PURSUIT-EVASION GAME MODELLING IN A GRAPH USING PETRI NETS

Adel Djellal[1], Hichem Mayache[1] and Rabah Lakel[2]

[1] EEA Department, National Higher School of Engineering and Technology, 23005 Annaba, Algeria
[2] Department of Electronics, Badji Mokhtar University, Annaba, Algeria

ABSTRACT

This paper explores Pursuit-Evasion scenarios within game theory, focusing on environments with finite spaces, a limited number of pursuers, and evaders. It introduces a novel approach employing Petri nets to model Pursuit-Evasion in a bidirectional graph. Each area of the environment is represented by a set of places in the Petri net, capturing its specific behavior. Petri nets, known for their efficacy in modeling finite state spaces, are utilized to combine sub-nets representing individual areas. The detailed modeling provides a structured and formal representation of Pursuit-Evasion dynamics, offering a powerful tool to validate search strategies in graph-based pursuit-evasion models and contributing to the broader understanding of game theory and system modeling.

KEYWORDS

Pursuit-Evasion; Petri Net; Graph Theory

1. INTRODUCTION

In game theory, the pursuit-evasion problem is conceptualized as a game with three key components: players (cleaners and intruders), actions (movements of antagonists in the environment), and a cost function encompassing factors such as the time taken for agents to neutralize intruders and the required number of agents [1–13]. The game involves players with distinct means of perception, granting advantages to intruders in certain scenarios, such as enhanced knowledge of the environment, precise awareness of cleaning agents' positions, and information on their travel strategies.

The objective for participants is to devise policies optimizing their cost functions. Robot agents aim to minimize the execution time and the number of agents engaged, while intruders strive to remain undiscovered and out of sight from cleaning agents for as long as possible. This study utilizes a classification diagram [2], characterizing the environment as a Discrete Finite Graph with Heterogeneous or Homogeneous searchers, Constrained motion with Transit Costs, Perfect Detection with Line-of-Sight sensor models, Multiple targets, and Stationary Target motion with Random Placement. The Pursuit-Evasion problem in a graph, initially formulated by [14], is classified into known environments [6] and unknown environments [3, 10, 4], requiring distinct algorithms and techniques due to visibility graph dynamics in unknown environments.

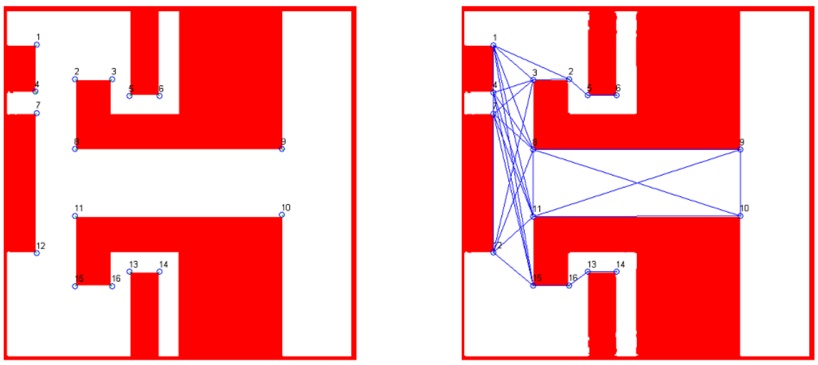
This paper is organized as follows. The environment model is presented in Section 2, Section 3 describes the offline search techniques. Section 4 presents model analysis using Petri net, conclusion is done.

2. ENVIRONMENT MODELLING

2.1. Visibility Graph

The chosen simulation environments consist of public spaces, comprising corridors, rooms, or offices, as depicted in Figure 1. The visual representation illustrates accessible areas in white, while non-accessible spaces or obstacles are highlighted in dark shading. Subsequently, the environment is modeled using a graph.

In our simulation application, environments are stored as images, and a straightforward image processing approach [15] is employed to extract outlines. Utilizing the vertex definition from [15], which considers a vertex as the intersection of two walls forming an angle greater than a certain threshold, we generate a set of nodes for the graph, as illustrated in Figure 1(a). By establishing pairs of nodes and employing a visibility definition inspired by [15] (where an edge signifies the line of sight between two vertices), the resulting visibility graph is presented in Figure 1(b). This graph serves as the topological model for the test environments, and it plays a pivotal role in this work, serving as the foundation for all search strategies developed.

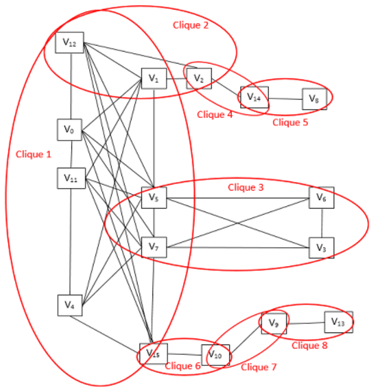


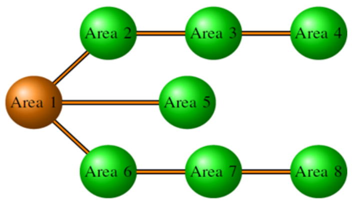
(A) Environment with Vertices (B) Environment with visibility graph

Figure 1. Generating visibility graph using building image

2.2. Area Graph

This stage involves extracting key information from the visibility graph, including the degree of each vertex (indicating the number of adjacent vertices), identifying complete subgraphs referred to as cliques or zones, establishing the connected area graph, and pinpointing critical areas. The algorithm employed for this purpose is thoroughly outlined in [15].



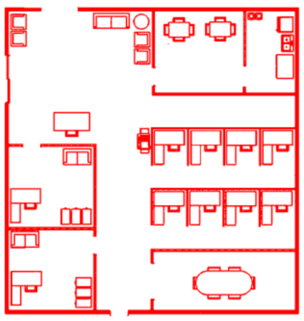


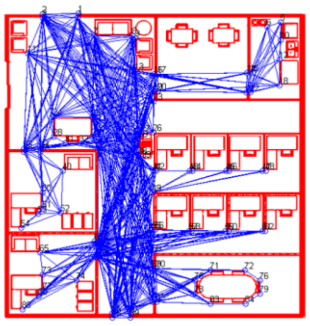
(A) Visibility graph converted into set of areas (B) Obtained area graph

Figure 2. Generating area graph using visibility graph

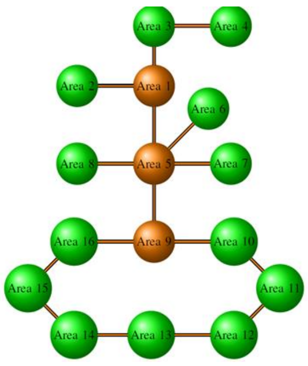
In Figure 2, we obtain a new graph, called Area Graph, whose vertices are the cliques and the edges represent the connection between areas. It is translated by vertices belonging to two adjacent cliques. For example, the edge joining $zone_1$ to $zone_2$ is represented by the vertices common to $clique_1$ and $clique_2$: $clique_1 \cap clique_2 = \{V_1, V_2\}$.

Figure 3 shows a real building with its decomposition into visibility graph, and the area graph is generated.





(A) Environment E (B) Visibility graph



(C) Area graph

Figure 3. Real building

2.3. Degree of an Area

The degree of a zone, akin to that of a vertex, corresponds to the number of connected adjacent zones.

- A zone with a degree of 1 is termed a leaf, representing a terminal zone.
- A zone with a degree of 2 is part of a path and is specifically known as a corridor, serving as a passage zone.
- A zone with a degree exceeding 2 is categorized as a critical zone, indicating its connection to more than two zones or paths.

3. PETRI NET MODELLING

Each zone can exist in one of two states: either the contaminated state or the decontaminated state. When a zone is in the decontaminated state, it requires protection either from an agent stationed directly on the zone or from agents positioned on the converging paths. It is crucial to note that an unguarded, decontaminated area is susceptible to recontamination, highlighting the importance of continuous monitoring and protection measures.

Area A_i has Boolean attributes: $agentA_i?$ / $guardedA_i?$ / $contamA_i?$ And $decontamAllA_i?$
The algorithm modelling the evolution in the change of state of a zone is presented in Algorithm 1.

Algorithm 1 State of Area A_i.

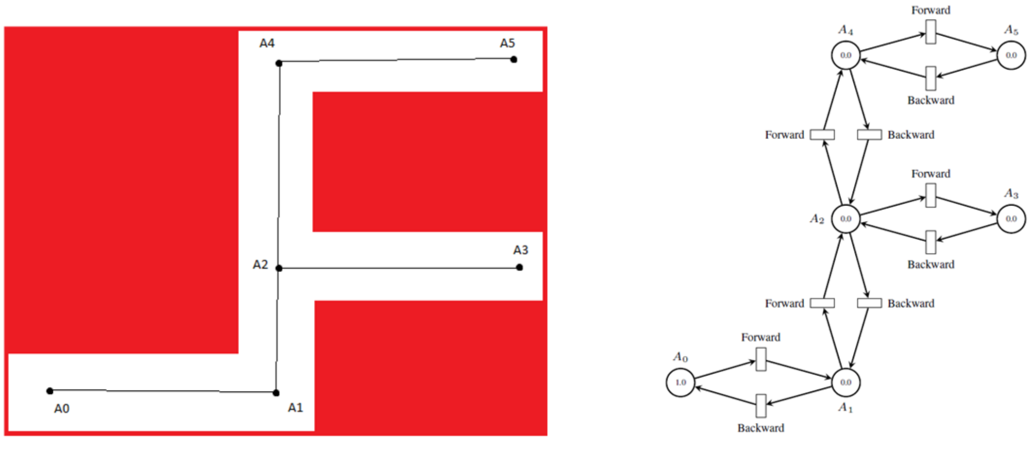
```
Area $A_i$ is initially $!guardedA_i?$, $contamA_i?$ and $!decontamAllA_i?$
if $agentA_i?$ then
    $guardedA_i?$
end if
if $guardedA_i?$ then
    $!contamA_i?$
end if
if $!agentA_i?$ and $\sum_j AgentA_j?^1$ then
    $guardedA_i?$
end if
if $\prod_j !contamA_j?^2$ then
    $decontamAllA_i?$
end if
if $decontamAllA_i?$ then
    $!contamA_i?$
end if
```

To transform the overall behavior of the system into a Petri net, it is necessary to break down each part of the algorithm into independent sub-nets. The comprehensive Petri net is then constructed by amalgamating all these individual sub-nets. This modular approach allows for a more structured representation of the system's dynamics, where each sub-net encapsulates a specific aspect of the algorithm, contributing to the holistic Petri net model.

3.1. Area Petri Net

The environment is translated into a Petri net, where places represent distinct areas, transitions correspond to arcs, and signify the movements of guardians between adjacent areas (refer to

Figure 4). In this representation, a token located in place Ai signifies the presence of an agent in that specific area ($guardedA_i?$).



(A) Environnement with area graph (B) Equivalent Petri net

Figure 4. Environment with area graph and equivalent petri net

Every place within the Petri net, representing a zone in the graph, is, in fact, a macro place. The dynamic evolution of the markings in this macro place is governed by three distinct sub-networks of Petri: the first sub-net models the visitation and guarding of the zone by an agent, the second sub-net represents the zone's contaminated-decontaminated states, and the final sub-net of Petri models the propagation of contamination through adjacent areas. This modular approach facilitates a comprehensive representation of the various aspects influencing the behavior of each zone in the overall system.

3.2. Sub-Net for the Behavior *Guarded?*

The two places $guardedA_i?$ and $!guardedA_i?$ of the Petri subsystems in Figure 5 correspond to the two possible states in which a zone can be found. The zone is initially unguarded (presence of a token in the place $!guardedA_i?$). And the arrival of an agent on the zone (presence of a token in the place A_4) triggers the crossing of the transition $guaA_4?$ causing a change of the state of the zone. It becomes a guarded area (presence of a token in the place $guardedA_4?$), And it will remain so as long as there is an agent present in the area.

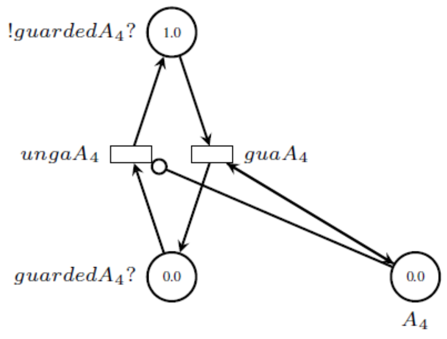


Figure 5. Behavior guarded?

3.3. Sub-Net for the Behavior *Contam?*

Both places $contamA_4?$ and $!contamA_4?$ correspond to one of two states: contaminated or decontaminated, representing the possible conditions of an area. Initially, the area is in a contaminated state, transitioning to a decontaminated state upon the visitation by an agent. This decontaminated state persists as long as the adjacent areas are maintained or transitioned to a decontaminated state, ensuring a sustained absence of contamination.

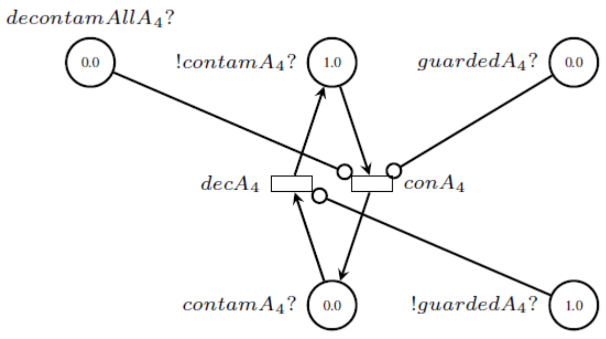


Figure 6. Behavior *contam?*

3.4. Sub-Net for the Behavior *Decontamall?*

The two places, *contamA4*? and !*contamA4*?, correspond to the contaminated and decontaminated states, respectively. This sub-net is designed to model the contamination of adjacent areas (refer to Figure 7). An area can only become contaminated if any of its neighboring areas is contaminated, as indicated by inhibiting arcs emanating from each *contamAi* place. If the entire zone is successfully decontaminated (*decontamAllA4*?), it remains in the decontaminated state (!*contamA4*?). An inhibitory arc extends from the *decontamAllA4*? place to the *conA4*? transition, ensuring the decontaminated state persists as long as all neighboring areas remain decontaminated. However, if any adjacent area becomes contaminated, the place !*decontamAllA4*? will hold a token, signifying the potential for *A4* to be contaminated again. This is facilitated by transitions (*T40* to *T43*), each connected to a neighboring zone *Ai*, featuring arcs from *decontamAllA4*? and *contamAi*? to !*decontamAllA4*? and *contamAi*?.

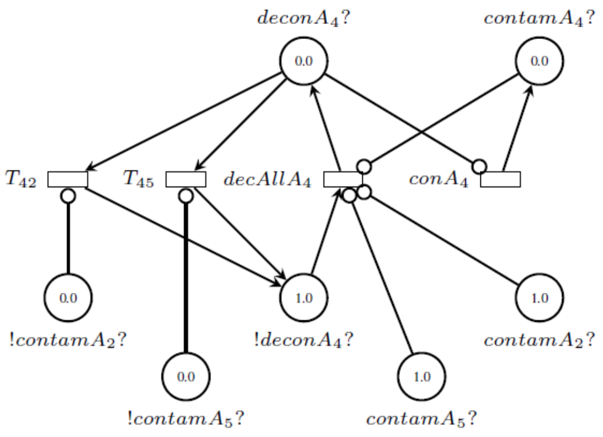


Figure 7. Behavior *Decontamall?*

3.5. Behavior's Combination

Integrating the three aforementioned sub-nets, which model the state of each area in the environment (as depicted in Figure 5(a)), yields the comprehensive Petri net representing the Pursuit-Evasion model. The resulting global net, illustrated in Figure 8, encapsulates the collective evolutionary behavior of various zones within the environment. This global Petri net serves as a holistic representation of the Pursuit-Evasion dynamics, capturing the interplay of agents, contamination states, and the propagation of actions across different zones.

4. MODEL ANALYSIS

4.1. Modelling tool PIPE

For the analysis of the proposed model, the PIPE tool was employed. PIPE is an open-source modeling and analysis tool utilizing Petri nets, specifically designed for creating and analyzing both immediate and stochastic Petri nets, including Generalized Stochastic Petri Nets (GSPNs) [18]. Implemented entirely in Java, PIPE ensures platform independence and provides a sleek and user-friendly graphical interface. This graphical tool facilitates the creation, backup, loading, and analysis of Petri nets. Originally developed at Imperial College London through various student projects, an enhanced version with significant improvements across different aspects has been further implemented at the University of the Balearic Islands.

4.2. Incidence Matrix W^+

The connecting arcs from Transitions to Places $Pre(P_i, T_j)$ can be represented in a matrix with $W^+(i,j) = Pre(P_i, T_j)$.

Then the index of the lines is the place and the index of the columns is the transition. The values of the matrix are:

$$W^+(i,j) = \begin{cases} 1 & \text{if there is arc from } T_j \text{ to } P_i \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

For the previous example (Figure 5(a)), the forward incidence matrix is a matrix of 42 rows and 50 columns.

4.3. Incidence Matrix W^-

The connection arcs from Places to Transitions $post(P_i, T_j)$ can be represented in a matrix with $W^-(i,j) = Post(P_i, T_j)$. Then the index of the rows is the place and the index of the columns is the transition.

The values of the matrix are

$$W^-(i,j) = \begin{cases} 1 & \text{if there is arc from } P_i \text{ to } T_j \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

In the previous example (Figure 5(a)), the back-incidence matrix is a matrix of 42 rows and 50 columns.

Inhibition matrix

Inhibition arcs from Places to Transitions $Inhib\left(P_i, T_j\right)$ can be represented as a matrix with $H(i,j) = Inhib\left(P_i, T_j\right)$.

Then the index of the rows is the place and the index of the columns is the transition. The values of the matrix are :

$$H(i,j) = \begin{cases} 1 & \text{if there is inhibition arc from } P_i \text{ to } T_j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

In the previous example (Figure 5(a)), the inhibition matrix is a matrix of 42 rows and 50 columns.

4.4. Initial Marking

The initial marking used in PIPE platform is done as a vector from Equation 4. This initial marking can be found in an intuitive way:

- Initially the robot(s) is(are) in one of the area places (here place A_4).
- All zones are initially:
  - not guarded ($!guardedA_i?$)
  - contaminated ($contamA_i?$)
  - not fully decontaminated ($!decontamAllA_i?$)

According to the model created in PIPE, the P-invariant subnets are:

$$\begin{aligned} M(!contamA_i?) + M(contamA_i?) &= 1 \\ M(!deconA_i?) + M(deconA_i?) &= 1 \\ M(!guardedA_i?) + M(guardedA_i?) &= 1 \\ \sum M(A_i) &= g \end{aligned} \quad (4)$$

With g is the number of needed guardians, and $i = 0 \cdots 5$ is the index of the Area A_i.

4.5. Discussion

The establishment of the initial marking in the PIPE platform follows a vectorized representation as per Equation 4. This initial marking can be intuitively derived through the combination of area sub-nets, where an area is characterized as either contaminated or uncontaminated, guarded or unguarded, fully decontaminated or not. In essence, this implies that:

$$contamA_i?.!contamA_i? = 0 \quad (5)$$
$$guardedA_i?.!guardedA_i? = 0 \quad (6)$$
$$decontamAllA_i?.!decontamAllA_i? = 0 \quad (7)$$

So that the pairs of places representing the previous states are P-invariant with a sum of tokens equal to 1, which verifies equation (4).

$$M(!contamA_i?) + M(contamA_i?) = 1 \quad (8)$$
$$M(!guardedA_i?) + M(guardedA_i?) = 1 \quad (9)$$

$$M(!decontamAllA_i?) + M(decontamAllA_i?) = 1 \quad (10)$$

Moreover, the Petri net modeling the zone graph (depicted in Figure 4(b)) operates autonomously, ensuring that its tokens will not be lost. However, tokens in this context represent guardians in the environment. Therefore, at any given time, the aggregate of tokens in this network equals 'n,' signifying the total number of agents present in the environment.

$$\sum_i M(A_i) = n \quad (11)$$

Boundedness

A marked network is deemed bounded when all its places are bounded. Networks classified as 1-bounded are referred to as safe networks. The network represented in Figure 5(a) is covered by positive T-invariants [15], indicating its potential for being both bounded and alive. Furthermore, the network is covered by positive P-invariants, reinforcing its status as a bounded network.

The size of global Petri net

In the example depicted in Figure 5(b), the network consists of 42 places and 52 transitions. By denoting as the number of zones and as the number of neighbors, it becomes feasible to infer details about the total number of places and transitions from the counts within each sub-net. In accordance with the findings in [15], the overall count of places and transitions is the summation of the places and transitions across the macros of each zone. Therefore,

$$\begin{cases} P = 7 \times n \\ T = 5 \times n + 4 \times nv \end{cases} \quad (12)$$

Applying the case of the environment in Figure 10(a) where $n = 6$ and $nv = 5$ then $P = 42$ and $T = 50$, which verifies the implementation on PIPE.

Reachability Graph

The reachability graph (R, M_0) is an oriented graph whose nodes are the markings of $A(R, M_0)$, and each arc links a mark to another, obtained by crossing a transition t_{ij}: then we have $M_i \xrightarrow{t_{ij}} M_j$. The Petri net representing the behavior of the environment is always bounded, so it always has a finite reachability graph. In the case of Figure 5(a), the graph contains 469 states and 669 arcs. It is obvious that its graphic representation remains difficult.

Nevertheless, for any method of decontamination, the markings of places A_i and $contamA_i?$, which are the most important to have an idea on decontamination behavior, can be represented in a two-column matrix; The initial marking of the places A_i is given by the left column vector, of which all the elements are null, except one and its value indicates the number of robots necessary for the decontamination. Theinitial marking of places $contamA_i?$ is placed in the right column vector and is equal to the unit vector. The final marking is reached, and the number of agents mobilized for the decontamination operation is validated, when the column vector relating to the $contamA_i$ places. becomes equal to the null vector.

Equation 13 shows the marking sequence for the Worst-Case technique applied on environment in Figure 5(a). We can see that the final marking must be as discussed before.

$$M_0 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_2} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} = M_f \tag{13}$$

With:

- $\sigma_1 = 2 \times Forward_{01} \rightarrow 2 \times Forward_{12}$
  $\sigma_2 = 1 \times Forward_{23/32} \rightarrow 1 \times Forward_{24} \rightarrow$
- $1 \times Forward_{45}$

Equation 14 illustrates the marking sequence for the Worst-Case technique applied to the environment depicted in Figure 3, while Equation 15 displays the marking for the improved technique. It is evident that the final marking aligns with the earlier discussion.

$$M_0 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 4 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_1} \begin{bmatrix} 4 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_2} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 3 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_4} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{14}$$

With:

- $\sigma_1 = 4 \times Forward_{43} \rightarrow 4 \times Forward_{31}$
  $\sigma_2 = 1 \times Forward_{12} \rightarrow 1 \times Forward_{21} \rightarrow 3 \times$
- $Forward_{15}$

  $\sigma_3 = 1 \times Forward_{56/65} \rightarrow 1 \times Forward_{57/75} \rightarrow$
- $1 \times Forward_{58/85} \rightarrow 1 \times Forward_{59}$

- $\sigma_4 = 1 \times Forward_{910} \cdots 1 \times Forward_{1516}$

$$M_0 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 2 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_2} \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_4} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_5} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_6} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{\sigma_7} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{15}$$

With:

- $\sigma_1 = 1 \times Forward_{43} \rightarrow 1 \times Forward_{31}$
- $\sigma_2 = 1 \times Forward_{43} \rightarrow 1 \times Forward_{31}$
- $\sigma_3 = 1 \times Forward_{12/21} \rightarrow 1 \times Forward_{15}$
- $\sigma_4 = 1 \times Forward_{15}$

- $\sigma_5 = 1 \times Forward_{56/65} \rightarrow 1 \times Forward_{57/75} \rightarrow 1 \times Forward_{58/85} \rightarrow 1 \times Forward_{59}$

- $\sigma_6 = 1 \times Forward_{59}$
- $\sigma_7 = 1 \times Forward_{910} \cdots 1 \times Forward_{15/16}$

4.6. Modeling Example Using Petri Net

Figure 14 shows and example of environment modelled with Petri net, the proposed model is for the environment presented in Figure 5(a). we can see that the model is complex even if the environment if simple, which is the main problem of this technique.

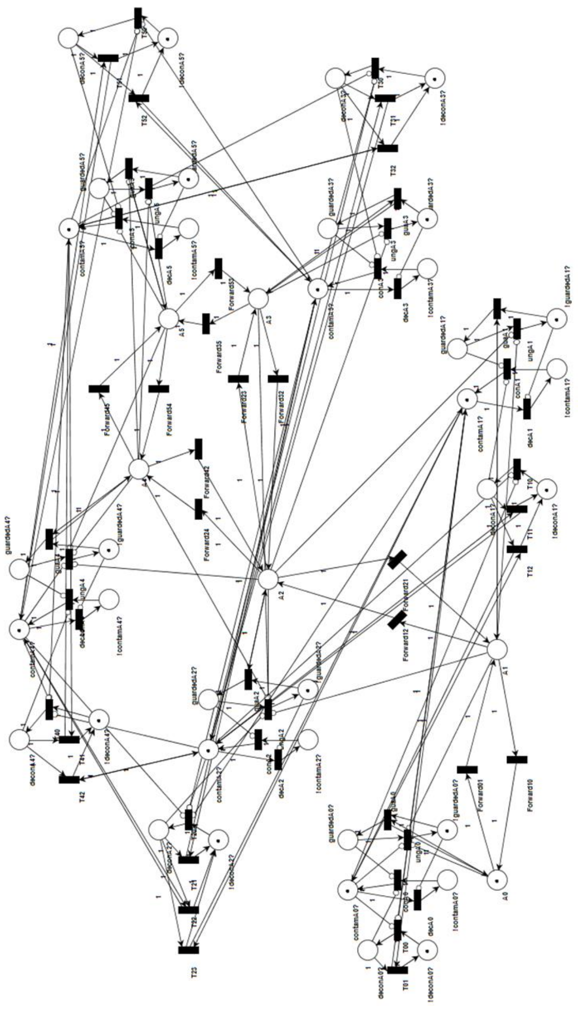


Figure 8. Petri net model for environment in figure 5(a)

5. CONCLUSIONS

This work elucidates the utilization of graphs for environment modeling and search methodologies in known environments. It introduces a Petri net model to validate research techniques applied to graphs. While the Worst-Case method proves highly efficient in known environments, it requires a constant mobilization of a substantial number of resource agents for environmental exploration.

The developed improved research approach dynamically optimizes the number of engaged agents. The validation through Petri nets affirms the effectiveness and robustness of these diverse research techniques. The implications of this work extend to the field of disaster management, with potential applications in dynamic environments, such as those involving the opening or closing of doors that alter the topological model. Future work can further validate navigation and decontamination techniques in 3-D environments closer to reality. Additionally, the study can be expanded by developing an interactive software platform for modeling and validating the various techniques presented in this work. These prospects open up promising avenues for advancing the practical applications and impact of the research.

REFERENCES

[1] A. A. Azamov, A. S. Kuchkarov, and A. G. Holboyev, "The pursuit-evasion game on the 1-skeleton graph of a regular polyhedron. ii."Automation and Remote Control, vol. 80, no. 1, pp. 164–170, 2019.

[2] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," Autonomous robots, vol. 31, no. 4, p. 299, November 2011.

[3] A. Djellal and R. Lakel, "Using graph search technique to solve the pursuit-evasion problem in unknown environment," in International Conference on Industrial Engineering and Manufacturing ICIEM'10, 2010.

[4] A. Djellal and R. Lakel, "Using graph theory to search an evader in unknown environments," in Conférence Internationale sur le Traitement de l'Information Multimédia (CITIM'2012), 2012.

[5] S. M. LaValle and J. E. Hinrichsen, "Visibility-based pursuit-evasion: the case of curved environments," in IEEE Int. Conf. Robot. & Autom., 1999.

[6] R. Lakel and A. Djellal, "Resolving the pursuit evasion problem in known environment using graph theory," Int. J. Bio-Inspired Computation, vol. 2, no. 6, pp. 434–439, 2010.

[7] L. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," International Journal of Computational Geometry and Applications, 1985.

[8] V. G. L. Mejia, F. L. Lewis, Y. Wan, E. N. Sanchez, and L. Fan, "Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors," IEEE Transactions on Automatic Control, 2019.

[9] A. V. Moll, D. W. Casbeer, E. Garcia, and D. Milutinovic, "Pursuitevasion of an evader by multiple pursuers," in International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2018.

[10] D. Pallier and H. Fiorino, "Coordinated exploration of unknown labyrinthine environments applied to the pursuit-evasion problem," AAMAS' 05, 2005.

[11] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani, "Finding an unpredicable target in a workspace with obstacles," in IEEE Int. Conf. Robot. & Autom., 1997.

[12] V. Sunkara, A. Chakravarthy, and D. Ghose, "Pursuit evasion games using collision cones," in AIAA Guidance, Navigation, and Control Conference, 2018.

[13] F. Yan, J. Jiang, K. Di, Y. Jiang, and Z. Hao, "Multiagent pursuit-evasion problem with the pursuers moving at uncertain speeds," Journal of Intelligent & Robotic Systems, pp. 1–17, 2019.

[14] T. D. Parsons, "Pursuit-evasion in a graph," Theory and Applications of Graphs, pp. 426–441, 1976.

[15] A. Djellal, "Modélisation et analyse des comportements d'une société d'agents mobiles `a l'aide des r´eseaux de petri," Ph.D. dissertation, Badji Mokhtar University – Annaba, 2016.

[16] MobotSoft, "Wall following robot "mobotsoft"," WordPress, Tech. Rep., 2016.

[17] P. Bonet, C. M. Llado, and R. Puigjaner, "Pipe v2.5: a petri net tool for performance modeling," in the 23rd Latin American Conference on Informatics (CLEI2007), 2007.

[18] M. M. Ajmone, Modelling with Generalized Stochastic Petri Nets. John Wiley & Sons inc., 1994.