International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Cellular Automata over Hexagonal and Triangular Tessellations for Path Planning

Jasmeena Tariq, A. Kumaravel

Abstract: ***Exploring unknown terrains demand for the sophisticated algorithms for path planning in case of multi-robot systems. Robots searching for common goals are complicated as they need to avoid collisions. We need to provide paths for all robots which should be collision free and should take less amount of time. Usually these algorithms use square tessellations in this type of investigations, which leads to the lack of variation and enlargement of neighborhood. In this paper we present the construction of a cellular automaton over triangular and hexagonal tessellations and show the complexity for the same.***

Index Terms: Cellular Automata, Complexity, Tessellations, Multi-Robot systems, Path planning

I. INTRODUCTION

So far, many different approaches have been introduced for path planning in multiple robots. The previous work [1] deals with RTS(real time strategy) games in which total time required for all agents to reach the goal is minimized by applying collision free strategy. This paper has used A* algorithm to solve the particular problem. Another significant work [2] deals with path planning algorithm of autonomous mobile robots through 'bug' algorithm. The bug algorithm makes use of minimal sensors in a robot and very simple methods are used.Since the constraint on mobile robots that have to move towards the same goal and all have same priorities, the method ought to be more challenging. Our paper deals with collision free path planning of multiple robots using cellular automata taking less time and travelling minimum distance over hexagonal and triangular tessellation. We have applied a slight modification of traditional A* algorithm adding some extended heuristics.

II. MODELLING THE PROBLEM

In order to plan a path for robots, to move in a geographical area, we make use of global path planning method. Here the obstacles are static and we have knowledge of distribution of free cells and obstacles. A central planner is designed for tracking the movements of all the concerned robots in the given environment. This helps us in planning a collision free path for multiple robots.

Manuscript published on November 30, 2019.
* Correspondence Author
Jasmeena Tariq*, research scholar at, Computer Applications, Bharath Institute of Higher Education and Research, Chennai.
A. Kumaravel, Professor and Dean, School of Computing, Bharath University, Chennai.

2.1. Framework

2.1.1. Triangular tessellation: We have studied papers related to path planning of multiple agents [1] where square tessellation is used to represent a geographical area. An entire new universe unfolds when we consider the grid (also called a *tessellation* or *tiling)* of equilateral triangles [3]. Here each cell has 12 touching neighbors: three on the edges and nine on the vertices as shown in Figure 1. For the sake of simplicity we assume that each movement through vertices costs 10 units of distance and each diagonal movement costs 14 units of distance.

Figure 1: Each cell in a triangular tessellation has 12 neighbors. There are two types of neighboring cells: cells which touch the vertices (represented as 10) and the cells which are connected diagonally (represented as 14), the robot cell is represented by 'a'.

2.1.2. Hexagonal tessellation: Hexagons are unique in a way that distance from its centre, of a particular cell, to any of its vertex is equal to the length of its each side. Hexagons can be described with Cartesian coordinates. Hexagonal tessellations have some very important advantages over square tessellation as hexagonal tessellation has equidistance of neighborhood, better representation of data, higher symmetry, most filters are somewhat circular (or at least directional) and their neighborhood also grows with some circularity, hexagonal filtering is computationally more efficient [4]. For simplicity we assume that each movement across cells costs 10 units of distance.

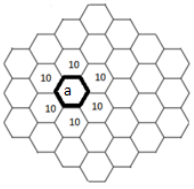

Figure 2: Each cell in a hexagonal tessellation has 6 neighbors. As the distance to any cell from the centre of a particular cell (here represented by 'a') is always same so we have represented its all neighbors by number 10.

In the above discussed two tessellations, a cell can contain any three of the following objects at a particular time:

1. Robots (represented by small letters)
2. Destination (represented by letter "D")
3. Barriers (represented by letter "B")

Number of robots is represented by 'n'. 'n' is less than the square of cells in a certain tessellation. A cell can be filled only when it contains a robot or it contains a barrier, else it is a free cell.

We assume that a given geographical area is represented by 'b' in both triangular as well as hexagonal tessellation. In a triangular tessellation of size b = w × N where as in hexagonal tessellation b = d × M, where, 'd' is the diagonal of any hexagonal cell. 'M' is maximum number of hexagonal cells horizontally covering the area of search and 'w' is the width of each triangle. 'N' is maximum number of triangular cells horizontally/ vertically covering the area of search.

Figure 3 and Figure 4 show the triangular environment and hexagonal environment with a certain configuration.

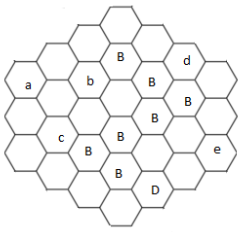

Figure 3: A hexagonal environment with a certain configuration

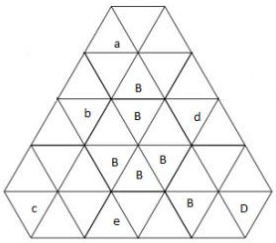

Figure 4: A triangular environment with a certain configuration

We define three sets of agents or robots as follows:

1. begin.
2. final.
3. filled.
4. waiting.

At the start all the robots belong to *begin* set. A robot is deleted from *begin* set only when it reaches the destination. After a robot reaches its destination, it is added in the *final* set. In *filled* set every robot fills one cell at a time; it contains the robots which are moving. The *waiting* set contains all the robots that are waiting for other ones to move ahead, it avoids any collision.

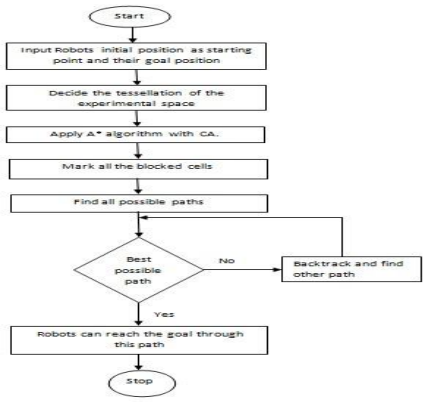

Figure 5: Flowchart of the proposed algorithm

2.2. Problem Formulation

Our central planner's objective is to lead all the robots towards goal by satisfying the constraints like less amount of time taken for the waiting and reaching the destination through shortest possible path without collision.

We assume, |*begin*|=n ('n' is the number of robots), 'F_t' is the time taken by all the robots such that |*final*|=n. For example we have a robot 'r' in any given environment, t_r defines the time step in which robot 'r' reaches its destination and d_r is the total length of path taken by robot 'r'.

$$F_t = t_1, t_2 \ldots t_r \ldots t_n.$$

We take some measure to calculate the total amount of time every robot (in this example 'r') spends in *waiting* set, and hence can compute its waiting time (in this example 'w_r').

The goal of central planner is to minimize $\sum_{r=1}^{n} t_r$, t_r includes the distance travelled by a robot 'r' (d_r) and waiting time of a robot 'r' (w_r), therefore, $t_r = d_r + w_r$. For multiple robots our algorithm makes a balance between'd', 'w' and't' of each robot, so that we can minimize F_t.

III. CELLULAR AUTOMATA

As already defined [5], Cellular automata (CA) over T are decentralized, discrete space-time systems that can be used to model physical systems [7, 8]. Cellular Automata can be formally defined as quadruple (N^n, Q, A, F), where:'N^n' is an 'n' dimensional working space(n=1, 2, 3).'Q' is a set of states, where Q={0,1,…,q-1},'A' is a set of positions of the neighbor cells, whose new state is being computed, A= {a_1,…,a_v}, a_i's are positions of the neighboring cells with respect to a particular cell where I can take any value from 1 to 'v'.

'f :- $Q^{|A|} \rightarrow Q$,' is the local function which gives the transition rule for each node.

A configuration 'c' is a function from N^n to Q and set of all configurations is $C=Q^{N^n}$, thus we can define functional (F) for a cellular automaton as:-

$\forall c \in C, \forall i \in N^n, F(c)(i) = f(c(i + a_1),..., c(i + a_v))$

IV. PROPOSED WORK

Here we focus on two types of tessellations namely triangular and hexagonal types. A cellular automaton, which calculates the minimum distance between each cell and destination dynamically, can be constructed as explained in the following sections.

When there are no further updates, our cellular automaton should terminate.

4.1 Triangular Tessellation

We assume that a given geographical area given by 'a' as in 2.1.

Figure 6: A sample geographical area with one robot (a) and destination(D) using triangular tessellation. Obstacles are represented by black triangles.

In Figure 5, we have shown only one robot with a destination, suppose we have to find minimum distance travelled by robot (a) towards the destination(D) , thus the transition rules are:

$$q^{t}_{x,y}=\begin{cases} 1 \quad \text{if} \quad q^{t}_{x,y}=1 \quad \text{(1 is used for obstacle)} \\ \min\{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{13}\} \quad \forall q^{t}_{x+k,y+m}\,; k=-1,0,1: \; m=-1,0,1; \end{cases}$$

where,

$p_1=q^{t}_{x,y}$
$p_2=q^{t}_{x,y+1}+10$
$p_3=q^{t}_{x-1,y-1}+14$
$p_4=q^{t}_{x-2,y+1}+14$
$p_5=q^{t}_{x-2,y}+14$
$p_6=q^{t}_{x-1,y}+10$
$p_7=q^{t}_{x-1,y-1}+14$
$p_8=q^{t}_{x,y-1}+14$
$p_9=q^{t}_{x+1,y-1}+14$
$p_{10}=q^{t}_{x+1,y}+10$
$p_{11}=q^{t}_{x+1,y+1}+14$
$p_{12}=q^{t}_{x+2,y+1}+14$
$p_{13}=q^{t}_{x+2,y}$

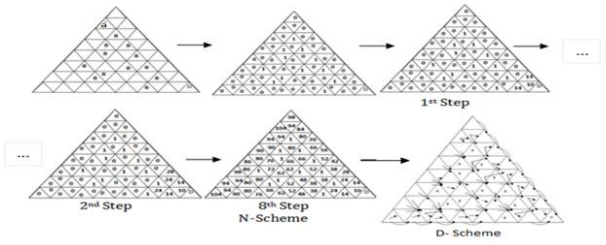

Figure 7: Dynamic Calculation of minimum distance between each cell using one robot in Cellular automata (Triangular tessellation)

4.2. Hexagonal Tessellation

As an example of path planning in cellular automata using hexagonal tessellation we use a static hexagonal tessellation given in figure below.

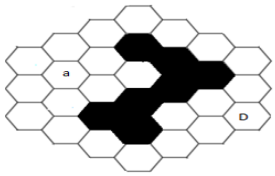

Figure 8: A sample geographical area with one robot (a) and destination (D) using hexagonal tessellation. Obstacles are represented by black hexagons.

The transition rules are

$$q^{t+1}_{x,y}=\begin{cases} 1 \quad \text{if } q^{t}_{x,y}=1 \quad \text{(1 is used to denote obstacle)} \\ \min\{p_1, p_2,\ldots, p_7\} \quad \forall q^{t}_{x+k,y+m}\,; k=-1,0,1: \; m=-1,0,1; \end{cases}$$

where,

$p1=q^{t}_{x,y}$
$p2=q^{t}_{x-1,y}+10$
$p3=q^{t}_{x,y-1}+10$
$p4=q^{t}_{x+1,y}+10$
$p5=q^{t}_{x-1,y+1}+10$
$p6=q^{t}_{x,y+1}+10$
$p7=q^{t}_{x+1,y+1}+10$

In Figure 8 the entire procedure has been illustrated.

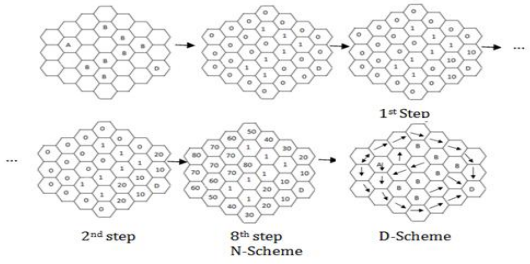

Figure 9: Dynamic Calculation of minimum distance between each cell using one robot in Cellular automata (Triangular tessellation)

In Figure 8 we have provided two types of schemes. N-Scheme contains minimum distance between each hexagonal cell and the destination. Now our central planner can take the shortest path towards destination from the start position of the robot, it is shown in D-Scheme which gives us the direction of this shortest path.

As shown in Figure 8 optimal direction is calculated by 7 comparisons, thus we can calculate D-Scheme in O(b).

In the given examples A* algorithm works efficiently, but for more than one robot A* algorithm tends to loose its efficiency [6]. Every robot can find its shortest path in θ(k) , where k=O(b) using N-Scheme, thus central planner can find the shortest path of all robots in O(b) where 'b' is determined by tessellations as mentioned in section 2.1. If we follow earlier methods, the central planner acts greedily by reducing distance only, without caring for collisions and waiting time.Thus, for more growing number of robots we use D-Scheme, where the central planner intelligently makes decisions based on the directions given in D-scheme. Now heuristics is used in order to get a shortest distance with less time taken (F_t) without collisions using D-scheme.

Central planner can decide what direction to take, in order to avoid collisions, easily based on our heuristics and hence we can solve this problem more intelligently. At the beginning of our algorithm all robots, with their current location, is added in Begin set. We have two different direction sets for hexagonal and triangular tessellation, based on their neighbors.

D_h={N, NE, NW, S, SE, SW} and

D_t={N, NW, NE, W, WW, E, EE, S, SSW, SW, SE, SSE}

where, D_h is the direction set for hexagonal tessellation and D_t is for triangular tessellation. Further these direction sets are shown in Figure 9 and Figure 10.

As per our heuristics, central planner decides directions for every robot, The *final cost* function can be defined as:

$$Fcost\ (x, y, d) = v\ (d) + V\ (x, y, d) + w_t \times c\ (x, y, d)$$

The parameters of '*Fcost*' are the location of current robot and a possible direction. This function returns a positive real number as the total time taken. Our central planner has to choose the directions with lower costs and no collisions or blocks for a robot in filled set in each time step. '*v'* determines the required units of time for a robot to get to a new cell. V returns the value of the neighbor cell in N-scheme, with respect to robot's location and given direction. Therefore for any robot having the cell position i, j we can say V (i, j, d) = N-scheme [i, j-1] (if the direction is NE in case of hexagonal tessellation) and V (i, j, d) = N-scheme [i-1, j] (if the direction is E in case of triangular tessellation), being the minimum distance from the cell towards destination. If these cells are already taken, c (i, j, d) returns an estimate of number of collisions.

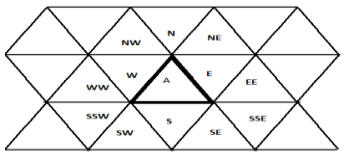

Figure 10: Direction marked in triangular tessellation

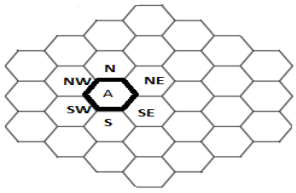

Figure 11: Direction marked in hexagonal tessellation

V. RESULTS

Estimated number of collisions given by 'c' helps the central planner to calculate waiting time (we must have an estimate of duration of time for a single step, represented by 'wt') of a robot. This time helps us in calculating estimated waiting time.

As the numbers of robots increase the value of 'w_t' also increases. Here we can say that for any robot 'r':

'V+v' $\longrightarrow$ d_r

'$w_t \times c$' $\longrightarrow$ w_r

For all robots, which are in *begin* set, our central planner minimizes 't_r' by taking certain decisions based on heuristics.

In worst case:

For triangular tessellation we have to calculate twelve probable paths and n-1 shortest path, for each robot in every time step.

For hexagonal tessellation we have to calculate six probable paths and n-1 shortest path, for each robot in every time step.

Thus the worst case time complexity is O(nb). If we bound possible number of steps by integer K, then the time complexity of our algorithm can be given by $O(Kn^2b)$. The value of K depends on the value of 'wt', and K<<nb for all values of 'wt'. Hence K=O(nb) and a bound for our algorithm happens to be $O(n^3b^2)$.

VI. CONCLUSION

In this paper we have a procedure to plan a path in a multi robot environment, without any collision, with a common goal. The algorithm that we introduced here is a slight modification of traditional A*. We created cellular automata for path planning of multi robots in an environment using two different tessellations and it was computationally bounded. We can use this algorithm in those multi- agent systems where agents have a common goal but fewer possible paths towards goal (giving rise to long queues) and hence the length of the queues can be reduced.

REFERENCES

1. Yashar Tavakoli, H. Haj Seyyed Javadi, Sepideh Adabi, "A Cellular Automata Based Algorithm for Path Planning in Multi-Agent Systems with A Common Goal", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.7, July 2008.
2. Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z., "A Simple Local Path Planning Algorithm for Autonomous Mobile Robots", INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING & DEVELOPMENT, Issue 2, Volume 5, 2011
3. Carter Bays, "Cellular Automata in the triangular Tessellation", Complex systems 8, 127- 150, 1994.
4. NUNO ALBUQUERQUE SOUSA, "Hexagonal Grid Image Processing, Algorithms in Neural Vision and Prediction", June 6, 2014.
5. B.Durand, E. Formenti, A. Grange and Z. Roka, "Number conserving cellular automata: new results on decidability and dynamics". Discrete Mathematics and Theoretical Computer Science, AB: 129-140, 2003.
6. R. Leigh, S.J.Louis, C.Miles, "Using a Genetic Algorithm to Explore A* -like pathfinding Algorithms", In Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007).
7. Kumaravel, K.Rangarajan, "Algorithm for Automaton Specification for Exploring Dynamic Labyrinths", Indian Journal of Science and Technology ,Volume 6, Issue 5S, 2013.
8. A. Kumaravel "An Application of Non-uniform Cellular Automata for Efficient Cryptography", Indian Journal of Science and Technology ,Volume 6, Issue 5S, 2013.

AUTHORS PROFILE

Jasmeena Tariq is a research scholar at, Computer Applications, Bharath Institute of Higher Education and Research, Chennai. Her research interests are Machine Learning, Image Processing, Data mining and Cellular Automata. You can email her at

Dr Kumaravel is working as a Professor and Dean, School of Computing, Bharath University, Chennai. His research interest includes Soft Computing, Cloud Computing, Machine Learning, Pervasive Computing and Knowledge Engineering. He is a life Member of ISTE and IET.