The synergy of non-manifold topology and reinforcement learning for fire egress

Wassim Jabi[1], Aikaterini Chatzivasileiadi[1], Nicholas Mario Wardhana[1], Simon Lannon[1], Robert Aish[2]
[1] Welsh School of Architecture, Cardiff University
[1] {jabiw|chatzivasileiadia|wardhanan|lannon}@cardiff.ac.uk
[2] The Bartlett School of Architecture, University College London
[2] robert.aish@ucl.ac.uk

Keywords: *Non-manifold topology, Topologic, Reinforcement Learning, Fire egress*

Abstract

This paper illustrates the synergy of non-manifold topology (NMT) and a branch of artificial intelligence and machine learning (ML) called reinforcement learning (RL) in the context of evaluating fire egress in the early design stages. One of the important tasks in building design is to provide a reliable system for the evacuation of the users in emergency situations. Therefore, one of the motivations of this research is to provide a framework for architects and engineers to better design buildings at the conceptual design stage, regarding the necessary provisions in emergency situations. This paper presents two experiments using different state models within a simplified game-like environment for fire egress with each experiment investigating using one vs. three fire exits. The experiments provide a proof-of-concept of the effectiveness of integrating RL, graphs, and non-manifold topology within a visual data flow programming environment. The results indicate that artificial intelligence, machine learning, and RL show promise in simulating dynamic situations as in fire evacuations without the need for advanced and time-consuming simulations.

1. Introduction

One of the important tasks in building design is to provide a reliable design for the evacuation of the users. As the complexity of the building increases this becomes an even harder task. In parallel, non-manifold topology (NMT) is an innovative computer-based representation of architectural entities and can be useful for spatial design, BIM-based design development and building performance analysis for complex geometries. Prior publications by the authors suggest that NMT provides topological clarity that has the potential to allow architects to better design, analyse, reason about, and produce their buildings (Aish and Pratap, 2012; Jabi, 2015; Jabi, 2016, Jabi et al., 2017; Jabi et al., 2018; Aish et al., 2018, Chatzivasileiadi et al., 2018).

Emergency situations are not static, but rather dynamic and uncertain. To provide the optimum support in such situations, not only is there a need there for an ideal evacuation and routing system, but the building itself also needs to be designed optimally. Thus, one of the motivations of this research is to provide a framework for architects and engineers to better design buildings with regard to the necessary provisions in emergency situations. This research aims to further current research on NMT, by hypothesising that the incorporation of machine learning (ML) and in particular reinforcement learning (RL) can help enhance the current design workflows and expand the applications of this representation to include effective evacuation planning in cases of emergency, e.g. fire egress planning. This is demonstrated through the use of an NMT modelling library developed by the authors, called Topologic [1].

2. Previous research

ML is a fast-emerging field and initial research (Krijnen and Tamke, 2015; As, Pal and Basu, 2018; Bloch and Sacks, 2018; Koo and Shin, 2018) has shown that it can be applied directly to provide architectural insights. In this respect, it has also been argued that the use of NMT could contribute towards a richer representation of spatial relationships in architectural design (As, Pal and Basu, 2018), which can be useful for emergency situation egress analyses. A wide range of evacuation systems based on ML has been presented in prior studies. For example, Inoue et al. (Inoue *et al.*, 2008) proposed an indoor evacuation system, generating static escape routes. However, in the case of sudden changes in access, for example unexpected obstructions, this system would be short of providing an alternative route and therefore lacks reliability. This is the case with traditional evacuation systems, which do not consider either the changing environmental conditions in the building or the individual features of the evacuees and as a result they can pose a life risk.

Other studies have considered human-related factors; however, GIS researchers have identified deficiencies due to the lack of convenient algorithms (Meijers, Zlatanova and Pfeifer, 2005; Lee, 2007; Lee and Zlatanova, 2008) or other researchers (Atila *et al.*, 2018) argue that these factors are not sufficient for proper navigation. Ozel (Ozel, 1992) was one of the early pioneer researchers who investigated the human perspective and more specifically the human behaviour in fire incidents through simulation, including the cognitive effects of the physical environment into the decision making process. However, the model ran on a limited CAD system and doesn't offer the possibility of training considering multiple goal structures. Moreover, there are studies that, although considering human factors in the design of their indoor navigation system, such as disabled or elderly and blind people (Khalifa, El Kamel and Barfety, 2010; Blattner, Vasilev and Harriehausen-Mühlbauer, 2015; Tsirmpas *et al.*, 2015), they do not address emergency cases. Atila et al. (Atila *et al.*, 2018) studied the combination of different human factors extensively; however they used building surveying devices for the building case study which can be more expensive and cumbersome than initiating a computer-based design model which can also benefit and be updated from the simulation feedback resulting in an optimised design. There exist a number of highly accurate and comprehensive studies, such as (Luo and Beck, 1994; Bong, 2000; Baum, 2011), including very detailed model inputs regarding fluid dynamics, thermodynamics, gas flow and specific information on materials and building geometry and structure. However, these approaches involve computationally intensive numerical simulations that would be too detailed, slow and expensive to use for conceptual design.

Knight et al. (Knight *et al., 1999)* developed a field modelling tool based on qualitative reasoning, called 'Smartfire', to be used by the fire safety engineering community members who are not experts in modelling techniques. It is applicability, though, is limited because it focuses on the set-up of fire models through metric diagrams and place vocabulary and only addresses single rectangular rooms. Champneys [2] explored the possibility of developing a simplified model to simulate the spread of fire; however, only 2D and no 3D effects were investigated. The majority of the evacuation modelling software tools, such as FDS+Evac [3], FPETool [4], EVACNET4 [5], TIMTEX (Harrington, 1996), WayOut [6], STEPS [7], SIMULEX [8], fall short of a comprehensive modelling environment, as they exhibit one or more of the following limitations: they use basic grid structures and a rectilinear numerical mesh, do not incorporate fire data or occupant behaviour and do not allow for visualisation or might allow for partial visualisation, e.g. 2D or 3D. Most of them are also not freely available to the public or open-source.

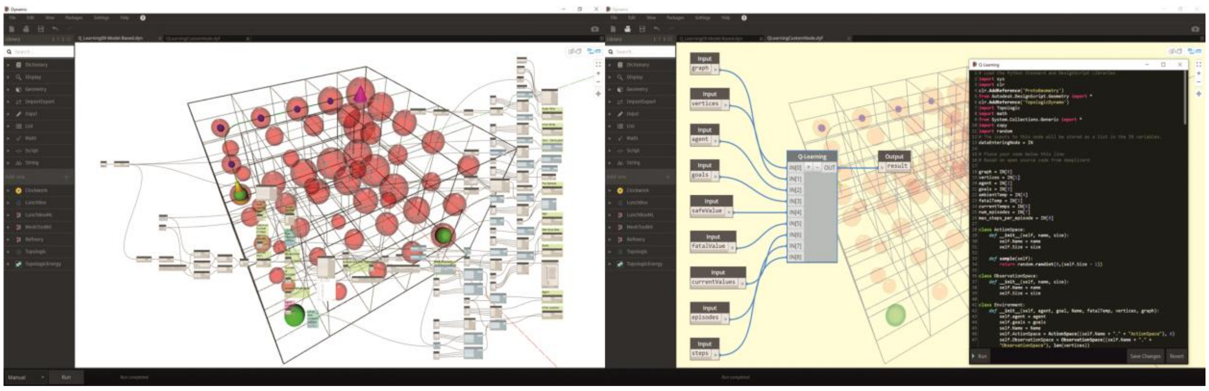
Based on the above, there is no study taking into account artificially intelligent agents that learn independently, a dynamic routing service, fire data and environmental factors at the same time, which can also contribute to informed design decisions at early design stages. It is also argued that the complexity and the size of buildings nowadays requires further research for efficient emergency egress and route selection (Ozel, 1992), especially considering recent disastrous fire incidents, such as the Grenfell Tower one [9]. In this study, therefore, we address this shortcoming by presenting a proof-of-concept experiment for a dynamic and intelligent indoor evacuation system in emergency situations, as part of the spatial design.

3. Reinforcement Learning

RL is a branch of artificial intelligence (AI) and, more specifically, ML, in which an agent optimizes their behavior within an environment to maximize an allocated award. Similar to how one may use treats to teach a pet to do tricks, RL agents learn to carry out a sequence of actions that maximize their final cumulative award. The difference between a pet doing tricks and an RL agent is that the latter learns how to do the trick on their own by observing its environment and learning from its own actions. In a way, it trains itself and thus can be thought of as an artificially intelligent agent. In doing so, RL agents also learn how to avoid actions that either do not reward them or give them a negative award (a punishment). The explanation of RL is based on [10].

To create an RL system, one must represent the following concepts:

- Agent: The artificially intelligent agent. This may be abstract. In our example, the agent is represented by the locations of the rooms that it visits.
- Environment: This is the overall environment in which an agent is operating. In a game of Chess or Go, the environment is the game board. In our case, a building (i.e. a CellComplex with an associated graph) is our environment (Figure 1).



```
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
clr.AddReference('TopologicDynamo')
import Topologic
import math
from System.Collections.Generic import *
import copy
import random
# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

# Place your code below this line
# Based on open source code from [illegible]

graph = IN[0]
vertices = IN[1]
agent = IN[2]
goals = IN[3]
ambientTemp = IN[4]
fatalTemp = IN[5]
currentTemps = IN[6]
num_episodes = IN[7]
max_steps_per_episode = IN[8]

class ActionSpace:
    def __init__(self, name, size):
        self.Name = name
        self.Size = size

    def sample(self):
        return random.randint(0,(self.Size - 1))

class ObservationSpace:
    def __init__(self, name, size):
        self.Name = name
        self.Size = size

class Environment:
    def __init__(self, agent, goal, Name, fatalTemp, vertices, graph):
        self.agent = agent
        self.goals = goals
        self.Name = Name
        self.ActionSpace = ActionSpace((self.Name + "." + "ActionSpace"), 4)
        self.ObservationSpace = ObservationSpace((self.Name + "." +
        "ObservationSpace"), len(vertices))
```

Figure 1. Overall environment using Dynamo and Topologic

- State: The different states or conditions the agent may find themselves in. In our case, we experimented with both an environment-based state model where the state is represented as the index into the list of vertices in the graph and an agent-based state model, where the state was represented as a vector of the adjacent room temperatures. The agent transitions from one state to another by taking an action.

- Action: Agents can take any number of actions. An action is usually represented as an integer number that identifies which action an agent has selected for his/her next move. In our case, we have allowed 6 possible actions representing the direction of travel of the agent (i.e. north, south, east, west, up and down).
- Reward: After every action, the agent usually receives an award represented as a number. This can be negative, zero, or positive. It is up to the designer of the RL environment to decide how frequently and the amount of reward to give the agent based on their action and the state of the environment.

It is beyond the scope of this paper to delve into the mathematics of RL which can be quite involved. If you are interested in the mathematical equations and logic of RL, we recommend the following references (Sutton and Barton, 2018; Thombre, 2018, [10]). In general, there exists probability distributions that can be assigned to all the states and all the rewards which depend on the preceding state and action. The main driving force for an RL system is to predict the overall expected return. This, in turn, is driven partially by how greedy you design the system to be. For example, you can design the algorithm to appreciate immediate rewards more than future rewards (instant gratification) by discounting future rewards. Finally, the probability that an agent will take an action given a certain state is governed by what is a 'policy'. A policy is a function that maps a state to the probability that an agent will select an action. Finally, a 'value' function is used to estimate the expected return of an action given a state. This helps the agent make a choice (e.g. "Should I buy the stock of company A or company B?"). Basically, an RL system learns to optimize their policies so that, after many episodes of training, they can make wise decisions that maximize their overall return and allow them to reach their goal.

Topologic: a non-manifold topology modelling library

Topologic [1] is an open-source software modelling library enabling hierarchical and topological representations of architectural spaces, buildings and artefacts through NMT. Topologic is designed as a core library and additional plugin to two host applications, namely Dynamo and Rhino3D/Grasshopper, which are parametric modelling platforms commonly used in architectural design practice. These applications provide workspaces with visual programming nodes and connections for architects to interact with Topologic and perform architectural design and analysis tasks.

Written using the Object-Oriented Programming (OOP) paradigm, at the top level of Topologic's class hierarchy is the Topology class, which is inherited by other classes representing topological entities, such as Vertex, Edge, Wire, Face, Shell, Cell, CellComplex and Cluster. With respect to building design, a space (or room) corresponds to a Cell and a building corresponds to a CellComplex (which is a set of Cells connected through their Faces). It should be noted that, for the most part, Topologic does not contain any geometrical entities or computations. Instead, it relies on those from the individual host applications to ensure compatibility with their internal geometry kernels. In addition, the topological classes can be used to represent their geometry counterparts. The architecture, capabilities and applications of Topologic can be found in prior publications by the authors (Aish *et al.*, 2018; Chatzivasileiadi *et al.*, 2018; Jabi *et al.*, 2018; Wardhana *et al.*, 2019).

Topologic provides a unified topological framework within which the designed model can serve as a driver for multi-disciplinary analysis, e.g. building fabric, energy analysis, structural analysis and spatial reasoning. Apart from the topological entity classes, and within the spatial reasoning domain, the authors have developed the Graph class, which provides methods to derive the dual graph from any Topology and use it for spatial analysis, including path planning.

Experiment Setup and Input Parameters

The experiment was conducted using Topologic v 0.8.5 within Dynamo version 2.3 running on a MacBook Pro 13 inch with an Intel® Core™ i5 @ 2.90GHz, with 8GB RAM. The laptop was running the Windows 10 Operating system under Apple's Bootcamp. The code is based on open source code provided at [10]. The code was modified from its focus on a 2D grid environment to enable navigation within an arbitrarily complex 3D CellComplex. While the original environment was static, in our adaptation, it is dynamic with a fire simulation that spreads through the building and incrementally increases the temperature in each room accordingly. We also experimented with changing the state model to be agent-based, while the original was strictly model-based. For the experiment we followed the workflow below:

1. Create a CellComplex of a hypothetical building: The first step was to design and build a hypothetical and abstracted building made of identical cubical cells each measuring 10 units. This model was then converted into a non-manifold CellComplex using Topologic so that adjacency information can be queried (Figure 2a).
2. Convert CellComplex to Graph: Topologic has a special class called "Graph" that implements graph theory. Graphs are made of vertices and edges. The dual graph of a CellComplex is a cluster of edges connecting the centroids of adjacent Cells in the CellComplex (Figure 2b).

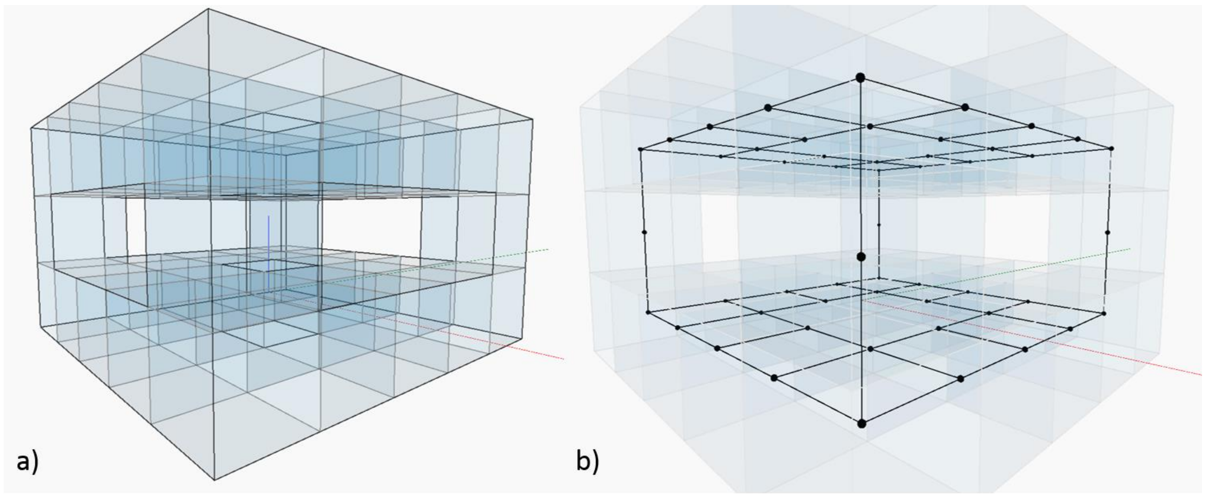


Figure 2. The a) CellComplex, and b) its dual graph created using Topologic

3. Define the initial temperatures. Each Cell in the CellComplex is assigned a temperature. To indicate the source of a fire a Cell is assigned a very high temperature (e.g. 120°C). The remainder of the Cells are assigned an ambient temperature of 20°C.
4. Simulate the spread of a fire in the building. A simplified formula was devised to simulate the spread of fire in the building for each time step in the simulation, using a heat transfer rate of 1.20. The topological adjacency information was used to compute the resultant increase in temperature for each cell and for each time step in the simulation (Figure 3).

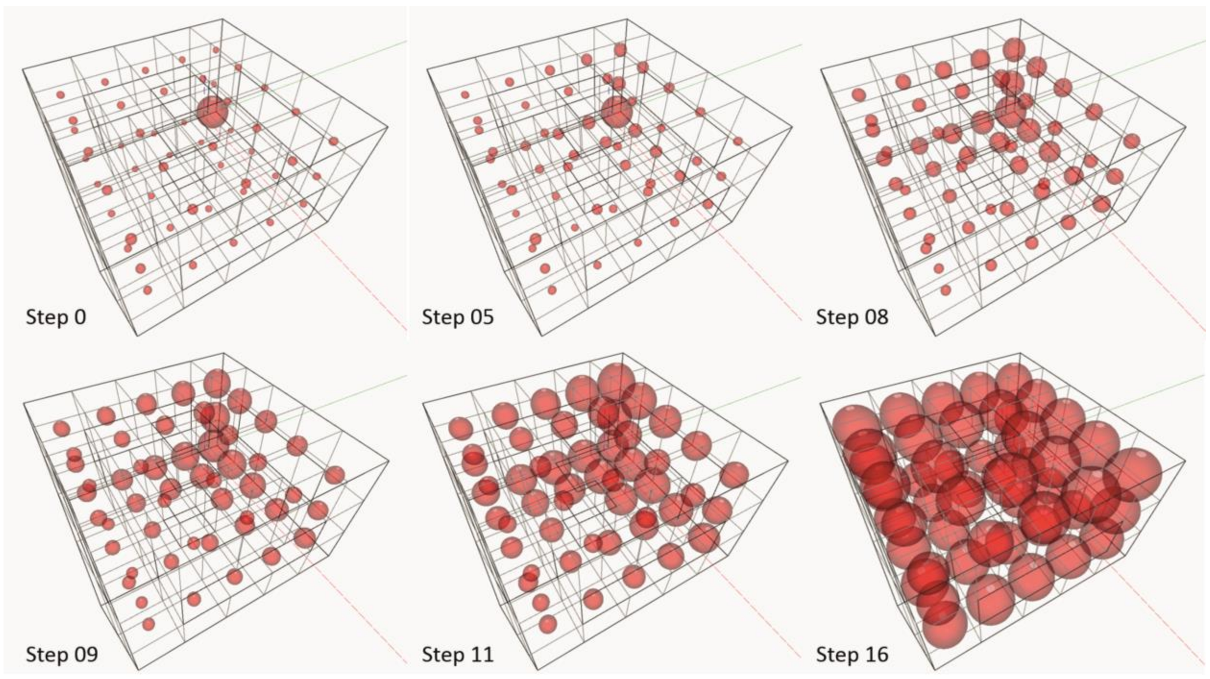


Figure 3. Fire spread simulation in the CellComplex.

5. Run the Reinforcement Learning algorithm: This algorithm allows an artificial agent located in a starting Cell to learn how to escape the expanding fire and find a fire exit. This algorithm requires the following as input: (a) The building graph that defines its topology, (b) Vertices: This is a list of Topologic Vertices that represent the location (centroid) of each room in the building. This is needed to synchronize the order of the room centroids with the list of temperatures and the location of agents and goals, (c) The initial location of an agent: This is expressed as an integer index into the list of Vertices, (d) A list of goals: This is a list of integers into the list of vertices that represents the locations of the fire exits. We experimented with one exit and three exits in the corners of the building. The simulation ends once an agent reaches a fire exit, (e) Ambient temperature: This is a number that indicates what is considered a safe temperature, (f) Fatal temperature: This is a number that indicates what is considered a fatal temperature, e.g. 120°C [11]. If an agent enters a room that has a fatal temperature, then the simulation ends, (g) Room temperatures: This is a list of temperatures for each time step resulting from the spread of fire simulation conducted in the previous step, (h) Number of episodes: This indicates the number of episodes in the simulation, which was set to 3000 in our case. The agent will learn from each episode and improve its performance, (i) Maximum number of episode steps: This number indicates the maximum number of steps allowed in one episode, which was set to 100 in our case. This allows the agent to explore its environment and learn from it for the next episode.
6. Environment-based vs. agent-based state models: As mentioned above in the description of the RL algorithm, an agent transitions from one state to another and collects positive, neutral, or negative rewards after each state transition. The role of the algorithm is to devise a policy that maximizes the probability that an action will yield a state that offers a larger reward than other options. The question then becomes: how should a state be represented? In a chess game, a state may be represented as the location of a chess piece. Indeed, in the first iteration of the algorithm, we used a global list of states each corresponding to a room in the building. Therefore, if a building had fifty (50) rooms, the RL algorithm created 50 states. Multiplied by six (6) possible actions for north, south, east, west, up, down, this yields

a search space of 50x60 = 3000. This was a limited approach as it required a different state for each building that one wishes to simulate. Additionally, the number of states and number of actions dramatically affects the time needed to complete a simulation. That is, this was not scalable. Therefore, it would be preferable to limit the number of states. While the initial iteration of the algorithm worked normally and produced good results, we felt that it is did not truly reflect what an agent would be aware of during a fire evacuation. Specifically, we felt an environment-based state model, whilst obeying the Markov property [12], gave the agent universal knowledge of its environment, the temperatures in each room even on different floors or ones that are quite far from its own location. To remedy this limitation, we decided to experiment with an "agent-based" state model. Specifically, the states that the agent can be in are no longer the rooms, but the condition it finds itself in based on the time of the simulation and its location in the building. The agent examines the six immediately adjacent rooms to its own location and assigns a 0 if it is a room with an ambient temperature or a 1 otherwise. This means that the agent can be in one of seven (7) states indicated by the range [0..7]. At each time step, the agent is only aware of its immediate surroundings. This is a reasonable assumption as a person in a real-life situation would be able to assess the danger in immediately adjacent rooms. The hypothesis is that the reinforcement learning algorithm will eventually allow the agent to learn to associate a lower state number with a higher reward and the agent will take actions that it predicts will have the highest probability to yield a state with a number as close to zero (0) as possible.

Experimental results

The learning curves for the two experiments (environment-based state model and agent-based state model) for the cases of a CellComplex with one or three exits are plotted in Figures 4 and 5. Training performance was monitored at different intervals during learning, so Figure 4 presents the average reward per 300 episodes and Figure 5 the average reward per 500 episodes.

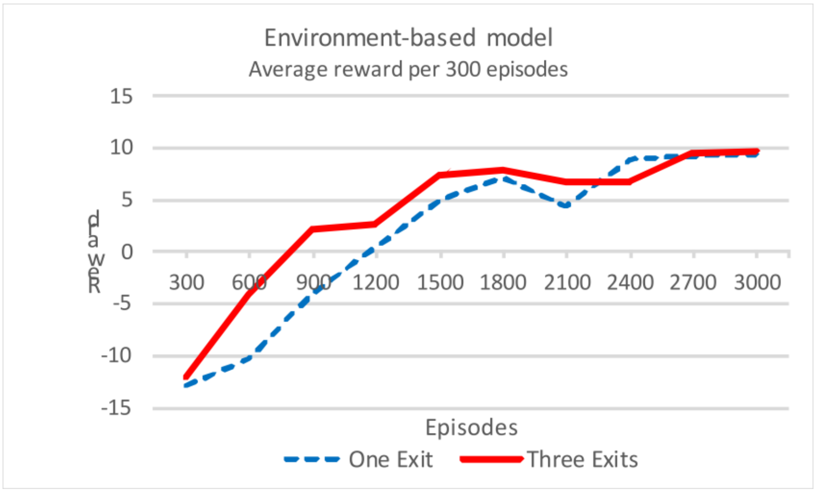


Figure 4. Average rewards per number of episodes for the environment-based state model

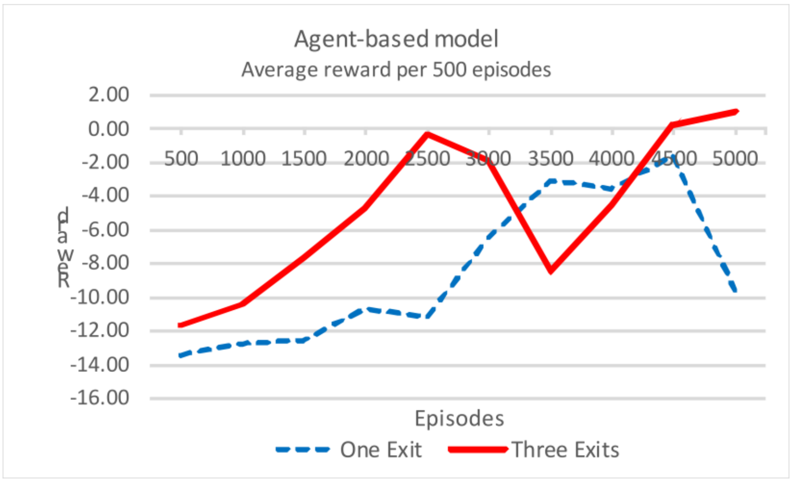


Figure 5. Average rewards per number of episodes for the agent-based state model

In both experiments the agent was able to find an exit, i.e. the goal, and the graphs in Figures 4 and 5 show that three exits were found easier than one exit. The graph on Figure 4 shows how the environment-based state model better performed as it obeyed the Markov property [12]. In the three exits case the agent achieved higher rewards compared to the one exit case in both experiments, achieving higher performance overall (about 43% higher in the agent-based model and 115% in the environment-based model, compared to the one exit case). In the first experiment, the highest average score per 500 episodes it got in the three exits case was 1.05 and in the one exit case -1.59, while in the second experiment the highest average score per 300 episodes it got in the three exits case was 9.63 and in the one exit case 9.40. As can be seen in Figure 6 as well, in the environment-based experiment, the agent gets to the nearest exit in the three exits scenario much quicker with lower overall temperatures (i.e. before the fire spreads). Moreover, Figure 7 presents the results for one and three exits for the environment-based model at episode 500. Even this early in the training, the one with three exits (b) finds the exit. The path is not optimal, but succeeds to find the exit nevertheless. The one with the one exit (a) fails miserably. The execution time for one exit was 50 seconds on average, while for the three exits was 32 seconds.

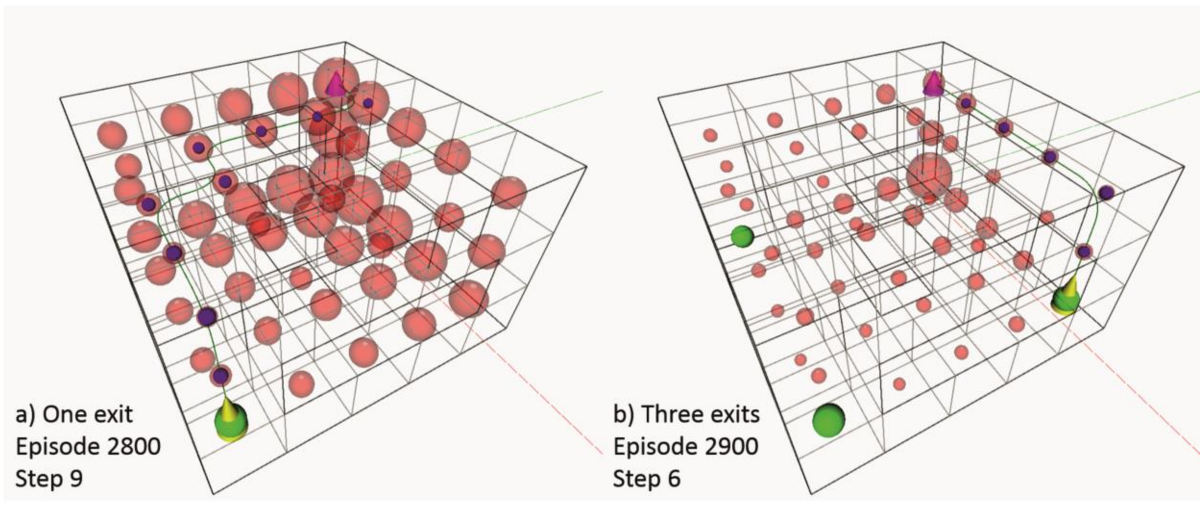


Figure 6. Screenshot of the environment-based state model experiment at later episodes: a) one exit, b) three exits

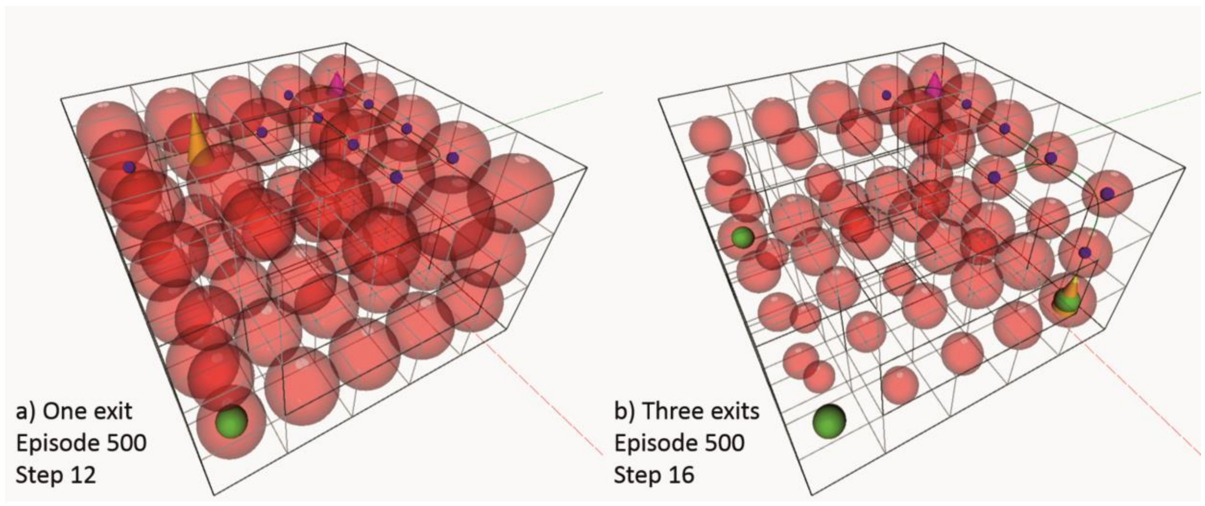


Figure 7. Screenshot of the environment-based state model experiment at early episodes: a) one exit, b) three exits

Although the environment-based state model yielded better results, we will continue to explore agent-based models since environment-based state models are not scalable. Further investigation into the appropriate number of episodes, number of steps or reward system is warranted for the agent-based experiment. These experiments present preliminary results, demonstrating what we already know intuitively that the inclusion of three exits rather than one is preferable. Further calibration of the model could provide more valuable insights into the safe and effective design of buildings at early stages.

Limitations and future work

This paper presented a proof-of-concept experiment. The inputs, e.g. for temperature and fire spread, are mainly based on abstract figures, with no true physics being incorporated. No material properties for the building elements, environmental factors such as humidity and CO_2 levels of cells (rooms) or human factors, e.g. disabled or elderly or blind people, have been taken into consideration. The fire spread model can be enhanced by linking it to a more sophisticated and pre-computed fire growth model, by including true physics to model the effect of buoyancy by giving different update rules for vertical or horizontal fire spread. Similarly, the agent speed of travel and decision making can be modelled after real observations of how humans behave in cases of emergency. While there was no real limitation on the complexity of the building, we chose a basic design including similar cubical cells in the interest of clarity and for the reduction of simulation time. In addition, as no validation method has been pursued, it is unknown if the number of steps and episodes are the most appropriate for the specific experiment. Moreover, the ratio between negative and positive rewards needs further investigation and could contribute to the improvement of the agent's learning.

While working within a limited game-like environment is useful to establish methodologies and investigate fundamental issues, ultimately, we are interested to know if the system can be applied to real-world data. To that end, we have started an investigation of how Q-Learning on graphs performs using a realistic building. We chose the "Sample Architectural Project" BIM model that comes bundled with Autodesk Revit as an example. We converted the BIM model into a Topologic model using custom scripts and derived its dual graph and vertices which we then used for the same Q-Learning algorithm as in the previous experiments. While we are not ready to share the results as this is an on-going investigation, we can report that we are witnessing that the agent learns to avoid the nodes that have high temperatures and seeks 'safer' routes to the exits (Figure 8). It is important to note that this is a very preliminary finding and further investigation is needed.

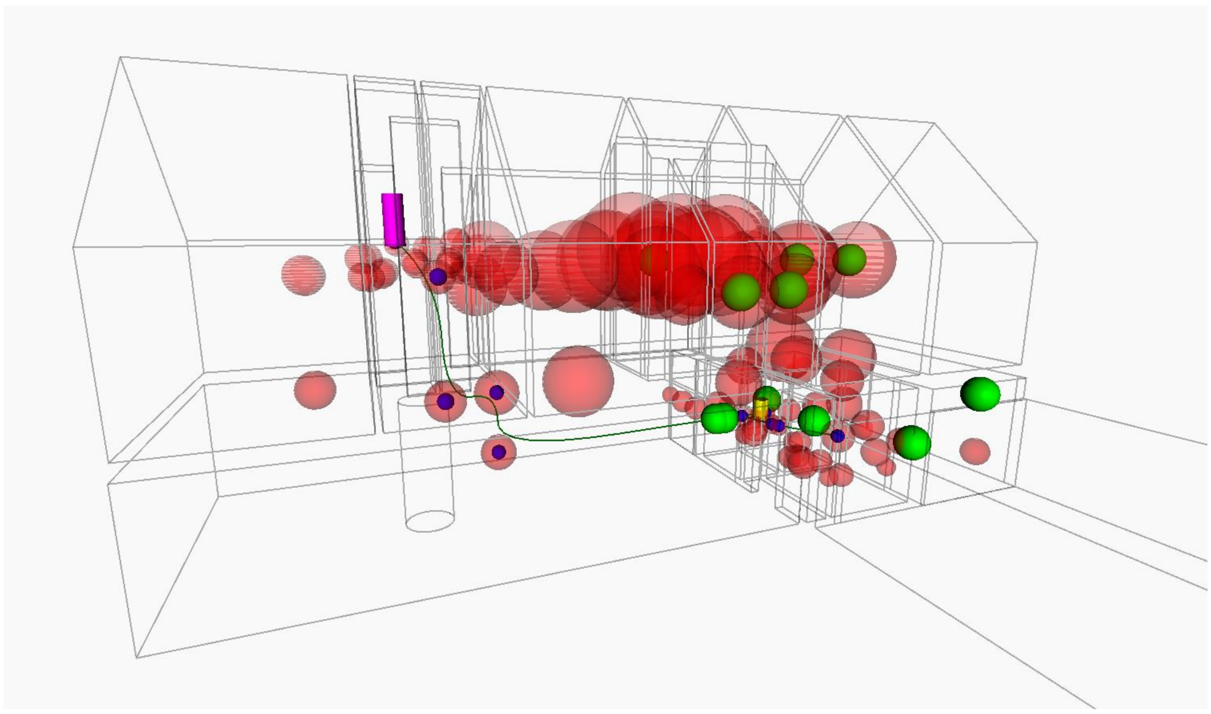


Figure 8. Initial results indicating that in later episodes, the agent avoids dangerous nodes and finds safer routes to its goals (indicated in green spheres).

Conclusions

This paper demonstrated a proof-of-concept about the application of reinforcement learning on graphs for fire egress situations, through the use of a non-manifold topology library, called Topologic. The experiment showed that the synergy of RL and NMT can provide informed decisions about the number and location of fire exits in buildings, contributing to safer environments. Future plans include a deeper analysis on the effective synergy of NMT and ML for spatial design in conjunction with energy analysis optimisation and BIM-based design development. Further research into this area could include higher complexity, varied geometry, multiple agents, multiple fires and different number and locations for exits. Further parameters could also be included to enhance the experiment and expand the application areas. These could be the use of different building types, shapes, internal layouts, windows and doors (Topologic apertures), different materials and internal firebreaks. The incorporation of fire and human data and characteristics could further improve the proposed model and make it both more realistic and user-centred. A sensitivity analysis in order to define the appropriate number of steps and episodes as well as reward model could also add to the refinement of the proposed experiment.

Acknowledgments

This project is funded by a Leverhulme Trust Research Project Grant (Grant No. RPG-2016-016).

References

Aish, R, Jabi, W, Lannon, S, Wardhana, NM and Chatzivasileiadi, A 2018 ’Topologic: tools to explore architectural topology’, Proceedings of eCAADe 2018

Aish, R and Pratap, A 2012 ’Spatial information modeling of buildings using non-manifold topology with ASM and DesignScript’, Advances in Architectural Geometry 2012

As, I, Pal, S and Basu, P 2018, ’Artificial intelligence in architecture: Generating conceptual design via deep learning’, International Journal of Architectural Computing, 16 (4), p. 306–327

Atila, U, Ortakci, Y, Ozacar, K, Demiral, E and Rakip Karas, I 2018, ’SmartEscape: a mobile smart individual fire evacuation system based on 3D spatial model’, International Journal of Geo-Information, 7 (6), p. 223–242

Baum, HR 2011, ’Simulating fire effects on complex building structures’, Mechanics Research Communications, 38 (1), p. 1–11

Blattner, A, Vasilev, Y and Harriehausen-Mühlbauer, B 2015, ’Mobile Indoor Navigation Assistance for Mobility Impaired People’, Procedia Manuf., 3, p. 51–58

Block, T and Sacks, R 2018, ’Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models’, Automation in Construction, 91, p. 256–272

Bong, FNP 2000, Fire spread on exterior walls, Master’s Thesis, University of Canterbury

Chatzivasileiadi, A, Lannon, S, Jabi, W, Wardhana, NM and Aish, R 2018 ’Addressing pathways to energy modelling through non-manifold topology’, SimAUD 2018

Harrington, SS 1996, Timtex: A hydraulic flow model for emergency egress, MS Department of Fire Protection Engineering, University of Maryland

Inoue, Y, Sashima, A, Ikeda, T and Kurumatani, K 2008 ’Indoor emergency evacuation service on autonomous navigation system using mobile phone’, Proceedings of the Second International Symposium on Universal Communication

Jabi, W 2015 ’The potential of non-manifold topology in the early design stages’, Proceedings of ACADIA 2015

Jabi, W 2016, ’Linking design and simulation using non-manifold topology’, Architectural Science Review, 59, pp. 323-334

Jabi, W, Aish, R, Lannon, S, Chatzivasileiadi, A and Wardhana, NM 2018 ’Topologic: enhancing the representation of space in 3D modelling environments through non-manifold topology’, Proceedings of eCAADe 2018

Jabi, W, Soe, S, Theobald, P, Aish, R and Lannon, S 2017, ’Enhancing parametric design through non-manifold topology’, Design Studies, 52, pp. 96-114

Khalifa, I, El Kamel, A and Barfety, B 2010, ’Real time indoor intelligent navigation system inside hypermarkets’, IFAC Proceedings, 43 (8), p. 461–466

Knight, B 1990, ’A knowledge-based system to represent spatial reasoning for fire modelling’, Engineering Applications of Artificial Intelligence, 12 (2), p. 213–219

Koo, B and Shin, B 2018, ’Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure Building Information Models’, Journal of Computational Design and Engineering, 5 (4), p. 391–400

Krijnen, T and Tamke, M 2015, ’Assessing implicit knowledge in BIM models with machine learning’, in Thomsen, M and et al., . (eds) 2015, Modelling Behaviour, Springer International Publishing

Lee, J 2007, ’A three-dimensional navigable data model to support emergency response in microspatial built-environments’, Annals of the Association of American Geographers, 97, p. 512–529

Lee, J and Zlatanova, S 2008, ’A 3D data model and topological analyses for emergency response in urban areas’, in Aplin, P (eds) 2008, Geospatial information technology for emergency response, Taylor and Francis

Luo, M and Beck, V 1994, ’The fire environment in a multiroom building— comparison of predicted and experimental results’, Fire Safety Journal, 23 (4), pp. 413-438

Meijers, M, Zlatanova, S and Pfeifer, N 2005 ’3D geoinformation indoors: Structuring for evacuation’, Proceedings of the 1st International Workshop on Next Generation 3D City Models

Ozel, F 1992, ’Simulation modeling of human behaviour in buildings’, Simulation: Transactions of The Society for Modeling and Simulation International, 58 (6), p. 377–384

Sutton, R and Barton, A 2018, Reinforcement Learning: an introduction, MIT Press Thombre, P 2018, Multi-objective path finding using reinforcement learning, Master’s Thesis, San Jose State University

Tsirmpas, C, Rompas, A, Fokou, O and Koutsouris, D 2015, ’An indoor navigation system for visually impaired and elderly people based on Radio Frequency Identification (RFID)’, Information Sciences, 320, pp. 288-305

Wardhana, NM, Chatzivasileiadi, a, Jabi, W, Aish, R and Lannon, S 2018, ’Bespoke geometric glazing design for building energy performance analysis’, FME Transactions, 47, p. 370–378

[1] https://topologic.app/

[2] http://www.maths-in-industry.org/miis/727/1/ESGI91-AWE_CaseStudy.pdf

[3] http://www.vtt.fi/

[4] https://www.nist.gov/file/67031

[5] http://www.iklimnet.com/hotelfires/fire_egress_software_10.html

[6] http://www.iklimnet.com/hotelfires/fire_egress_software_14.html

[7] https://www.steps.mottmac.com/

[8] https://www.iesve.com/software/virtual-environment/module/Simulex/480

[9] https://www.bbc.co.uk/news/uk-40301289

[10] www.deeplizard.com

[11] https://www.quora.com/What-is-the-highest-temperature-a-human-being-can-survive

[12] https://en.wikipedia.org/wiki/Markov_property