Dr. Shyama Prasad Mukherjee International Institute of Information Technology, Naya Raipur
(A Joint Initiative of Govt. of Chhattisgarh and NTPC)
Email: iiitnr@iiitnr.ac.in, Tel: (0771) 2474040, Web: www.iiitnr.ac.in

Coordination of Intelligent Agents Following a Pre-Defined Path

5th Semester Minor Project -1
Final Report

Submitted by,

Ashish Upadhyay, CSE, 15100007
Bhupendra Kumar Sonwani, CSE, 15100008
Vimal Anand Baghel, CSE, 15100037

Faculty Mentor,

Dr. Muneendra Ojha
Assistant Professor, CSE

Coordination of Intelligent Agents Following a Pre-Defined Path

Ashish Upadhyay
15100007

Bhupendra Kumar Sonwani
15100008

Vimal Anand Baghel
15100037

Department of Computer Science and Engineering
Dr. SPM International Institute of Information Technology – Naya Raipur (IIIT-NR)
C.G., India – 493 661

Abstract— Inspired by the "Robot Sheepdog Project" this paper presents a set of externally excited formation of autonomous agents by a sheepdog (extra intelligent agent) leading a flock (formation) in the environment to achieve a goal. It consists of two decentralized control levels, the lower level which is termed as sheep, maintains the formation and remains intact during the task while a higher level which is termed as the sheepdog, consists of finding a trajectory for both formation and itself. Sheepdog is also responsible for coordinating the formation motion along this computed path sequence. The sheepdog will use A-star (A*) search algorithm to compute the path between source and destination whereas the flock will use the modified version of Particle Swarm Optimization (PSO) algorithm to follow the sheepdog. We propose the new algorithm for multi-agent coordination and also compare our algorithm with the existing algorithms.

Keywords—Agent; Multi-agent environment; A*; PSO; Autonomous Agents;

I. Introduction

An agent is anything that percept its environment through its sensor and acts in the environment through its actuators in this sense we can say an agent is intelligent. In this project agent has the prior knowledge of the environment, therefore, the environment is known and fully-observable. An agent has simple agent function which is mathematically defined used by the program called agent program.

A multi-agent environment is a system of multiple intelligent agents interacting with each other. They use intelligence like functional, procedural approach or algorithmic search or reinforcement learning to solve the problems which cannot be solved or are difficult to solve by a single agent. In a multi-agent environment, all the agents are autonomous and have the capability to communicate with each other. They pass their information within themselves to maximize their performance.

With the evolution and advancement of "Artificial Intelligence," the topics like the mutual cooperative behaviour of multiple agents in a collaborative environment has gained popularity. Such an environment has many modern applications in scientific data gathering, surveillance, search and rescue operation as well as in neutralizing the enemy in a war zone.

In Artificial Intelligence, games and replicating the behaviour of living things are two hot topics that have always attracted the attention of research community towards them. The Robot Sheepdog Project is one of them. It implements the behaviour of a dog that gathers a flock of ducks in a circular area and guides them safely to a goal position by maintaining the flock formation. In this project, there is a robot acting like a dog, which tries to first gather all the ducks in the area at one place and form a flock. Then it pushes the flock towards the goal position.

The implementation of these types of problem uses a huge amount of computational resources in terms of space and time complexity.

In our project, we are modifying the idea presented by [2], to reduce their time complexity. The sheepdog will use A-star (A*) search algorithm to find the path from source to destination and at the same time the flock will use modified Particle Swarm Optimization (PSO) algorithm to follow the sheepdog.

The application of our project is to use this technique in neutralizing the enemies' battleships in war-zone. The sheepdog which is an extra intelligent agent which is autonomous and has the capability of image recognition and pathfinding in a static environment, it will find a collision-free trajectory for itself as well as for its flock. It acts as the leader of its flock. As soon as it finds the path between the source and goal it will communicate the message to its flock and ask them to reach the goal to bomb the target. In our case target is the battleship. The flock is a formation of multiple dumb agents known as sheep having the capability only to follow the path (in the form of a sequence of waypoints) given by sheepdog.

II. Literature Survey

In [1], the "Robot Sheepdog Project" which uses machine vision and robot building to present an idea of having a sheepdog robot that gathers a flock of ducks in a circular arena and pushed them safely to a predetermined goal. This idea was implemented in [2] by using the A* search algorithm to find the sequence of waypoints that should be followed by the sheep agents to reach the goal position. The idea of [2] was different from [1] in the only way that in [2] the sheepdog guides the sheep towards the goal rather than pushing them. In [3], three UAVs were repelled towards a goal position and then they form an equilateral triangle around the target. This methodology is used for reconnaissance of a close target. The paper has also implemented a technique of having dynamic obstacle which changes its position with the passage of time.

The formation collision-free, avoiding obstacles [2]. Coordination of autonomous multi-agent system is given in [4] which includes dynamic control strategies, formation control tasks and information flow between the agents in the formation. They can mimic the natural swarm of birds, fish and sheep. Particle swarm optimization and grey wolf algorithm are the examples of utilizing the flocking behaviour to achieve the goal. Formation shape dealing with the environment is important. Splitting and remerging the formation is the major issue in [4] in the avoidance of obstacle. All the constraint within the formation should satisfied. In [4] and [2] they defined the co-leader system for the formation and structure of the formation is defined with DOF (degree of freedom) which is at most three. Control laws for leading the cohesive motion [2] & [4] to the desired destination. First Law present the avoiding inter-agent collision and in second law conveys to avoid obstacle collision using virtual vector field approach.

In [5] the TAO-MTP algorithm is used by the hunter agent which uses Real-Time Adaptive A* (RTAA*) algorithm to capture the escaping prey. Here [5] hunter uses minimum pursuit cost, reducing the trajectory storage in the complete known environment. In [6] mobile agents are able to localize themselves in a plane where the inter-agent distance and angle subtended by each agent is known.

In surveillance method, our aim is to monitor the goal for a period of time and gather some information about it. In [7] & [8] an agent is designed to find a path between source and destination. The agent reaches the goal by circumnavigating the path and after determining the goal it keeps on rotating around it. These works present the idea of dual control problem in which we wish to find an unknown goal characteristics and solve the control problems simultaneously.

In circumnavigation using distance measurement method, the position of the destination point is unknown. In this only one agent is involved and the agent has only knowledge of the distance between its own position and the distance. In this problem, the main objective of the agent is to search and find the position of destination point using the given distance between agent's position and destination point. In this paper, there is dual control problem in which agent, the motive is to identify the unknown characteristics of the destination (system) and also gain the control objective. The agent described in this paper must follow the trajectory path to locate the target. In this paper mobile agent has to localize the unknown target using bearing measurement by circumnavigation. The mobile agent has only knowledge of the bearing angle of the target.

These days the development of UAVs is the field of high interest which gives limelight majorly on the autonomy of the vehicles another paper use the parallel comparison of particle swarm optimization algorithm and genetic algorithm.Both of algorithm gives the path to reach the goal in three-dimensional space. They use the concept of parallel programming. As we know that the path planner is the key element is the autonomous vehicle to reach the goal. In [9] first, they discussed the cost function and their feasibility criteria. The cost function is developed to evaluate the quality of candidate trajectories. The non-deterministic algorithm has been used to cope up with the complexity of UAV path planning. The environment and the trajectory representation is in the form of two-dimensional matrix separately where no-fly zones are declared with very high numeric values. The genetic algorithm is used in such a way that the path produced after the crossover having minimum cost will be taken with some extra functionality of addition, deletion and modification of the path. The steps include in this algorithm are initialize random population, compute their fitness function selection of parent to perform the crossover then mutant is developed, replacement is done, path smoothing is performed if the goal is reached then algorithm terminates and if not it goes on. These paths will replace the old generation of their parents having the larger cost function than them. In [9] PSO used to simulate the movement of the swarm in multi-dimensional search space converging towards the optimal solution. They found that genetic algorithm did better and PSO but in some scenario, PSO did significantly better than a genetic algorithm.

In [10], DMAPP was proposed for multi-agent coordination but was not complete. It sometimes failed to provide the solution even if the existing.
In [11] the improvement of the DMAPP algorithm which is based on the decoupled approach of Multi-agent path planning. This algorithm has three phases

- Individual path planned by the agents
- Decision making on priority
- Restructuring of plan

So, in [11] DiMPP algorithm is proposed which guaranty the finding solution. This paper uses SIC (sum of individual costs) as a cost function. This algorithm also works in three phases first two phases are same as the DMAPP. Firstly all agents compute their individual paths using A* algorithm. After computing the path the agents need to restructure their plan to check whether they are conflicting or not. It uses distributed approach to decide the priority of agents based on the individual plan length and the agent having longest plan length is given highest priority. In this phase, the main goal of the agents is to know about the path of the remaining agents. After completing this phase all agents have the knowledge of the priority of agents and path planned by the remaining agents. Now the restructuring of the plan is done by the agents, it starts from the highest priority agent and sequentially goes towards the lowest one. So this way the algorithm get completed with solution and time complexity $O(n^{2T})$.

III. Problem Definition

The aim of our project is to reduce the time complexity of the computation process in order to find the goal. If we use PSO for finding the path from source to goal, it will be in the form of O(nlogn) since the PSO algorithm depends on the number of particles. It sorts the current position of all the particles in the swarm and whichever is the best amongst them becomes the global best (gbest).

IV. Proposed Methodology

In an environment, a sheepdog agent will be finding its goal and directing its flock toward the goal. The sheepdog will use A* algorithm to find the trajectory from source to goal. The flock will use modified PSO algorithm to follow the sheepdog.

If flock would have used the PSO algorithm, then the time complexity will rise to O(nlgn) because the time complexity of PSO depends on the number of particles in swarm and number of iteration allowed for computation. If we keep the number of iteration and vary the number of particles, then the sorting used to find the global best will rise the complexity. If we use the best sorting algorithm, then also we could reach the limit of O(n logn).

But, in our proposed algorithm, we are replacing the idea of global best of PSO with the current position of a sheepdog. In this case, the time complexity will become independent of a number of particles and will only depend on the A*. Also, we have mitigated the extra overhead of sorting of flock's local best.

The two algorithms which we have used to make our modified algorithm are explained below.

- *A* search algorithm*

A* is most popular best first search which is widely used to find the shortest and optimal path between the source and destination. It solves the problem by searching all the possible path to find the goal. It expand the nodes by using cost function: f(n) = g(n) + h(n) where n is a node present in the path, g(n) is the actual cost to reach that node from the source and h(n) is the heuristic function, the estimated cost from that node to the goal. Implementation of A* algorithm is done by use of priority queue or heap. A* algorithm always searches optimal path.

- *Particle Swarm Optimization (PSO)*

Particle swarm optimization is iterative algorithm having the swarm of particle that moves around the search space according to particle function act as swarm intelligence to optimize the global objective value (gbest).This is also used in the navigation strategy for autonomous agents where the prior knowledge about the environment is lacking. For every particle, there is local optimizer (lbest) which is compared to global objective value if the local optimizer is greater than gbest then gbest take the value of lbest. Every particle is having velocity and position vector which change at every point in time. These particles can be treated as mobile agent sometimes called "seekers".

The algorithms which we have proposed for our work is given in figure 3.

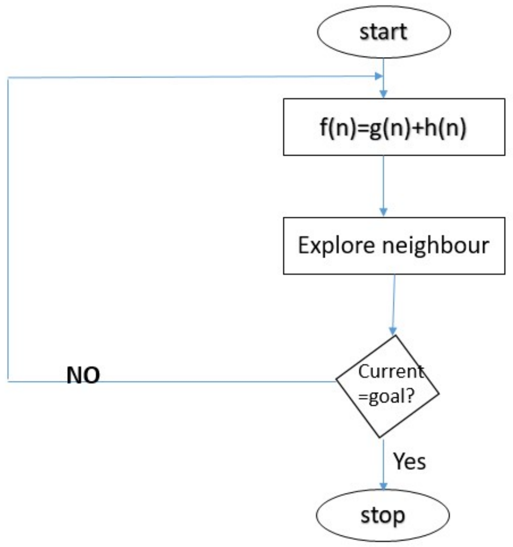


Figure 1: Block diagram for A-star algorithm.

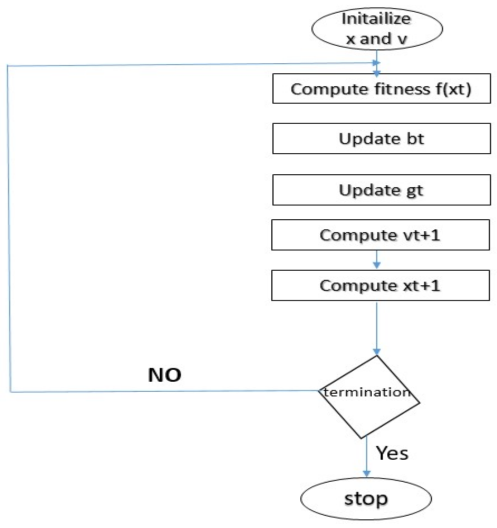


Figure 2: Block diagram for PSO algorithm

Procedure 1	A-star search algorithm with PSO() calling

```
function Astar(start, goal):
     return SUCCESS or FAILURE
     frontier <- Heap.new(initial)
     explored <- Set.new()
     while not in frontier.isEmpty():
          state <- frontier.deleteMin()
          explored.addstate)
          if goaTest(state)
               return SUCCESS(state)
          for neighbour in state.neighbours():
               if neighbour not in (frontier U explored):
                    frontier.insert(neighbour)
                    PSO(initial,current,bounds,num_particles)
               else if neighbour in frontier:
               frontier.decreaseKey(neighbour)
     return FAILURE
```

Procedure 2	Heuristic function used for A-star	*Procedure 3*	Testing the goal condition

```
function h(current,goal):
     (x1, y1) <- current
     (x2, y2) <- goal
     h<-a(x1-x2)+a(y1-y2)
     g <- g + 1
     f <- g + h
     return f
```

```
function gT(pos, goal):
     if (goal = pos):
          return True
     else
          return False
```

Procedure 4	PSO() used by flock to follow the sheepdog

```
function PSO(initial, SD_current, bounds, num_particles):
     Initialize the flock
     bounds <- limit of randomness
     SDcurrent <- current position of sheepdog
     initial <- initial position of flock
     gBest <- SD_current
     update_velocity(flock, SD_current)
     update_position(flock, SD_current)
     return swarm_position
```

Procedure 5	Update position procedure used for updating the position of flock

```
function update_position(flcok, SD_current):
     for i in range (0, num_dimensions):
          position <- position + velocity
     return position
```

Procedure 6	Update velocity procedure used for updating the velocity of flock
	```
	function update_velocity(flock, SD_current):
	     for i in (0, num_dimensions):
	          r1 <- random()
	          r2 <- random()
	          vel_cognitive = c1 * r1 *(best_pos - current_pos[i])
	          vel_flock = c2 * r2 * (SD_current - position[i])
	          velocity = vel_cognitive + vel_flock
	     return velocity
	```

Figure 3: This figure shows the algorithm proposed in our methodology. **Procedure 1** shows the starting of sheepdog for pathfinding using A-star search. Here PSO() for the flock is called in line number 13. **Procedure 2** shows the heuristic function used for astar search. **Procedure 3** shows the goal test function of astar search. **Procedure 4** shows the PSO() function used by flock to follow the sheepdog. **Procedure 5** shows how flock will update its position with respect to the current position of a sheepdog. **Procedure 6** shows how the flock will update its velocity with respect to the current position of a sheepdog.

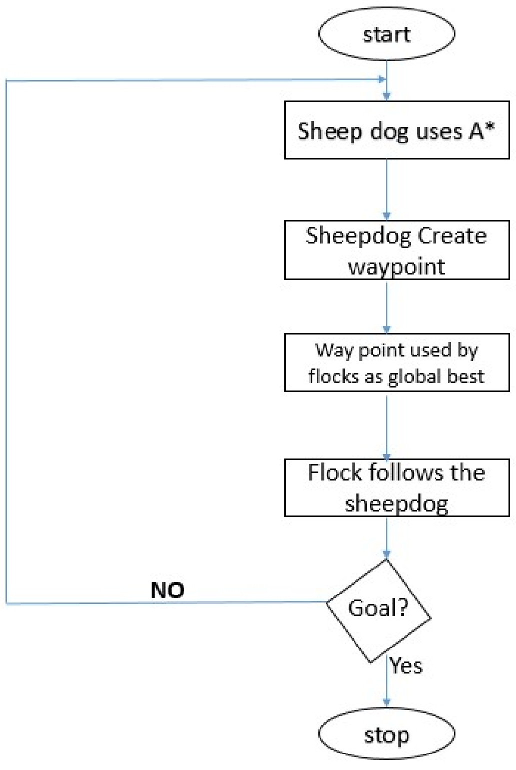

start
Sheep dog uses A*
Sheepdog Create waypoint
Way point used by flocks as global best
Flock follows the sheepdog
NO
Goal?
Yes
stop

Figure 4: Block diagram of our proposed algorithm

A short description of all the procedures used above is given here.

Procedure 1: Here sheepdog uses the A-star algorithm to find the path from source to destination. First of all, it creates a heap named frontier. This will store the unexplored neighbours and its first value is always

taken into consideration for exploration of goal state possibility. It will create a new set named explored which will contain the blocks which have been searched and has been found that they will not lead to the goal state.
Now, a loop will start and search until the end of the frontier. It will take the current position as the first value of frontier and will add it to explored. Now the current state will be checked whether it is a goal or not? If it is the goal then return success otherwise check for its neighbours. Now the neighbours will be checked f they are already in the explored set, if yes then do nothing otherwise, if in frontier then decrease the key. During this checking condition when the sheepdog is taking a step towards the goal, the flock will be called from its current position to follow the steps of sheepdog using PSO algorithm.

Procedure 2: The heuristic function will calculate the cost function of the path which is equal to the sum of shortest line distance (h_{eld}) and the cost to reach that path from the initial position.

Procedure 3: In procedure 3, the goal condition is being checked by the sheepdog. Here sheepdog will check if its current position is equal to the specified goal state or not? If yes return success else returns false.

Procedure 4: Particle Swarm Optimization (PSO) will first randomly initialize the position and velocity vector of the flock (swarm).The global best will take the values from the current position of the sheepdog, one thing is to be noted that no calculation of local best has been done because the flock has to move in the direction of sheepdog's current position vector the velocity vector also needs to be updated with it. As no particle has its local best, so it mitigates the problem of sorting automatically which is the main cause to reduce the time complexity and hence the performance of pathfinder algorithm increase significantly.

```
velocity = cognitive_velocity + SD_current_position + selfvelocity
          position = velocity + current_position
```

Procedure 6: Here, every particle will try to change its position by seeking the intelligence from flock's velocity and current position. This will help particle to move towards the goal.

Procedure 5: This procedure is being used for updating the velocity of the flock which is the part of PSO algorithm. In this part, flock's every particle will take three things into consideration in order to change their velocity.

First, the global best. In pure PSO algorithm, global best was decided by sorting the current position of every particle in the flock and whichever is best amongst them is termed as the global best. But in our algorithm, global best is replaced by the current position of the sheepdog in order to remove the time complexity of sorting the flock's current position.

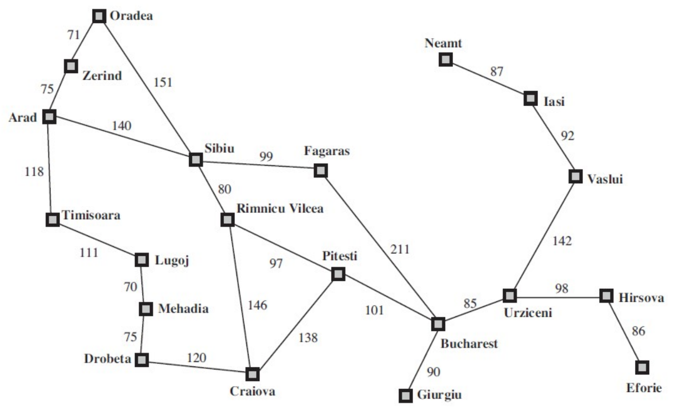
Second, the cognitive part. In this part, every particle will try to seek some intelligence from the positions of other particles in the swarm.
In the third part, the particle will consider its own intelligence and try to guess the best move towards the goal. All these three submodules will be combined in order to make the decision of velocity change of every particle of the flock.

V. EXPERIMENT

In our previous work, we had compared four traditional algorithms of pathfinding in terms of a number of iterations they take to find a path between source and destination. These algorithms are:

- A* Search algorithm (A*)
- Breadth First Graph Search (BFS)
- Breadth First Tree Search (BFS Tree)
- Uniform Cost Search (UCS)

The map used for calculating the distance:



<u>Figure 5</u>: Graph used for comparing four algorithms stated above.

The table shows the data collected for different goals but same source:

- Arad to Fagaras

Algorithm	Number of iterations
A*	14
BFS	11
BFS Tree	36
UCS	28

- Arad to Bucharest

Algorithm	Number of iterations
A*	24
BFS	25
BFS Tree	104
UCS	41

- Arad to Hirsova

Algorithm	Number of iterations
A*	41
BFS	45
BFS Tree	51
UCS	810

The number of iterations is directly proportional to the time complexity as well as space complexity. So, this data gives us a clear picture that we cannot use last two algorithms, i.e. BFS Tree search and UCS although UCS provides the optimal solution it takes so much time in computations.

Also, we know mathematically that A* gives the optimal solution but BFS does not. So, we can filter out the remaining three algorithms based on this conclusion and can use A* for pathfinding in our scenario.

We also compared our proposed algorithm with PSO by varying the number of particles in the flock. The table of comparison is given below:

- Time complexity

Number of sheep	Time Taken	
	PSO	A*-PSO
5	10	9
10	10	9
15	12	9
20	16	9
25	20	9
50	30	9

- Reached goal or not?

Number of sheep	Reached Goal	
	PSO	A*-PSO
5	No	Yes
10	No	Yes
15	Yes	Yes
20	Yes	Yes
25	Yes	Yes
50	Yes	Yes

Here, we can observe that in case of PSO as we increase the increase the number of particles, the time complexity of finding the solution increases. But in case of our proposed algorithm, the time complexity for solution finding remains constant no matter how much we vary the number of particles. Also, we can see that in case of PSO, if we keep the number of particles low from a limit, then we may decrease the time complexity but we will no be able to find the solution. But in case of our algorithm, the solution is guaranteed if exists.

VI. Results And Discussion

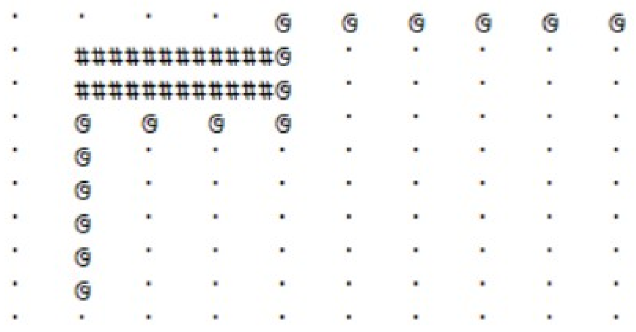
In our work, we tried two types of algorithm for multi-agent coordination. In the first phase, we used A-star search algorithm for both the sheepdog as well as a flock for pathfinding. For sheepdog, the goal was known and it was calculating its path based on that information. Whereas for the flock, the destination was unknown. So, they were trying to follow the sheepdog and the current position was sheepdog was acting as a goal for them.

In this methodology, the A-star was being used extensively and hence the space complexity was being increased. A-star algorithm stores every node which it has visited as well as the neighbours which it is going to visit. In that case, if we apply A-star search on so many particles then the space complexity will create a problem.

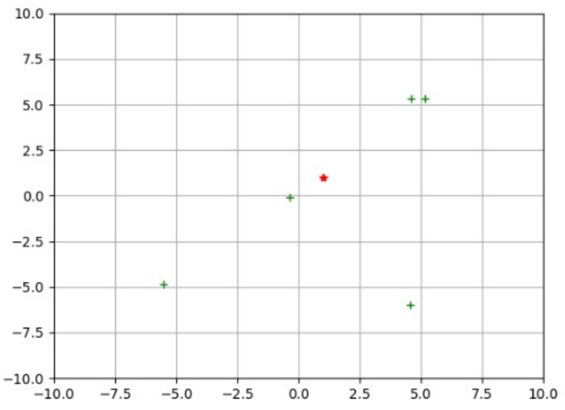
So, we used another algorithm that is Particle Swarm Optimization to derive the path for the flock. In this algorithm, we used A-star to find the path of sheepdog from source to destination and PSO is used by flock to follow the sheepdog.

Now, we know that PSO uses sorting technique to calculate the global best. That can increase the time complexity of the computation and hence we proposed a new method in which global best for PSO part will be replaced by the current position of the sheepdog and hence eliminating the sorting problem leading to the reduction of time complexity.

Some snapshots of our work are shown below:



```
.    .    .    .    @    @    @    @    @    @
.    #############@    .    .    .    .    .
.    #############@    .    .    .    .    .
.    @    @    @    @    .    .    .    .    .
.    @    .    .    .    .    .    .    .    .
.    @    .    .    .    .    .    .    .    .
.    @    .    .    .    .    .    .    .    .
.    @    .    .    .    .    .    .    .    .
.    @    .    .    .    .    .    .    .    .
.    .    .    .    .    .    .    .    .    .
```



<u>Figure 6</u>: This is the 10x10 grid environment for the sheepdog. # represents the obstacle in the path. @ represents the path traced by sheepdog using A* algorithm. Initial is (1, 1) and final is (9, 9).

<u>Figure 7</u>: In this figure, we can see the sheepdog is initially at (1, 1) position (denoted by red *) and sheep of the flock are randomly separated in the environment (sheep denoted by green +).

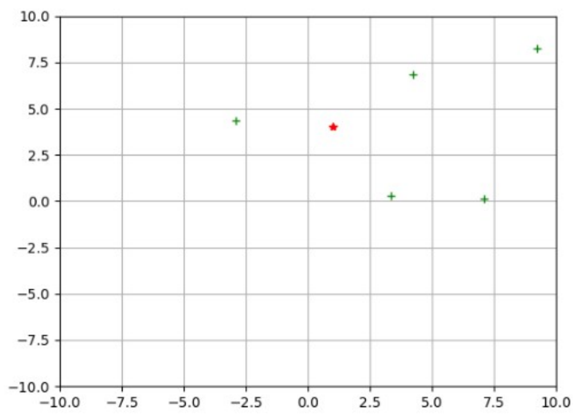


Figure 8: Here sheepdog has moved some steps towards the goal and hence the flock is also changing its position.

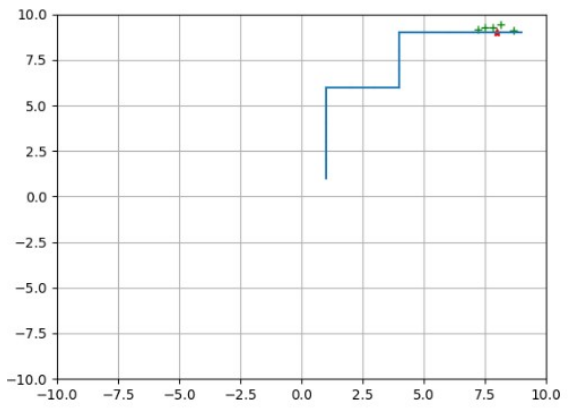


Figure 9: Here sheepdog has found the goal and reached to it. The path is indicated by a blue line. We can also see that the flock has also converged at the goal point.

VII. Future Works

This work has a limitation of the negligence of obstacle by the flock. The flock is not considering any obstacle in its path whereas the sheepdog does.

This work can be extended in future by implementing the robust flock coordination and their communication to tackle the problem of collision. In the 3-D model, the agent will have to avoid an obstacle in the vertical direction from the ground also.

VIII. References

[1] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, “Experiments in automatic flock control,” *Rob. Auton. Syst.*, vol. 31, no. 1, pp. 109–117, 2000.

[2] I. Shames, C. Yu, B. Fidan, and B. D. O. Anderson, “Externally excited coordination of autonomous formations,” *2007 Mediterr. Conf. Control Autom. MED*, 2007.

[3] I. Shames, B. Fidan, and B. D. O. Anderson, “Close target reconnaissance using autonomous UAV formations,” *Proc. IEEE Conf. Decis. Control*, pp. 1729–1734, 2008.

[4] C. Yu, B. Fidan, I. Shames, S. Sandeep, and B. D. O. Anderson, “Collision free coordination of autonomous multiagent systems,” *2007 Eur. Control Conf. ECC 2007*, pp. 900–907.

[5] M. Xu, Z. Pan, H. Lu, Y. Ye, P. Lv, and A. El Rhalibi, “Moving-target pursuit algorithm using improved tracking strategy,” *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 27–39, 2010.

[6] I. Shames, B. Fidan, B. D. O. Anderson, and H. Hmam, “Self-localization of mobile agents in the plane,” *3rd Int. Symp. Wirel. Pervasive Comput. ISWPC 2008, Proc.*, pp. 116–120, 2008.

[7] I. Shames, S. Dasgupta, and B. D. O. Anderson, “Circumnavigation Using Distance Measurements,” pp. 2444–2449, 2009.

[8] Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995): 27.

[9] M. Deghat and I. Shames, “Target localization and circumnavigation using bearing measurements in 2D,” *Decis. Control (…*, pp. 334–339, 2010.

[10] Roberge, Vincent, Mohammed Tarbouchi, and Gilles Labonté. "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning." *IEEE Transactions on Industrial Informatics* 9.1 (2013): 132-141.

[11] Chouhan, Satyendra Singh, and Rajdeep Niyogi. "DMAPP: A Distributed Multi-agent Path Planning Algorithm." *Australasian Joint Conference on Artificial Intelligence*. Springer International Publishing, 2015.

[12] Chouhan, Satyendra Singh, and Rajdeep Niyogi. "DiMPP: a complete distributed algorithm for multi-agent path planning." *Journal of Experimental & Theoretical Artificial Intelligence* (2017).

[13] Goldenberg, Meir, et al. "Enhanced Partial Expansion A." *J. Artif. Intell. Res.(JAIR)* 50 (2014): 141-187.

[14] Yu, Wentao, Jun Peng, and Xiaoyong Zhang. "A prioritized path planning algorithm for MMRS." *Control Conference (CCC), 2014 33rd Chinese*. IEEE, 2014.