associated variants of gaskinetic model proposed by Helbing. This model has been extended to include further individual-level details such as familial ties and altruism [4].

The other approach is an agent-based model where individuals are modeled as intelligent agents with (limited) perception and decision-making capabilities. Some of the earliest applications of simple agent-based behaviors were seen in Reynolds' flocking model – the "boids" [7]. In this and related works, each agent is endowed with a mix of simple steering behaviors, that produce complex macroscopic (group-level) behaviors as *emergent phenomena*. The ba-

[1] We do not discuss user-guided or scripted crowd behaviors since they are not relevant to our application domain, viz., stadium egress

sic idea of emergent behaviors has been extended to rule-based systems [2, 3] that offer the added advantages of efficiency and variety in behaviors. In this paper, we adopt the agent-based approach, but our focus is to advance the state-of-the-art by solving the associated problem of congestion avoidance in an efficient manner.

Cellular automata [5] underlie the third major approach, with recent improvements [8] for pedestrian room evacuation. Since our intended application is egress from large stadiums, a CA approach is not quite appropriate due to scalability concerns on very large maps.

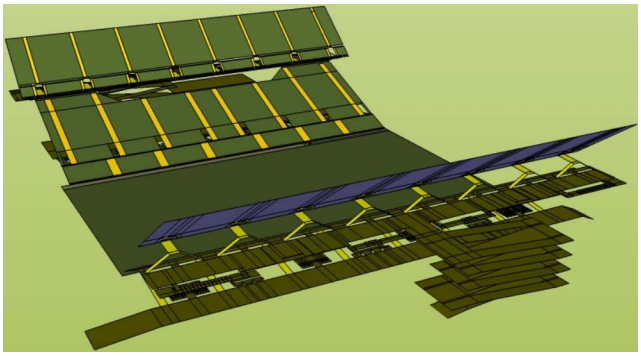


Figure 1. Partitioning of the walkable surface into convex polygons.

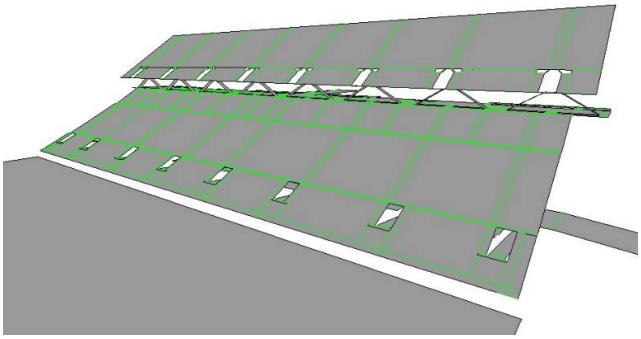


Figure 2. Portals shown in green on one section of a stadium.

3 Egress Simulation

In this paper, we focus on simulating the egress behavior of spectators from a sports stadium. The input to our simulation is a sketch-up of a stadium with all walkable surfaces partitioned into a set of convex polygons, P. Figure 1 shows the partitioning of the walkable surface into rectangles of various sizes (the different colors are not relevant here). We have developed a GUI to populate the stadium with agents of different sizes and speeds. The main task is to make the agents egress the stadium in a human-like manner. In particular, the agents should be able to

- Choose the shortest path to exits if there are no congestions in bottlenecks, or no dynamically added obstacles.
- If there is a congestion at a bottleneck, the agents should be able to replan an alternate path *cheaply*. This problem is the main focus of this paper.
- If there are dynamically added obstacles, such as debris from an explosion, or barricades placed by emergency personnel, the agents should be able to replan an alternate path. Although the handling of such obstacles is no different from dynamic congestions (previous bullet), this is a lesser challenge since the locations of such obstacles are user-defined. Hence there is no need to autonomously locate such obstacles, as is the case with dynamic congestion.

We call the junction (plane) between two polygonal regions a *portal* if it is passable. It is possible that two regions are connected by a physical device such as a door, or just an open connection between two polygons in a contiguous open region; we refer to all such polygon-to-polygon connections as portals. Figure 2 shows the portals in green, for one face of a stadium.

Given a partition of the walkable surface into a set of polygons (P) connected through portals, it is simple matter to calculate all-pair-shortest-paths by the Floyd Warshall algorithm, *offline*, based on the polygon-graph. While the polygons form the nodes of the graph, its edges are formed by the portals with the edge weights determined by the sum of Euclidean distances between the centroids of the adjacent polygons and the center of the intervening portal. We save the resulting distances and path-lookup tables for online use during the simulation.

The agents' pathfinding is done in two levels. At the top (abstract) level, a path specifies the series of polygons that an agent must step through to reach its goal, as discussed in the previous paragraph. At the lower level, an agent needs to navigate *within* a polygon. This navigation task is simplified by the knowledge that there are no static obstacles within a polygon, since all such obstacles are skirted during the initial partitioning. Furthermore, since all polygons are convex, we are guaranteed that an agent can navigate from any point in a polygon to any other point in that polygon, without encountering a static obstacle. The only obstacle that an agent may encounter within a polygon is another agent, or a dynamically added obstacle.

We use the *seek* steering behavior [7] to let an agent seek an appropriate point on a portal connecting to the next poly-

gon on the agent's path. We also use the *collision-avoidance* steering behavior [7] to enable an agent avoid running into other agents or dynamically placed obstacles.

While steering behaviors allow an agent to navigate realistically within a polygon, a common pathing problem in egress scenarios is the congestion at narrow portals that should prompt an agent to repath. Ignoring this aspect will produce ever-increasing congestions that are not only visually unrealistic, but also yield erroneous egress rates, invalidating the simulation. We call this problem *congestion avoidance* and outline our solution to this problem in the next section.

4 Congestion Avoidance

There are three major subproblems under congestion avoidance:

Blocked portal identification: Here the problem is to identify which portals are congested, i.e., *congestion location*.

Re-routing candidate selection: Given that a blocked portal has been identified, this problem deals with identifying a subset of agents who can visually locate this congestion and decide to repath.

Re-routing: Finally, given the subset of agents who have decided to repath, how do we actually compute an alternative path for such an agent, which must be the shortest path under the constraint that the congested portal is (temporarily) unpassable?

The main focus of this paper is the first problem, viz., blocked portal identification. We offer a novel and efficient (linear in the number of agents and portals) solution to this problem. For the remaining two problems, we offer simple heuristic solutions that are to be improved in the future. We describe these solutions below.

4.1 Blocked Portal Identification

A first-cut solution to this problem would be portal-driven. Each portal checks its vicinity and if it sees a large enough crowd then it announces that it is closed. This solution ($O(np)$) is more efficient than an agent-driven solution ($O(n^2)$) where agents check their vicinity for crowds, since there are usually fewer portals than agents. Although the portal-driven solution is in the spirit of *smart terrain* [9] it still involves significant computation (potentially $O(n)$) per portal, and allows for flip-flop decisions when using a hard cut-off for crowd size to decide on closure.

We offer a novel solution, called *portal clearinghouse* (PCH), that is both agent-driven and portal-driven, thus distributing the computational load, as well as making the overall computations simple. This solution has a complexity of $O(n + p)$. The portal clearinghouse is a table that records *number of complaints (c_p) against each portal, p.* A *complaint* is issued by an agent i if it is unable to pass through a portal (p) connecting its current polygon with the next polygon on its path, within a given time d_{ip}. Since every agent uses sophisticated navigation technique within a polygon, it is reasonable to assume that the only reason an agent fails to pass through a portal on its path is a crowd blocking that portal, if d_{ip} is reasonably chosen. The portal clearinghouse iterates through the list of portals and for every c_p that exceeds a threshold C, p is closed.

Since d_{ip} has to be chosen for every agent i, it is necessary to keep the computation of d_{ip} simple, to manage the overall complexity of the blocked portal identification process. We use a heuristic approach that is simple yet effective. When an agent i enters a polygon, it computes the total time, T_{ip}, that it would take to reach the portal [2] p (connecting to the next polygon) from its current position, at its current speed, assuming a straight line path. Its actual path would most likely be different since there could be other agents to avoid collision with. So, for some choice of $m > 1$, we say

$$d_{ip} = m \cdot T_{ip}$$

Everytime an agent enters a new polygon, d_{ip} is computed and a timer is started. When the timer exceeds d_{ip} with the agent still in the same polygon, the agent issues a complaint and the portal clearinghouse increments c_p. Since a crowd

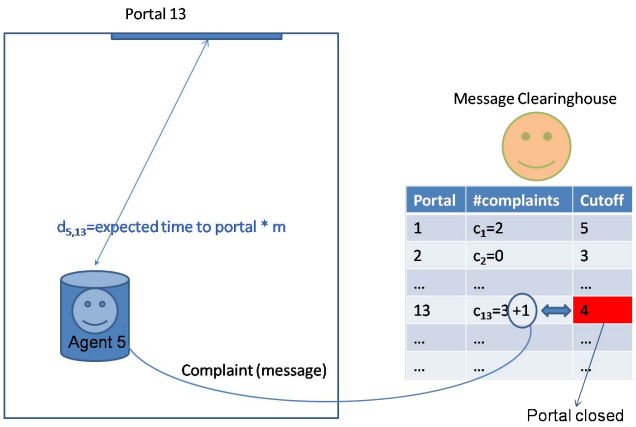


Figure 3. Overview of the Portal Clearinghouse (PCH) technique.

blocking a portal is a temporary obstacle, each complaint received by the portal clearinghouse is also timed. Every message (from agent i about portal p) expires after a time

[2] More specifically, the time to reach the *seek* point on the portal by the *seek steering behavior*.

t_{ip} from the time it is issued, and c_p is decremented. Consequently, an agent i must keep its timer running and reissue a complaint every t_{ip} units of time after issuing a complaint about p for the first time, unless it manages to move into the next polygon on its path, at which point its timer is reset. Figure 3 illustrates the basic process.

The cutoff C is chosen based on the relative sizes of the portal and the agents. Each agent is displayed as a cylinder of a chosen radius, but fixed height. The approximate maximum number of agents that can stand side to side along the length of a portal is given by the ratio (length of a portal / average agent radius). The value of C is chosen to be this maximum scaled by a fixed constant.

It is important to note that a *closed* portal does not mean it is unusable. The agents that are currently crowding a *closed* portal will continue to attempt to pass through it. It is only the agents that intend to pass through the same portal, and are *in the vicinity* during the period that the portal is closed, that will try to avoid it. We turn to identifying this set of agents in the next section.

4.2 Re-routing Candidate Selection

Once a closed portal is identified, the next task is to identify the set of agents (other than the set of agents that is already crowding this portal) who should be able to "see" this congestion. Since we do not perform expensive agent-vision, we again use a heuristic approach for this step. Given the farthest distance that an agent can see (the visible distance), each polygonal region with a closed portal performs a *distance limited depth-first search* [6] to identify which other polygons are in its *Potentially Visible Set (PVS)*, using the visible distance as the distance limit. The PVS is the set of all polygons such that agents located in these polygons can *potentially* "see" the congested portal. This is, however, a crude measure since polygons that are separated by a wall could be included in the PVS of each other, if the route between them is shorter than the visible distance. So the PVS set needs to be refined, by eliminating such candidate regions. In this paper, we leave this refinement as a future work, and accept the PVS as the *actual* visible set.

All agents in the visible set are potentially candidates for re-routing. It is rather straightforward to verify if the closed portal lies on the path of a potential candidate, by checking if the pair of regions that the portal connects appears in its path. Every agent in the visible set that passes this test is a re-routing candidate.

4.3 Re-routing

In a stadium egress scenario, especially in a panic situation, most (if not all) agents in a polygon have the same destination. As a result, we can reduce the problem of re-routing candidates to the problem of re-routing groups of candidates. For each such group, we perform A* search under the constraint that the given portal is closed. It is possible that several more portals along new paths are also closed, but the agents should not know about them until they navigate to the PVS of the associated polygons. This naturally limits the complexity of re-routing.

5 Demonstration

To demonstrate how the system works, we show a series of snapshots of the crowd behavior on a small map, shown in Figures 4, 5. The map consists of a source region (where the agents are initially located) and a sink region (shown in blue), with 3 corridors connecting these regions. The corridors are created such that the progressively shorter routes pass through the progressively narrower corridors, thus forcing the agents to invoke the re-routing behavior at various times. The green lines on the maps show the portals. Figure 4(a) shows the initial configuration. In Figure 4(b), bulk of agents approach the narrowest corridor, thus closing it quickly, prompting a branch to approach the middle corridor. In Figure 4(c) we see the middle corridor gets more crowded, but since it is wider, it takes a while for it to get closed, while the crowd spills toward the bottom corridor (Figure 4(d)).

Figure 5(e) shows an interesting side effect of our PCH system. Since many agents had spilled over toward the bottom corridor while waiting to pass through the middle corridor, they were already in a polygon with the red portal (Figure 5(e)). When these agents find the portal to the middle corridor closed, they re-route and their new path passes across the red portal. However, since they were already located in this polygon, their timers had started, and agents start complaining even before they reach this (red) portal. With sufficient number of complaints, the portal gets closed (a portal turns red when it gets closed). This allows the trailing agents in the adjacent region (see Figures 5(f) and (g)) to re-route and turn back to the middle corridor since that portal had opened by this time. This behavior emulates the ability of humans to re-route not only when an exit is blocked, but also when we see a huge crowd approaching a narrow exit on our way, and re-route *in anticipation*. Finally, Figure 5(h) shows the agents approaching the target region via all three corridors.

The above side effect that emulates re-routing in anticipation, is actually the result of a combination of timer-based PCH and appropriate polygonal partitioning, rather than PCH alone. It might not have arisen with a different scheme of partitioning. Hence we do not tout this as a guaranteed benefit of the PCH system, or a potential replacement for costly vision-based techniques. However, the demonstra-

tion does indicate that the PCH system works seamlessly with imperceptible computational delay in a 1000 agent simulation.

6 Conclusions and Future Work

We have presented the *Portal Clearinghouse* algorithm for efficient congestion location in a multi-agent-based stadium egress simulation scenario, and demonstrated its effectiveness in a simulation with 1000 agents. For the two related subproblems of re-routing candidate selection, and re-routing, we have used simple heuristic solutions that we intend to refine in the future. Although we do not employ expensive vision algorithms, we have demonstrated that our technique can display sophisticated behaviors that emulate vision based complex AI, on some maps. In the future, we would like to refine the PVS identification approach, and explore more efficient re-routing algorithms.

7 Acknowledgments

This work was supported in part by a Department of Homeland Security grant (BOA:4200000225, Task Order 2 63894), managed by the Oak Ridge National Lab through the SERRI program, and a start-up grant from the University of Southern Mississippi. All views expressed in this paper are solely of the authors, and in no way reflect the views of the US government, or the University of Southern Mississippi.

References

[1] Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physical Review E **51** (1995) 42–82

[2] Ulicny, B., Thalmann, D.: Towards interactive real-time crowd behavior simulation. Computer Graphics Forum **21**(4) (2002)

[3] Pan, X., Han, C., Dauber, K., Law, K.: A multi-agent based framework for simulating human and social behaviors during emergency evacuations. In: Social Intelligence Design, Stanford University (March 2005)

[4] Braun, A., Musse, S.R., de Oliveira, L.P.L., Bodmann, B.E.J.: Modeling individual behaviors in crowd simulation. In: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA, Los Alamitos, CA, USA, IEEE Computer Society (2003) 143–148

[5] Fukui, M., Ishibashi, Y.: self-organized phase transitions in CA-models for pedestrians. J. Phys. Soc. Japan (1999) 2861–2863

[6] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (1995)

[7] Reynolds, C.: Flocks, herds and schools: A distrtibuted behavior model. In: Proceedings of ACM SIGGRAPH. (1987)

[8] Gudowski, B., Was, J.: Some criteria of making decisions in pedestrian evacuation algorithms. In: Proc. 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07), IEEE (2007)

[9] Rabin, S.: Promising Game AI Techniques. In: AI Game Programming Wisdom. Volume 2. CHARLES RIVER MEDIA (2003) 15–28

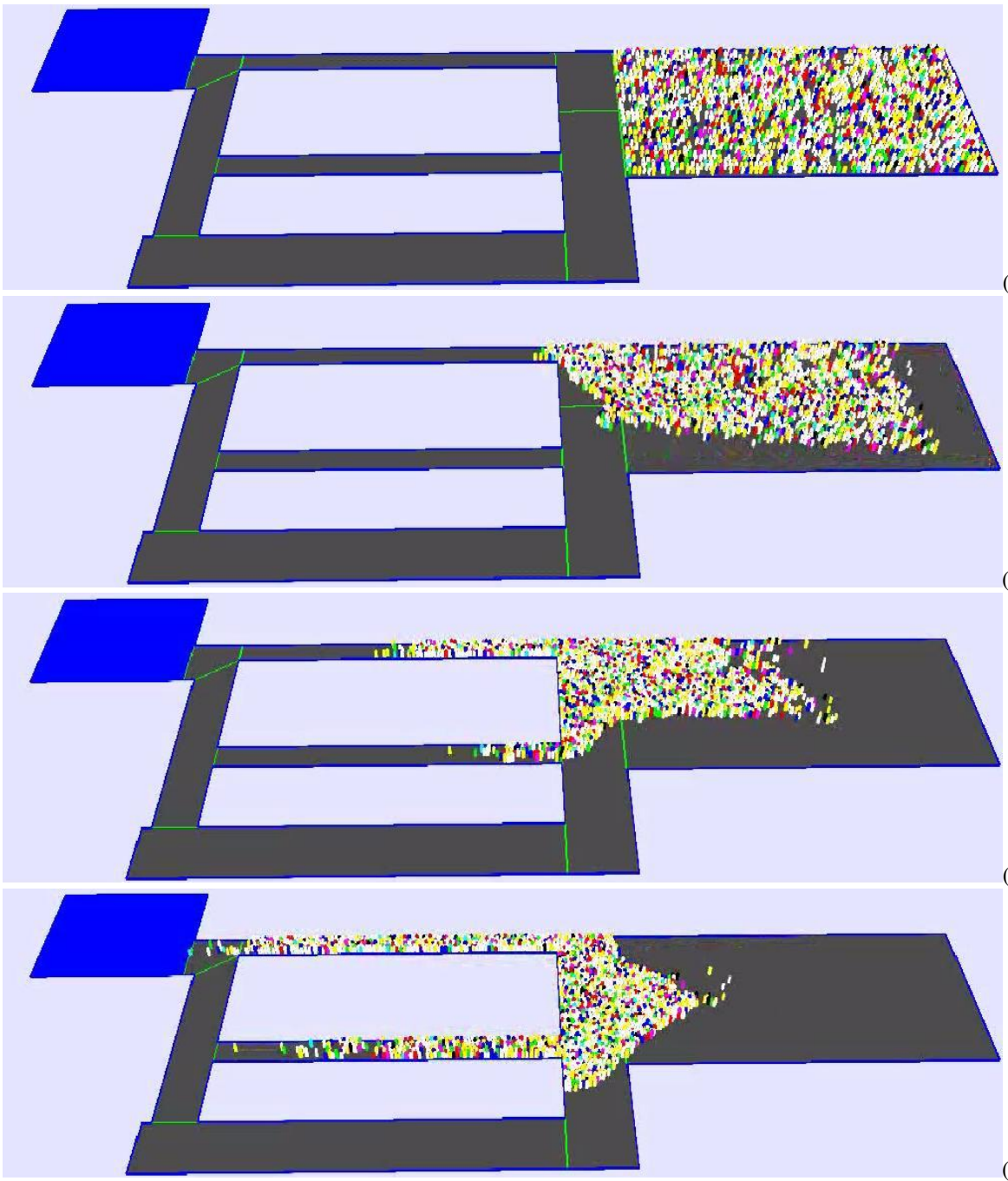
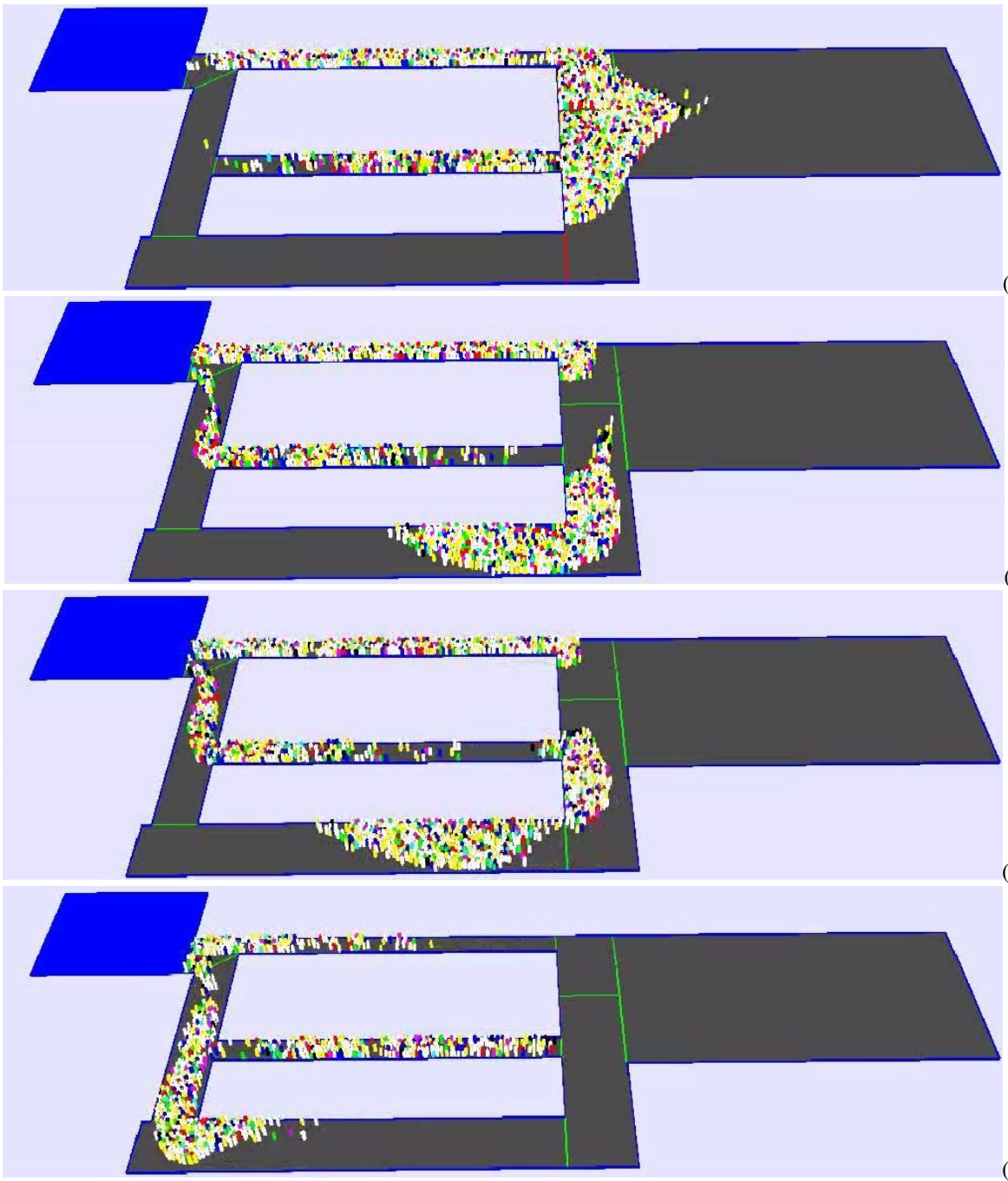


Figure 4. Demonstration of PCH technique on a small map



(e) (f) (g) (h)

Figure 5. Demonstration of PCH technique on a small map (continued from Figure 4).