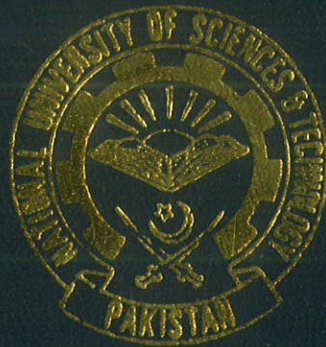


**Modeling Complex Multi Agent Systems:  
An Algorithm for Robotic Swarm Emergent  
Behavior**



**By**

**SAROSH HASHMI**

**2004-NUST-MSIT-04**

**NUST Institute of Information Technology  
National University of Sciences & Technology  
Rawalpindi, Pakistan  
May, 2006**



# **Modeling Complex Multi Agent Systems: An Algorithm for Robotic Swarm Emergent Behavior**

By

**Sarosh Hashmi**

(2004-NUST-MSIT-04)



A thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Science in Information Technology

**NUST Institute of Information Technology**


**National University of Science and Technology**

**Rawalpindi, Pakistan**

(May, 2006)

Certified that the contents and form of thesis entitled "Modeling Complex Multi Agent Systems: An Algorithm for Robotic Swarm Emergent Behavior" submitted by Mr. Sarosh Hashmi have been found satisfactory for the requirement of the degree.

Supervisor: \_\_\_\_\_

 23/5/06

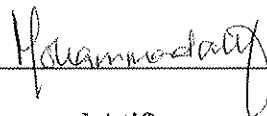
(Dr. Waqar Mahmood)

Member: \_\_\_\_\_

 24/5/06

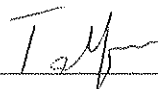
(Dr. N.D Gohar)

Member: \_\_\_\_\_



(Mr. Muhammad Atif)

Member: \_\_\_\_\_




(Mr. Tashfeen Khan)

## CERTIFICATE OF ORIGINALITY

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NIIT or any other education institute, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at NIIT or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged.

Signature:  \_\_\_\_\_

Sarosh Hashmi

## ACKNOWLEDGEMENTS

I bow my head in gratitude before Allah Almighty for giving me the strength, steadfastness and perseverance to pursue and complete my thesis. Without His blessings, it would not have been possible for me to complete my thesis work in its entirety in due time.

I owe my special thanks to my supervisor, Dr. Waqar Mahmood for his guidance and support. Without his worthy comments and directions, I could have faced lot of difficulties in completing my thesis. I would like to thank Mr. Muhammad Atif for his guidance and dedication in the completion of this work. Then I would like to thank Dr. N.D Gohar, Mr. Tashfeen Khan and Mr. Rizwan for all of there time, suggestions and the fine tuning where ever required.

In this entire journey to completing this work I would like to pay my special thanks to my parents for their unflinching support and encouragement while I worked day in and day out. And also I am grateful to every individual whose name is not mentioned but has been helpful to me in completing this project in any respect.

# Table of Contents

ABSTRACT.....	1
<b>1 INTRODUCTION.....</b>	<b>3</b>
1.1. RESEARCH DOMAIN.....	3
1.2. MOTIVATION.....	4
1.3. RESEARCH PROBLEMS.....	4
1.4. PROBLEM DEFINITION.....	4
1.5. THESIS STRUCTURE.....	5
<b>2 LITERATURE REVIEW.....</b>	<b>7</b>
2.1. GEOMETRICAL ANALYSIS BASED PROBABLISTIC MODELING.....	7
2.2. MODELING THROUGH MARKOV CHAIN BASED RATE EQUATIONS.....	8
2.3. FUSION OF FORMAL METHODS FOR MOEDLING EMERGENT BEHAVIORS.....	9
2.4. TEMPORAL LOGIC TO MODEL EMERGENT BEHAVIORS.....	9
2.5. DISCRETE EVENT SYSTEMS MODELING.....	10
<b>3 THE MATRIX FORMULATION.....</b>	<b>12</b>
3.1. THE CASE STUDY.....	12
3.2. MATRIX FORMULATION.....	14
3.3.1 <i>The Connect Matrix</i> .....	14
3.3.2 <i>The State Matrix</i> .....	15
3.3.3 <i>The Direction Matrix</i> .....	16
3.3.4 <i>Equation For Next State</i> .....	16
3.3.5 <i>Equation For Next Direction</i> .....	16

3.3.6	<i>The Initial Model</i> .....	18
3.3.	MICROSCOPIC MODEL.....	18
<b>4</b>	<b>CONTROLLABILITY AND OBSERVABILITY</b> .....	<b>20</b>
4.1.	OBSERVABILITY AND COLLISION AVOIDANCE .....	20
4.2.	CONTROL INPUT AND DIRECTED SWARMING .....	22
4.3.	THE COMPLETE MODEL .....	24
4.4.	THE LAYERED ARCHITECTURE .....	24
4.5.	SIMULATION OBSERVATIONS OF COMPLETE MODEL .....	25
<b>5</b>	<b>THE IMPROVED ALGORITHM</b> .....	<b>27</b>
5.1.	BACKGROUND.....	27
5.1.1	PROBLEMS WITH $\alpha$ ALGORITHM.....	27
5.1.2	THE BETA ALGORITHM.....	27
5.2.	THE IMPROVED ALGORITHM.....	28
5.3.	THE BRIDGE AVOIDANCE STATE .....	29
5.4.	NEIGHBOR DENSITY BASED INTELLIGENT TURNS .....	30
5.5.	THE MODIFIED LAYERED ARCHITECTURE .....	35
5.6.	RELATIVE ANGLE MEASUREMENT HARDWARE .....	36
<b>6</b>	<b>RESULTS EVALUATION</b> .....	<b>37</b>
6.1.	PLATFORM AND HARDWARE .....	37
6.2.	METRICS .....	37
6.3.	SIMULATION PARAMETERS .....	38
6.4.	COMPARITIVE ANALYSIS OF $\alpha$ AND IMPROVED ALGORITHM .....	39
6.5.	COMPARITIVE ANALYSIS OF IMPROVED ALGORITHM WITH	

	COLLISION AVOIDANCE OFF AND 25% NOISE .....	45
6.6.	COMPARITIVE ANALYSIS OF IMPROVED AND BETA ALGORITHM .....	48
6.7.	SUMMARY .....	50
<b>7</b>	<b>CONCLUSION .....</b>	<b>51</b>
7.1.	RESEARCH CONTRIBUTIONS .....	51
7.2.	FUTURE WORK .....	52
<b>8</b>	<b>REFERENCES .....</b>	<b>53</b>
	<b>INDEX.....</b>	<b>55</b>



## Lit of Figures

FIG 1: FINITE STATE MACHINE OF INDIVIDUAL ROBOT IN SWARM .....	12
FIG 2: COORDINATES FOR THE NEXT STEP, COMPUTED ON THE BASIS OF NEXT DIRECTION, ACCORDING TO $\alpha$ ALGORITHM.....	13
FIG3: THE COMPLETE INITIAL MODEL .....	18
FIG 4: THE COMPLETE MODEL.....	24
FIG 5: THE LAYERED ARCHITECTURE .....	25
FIG 6: A GRAPH WITH CUT VERTICES $v, x, y, w$ AND BRIDGE $e$ .....	27
FIG 7: MODIFIED FINITE STATE MACHINE OF INDIVIDUAL ROBOT IN SWARM .....	28
FIG 8: THE BRIDGE AVOIDANCE STATE .....	30
FIG 9: THE COORDINATE SYSTEM OF ROBOT $m$ IS DEFINED BY CHOOSING ROBOTS $r$ AND $s$ .....	33
FIG 10: AN EXAMPLE ILLUSTRATING THE WAY TO OBTAIN THE POSITION OF ROBOT $n$ IN THE COORDINATE SYSTEM OF ROBOT $m$ .....	34
FIG 11: MODIFIED LAYERED ARCHITECTURE .....	35
FIG 12: BEARING AND ORIENTATION MEASUREMENT SENSOR .....	36
FIG 13: NUMBER OF CONNECTED ROBOTS VERSUS TIME FOR $\alpha$ ALGORITHM FOR SWARM SIZE = 8.....	39
FIG 14: NUMBER OF 1-CONNECTED SWARMS VERSUS TIME FOR $\alpha$ ALGORITHM, SWARM SIZE = 8 .....	40
FIG 15: NUMBER OF CONNECTED ROBOTS AND NUMBER OF 1-CONNECTED SWARMS FOR IMPROVED ALGORITHM, SWARM SIZE = 8.....	42
FIG 16: NUMBER OF CONNECTED ROBOTS FOR $\alpha$ ALGORITHM, SWARM SIZE = 20 ....	43
FIG 17: NUMBER OF 1-CONNECTED SWARMS FOR $\alpha$ ALGORITHM, SWARM SIZE = 20	44

FIG 18: NUMBER OF CONNECTED ROBOTS AND NUMBER OF 1-CONNECTED SWARMS FOR IMPROVED ALGORITHM, SWARM SIZE = 20 .....	45
FIG 19: NUMBER OF CONNECTED ROBOTS AND NUMBER OF 1-CONNECTED SWARMS FOR IMPROVED ALGORITHM, SWARM SIZE = 8 WITH COLLISION AVOIDANCE OFF .....	46
FIG 20: NUMBER OF CONNECTED ROBOTS AND NUMBER OF 1-CONNECTED SWARMS FOR IMPROVED ALGORITHM, SWARM SIZE = 20 WITH 25% NOISE .....	47
FIG 21: DISTANCE TRAVELED VERSUS TIME FOR BETA ALGORITHM, SWARM SIZE = 8 .....	48
FIG 22: DISTANCE TRAVELED VERSUS TIME FOR IMPROVED ALGORITHM, SWARM SIZE = 8 .....	49

## List of Tables

TABLE 1: $(n+1)th$ STATE DERIVED FROM $nth$ STATE AND CONNECT MATRICES. ....	16
TABLE 2 : $(n+1)th$ DIRECTION DERIVED FROM $nth$ DIRECTION AND STATE-CONNECT MATRICES. ....	17
TABLE 3 : $(n+1)th$ DIRECTION DERIVED FROM $nth$ DIRECTION AND OBSERVER- DUMMY MATRICES. ....	22
TABLE 4 : $(n+1)th$ DIRECTION DERIVED FROM $nth$ DIRECTION AND INPUT-DIRECTION MATRICES. ....	23
TABLE 5 : PLATFORM AND HARDWARE SPECIFICATION .....	37

## List of Abbreviations

[B]

BOM: Bearing and Orientation Measurement Sensor

[C]

CSP: Communicating Sequential Processes

[D]

DES: Discrete Event Systems

[I]

IR: Infra Red

[L]

LCS: Local Coordinate System

[S]

SI: Swarm Intelligence

[W]

WSCCS: Weighted Synchronous Calculus of Communicating  
Systems

## ABSTRACT

Multi Agent systems comprise of a number of agents, interacting together to perform a specific task. Swarm robotics systems are an important class of multi agent systems. These systems are based on the principles of swarm intelligence, whereby sequence of local events between simple robotic agents or agents and their environment lead to the emergence of overall complex behavior following certain design goals. Such systems tend to be robust and scaleable due to their purely distributed nature. However, they require sound mathematical models that would guarantee desired emergent behavior.

Due to their characteristics of having discrete state space and event driven, swarm robotic systems can be categorized as discrete event systems (DES) and can therefore be subjected to DES modeling and analysis tools. Taking on the this approach and considering successful application of matrices to model complex DES like flexible manufacturing systems, a novel matrix formulation is derived to model emergent coherent movement of a wirelessly connected swarm of robots.

Control theoretic concepts of controllability and observability are employed to further refine the model and consequently an observer matrix is defined to model collision avoidance of swarm robots by keeping track of distances between them. Then, to move the swarm in a desired direction, an external control input is defined that changes the direction of every robot to desired direction. This approach can be applied not only to model swarm dynamics but also to quickly verify its emergent behavior. The simulation in MATLAB provides the dynamic behavior of swarm well captured through the modeling approach.



We have also proposed an improved algorithm for coherent emergent motion of a robotic swarm. The strength of this algorithm lies in its potential to allow very fast and stable emergent movement of a robotic swarm using inexpensive hardware and no beacons or milestones at all. The algorithm therefore has great potential to be applied to real world problems like area surveillance by a group of robots, space exploration missions and mines sweeping tasks etc.

## **INTRODUCTION**

### **1.1. RESEARCH DOMAIN**

We in this thesis consider a class of multi-agent systems where agents comprising the system are simple but the sequence of local events between these agents or agents and their environment lead to the emergence of desired complex behaviors and hence such systems are called as Complex Multi-Agent Systems. Distributed manufacturing, distributed peer to peer and swarm robotic systems are examples of such systems.

Swarm robotics systems based on the principles of swarm intelligence, draw inspiration from biological domain examples of social insects (like ants, birds etc.) [1]. The field of swarm robotics comes under the category of collective robotics. Here researchers are interested to discover how a group of simple, inexpensive and autonomous robots can be employed to perform complex tasks, which are impossible or inefficient to be completed by a single robot. The constraint of using autonomous robots with limited sensing capabilities and without any centralized control makes the task of designing and verifying such system very difficult. We in this thesis present a matrix formulation for designing and verifying behaviors of a swarm robotic system. Further an improved algorithm is presented that uses inexpensive hardware and provides very fast and stable emergent coherent movement of a swarm of robots.

## **1.2. MOTIVATION**

Efficient modelling of multi-agent systems is most important for their control, reconfiguration and optimization. Comprehensive modeling methodologies need to be developed to engineer such systems with high degree of dependability and reliability thereby guaranteeing desired collective emergent behavior, preventing undesirable behavior and determining parameters that optimize overall system performance. The individual actions of autonomous entities in such systems lead to collective emergent behavior, which needs to be validated against set goals thereby, necessitating a sound integrated modeling approach for specifying, proving and testing individual robots as well as overall system emergent behavior.

## **1.3. RESEARCH PROBLEMS**

Robotic systems based on swarm intelligence are difficult to model due to the following challenges:

- 1- These systems are highly distributed and parallel
- 2- There are large number of interacting agents
- 3- Swarm intelligence based system show emergent behaviour
- 4- Total or near total autonomous agents
- 5- Worse than exponential growth in the interactions of agents

## **1.4. PROBLEM DEFINITION**

A modeling methodology needs to be developed, which not only captures individual behaviors of the agent comprising the swarm, but also present the emergent behavior arising due to the interactions of these autonomous agents. The methodology should be scalable to allow addition of large number of interacting agent. It should be able to handle bounded as well as unbounded environments.

Also it should not be very resource intensive and provide fast simulations. Keeping in view these requirements of a modeling methodology, we present a model based on matrices (Chapter 3).

Another issue needs to be addressed is the problems suffered by current algorithms for purely distributed coherent motion of robotic swarms. The two algorithms considered are alpha and beta algorithms [20] and each have different problems. The alpha algorithm while providing fast emergent motion doesn't guarantee swarm coherence, whereas beta algorithm does guarantee swarm coherence but provides very slow emergent motion. We present an improved algorithm, which overcomes the problems of alpha and beta algorithms and provides very fast and stable emergent motion (Chapter 5).

## **1.5. THESIS STRUCTURE**

An analysis of existing modelling methodologies is presented in the next chapter i.e. chapter 2. It explains the problems addressed by these methodologies along with the un-addressed issues in each system. Matrix formulation for modelling robotic swarm emergent behaviour is presented in chapter 3. Chapter 4 discusses control theoretic concepts of controllability and observability. Control input is introduced to move the swarm in desired direction and an observer matrix is defined that notifies possible collision in the next step, thereby triggering a collision avoidance input. Chapter 5 presents an improved algorithm for emergent coherent motion of a robotic swarm. Chapter 6 gives a thorough evaluation of the improved algorithm in terms of number of connected robots, number of sub-swarms and distance travelled observed on running the simulation. Chapter 7

concludes this work by summarizing the contribution of our research and discussing some future directions.



## LITERATURE REVIEW

There has been very little research available in literature on the domain of swarm robotics modeling. The models available in literature capture swarm behaviors in some specific scenarios. However majority of them lack in presenting a comprehensive solution which would address the issues highlighted in Section 1.3.

### 2.1. GEOMETRICAL ANALYSIS BASED PROBABLISTIC MODELING

A non-spatial probabilistic model of distributed manipulation tasks by a swarm of robots was presented in [2],[3] based on probabilities calculated by geometrical analysis of the objects involved like, walls, other robots, sticks to be pulled out etc. For example, the probability of coming across a stick in the arena will be equal to area of stick divided by area of arena; similarly the probabilities of coming across other robots, walls etc. are calculated as described by the following equations:

$$p_{st} = A_{st}/A_a,$$

$$p_w = A_w/A_a$$

$$p_r = A_r/A_a$$

Where  $p_{st}$ ,  $p_w$ , and  $p_r$  are the probabilities of coming across a stick, wall or another robot respectively and  $A_{st}$ ,  $A_w$ ,  $A_r$ ,  $A_a$  are the areas of stick, wall, robot and arena respectively. Now the equation for macroscopic model for search and avoidance sub-chain can be written as:

$$N_s(k+1) = N_s(k) - p_a N_s(k)$$

$$N_a(k+1) = N_0 - N_s(k+1)$$

Where  $N_s$ ,  $N_a$  are the number of robots in search and avoidance states respectively and  $N_0$  is the total number of robots in the swarm.

While this methodology has the advantage of not being relying on computation of orientation and sensory information of individual robots; it is not applicable in scenarios where the arena is overcrowded with robots or for robots with individual sophisticated capabilities like wireless communication. Wireless communication may result in enhancing the decision power of robots based on the information exchanged, thereby increasing the number of probabilities to be computed, making the system very complex and difficult to model. Also the application of methodology is limited to bounded arenas only. If unbounded environments are considered than it will be very difficult to calculate the probabilities involved.

## **2.2. MODELING THROUGH MARKOV CHAIN BASED RATE EQUATIONS**

Lerman et al [4],[5] worked on characterizing each agent behavior as being stochastic and Markovian and proposed a stochastic master equation from which rate equations were derived describing the change at macroscopic level of swarm over time. The approach was applied to a scenario of coalition formation of agents in an electronic marketplace where rates of agents joining or leaving coalitions were modeled as function of utility gain of becoming the member of a particular coalition, implying that agents will probably join larger coalitions with better utility gain rather than smaller ones. The methodology was also applied to study the behavior of a group of robots involved in a foraging task to collect pucks from an

arena and deliver them to a specified location, with rate equations giving number of robots in searching, homing and avoiding states at a particular instant. However, again this methodology is non-spatial and is not applicable to the cases where specification and verification of design goals is dependent on robots locations.

### 2.3. FUSION OF FORMAL METHODS FOR MOEDLING EMERGENT BEHAVIORS

Rouff et al [6],[7],[8] after evaluating various formal methods proposed a fusion of Communicating Sequential Processes (CSP) [9], Unity logic, Weighted Synchronous Calculus of Communicating Systems (WSCCS) [10] and X-Machines [11] to develop an integrated formal method for specifying and verifying emergent behaviors of swarms of large number of cooperating spacecrafts for NASA's future space exploration missions.

### 2.4. TEMPORAL LOGIC TO MODEL EMERGENT BEHAVIORS

Winfield et al [12] have considered temporal logic for specifying robotic swarm behavior and possibly using a temporal monodic prover [13],[14] to verify probable emergent behaviors. Temporal logic is an extension to conventional predicate logic and its key operators include:

- $\bigcirc\varphi$  is satisfied if the formula  $\varphi$  is satisfied at the *next* moment in time
- $\diamond\varphi$  is satisfied if  $\varphi$  is satisfied at *some* future moment in time,
- $\square\varphi$  is satisfied if  $\varphi$  is satisfied at *all* future moments in time.

For example the moving forward task for robots can be specified in temporal logic as follows:

$$\begin{aligned}
 \text{move}F(i) := & \\
 & (\theta_i = N \wedge (Ox_i = x_i) \wedge (Oy_i = y_i + a)) \vee \\
 & (\theta_i = S \wedge (Ox_i = x_i) \wedge (Oy_i = y_i - a)) \vee \\
 & (\theta_i = W \wedge (Ox_i = x_i - a) \wedge (Oy_i = y_i)) \vee \\
 & (\theta_i = E \wedge (Ox_i = x_i + a) \wedge (Oy_i = y_i))
 \end{aligned}$$

The above equation says that if the current orientation of robot ( $\theta_i$ ) is north then moving forward further will require that at next moment (consider the usage of  $\bigcirc$  operator) its x coordinates should remain unchanged while y coordinates should be incremented by one step movement of 'a' units. Similarly the specifications if robot current direction is South, East or West are written accordingly.

Formal specification of swarm properties using this method has been described in this work, but verification of such properties using temporal prover is currently the work in progress and it is anticipated that application of temporal prover for verification will require reformatting of these specifications which may result in large number of clauses, thus making this methodology very resource intensive to implement.

## 2.5. DISCRETE EVENT SYSTEMS MODELING

The systems having discrete state space and in which state transitions are event driven are known as discrete event systems (DES) [15]. Computer, communications, traffic and manufacturing systems are few examples of DES.

Swarm robotic systems are also discrete state i.e. their state set has discrete values (like Forward, Coherent, Avoidance etc.) and also transition from one state to another is event driven. For example a state transition function can state that robot

has to come into coherence state in the event of disconnection from the swarm. Therefore swarm robotic systems can be rightfully categorized as DES. Many methodologies exist for modeling DES, like timed automata, Petri nets, min-max algebra etc. Matrix formulation also has been shown to be successfully employed for modeling, simulation and analysis of complex DES like modern flexible manufacturing systems with reentrant flow lines [16],[17]. It enables fast design and reconfiguration of rule based controllers for such manufacturing systems and state equation of these controllers give comprehensive dynamical description of these DES [18], [19]. Taking on the same approach we present in the next chapter a novel matrix formulation [21] to model emergent coherent movement of a wirelessly connected swarm of robots.



## THE MATRIX FORMULATION

### 3.1. THE CASE STUDY

We model the emergent behavior of a wireless connected swarm presented by Winfield et al in [12] which uses the simplest ‘alpha algorithm’ of all algorithms presented in [20] for swarm coherence. Robots are equipped with proximity sensors and memory, and can store proximity sensor data of one previous state. The finite state machine of individual robot in the swarm is shown in figure below and we accordingly model the robots to have following three states:

1. Forward State (Abbreviated as Fw)
2. Coherence State (Abbreviated as Coh)
3. Avoidance State

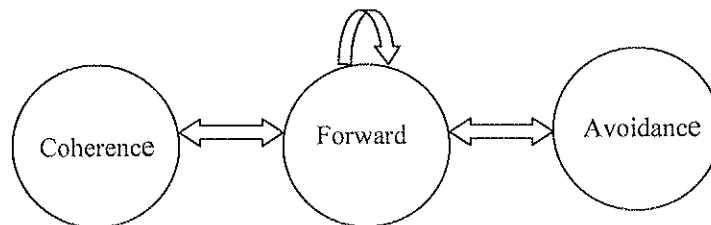


Fig 1: Finite State Machine of Individual Robot in Swarm

The alpha algorithm states that by default at each time step, the robots move ‘ $d$ ’ units in the forward direction on a grid space and their orientation can have values from the set of North, South, East and West only. A robot can also turn left  $90^0$ , turn right  $90^0$  or take  $180^0$  u turn. If direction of robot at next step will be north, its x coordinate is kept constant while y coordinate is increased  $d$  units, if next direction is south, again x coordinate is kept same while y coordinate is decremented  $d$  units, if next direction is east then x coordinate is increased  $d$  units

while  $y$  coordinate is kept constant and finally if next direction is west, the  $x$  coordinate is decreased  $d$  units keeping  $y$  coordinate constant as depicted in figure 2.

A variable " $\alpha$ " specifies the minimum number of neighbors (connections) which a robot <sub>$i$</sub>  has to maintain, falling below which the robot <sub>$i$</sub>  is forced to take a  $180^\circ$  turn and change to coherence state. For example if  $\alpha = 1$ , it means that loss of last connection with another robot in the swarm triggers the coherence state. If the number of neighbors increase above  $\alpha$  the robot takes right or left turn to avoid swarm collapsing on itself.

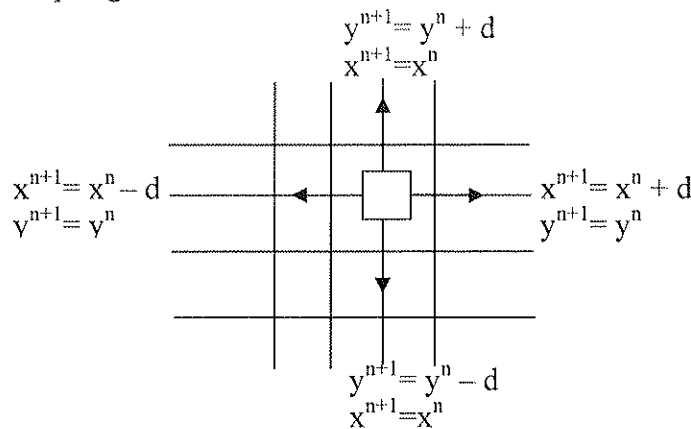


Fig 2: Coordinates for the next step, computed on the basis of next direction, according to  $\alpha$  algorithm

A robot is said to be connected to another robot, if later is in its connectivity radius ' $r$ ' and likewise a robot is said to be connected (abbreviated as 'con') to the swarm if it is connected with at least  $\alpha$  other robots and said to be disconnected otherwise (abbreviated as  $\sim$ con).

The state transition functions are (excluding avoidance state which is modeled separately by defining an observer matrix):

1.  $Fw + con \rightarrow Fw$
2.  $Fw + \sim con \rightarrow \text{turn } 180^\circ \rightarrow Coh$

3. Coh +  $\sim$ con  $\rightarrow$  Fw
4. Coh + con  $\rightarrow$  Left or Right turn  $\rightarrow$  Fw

### 3.2. MATRIX FORMULATION

Let the swarm size (denoted by variable name of 'TotalRobots') equal to four robots and x, y coordinates of the four robots be  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$  and  $(x_4, y_4)$  respectively.

#### 3.3.1 The Connect Matrix

We generate a proximity matrix A by finding the Euclidean distances between the respective robots:

$$A = \begin{matrix} & \begin{matrix} R1 & R2 & R3 & R4 \end{matrix} \\ \begin{matrix} R1 \\ R2 \\ R3 \\ R4 \end{matrix} & \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} \end{matrix}$$

Where  $a_{ij}$  = Euclidean distance between robot<sub>i</sub> and robot<sub>j</sub>.

As stated earlier, let  $r$  = maximum connected distance

We subtract the proximity matrix A by a matrix B having all elements equal to  $r$ , resulting in matrix C:

$$\begin{matrix} & \begin{matrix} R1 & R2 & R3 & R4 \end{matrix} \\ \begin{matrix} R1 \\ R2 \\ R3 \\ R4 \end{matrix} & \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} \end{matrix} - \begin{matrix} & \begin{matrix} r & r & r & r \end{matrix} \\ \begin{matrix} r & r & r & r \\ r & r & r & r \\ r & r & r & r \\ r & r & r & r \end{matrix} \end{matrix} = \begin{matrix} & \begin{matrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{matrix} \end{matrix}$$

Converting the negative values into zeros and positive values into ones of the above resultant matrix C gives a matrix D which has all binary values, where:

$$d_{ij} = \begin{cases} 0 & \text{if robot}_i \text{ and robot}_j \text{ are connected, and} \\ 1 & \text{otherwise.} \end{cases}$$

$$D \text{ (Binary values matrix)} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix}$$

We assume a value of  $\alpha = 1$  so that the loss of single connection to the swarm triggers the coherence state. In other words if there are less than two zeros in any row of the above matrix then the respective robot is not connected. Therefore, we can reduce the above matrix  $D$  into a column matrix 'Connect' by writing '0' in the row if  $(\alpha + 1)$  values in the corresponding row of matrix  $D$  are zeros and '1' otherwise:

$$\text{Connect}^n = \begin{bmatrix} \text{connect}_{11} \\ \text{connect}_{21} \\ \text{connect}_{31} \\ \text{connect}_{41} \end{bmatrix}^n$$

Where:

$$\left. \begin{array}{l} \text{connect}_{i1} = 0 \\ = 1 \end{array} \right\} \begin{array}{l} \text{if robot}_i \text{ is connected to the swarm, and} \\ \text{otherwise} \end{array}$$

$n$  is the time variable indicating discrete time instance.

### 3.3.2 The State Matrix

We define a column matrix 'State' where:

$$\left. \begin{array}{l} \text{state}_{i1} = 0 \\ = 1 \end{array} \right\} \begin{array}{l} \text{if robot}_i \text{ is in 'Forward' state} \\ \text{if robot}_i \text{ is in 'Coherence' state} \end{array}$$

$$\text{State}^n = \begin{bmatrix} \text{state}_{11} \\ \text{state}_{21} \\ \text{state}_{31} \\ \text{state}_{41} \end{bmatrix}^n$$

### 3.3.3 The Direction Matrix

Finally, we define a Direction matrix as follows:

$$\text{Direction}^n = \begin{bmatrix} \text{direction}_{n1} & \text{direction}_{n2} \\ \text{direction}_{n1} & \text{direction}_{n2} \\ \text{direction}_{n31} & \text{direction}_{n32} \\ \text{direction}_{n41} & \text{direction}_{n42} \end{bmatrix}$$

Where:

$$\begin{array}{l} \text{direction}_{i1} = 0 \ \& \ \text{direction}_{i2} = 0 \\ \qquad \qquad \qquad = 0 \\ \qquad \qquad \qquad = 1 \\ \qquad \qquad \qquad = 1 \end{array} \left. \begin{array}{l} \\ \\ \\ \\ \end{array} \right\} \begin{array}{l} \text{if robot}_i \text{ is pointing 'North'} \\ \text{if robot}_i \text{ is pointing 'South'} \\ \text{if robot}_i \text{ is pointing 'East'} \\ \text{if robot}_i \text{ is pointing 'West'} \end{array}$$

### 3.3.4 Equation For Next State

We compute next state vector  $\text{State}^{n+1}$  by the following equation:

$$\text{State}^{n+1} = (\text{State}^n \wedge \text{Connect}^n) \oplus \text{Connect}^n$$

Note:  $\oplus$  denotes the 'XOR' operation

State <sup>n</sup>	Connect <sup>n</sup>	State <sup>n+1</sup>
0	0	0
0	1	1
1	1	0
1	0	0

Table 1 (n+1)th state derived from nth State and Connect matrices.

### 3.3.5 Equation For Next Direction

We combine  $\text{State}^n$  and  $\text{Connect}^n$  matrices to form State-Connect augmented matrix and convert any row of this augmented matrix containing all ones to all zeros. Lastly we 'XOR' this modified State-Connect augmented matrix with  $\text{Direction}^n$  matrix to give  $\text{Direction}^{n+1}$ .



$$\text{Direction}^{n+1} = \text{State-Connect} \oplus \text{Direction}^n$$

It may be noted that the conversion of any row in State-Connect matrix containing all ones to all zeros does not affect the behavior of system as the next action to be taken in both cases is same (i.e. to move in forward direction). Following table describe the transition from  $n$ th direction to  $(n + 1)$ th direction using the above equation:

State Transition Function	State-Connect		Direction <sup>n</sup>		Direction <sup>n+1</sup>	
Fw + con → Fw	0	0	0	0	0	0
			0	1	0	1
			1	0	1	0
			1	1	1	1
Fw + ~con → turn 180° → Coh	0	1	0	0	0	1
			0	1	0	0
			1	0	1	1
			1	1	1	0
Coh + ~con → Fw	1	1	State-Connect entries converted to '0 0'. See entries against '0 0'			
Coh + con → Left or Right turn → Fw	1	0	0	0	1	0
			0	1	1	1
			1	0	0	0
			1	1	0	1

Table 2  $(n+1)$ th direction derived from  $n$ th Direction and State-Connect matrices.

We explain here the modeling of state transition no. 4. The last four rows of the above table state that if a robot is in coherence state '1' and connected to swarm '0' and if its current direction is north '0 0' it will turn right to change its direction to east '1 0'. If the robot is in south direction '0 1', it will turn right to change its direction to west '1 1'. If the robot is in east direction '1 0', it will take a left turn to change its direction to north '0 0' and finally if the robot is currently in west direction '1 1' it will turn left to south direction '0 1'.

In other words, the random behavior of state transition number 4 described earlier is modeled by computing the next direction of robot based on its current direction.

### 3.3.6 The Initial Model

The complete initial model is shown in the figure below. It describes that input to the model are current coordinates of the robot ( $X^n$  and  $Y^n$ ). This input is utilized to compute  $Connect^n$  matrix and subsequently  $State^{n+1}$  and  $Direction^{n+1}$  matrices to finally give the coordinates for next step i.e.  $X^{n+1}$  and  $Y^{n+1}$ .

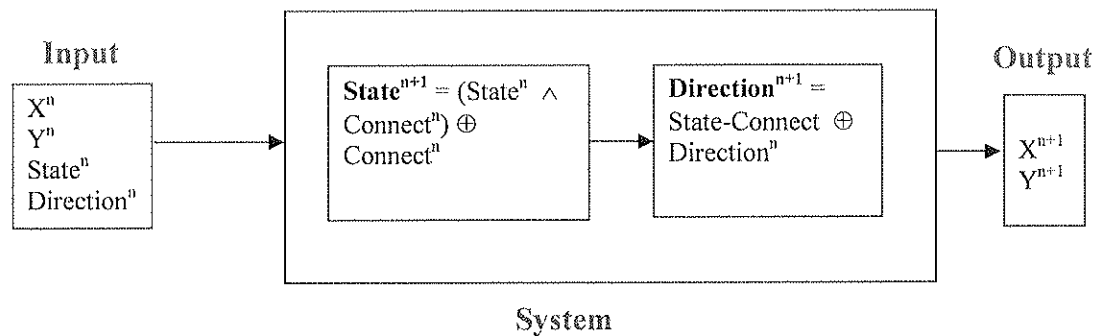


Fig3: The Complete Initial Model

### 3.3. MICROSCOPIC MODEL

The model presented is the macroscopic model of the whole swarm. To derive microscopic model, or capture the dynamics of individual robots we have to modify these matrices accordingly, like the proximity matrix for robot 1 can take the shape of the following row matrix:

$$\text{Proximity R1} = [0 \ 3 \ 1 \ 100]$$

(i.e. the first row of proximity matrix for the whole swarm)

The robot will fill the matrix values by locally sensing within its connectivity radius and finding the Euclidean distances for whatever other robots it finds in the

neighborhood. For the robots which it doesn't sense any information, it simply put a predefined infinity value like '100' in the above case in order to facilitate the later computation of *Connect*<sup>n</sup> matrix. Similarly all other matrices are modified accordingly.

## CONTROLLABILITY AND OBSERVABILITY

### 4.1. OBSERVABILITY AND COLLISION AVOIDANCE

We define avoidance radius ' $r_a$ ' be the radius falling within which, the robots assume to be heading towards a collision and need to change their directions [23].

At each forward movement of ' $d$ ' units, we subtract the proximity matrix  $A$  by the matrix  $E$  having all elements equal to  $r_a$  except the diagonal ones, resulting in matrix  $F$ :

$$F^n = A^n - E^n$$

$$\begin{bmatrix} 0 & f_{12} & f_{13} & f_{14} \\ f_{21} & 0 & f_{23} & f_{24} \\ f_{31} & f_{32} & 0 & f_{34} \\ f_{41} & f_{42} & f_{43} & 0 \end{bmatrix} = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} - \begin{bmatrix} 0 & r_a & r_a & r_a \\ r_a & 0 & r_a & r_a \\ r_a & r_a & 0 & r_a \\ r_a & r_a & r_a & 0 \end{bmatrix}$$

Converting the negative values into ones and positive values into zeros of the above resultant matrix  $F$  gives a matrix  $InRadius^n$  which has all binary values, where:

$$\text{inradius}_{ij} = \begin{cases} 1 & \text{if robot}_i \text{ and robot}_j \text{ are in the avoidance radius of each} \\ & \text{other, and} \\ 0 & \text{otherwise} \end{cases}$$

At each forward movement, the observer also checks whether the distance between two robots is increasing or decreasing by subtracting the distance at current step  $n$  with the distance at one previous step  $n-1$  giving matrix  $G^n$ .

$$G^n = A^n - A^{n-1}$$

Converting negative values into ones and positive values into zeros of the above resultant matrix  $G$  gives a matrix  $DistanceChange^n$  which has all binary values, where:

$$distancechange_{ij} = \begin{cases} 1 & \text{if distance between robot}_i \text{ and robot}_j \text{ is} \\ & \text{decreasing between two successive steps, and} \\ 0 & \text{otherwise} \end{cases}$$

We 'NAND' matrices  $InRadius^n$  and  $DistanceChange^n$  to give matrix  $H^n$  :

$$H^n = InRadius^n \otimes DistanceChange^n$$

Note:  $\otimes$  denotes 'NAND' operation

Where:

$$h_{ij} = \begin{cases} 0 & \text{if robot}_i \text{ and robot}_j \text{ are heading towards collision} \\ 1 & \text{otherwise} \end{cases}$$

This implies that if there is even a single zero in any row of the above matrix then the respective robot is supposed to be heading towards collision and need to change its direction. Therefore, we can reduce the above matrix  $H$  into a column matrix '*Observer*' by writing '1' in the row if all values in the corresponding row of matrix  $H$  are ones and '0' otherwise:

$$Observer^n = \begin{bmatrix} observer_{11} \\ observer_{21} \\ observer_{31} \\ observer_{41} \end{bmatrix}^n$$

Where:

$$observer_{r11} = \begin{cases} 0 & \text{if robot}_r \text{ is heading towards a collision, and} \\ 1 & \text{otherwise} \end{cases}$$

Further, we combine  $Observer^n$  with a matrix *Dummy* of order 'TotalRobots \* 1' of all ones to form Observer-Dummy augmented matrix. This matrix acts as an input which we 'XOR' with  $Direction^n$  matrix to give  $Direction^{n+1}$ .

$$Direction^{n+1} = Observer-Dummy \oplus Direction^n$$

The above equation is applied only to those robots for which an Observer value of '0' (and thus an Observer-Dummy value of '0 1') is noticed, whereas all other robots are treated as non colliding.

Following table describe the transition from  $n^{th}$  direction to  $(n + 1)^{th}$  direction using the above equation:

Observer-Dummy		Direction <sup>n</sup>		Direction <sup>n+1</sup>	
0	1	0	0	0	1
		0	1	0	0
		1	0	1	1
		1	1	1	0

**Table 3 (n+1)th direction derived from  $n^{th}$  Direction and Observer-Dummy matrices.**

The above table states that if the observer notes that a robot is heading towards a collision (an 'Observer-Dummy' value of '0 1'), the robot is forced to take a u turn. So if the robot is currently pointing towards north '0 0' then it will turn to south '0 1', if it is in south '0 1' it will turn towards north '0 0', and so on.

## 4.2. CONTROL INPUT AND DIRECTED SWARMING

To move the swarm in the desired direction we have to control the variable *Direction*. For this we define an input matrix as follows, that changes the orientation of all robots to the desired direction:

$$Input-Direction = \begin{bmatrix} input\_direction_{11} & input\_direction_{12} \\ input\_direction_{21} & input\_direction_{22} \\ input\_direction_{31} & input\_direction_{32} \\ input\_direction_{41} & input\_direction_{42} \end{bmatrix}$$

Where:

$$\text{input\_direction}_{i1} = 0 \ \& \ \text{input\_direction}_{i2} = 0$$

$\forall i = 1$  to TotalRobots if desired direction of swarm is 'North'

$$\text{input\_direction}_{i1} = 0 \ \& \ \text{input\_direction}_{i2} = 1$$

$\forall i = 1$  to TotalRobots if desired direction of swarm is 'South'

$$\text{input\_direction}_{i1} = 1 \ \& \ \text{input\_direction}_{i2} = 0$$

$\forall i = 1$  to TotalRobots if desired direction of swarm is 'East'

$$\text{input\_direction}_{i1} = 1 \ \& \ \text{input\_direction}_{i2} = 1$$

$\forall i = 1$  to TotalRobots if desired direction of swarm is 'West'

We get the desired direction of all robots by the following equation:

$$\text{Direction}^{n+1} = (\text{Input-Direction} \oplus \text{Direction}^n) \oplus \text{Direction}^n$$

Input-Direction		Direction <sup>n</sup>		Direction <sup>n+1</sup>	
0	0	0	0	0	0
		0	1	0	0
		1	0	0	0
		1	1	0	0
0	1	0	0	0	1
		0	1	0	1
		1	0	0	1
		1	1	0	1
1	0	0	0	1	0
		0	1	1	0
		1	0	1	0
		1	1	1	0
1	1	0	0	1	1
		0	1	1	1
		1	0	1	1
		1	1	1	1

Table 4 (n+1)th direction derived from nth Direction and Input-Direction matrices.

Given the current state and direction of swarm, we can now move the swarm in the desired direction by giving the aforementioned input. A destination

with a given x,y coordinates is thus reachable through the control input for specific number of steps, given the present location of swarm is known.

### 4.3. THE COMPLETE MODEL

The complete model is shown in the figure below. It can be seen that observer is introduced as a feed back mechanism in the system i.e. it checks the system output and any deviation from the desired behavior is observed (e.g. pair of robots coming into avoidance radius of each other) then it triggers a collision avoidance input to prevent collisions. The control input is introduced as a feed forward mechanism in the system i.e. system responds to this known clue in the form of control input and direction of every robot is changed to the desired one.

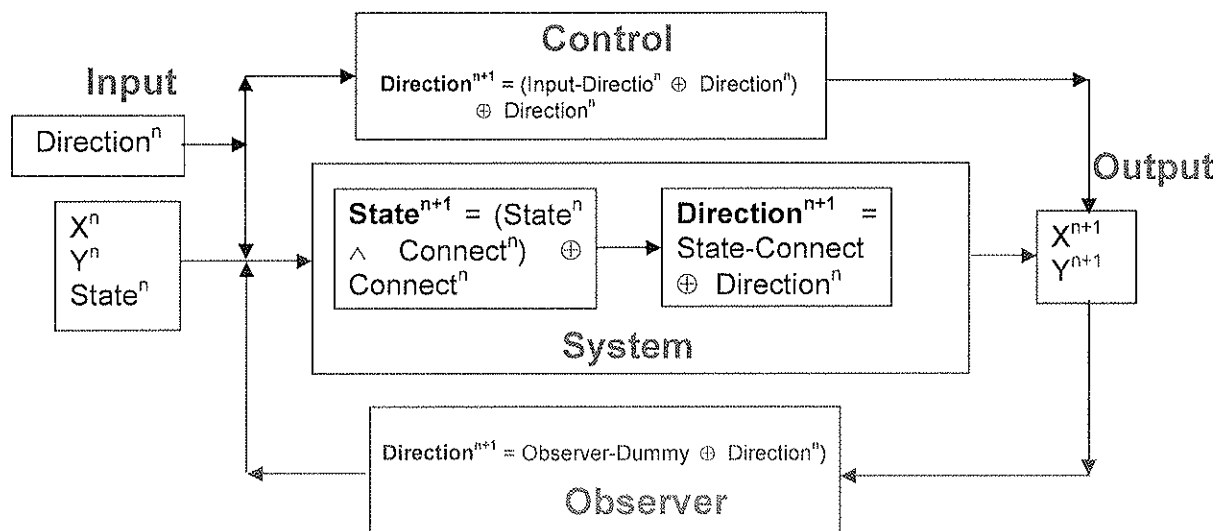


Fig 4: The Complete Model

### 4.4. THE LAYERED ARCHITECTURE

Our control system has three layers as shown in figure 5. The obstacle avoidance layer gets the top priority. If this layer is activated (incase observer notes a possible



collision) than commands from subsequent layers are not executed. Second priority goes to control input layer. If swarm coherence layer requires a robot to take a turn but if at the same time control input is also activated requiring it to continue moving forward, then due to control layer dominance the robot will not take a turn and continue moving forward. Thus swarm coherence rules layer has the least priority in this architecture.

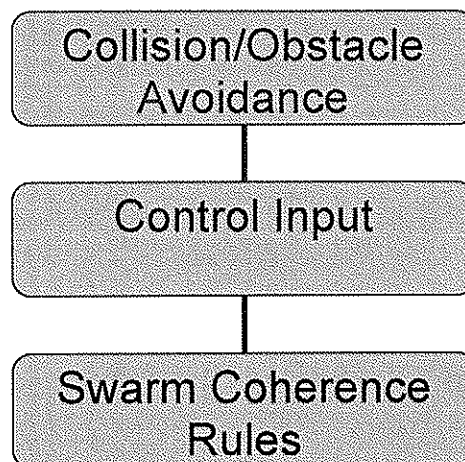


Fig 5: The Layered Architecture

#### 4.5. SIMULATION OBSERVATIONS OF COMPLETE MODEL

- The swarm remained coherent and move forward at a fast pace provided the initial states and directions of all robots were same at start.
- If randomness is introduced in the initial states and directions of individual robots, the swarm may divide itself into sub swarms depending on the value of alpha. In such case there may be sub swarm formation of different sizes.
- Increasing the value of alpha may increase the size of sub swarms but a very high alpha value results in a very over-reactive swarm, which just remains occupied in rearranging itself and never moves forward.

- On the application of control input, the whole swarm moves in the desired direction, however, due to simplistic dynamics of the system the control input dominates swarm dynamics. This results in swarm losing capability to regain coherence in case there exist non-zero disconnected robots at the time of control application or later.
- The robots always take u turn whenever they come inside avoidance radius of each other with gradually decreasing distance, to avoid collision.

## THE IMPROVED ALGORITHM

### 5.1. BACKGROUND

In this section the rationale for our proposed algorithm is made clear by first describing the problems identified with alpha algorithm. Then one solution to these problems i.e. the beta algorithm is described. Finally the problems with beta algorithms are discussed which necessitate the formulation of an improved algorithm described in detail in section 5.2.

#### 5.1.1 PROBLEMS WITH $\alpha$ ALGORITHM

As described in the simulation observations, the main problem with alpha algorithm is that at a lower alpha value, coherence of swarm is not guaranteed and it divides into sub-swarms. The underlying reason for this division is the existence of nodes which in graph theoretic term are known as 'bridges' and 'cut vertices'[24].

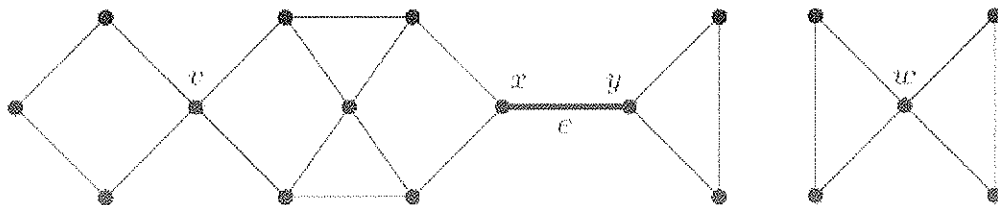


Fig 6: A graph with cut vertices  $v$ ,  $x$ ,  $y$ ,  $w$  and bridge  $e$  [24]

#### 5.1.2 THE BETA ALGORITHM

This algorithm [25] states that for each lost connection a robot checks how many of its remaining neighbors still have the lost robot in their neighborhood. If this number is less than or equal to beta, the robot takes a u turn and comes

back. In contrast, if its degree of connection is rising the robot chooses a random heading. So, the robot tries to maintain beta shared connections with each neighbor and formation of bridges and cut vertices is prevented if  $\beta \geq 2$ .

Beta algorithm overcomes the problem of swarm division by preventing the formation of bridges and cut vertices but yields another problem. Though many connections are shared, yet the strict connectivity constraints (i.e. even for  $\beta = 1$ , a robot has to maintain triangulation for each of its other neighbors, for  $\beta = 2$  it has to maintain rectangulation and so on) results in a very slow emergent motion, thereby minimizing the chances for applying the algorithm for any practical application.

## 5.2. THE IMPROVED ALGORITHM

The finite state machine of individual robot in the swarm is modified and is shown in the figure below. Accordingly we model the robots to have following four states:

1. Forward State (Abbreviated as Fw)
2. Coherence State (Abbreviated as Coh)
3. Avoidance State
4. Bridge Avoidance State

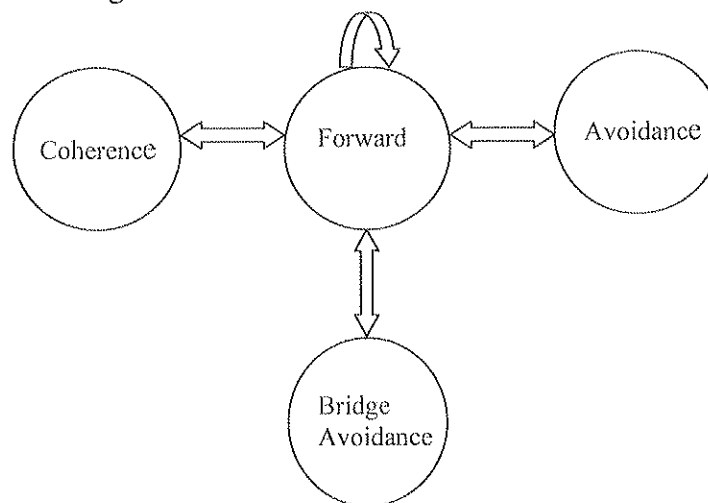


Fig 7: Modified Finite State Machine of Individual Robot in Swarm

According to this improved algorithm, by default at each time step, the robots move 'd' units in the forward direction and their orientation can have any value from 0° to 360°. A robot can turn at any angle between 0° to 360°. At each step, the coordinates for next step ( $x^{n+1}$ ,  $y^{n+1}$ ) are computed as follows:

$$x^{n+1} = x^n + (d * \cos \theta^n)$$

$$y^{n+1} = y^n + (d * \sin \theta^n)$$

Where  $\theta^n$  is the orientation of robot; at time step n, and can have value from 0° to 360° and 'd' is one step movement.

The modified state transition functions are:

1. Fw + con → Fw
2. Fw + ~con → *Turn towards Neighbor Density* → Coh
3. Coh + ~con → Fw
4. Coh + con → Left or Right turn → Fw
5. *Bridge Avoidance* + con → *Adopt bridge robot direction* → Fw

The details of bridge avoidance state and mechanism to compute neighbor density are described in the following sections.

### 5.3. THE BRIDGE AVOIDANCE STATE

Consider the figure below; the outer circle is the connectivity radius of the robot shown as small circle in the centre and the shaded zone is called the bridge avoidance zone. Whenever the robot in the center finds any robot(s) in shaded zone, it sends 'Follow' signal to this robot, meaning that robot(s) in this zone should change its angle to the same as the bridged robot in the center.

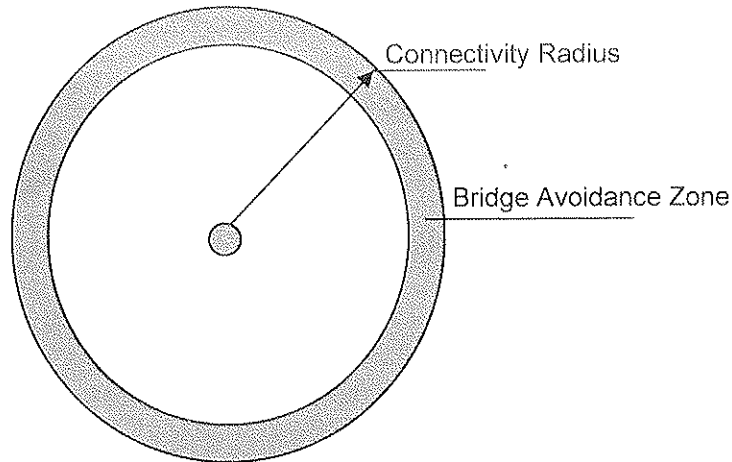


Fig 8: The Bridge Avoidance State

If deadlock occurs between two robots acting as bridge, than the robot with higher ID gets priority.

It should be noted that this bridge avoidance mechanism alone can't guarantee that swarm division will be averted. This is due to the fact in case of disconnection the robot will take  $180^\circ$  u turn, which may result in increase in its distance from bridge robot. After few such turns, the robot will eventually come out of the bridge avoidance zone and will be disconnected from the bridge robot, which is its only link to the swarm. Hence the need arises for changing these blind u turns into intelligent turns.

#### 5.4. NEIGHBOR DENSITY BASED INTELLIGENT TURNS

Robots compute their next state using the same procedure as described in section 3.3. Now for direction computation, if robot is in state as dictated by transition rules one and three, then its direction will remain unchanged. If robot finds fourth state transition function applicable, then it will randomly add or subtract  $90^\circ$  from its current angle and move forward. The most important point comes when the robot is in state as dictated by transition function number two. This rule is at the heart of swarm coherence and now the robot has to decide, in which

direction it has to turn in order to reconnect with the swarm. For this the robot has to check with respect to itself, in which direction its maximum neighbor density is lying. We follow the approach presented in [26] to build Local Coordinate System of robot<sub>m</sub> and find the relative positions of its neighbors.

In this approach the robot becomes the centre of its own coordinate system with the position (0, 0) and the positions of its neighbors are computed accordingly. If a robot<sub>m</sub> can communicate directly (in one hop) with robot<sub>n</sub>,  $n$  is called a one-hop neighbor of  $m$ . Let  $N$  be the set of all the robots in the swarm.  $\forall m \in N$ , we define  $K_m$  as the set of one-hop neighbors of  $m$ . Likewise,  $\forall m \in N$ , we define  $D_m$ , as a set of distances between  $m$  and each robot  $n \in K_m$ . The neighbors can be detected by using IR sensors.

The following procedure is performed at every robot<sub>m</sub>:

- detect one-hop neighbors ( $K_m$ )
- measure the distance to one-hop neighbors ( $D_m$ )
- send the  $K_m$  and  $D_m$  to all one-hop neighbors

Thus, every robot knows its two-hop neighbors and some of the distances between its one-hop and two-hop neighbors. A number of distances cannot be obtained due to the connectivity limitations or the obstacles between the robots. Fig. 9 shows robot<sub>m</sub> and its one-hop neighbors. Continuous lines represent the distances that robot<sub>m</sub> can obtain (by either direct sensing or extracting from exchanged neighbor ID and distances list) while dashed lines represent the distances that cannot be obtained.

By choosing two nodes  $r, s \in K_m$ , such that the distance between  $r$  and  $s$  ( $d_{rs}$ ) is known and larger than zero and such that the robots  $m, r$  and  $s$  do not lie on

the same line, robot<sub>m</sub> defines its Local Coordinate System. The latter is defined such that robot<sub>r</sub> lies on the positive x axis of the coordinate system and robot<sub>s</sub> has a positive  $s_y$  component. In this way, the Local Coordinate System of  $m$  is uniquely defined as a function of  $m$ ,  $r$  and  $s$ .

Thus, the coordinates of the robots  $m$ ,  $r$  and  $s$  are:

$$m_x = 0; m_y = 0;$$

$$r_x = d_{mr}; r_y = 0;$$

$$s_x = d_{ms} \cos \theta_{rs}; s_y = d_{ms} \sin \theta_{rs}$$

Where  $\theta_{rs}$  is the angle  $\angle(r, m, s)$  and it is obtained by using a cosines rule for triangles:

$$\theta_{rs} = \arccos \frac{d_{ms}^2 + d_{mr}^2 - d_{rs}^2}{2d_{ms}d_{mr}}$$

The positions of the nodes  $n \in K_m$ ,  $n \neq r, s$ , for which the distances  $d_{mn}$ ,  $d_{ns}$  and  $d_{rn}$  are known, are computed by triangulation. Therefore, we obtain

$$n_x = d_{mn} \cos \theta_{rn}$$

$$\text{If } \theta_{sn} = |\theta_{rn} - \theta_{rs}| \Rightarrow n_y = d_{mn} \sin \theta_{rn}$$

$$\text{Else } \Rightarrow n_y = -d_{mn} \sin \theta_{rn}$$

Where  $\theta_{rn}$  is the angle  $\angle(r, m, n)$ , and  $\theta_{sn}$  is the angle  $\angle(n, m, s)$ . In practice,  $\theta_{sn}$  will never be exactly equal to  $|\theta_{rn} - \theta_{rs}|$  due to the errors in distance measurements. The purpose of this exercise is to find on which side of the x axis robot  $n$  is located and some difference between those two values needs to be tolerated.



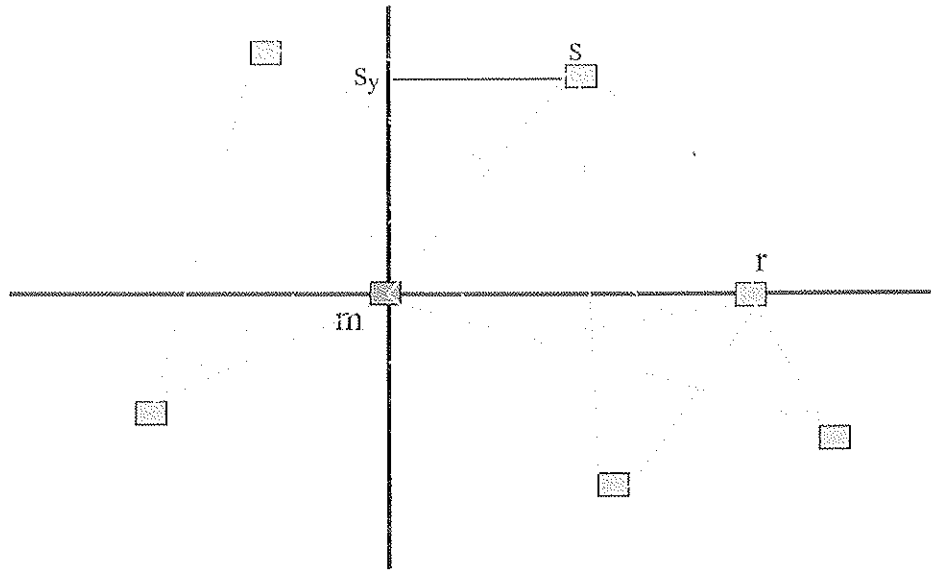


Fig 9: The coordinate system of robot  $m$  is defined by choosing robots  $r$  and  $s$

We obtain the values of  $\theta_m$  and  $\theta_{sn}$  by using the cosine rule

$$\theta_m = \arccos \frac{d_{mn}^2 + d_{mr}^2 - d_{rn}^2}{2d_{mn}d_{mr}}$$

$$\theta_{sn} = \arccos \frac{d_{ms}^2 + d_{mn}^2 - d_{sn}^2}{2d_{ms}d_{mn}}$$

The angles  $\theta_{rn}$ ,  $\theta_{sn}$  and  $\theta_{rs}$  are placed within the triangles  $(r, m, n)$ ,  $(n, m, s)$  and  $(r, m, s)$ , respectively and thus we observe just their absolute values and not their directions.

Fig. 10 shows the example of this computation for robot  $n$ . The positions of the robots  $k \in K_m$ ,  $k \neq r, s$ , which are not the neighbors of robots  $r$  and  $s$ , can be computed by using the positions of the robot  $m$  and at least two other robots for which the positions are already obtained, if the distance from the robot  $k$  to these robots is known.

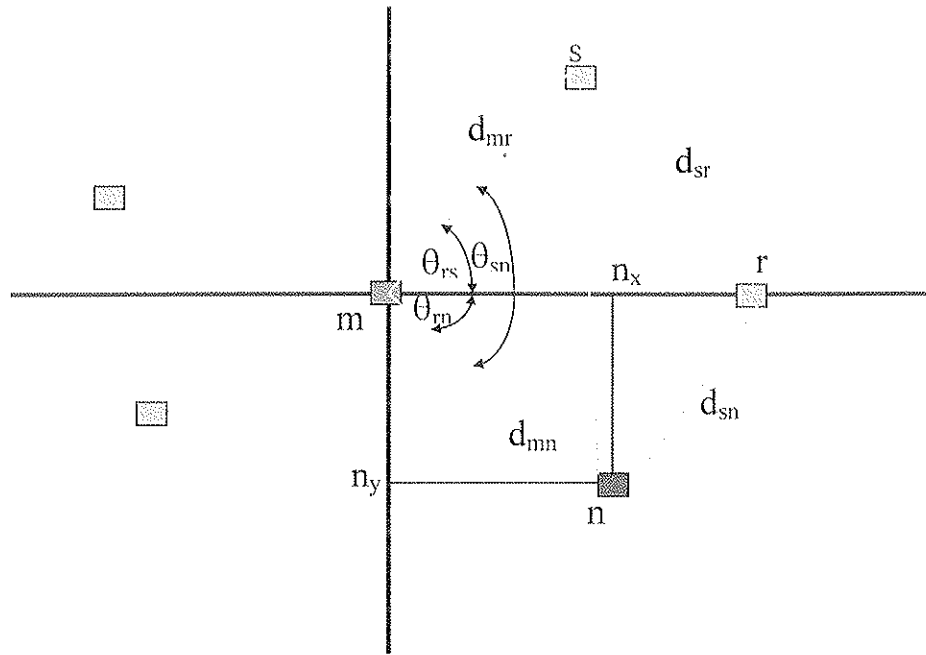


Fig 10: An example illustrating the way to obtain the position of robot *n* in the coordinate system of robot *m*

Once the robot<sub>*m*</sub> has localized all of its neighbors, it then computes its number of 2 hop neighbors by calculating the neighbors of each 1 hop neighbor, through exchanged neighbor ID lists. Finally robot<sub>*m*</sub> computes the total number of 1 and 2 hop neighbors in each quadrant. The quadrant having maximum number of neighbors is designated as the quadrant of maximum neighbor density. The mathematical description of this process is as follows:

We define a matrix *Quadrant*

$$\text{Quadrant} = \begin{bmatrix} \text{quadrant}_{11} & \text{quadrant}_{12} & \text{quadrant}_{13} & \text{quadrant}_{14} \\ \text{quadrant}_{21} & \text{quadrant}_{22} & \text{quadrant}_{23} & \text{quadrant}_{24} \\ \text{quadrant}_{31} & \text{quadrant}_{32} & \text{quadrant}_{33} & \text{quadrant}_{34} \\ \text{quadrant}_{41} & \text{quadrant}_{42} & \text{quadrant}_{43} & \text{quadrant}_{44} \end{bmatrix}$$

Where:

$$\text{quadrant}_{mm} = \text{total number of neighbors (1 and 2 hop) in quadrant}_m \text{ of robot}_m$$

We find the maximum value in each row and thus reduce the above matrix into a matrix *Ndensity*:

$$Ndensity = \begin{bmatrix} ndensity_{11} \\ ndensity_{21} \\ ndensity_{31} \\ ndensity_{41} \end{bmatrix}''$$

Where:

$ndensity_{m1}$  = Quadrant having maximum neighbor density w.r.t robot<sub>m</sub>

If neighbor density is maximum in quadrant 'n', robot *m* will compute the mean angle of neighboring robots present in quadrant 'n' and then turns in that mean angle.

### 5.5. THE MODIFIED LAYERED ARCHITECTURE

The modified layered architecture is shown in the figure below. Now one more layer i.e. the bridge avoidance layer is added to the system.

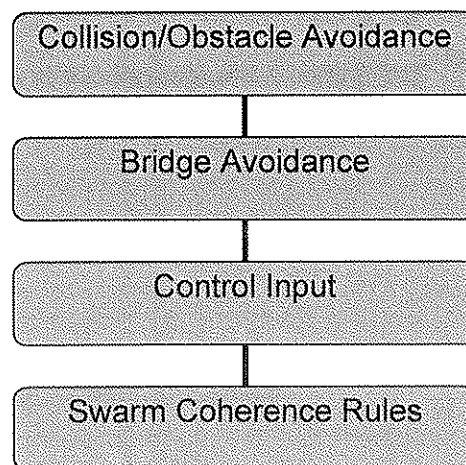


Fig 11: Modified Layered Architecture

## 5.6. RELATIVE ANGLE MEASUREMENT HARDWARE

In section 5.4 we have described how a robot<sub>m</sub> defines its LCS by arbitrarily choosing two neighboring robots  $r$  and  $s$  and assuming that  $r$  lies at positive x-axis while  $s$  has positive y-component. After robot<sub>m</sub> has defined its LCS, it requires actual relative angles of robot  $r$  and  $s$  with itself, in order to rotate its LCS accordingly and find actual relative locations of its neighbors. Bearing and Orientation Measurement Sensor (BOM) [27] shown in figure 12 is the hardware that can be used for such relative angle measurements. It consists of a ring of 16 infrared (IR) emitter/detector pairs and is a quite cheap hardware, costing approximately \$17 (2005 estimate). It is developed at Carnegie Mellon University.

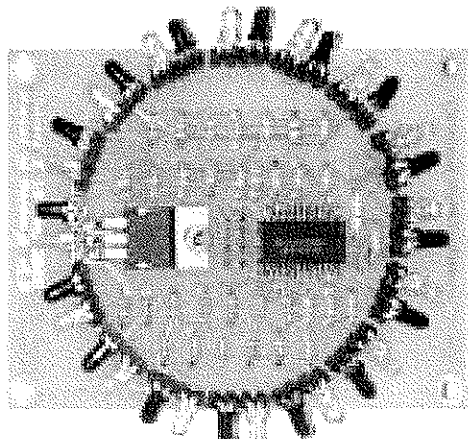


Fig 12: Bearing and Orientation Measurement Sensor

These sensors work in a such a way that one robot lights all of its emitters, creating a ring of uniform IR light, while the rest of the robots in the swarm use their IR detectors to determine the angle of the light source relative to the detecting robot. The use of only 16 detectors provides discrete angle information (accurate to  $\pm 11.25$  degrees).

## RESULTS EVALUATION

This chapter presents the results of simulation carried out in MATLAB to measure the performance of the proposed algorithm. It includes the results which help in identifying the benefits gained through the introduction of neighbor density based intelligent turns and bridge avoidance state. These results highlight the improved performance of the algorithm in terms of increased coherence of swarm and fast emergent motion.

### 6.1. PLATFORM AND HARDWARE

<b>OS</b>	Microsoft Windows XP Professional
<b>Software</b>	MATLAB 7
<b>Processor</b>	Pentium IV
<b>RAM</b>	512 MB

Table 5 Platform and Hardware specification

### 6.2. METRICS

The performance of improved algorithm is analyzed by using the following metrics:

1. Number of connected robots (robots which satisfy alpha value) with respect to time. This metric provides information about the stability of the swarm. If this number is constantly changing and never settles down on a single value, it implies that either the swarm is not moving at all and is just occupied

in rearranging itself in a confined location or is showing very slow haphazard, unstable emergent motion.

2. Number of sub swarms (called as 1-connected swarm in graph theoretic terminology) with respect to time. This metric gives a measure of swarm coherence. A value of one indicates fully coherent swarm, whereas an increased value implies decreasing coherence and robot(s) loss.
3. Distance traveled with respect to time. This metric indicates the type of motion observed. A straight line with constantly increasing slope with respect to time indicates fast emergent motion whereas the opposite indicates slow haphazard motion.

### 6.3. SIMULATION PARAMETERS

Simulations of alpha, beta and improved algorithms were performed by considering the following parameters:

Parameter	Value
Swarm Size	8, 20
Alpha	1 to 15 (depending on swarm size)
Beta	1,2
Initial Positions of Robots	3 different position for each alpha, beta value and 3 readings in each position
Initial Orientation of Robots	Any value from $0^{\circ}$ to $360^{\circ}$

Table.1 Simulation parameters

The results obtained are shown in the graphs described in the following sections.

## 6.4. COMPARITIVE ANALYSIS OF $\alpha$ AND IMPROVED ALGORITHM

The graph shown in figure 13, describes the number of connected robots versus time for alpha algorithm with swarm size of 8 robots. The graph shows that number of connected robots decreases gradually with the increase in alpha value i.e. at alpha value of one, when the swarm gets stable, we have 6 connected robots. On the other hand for alpha value of 6, we just get one to two connected robots. The graph gives information about stability of swarm with time. It can be seen that for lower alpha values the swarm gets stable after some initial adjustments, but for very high alpha value i.e. 6, the swarm never gets stable and just remains occupied in rearranging itself due to the robots unsuccessfully trying to satisfy the high alpha value.

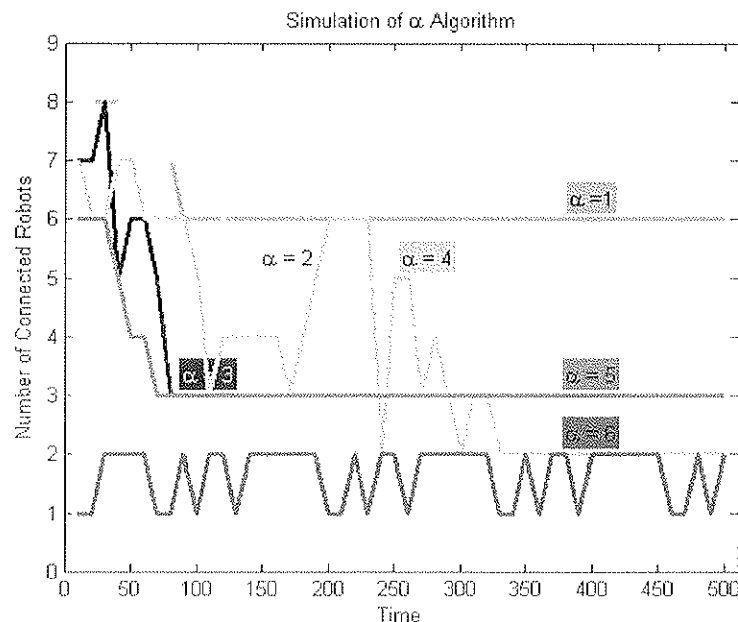


Fig 13: Number of Connected Robots versus time for  $\alpha$  algorithm for swarm size = 8

The second graph in figure 14 shows the number of sub swarms formation with time using alpha algorithm for a swarm size of 8 robots. It can be seen that

number of sub swarms decreases with the increase in alpha value. At alpha value of 1, number of sub swarms observed is 4 but as alpha approaches to higher values of 5 and 6, it can be seen that swarm remains coherent and consequently the number of sub swarms observed is only 1.

Figure 13 and 14, give a complete picture of swarm at any point in time e.g. at 500<sup>th</sup> time step, for  $\alpha=1$  there are total of 6 connected robots as indicated in Fig.13 divided into 4 sub swarms as shown in Fig. 14. It may be noted that six connected robots are divided into 2 sub swarms of size 3 robots each and 2 sub swarms of 1 each.

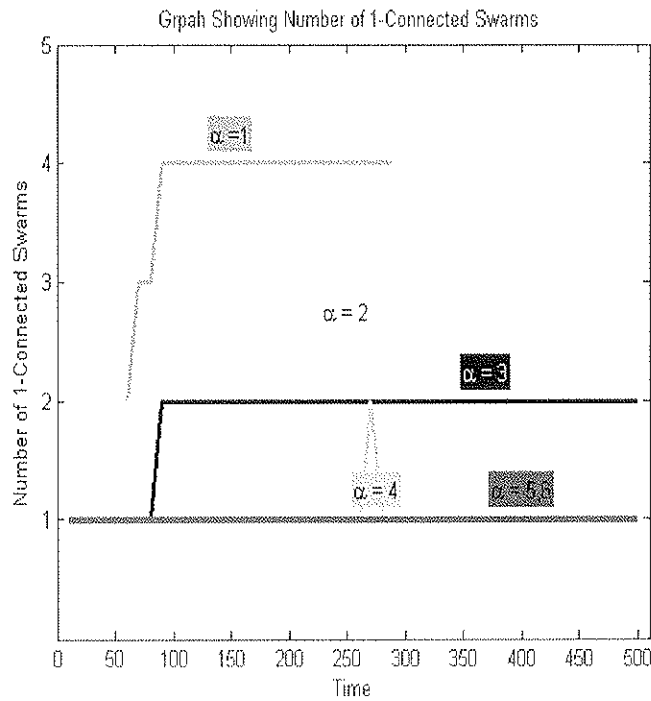


Fig 14: Number of 1-Connected swarms versus time for  $\alpha$  algorithm, swarm size = 8

Figure 15 is a multiple y-axis graph for improved algorithm. Left y-axis along with solid lines show number of connected robots, while right y-axis along with dotted line shows number of 1-connected swarms. The graph indicates that for alpha



values 1 to 3, the swarm stabilizes very fast and all the robots remain connected throughout (satisfy the alpha value). However for alpha value 4 and onwards a decrease in number of connected robots is observed with a corresponding increase in the time to stabilize. Consequently at alpha value 6 the swarm stabilizes at about 150<sup>th</sup> time step with 4 robots connected. The right y-axis shows that swarm remained coherent as one group from alpha value 1 to 6, with no subdivisions observed.

If we compare the results shown in this graph of improved algorithm with ones for alpha algorithm, it is clear that improved algorithm has overcome the problem of swarm sub division, as it remains coherent at all alpha values. Another improvement is that due to introduction of neighbor density based turns, at large alpha values the swarm (even though taking some more time) does stabilize and starts moving, in contrast to alpha algorithm where at higher alpha values, the swarm never stabilizes or shows any emergent motion.

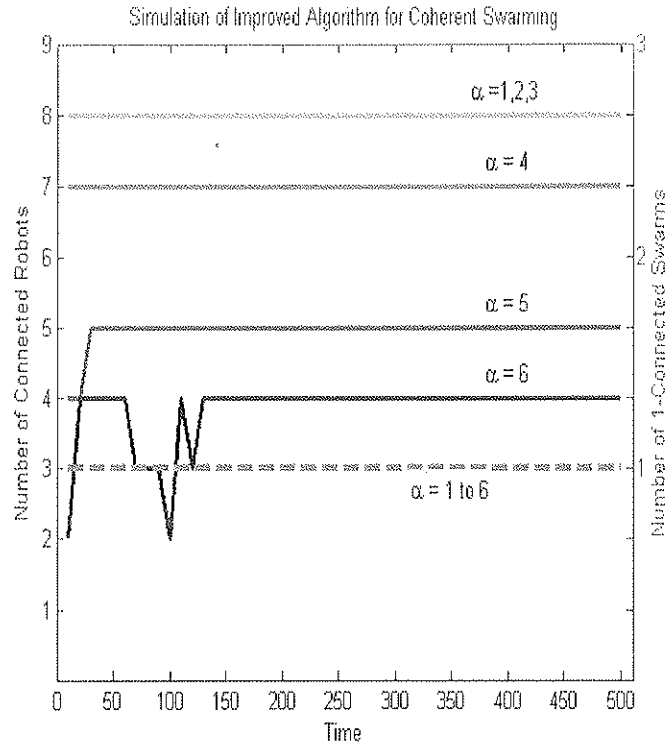


Fig 15: Number of Connected Robots and Number of 1-Connected swarms for Improved algorithm, Swarm Size = 8

Figure 16 shows number of connected robots for alpha algorithm for swarm size of 20 robots. The graph shows similar patterns as observed with the swarm size of 8. The number of connected robots decreases with the increase in alpha value, reaching to zero for alpha value 15. Stability also decreases with the increase in alpha value, as can be seen for the line of alpha value 5. Though straight lines are observed at alpha value 10 and 15 but these don't imply that swarm remains stable. Indeed in this situation where just one robot satisfies the alpha value, all other robots obviously remain occupied in repeatedly moving forward and backward in unsuccessfully trying to satisfy the high alpha value. Hence the swarm never stabilizes or shows any appreciable movement.

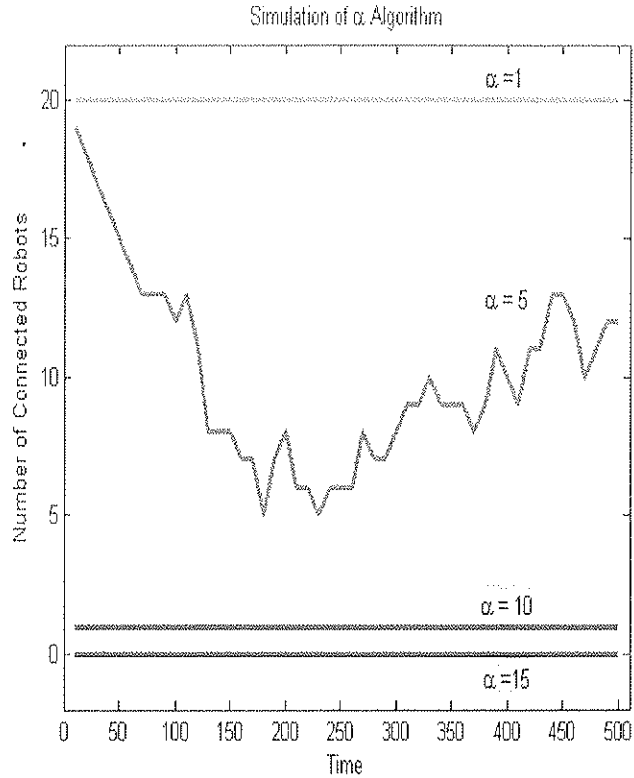


Fig 16: Number of Connected Robots for  $\alpha$  algorithm, Swarm Size = 20

Figure 17 shows graph of number of 1-connected swarms for alpha algorithm, with swarm size of 20. Again due to flaws in alpha algorithm, it can be seen that for alpha value 1, the swarm divides into 5 sub swarms. This number decreases with the increase in alpha, the same trend as seen for swarm size of 8. At alpha value of 10 and onwards, the swarm remains coherent.

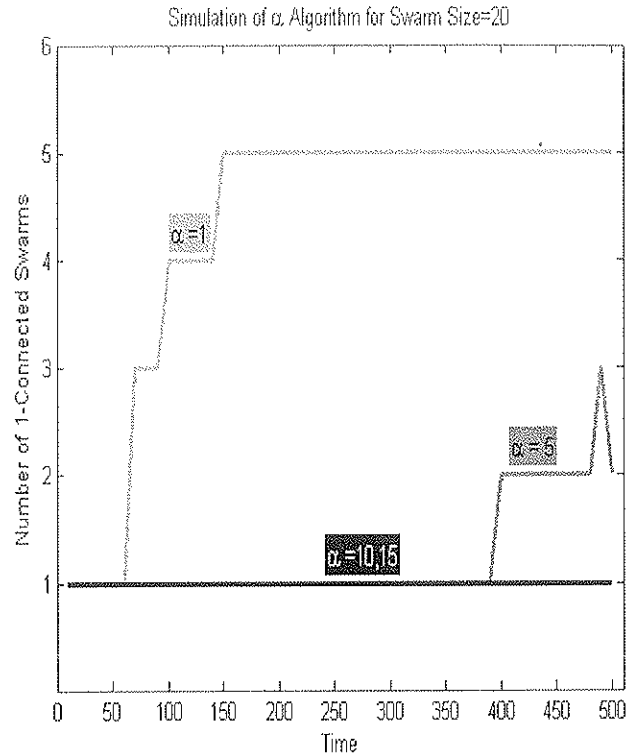


Fig 17: Number of 1-Connected swarms for  $\alpha$  algorithm, Swarm Size = 20

Figure 18 is again a multiple y-axis graph. Left y-axis shows number of connected robots for the improved algorithm, when swarm size is 20. The right y-axis shows number of sub swarms from alpha 1 to 6 and indicates that swarm remained coherent as one group throughout with no subdivisions observed at any time.

It can be seen that at alpha value 1, the swarm stabilized very fast with all the robots being connected throughout. At alpha value 5, little increase in turbulence (resulting in increased time to stabilize) and decrease in number of connected robots is observed. Though number of connected robots reduces to zero at alpha value 15, but still very fast emergent motion is observed in simulations. This is due to the fact that although none of the robot satisfy the alpha value, but this does not result in the swarm robots being engaged in forward and backwards turns, as was

the case with alpha algorithm, instead due to neighbor density based turns and additionally the bridge avoidance mechanism, the robots get stabilize at a common direction quickly, resulting that even at higher alpha values fast and stable emergent motion is observed.

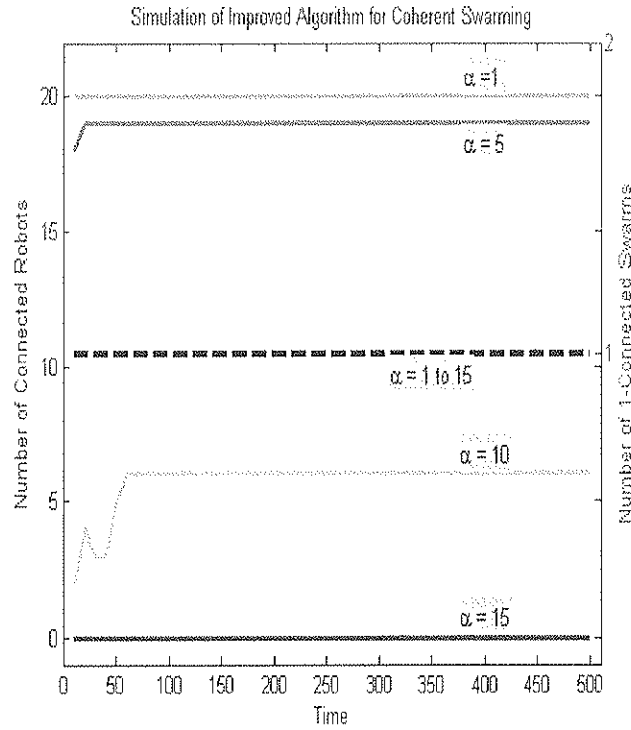


Fig 18: Number of Connected Robots and Number of 1-Connected swarms for Improved algorithm, Swarm Size = 20

### 6.5. COMPARITIVE ANALYSIS OF IMPROVED ALGORITHM WITH COLLISION AVOIDANCE REMOVED AND 25% NOISE ADDED

Figure 19 shows graph of connected robots and 1-connected swarms for improved algorithm when swarm size is 8 and collision avoidance is off. The comparison of this graph with graph 3 of improved algorithm with collision

avoidance on indicates that, no difference is observed for alpha value 1 till 3, but from alpha 4 onwards a decrease in number of connected robots and stabilization time is observed. Consequently it can be inferred that removal of collision avoidance layer resulted in swarm getting stabilized more quickly, regardless of number of robots at that time. Should the avoidance layer was on; it would have resulted in more time for robots to stabilize, thereby also increasing the chance for greater number of robots to satisfy alpha value.

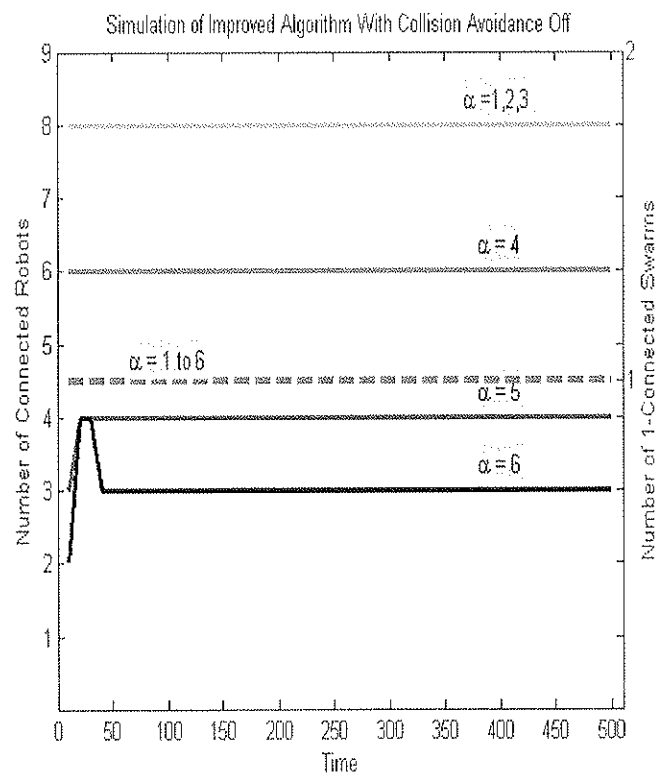


Fig 19: Number of Connected Robots and Number of 1-Connected swarms for Improved algorithm, Swarm Size = 8 with Collision Avoidance Off

Graph of connected robots and 1-connected swarm for improved algorithm with 25% noise is shown in figure 20. Noise is introduced in the simulation by randomly adding or subtracting 25% distance, from the normal Euclidian distance

computed for that time step. The graph indicates that sensor noise has no effect on the output for alpha value 1 but as alpha is increased the effect of erroneous sensor values becomes obvious with a sharp decrease in number of connected robots. Accordingly at alpha value 5, we get 16 connected robots on stabilization compared to 19 when collision avoidance was on. Similarly at alpha 10 zero connected robots are observed in contrast to 6 observed when collision avoidance was on. Overall it can be concluded that algorithm copes gracefully with sensor noise, as number of 1-connected swarm still remain 1 for alpha 1 to 15 (dotted line and right y-axis).

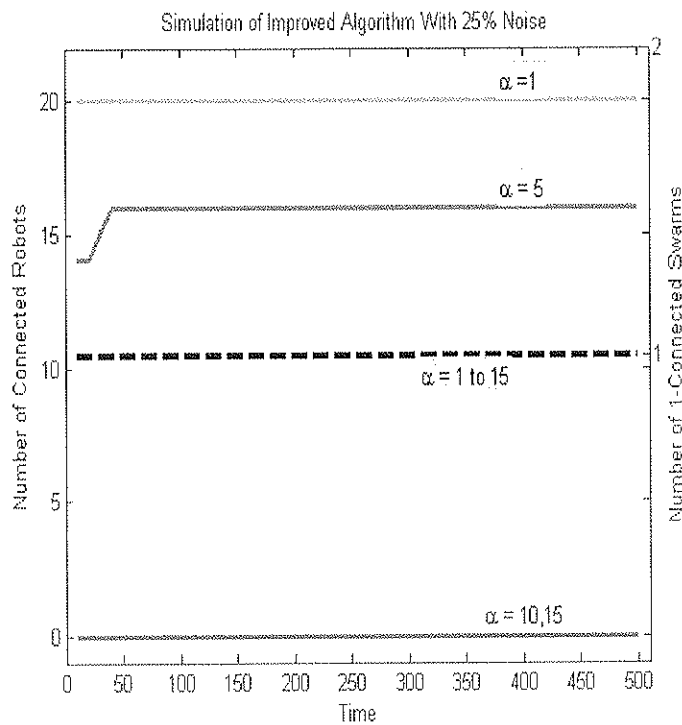


Fig 20: Number of Connected Robots and Number of 1-Connected swarms for Improved algorithm, Swarm Size = 20 with 25% Noise Added

## 6.6. COMPARITIVE ANALYSIS OF IMPROVED AND BETA ALGORITHM

Figure 21 shows graph of distance traveled with respect to time for beta algorithm. Dark grey line is for beta=1 and is indicative of a haphazard motion. As it can be noted from the graph that swarm covered the distance from 5<sup>th</sup> unit to 10<sup>th</sup> unit (total of 5 units traveled) in about 500 time steps while traveling from 20<sup>th</sup> unit to 25<sup>th</sup> unit took a whopping 1000 time steps. This implies that swarm never gets stabilized and exhibits a total random motion. Same haphazard pattern of motion is observed for beta=2 as well. Also it should be noted that these are the results when collision avoidance mechanism is not enabled in the swarm robots.

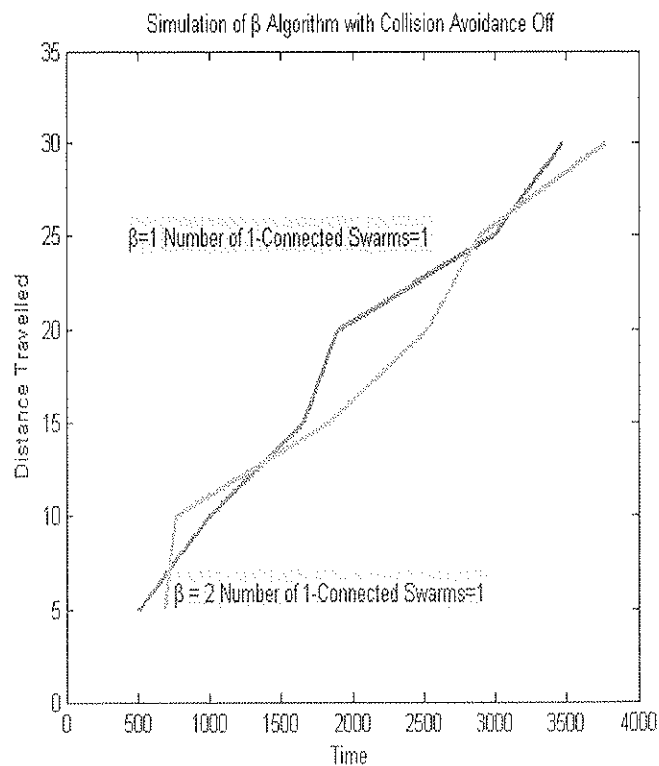


Fig 21: Distance Traveled versus time for beta Algorithm, Swarm size = 8



Figure 22 shows graph of distance traveled with respect to time for improved algorithm. The graph shows that for  $\alpha=1$  (dark grey line), initially the swarm moves with a relatively slow speed as it covers 5 distance units from (5<sup>th</sup> to 10<sup>th</sup> unit) in about 90 time steps, but afterwards it covers the distance with a constant rate of 5 distance units per 50 time steps, implying a stable motion. For  $\alpha=2$ , a constant rate of distance traveled is observed throughout.

If we compare the graph of beta algorithm with improved algorithm, it is clear that beta algorithm is very slow. As for beta algorithm, the swarm covers the

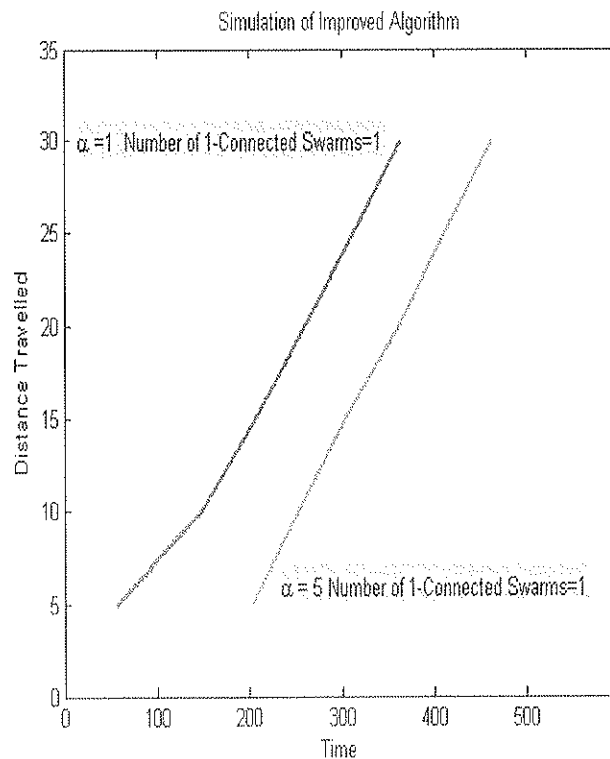


Fig 22: Distance Traveled versus time for Improved Algorithm, Swarm size = 8

distance of 30 units in about 4000 time steps, whereas in case of improved algorithm, the same distance is covered in about 500 time steps. It should be noted further that as these results for beta algorithm are with collision avoidance

off, so it is obvious that enabling collision avoidance layer will have a further slowing effect on the output of beta algorithm. Therefore, it can be asserted that improved algorithm is at least 8 times faster than beta algorithm.

## **6.7. SUMMARY**

Thorough comparison of improved algorithm with alpha and beta algorithms presented in this chapter has proved that due to the introduction of bridge avoidance state and neighbor density based intelligent turns, the improved algorithm has indeed overcome the problem of swarm division encountered in alpha algorithm and slow emergent motion in beta algorithm.

## CONCLUSION

### 7.1. RESEARCH CONTRIBUTIONS

A control theoretic approach based on matrices is presented to model the emergent behavior of a wirelessly connected swarm of robots. We modeled important parameters of the system like connection status of robots, their state and direction as individual matrices and then presented linear equations to model the relationship between these parameters. Collision avoidance between robots is modeled by applying observability concept to the system and consequently an Observer matrix is presented. The Observer matrix notifies a possible collision in the next step thereby triggering a collision avoidance input. Further a control input is defined that directs swarm in the desired direction. The simulation of complete macroscopic model in MATLAB elucidated emergent behavior of swarm over time that would have otherwise obtained through expensive experimentation.

The simulations showed that for alpha algorithm the swarm did not remain coherent and divided into sub swarms with time. For beta algorithm, a very unstable haphazard and slow swarm motion was observed. These observations prompted the need to rectify these problems in order to employ such robotic swarms for some useful practical purpose. Consequently, an improved algorithm was presented for guarantying fast coherent motion. The algorithm overcome the problem of swarm division and slow emergent motion by introducing a bridge avoidance state and intelligent neighbor density based in case of disconnection from the swarm. The results indicate complete swarm coherence at even very low

## REFERENCES

- [1] A. Winfield, C. Harper, and J. Nembrini, 'Towards Dependable Swarms and a New Discipline of Swarm Engineering', in SAB'04 Swarm Robotics workshop, eds. Sahin E and Spears W, Springer-Verlag, LNCS 3342, pp. 126-142, 2004.
- [2] A. Martinoli, K. Easton, and W. Agassounon, "Modeling swarm robotic systems: A case study in collaborative distributed manipulation". In *Int. Journal of Robotics*, volume 23(4), pp. 415-436, 2004.
- [3] A. Martinoli and K. Easton, "Modeling Swarm Robotic Systems". In B. Siciliano and P. Dario, editors, *Proc. of the Eight Int. Symp. on Experimental Robotics ISER-02, Sant'Angelo d'Ischia, Italy*. Springer Tracts in Advanced Robotics 5, pp. 297-306, 2003.
- [4] K. Lerman and A. Galstyan, *A General Methodology for Mathematical Analysis of Multi-Agent Systems*, USC Information Sciences Technical Report ISI-TR-529, 2001.
- [5] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems". *Proc. of the Swarm Robotics Workshop at the Eight Int. Conference on the Simulation of Adaptive Behavior SAB-04*, E. Sahin and W. Spears, editors, Los Angeles, CA, 2004.
- [6] C. Rouff, A. Vanderbilt, M. G. Hinchey, W. Truszkowski, J. Rash, "Properties of a Formal Method for Prediction of Emergent Behaviors in Swarm-Based Systems". *SEFM 2004*: pp. 24-33, 2004.
- [7] C. Rouff, W. Truszkowski, J. Rash, and M. Hinchey, "Formal approaches to intelligent swarms". In *IEEE/NASA Software Engineering Workshop*, pp. 51-57. 2003.
- [8] C. Rouff, M. G. Hinchey, W. Truszkowski, J. Rash, "Verifying Large Numbers of Cooperating Adaptive Agents". *ICPADS (I)*, pp. 391-397, 2005.
- [9] C.A.R. Hoare, *Communicating Sequential Processes*. *Communications of the ACM*, 21(8), pp. 666-677, 1978.
- [10] C. Tofts, "Describing social insect behaviour using process algebra". *Transactions on Social Computing Simulation*, pp. 227-283, 1991.
- [11] M. Holcombe, "Mathematical models of cell biochemistry". Technical Report CS-86-4. Dept of Computer Science, Sheffield University, UK, 1986.
- [12] A. Winfield, J. Sa, M.C. Gago, C. Dixon and M. Fisher, "On Formal Specification of Emergent Behaviours in Swarm Robotic Systems", *Int. J. Advanced Robotic Systems*, 2 (4), pp. 363-370, 2005.
- [13] U. Hustadt, B. Konev, A. Riazanov, and A. Voronkov, "TeMP: A temporal monodic prover". In D. A. Basin and M. Rusinowitch, (Eds.), *Proceedings of the Second International Joint Conference on Automated Reasoning (IJCAR 2004)*, volume 3097 of LNAI, Springer, pp. 326-330, 2004.
- [14] B. Konev, A. Degtyarev, C. Dixon, M. Fisher, and U. Hustadt, "Mechanising first-order temporal Resolution". *Information and Computation*, 199(1-2), pp. 55-86, 2005.
- [15] C.G. Cassandras, "Discrete Event Systems, Modelling and Performance Analysis", IRWIN Aksen Associates, Boston, 1993.

- [16] F.L. Lewis, H.-H. Huang, O.C. Pastravanu, A. Gürel, "A Matrix Formulation for Design and Analysis of Discrete Event Manufacturing Systems with Shared Resources". 1994 IEEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology (Cat. No.94CH3571-5). IEEE. Part vol. 2, New York, USA. pp. 1700-5, 1994.
- [17] J. Mireles and F.L. Lewis, "On the development and implementation of a matrix-based discrete event controller," Proc. Med. Conf. Control and Automation, paper MED01-012, Croatia, 2001.
- [18] D.A. Tacconi, F.L. Lewis, and H.-H. Huang, "Modeling and simulation of discrete event systems using a matrix formulation," Proc. IEEE Mediterranean Symp. Control and Automation, pp. 590-594, Crete, 1996.
- [19] Waqar Mahmood, "Intelligent modeling for control, reconfiguration and optimization of discrete event systems". PhD thesis, Georgia Institute of Technology, 1997.
- [20] J. Nembrini, A. Winfield, and C. Melhuish, "Minimalist coherent swarming of wireless connected autonomous mobile robots". In Proc. Simulation of Artificial Behaviour (SAB'02), Edinburgh, 2002.
- [21] Sarosh Hashmi, Waqar Mahmood, "A Matrix Based Approach for Modeling Robotic Swarm Behavior", In Proceedings of International Conference on Artificial Intelligence, ICAI 06, Las Vegas, USA, 2006.
- [22] Waqar Mahmood, Sarosh Hashmi, Modeling Complex Emergent Discrete Event Systems: A Case Study in Robotic Swarm Motion, 8th WSEAS International Conference on Automatic Control, Modeling and Simulation, ACMOS 06, Czech Republic, pp. 117-122, 2006.
- [23] Waqar Mahmood, Sarosh Hashmi, Modeling Robotic Swarm Behavior with Collision Avoidance: A Discrete Event System Perspective, WSEAS Transactions on Systems, Issue 5, Volume 5, pp. 1137-1143, 2006.
- [24] R. Diestel, "Graph Theory", Springer-Verlag, New York, 2000.
- [25] J. Nembrini, "Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots", PhD thesis, University of the West of England, Bristol, 2005.
- [26] S. Capkun, M. Hamdi, and J. P. Hubaux. "GPS-free Positioning in Mobile Ad Hoc Networks". In 34th International Conference on System Sciences, Hawaii, 2001.
- [27] A. Johnson, C. Atwood, F. Duvall et al., "Relative Localization in Colony Robots", In 19<sup>th</sup> National Conference on Undergraduate Research, NCUR 2005, Virginia, 2005.

# INDEX

## A

Alpha algorithm, 22, 37, 49, 51, 52, 53

Avoidance radius, 30, 34, 36

## B

Beta algorithm, 37, 58

Bridge avoidance, vii, 39, 40, 45, 47, 53, 59

Bridges, 37, 38

## C

Collision avoidance, 57

Communicating Sequential Processes, 19

Control input, 13, 34, 35, 36, 59

Controllability, 16

Cut vertices, ix, 37, 38

## D

Discrete event systems, 13, 20, 61

## F

Feed back, 34

Feed forward, 34

## I

Improved algorithm, 13, 14, 16, 37, 38, 47, 50, 51,

53, 54, 55, 57, 58, 59

## M

Macroscopic model, 17, 28, 59

Microscopic model, 28

Multi-agent systems, 14, 15

## N

Neighbor density, 39, 41, 45, 47, 51, 53, 59

## O

Observability, 16, 59

## S

Swarm intelligence, 15

Swarm robotics, 13, 14

## T

Temporal logic, 19

## U

Unity logic, 19

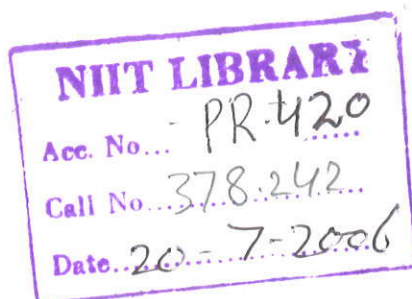
## W

Weighted Synchronous Calculus of

Communicating Systems, 19

## X

X-Machines, 19



SAR  
55

MIT-5

