

# Vector based Control Routines for Swarms of Path Finding Robotic Devices

Colin Chibaya<sup>a</sup>

*Sol Plaatje University, Chapel Road, Kimberley, South Africa*

**Keywords:** XSet, Control Routines, Message Passing, Robotic Devices, Emergent Behaviour.

**Abstract:** Swarm intelligence systems where robotic devices encoded with primitive actions executed at individual levels in order to cause swarm level emergent behaviour are appealing to the fields of nanotechnology and bioinformatics. Interaction between robotic devices allow improved swarm level properties with features more than the sum of the contributions of the individual robotic devices that form the swarm. However, it is challenging to pinpoint particular primitive actions which drive robotic devices towards deliberately engineered emergent behaviour. We propose an XSet model inspired by the behaviours of message passing agents. The proposed XSet model supports direct device to device interactions in which implicit communication spaces arise. In this context, an XSet puts together primitive actions, parameters, and meta information which stipulates when primitive actions are useful to robotic devices. We assess path finding and path following abilities of message passing robotic devices and compared the measures thereof to the relative performances of the stigmergic counterparts. Better message passing performances are observed when time in simulation is sufficiently long, when the population of robotic devices in the swarm is high. Besides giving a new swarm control model, message passing XSets bring us closer to more generalized swarm control rules.


## 1 INTRODUCTION

The aptitude to form paths between selected search points in simulated swarms of robotic devices emanates from particular control routines that are collectively designed to cause emergent behaviour (Chibaya, 2014; Chibaya, 2015; Chibaya, 2019). In this context, a robotic device is an autonomous artificial agent designed with a nanite architecture in mind (Chibaya, 2014). Nanites are tiny electronic devices assembled at nanometre scale. Control routines, on the other hand, are computational codes implemented to characterize robotic devices' individual-level actions (Negulescu and Barbat, 2004). Emergent behaviour is then non-reducible phenomena observed in swarms of robotic devices which cannot be traced back to the individual-level actions of the robotic devices thereof (Chibaya, 2015).

The design of control routines is, dominantly, inspired by some phenomena in nature, such as the map-reading views known from geography (Werfel, 2002; Wehmer et al., 2006), biological metaphors

(Chibaya, 2014), geometric views (Ngo et al., 2005), mathematical matrices (Harris, 2007), or physics (Spears et al., 2004a; Spears et al., 2004b; Spears et al., 2005). Successful control routines have been, commonly, built on stigmergic ant-like colonies (Chibaya, 2015), artificial bee colonies, or flocking birds. However, although related emergent behaviours are plausible, the individual-level actions of swarm members are blurredly explained. Without grasping the logic underpinning individual-level actions of swarm members, simulated swarm applications will remain naïve. In fact, chances are high of evolving undesirably hazardous outcomes from these swarms.

Although they are architecturally autonomous and naïve, robotic devices' interactions give rise to emergent behaviour whose properties are more than the sum of the contributions of the individual robotic devices (Chibaya, 2015). Collections of control routines, together with related parameter values, and meta information, forms the verbs robotic devices depend on in order to complete individual-level tasks. However, what are the white-box design features of

<sup>a</sup> <https://orcid.org/0000-0001-6995-605X>

those control routines included in path finding and path following robotic devices' dictionary?

Formalizing the component units of appropriate sets of control routines with which robotic devices yield emergent behaviour is an ambitious task (Chibaya, 2015). The work presented in Chibaya (2015) proposed the use of an XSet model built on the characteristics of stigmergic ant-like robotic devices. In this context, an XSet is an eXtended Set comprising control routines, parameter values, and meta information stipulating how and when control routines are useful to robotic devices (Chibaya, 2014; Chibaya, 2015). Stigmergic robotic devices interact indirectly via the environment using virtual pheromone cues (Chibaya, 2014; Chibaya 2015). The environment is the shared memory of the swarm. However, stigmergic XSets do not provide details regarding how to arrive at generalizable quantities of pheromone level required by individual robotic devices at a time. Also, they ignore the effects of pheromone dissipation to swarm convergence speed and quality. In fact, the quantities of pheromone levels used are hard-coded (Chibaya, 2014).

Alternative non-stigmergic XSet models can be tried for the same task domain. This paper investigates the design of an XSet model built on the behaviours of message passing robotic devices (Chibaya, 2019). The proposed XSet supports swarms of robotic devices that can directly interact with one another one-on-one (Chibaya, 2014). Message passing robotic devices can explicitly share direction vectors and confidence measures in the vectors they follow, and use this information to determine resultant vectors that determine the next path to follow. Shared vectors, together with related confidence measures, are used to modify robotic devices' perceptions of the direction to the target at the time. Precisely, robotic devices upgrade or downgrade their confidence measures depending on the quality of the information shared amongst neighbours. With time, the swarms thereof converge on deterministic paths towards desired targets.

### 1.1 Statement of the Problem

The particular problem addressed in this paper can be re-phrased into two questions as follows:

- Which control routines form valid message passing XSets? In responding to this question, we investigate the individual actions, the parameter values, and meta information which guide message passing robotic devices towards predictable emergent behaviour. As a case study and proof of concept, we investigate

those control routines useful for the path finding and following behaviour in swarms of robotic devices.

- How do message passing swarms perform relative to the stigmergic counterpart? In responding to this question, we administer an experiment in which we measure speed and quality of emergence that arise from using the message passing XSet versus the speed and quality of emergence yield when a stigmergic XSet model is used for the same path finding and path following task. Speed of emergence evaluates the time it takes a swarm to converge on a trodden path (Chibaya, 2014). On the other hand, quality of emergence establishes the tendencies of robotic devices to deterministically follow the established paths (Chibaya, 2014).

While answers to these two questions may not respond to the very general robotic device control problem, the message passing XSet model is hoped to provide features of an alternative approach to the stigmergic model, which brings us closer to generalized rules for addressing the broader swarm control problem.

### 1.2 Assumptions

We assume message passing robotic devices designed with abilities to use control routines listed in a message passing XSet model. The task at hand is to find and follow paths between selected points situated in a simulated environment. At any time in simulation, each robotic device is either searching for a food-like target or it will be travelling to a nest-like starting point. Perpetual knowledge of the task at hand defines a robotic device's internal state. Internal state information is kept in robotic devices' basic memories, together with the information regarding the direction vectors being followed, as well as the robotic device's confidence measures in those vectors. Robotic devices are able to interpret the vectors and confidence measures held in neighbours' memories by computing their own resultant vectors which fairly represents the directional views of all the neighbour robotic devices around. A searching robotic device uses direction vectors pointing to the food-like target, while a returning counterpart uses direction vectors pointing to the starting point. Robotic devices can relocate towards the direction of the found resultant vector, detecting targets in each step, and checking if they should flip between different internal states when it becomes necessary.

These assumptions connote a message passing XSet model with a finite cardinality, which supports robotic devices with a finite number of internal states.

### 1.3 Overview

Section 2 reviews related works, emphasizing on the design of robotic device control views in which formalization of the rules followed is the key problem addressed. The method we follow in coming up with the message passing XSet model is presented in section 3, focusing on the identification and interpretation of the key control routines required by message passing robotic devices in computational terms. The setup of the message passing XSet model is described in this section. Thereafter, section 4 describes the design of an experiment with which we evaluate and validate the message passing XSet model against the stigmergic counterpart. Speeds and qualities of emergence are measured and reported in section 5. Section 6 concludes the paper, highlighting our key observations and the contributions arising.

## 2 RELATED WORK

Use of XSets to control swarms of robotic devices towards desired emergent behaviour was first reported in Chibaya (2014). Subsequent papers then followed (Chibaya, 2015). However, emphasis has been on the use of stigmergic XSets which lack details regarding identification of generalizable views on the quantities of pheromone levels used by individual robotic devices in each step. More so, they ignore the effects of pheromone dissipation to swarm convergence (Chibaya, 2015). A mathematical model which captures the quantities of pheromone levels used by robotic devices, the size of the scene in which the swarm is deployed, the population of robotic devices required, the amount of time the swarm requires in simulation, and pheromone dissipation factors may, hopefully, lead to more generalizable views. However, such a mathematical model has not been found as yet.

Compelling alternatives to the XSet model exist. Most alternatives, like the stigmergic model, rely on indirect robotic device interactions coordinated via the environment (Di Caro et al., 2004; Negulescu and Barbat, 2004; Bonaneau et al., 1999; Chibaya and Bangay, 2007; Dorigo, 1992; Dorigo et al., 1999). Other alternatives are built on cell propagation theories (Nagpal, 2006), cellular automata (Geer et al., 2003; Green, 1994; Sanders and Smith, 2009), cell growth and morphogenesis theories (Nagpal and

Kondacs, 2002), or origami theories (Rothenmund, 2006). However, related robotic devices must possess substantial memory and extra elitist abilities to be able to handle the computations thereto (Chibaya, 2014). Robotic device orientation is usually based on hard-marked beacons in the scene (Werfel, 2002), landmarks (Wehmer et al., 2006), mathematical models (Ngo et al., 2005), physicomimetic forces (Spears et al., 2004a; Spears et al., 2004b; Spears et al., 2005), or some Jacobian matrices (Harris, 2007). In these cases, robotic devices are able to solve and convert mathematical equations into directional cues. In some cases, robotic devices have physically mounted sensors with which to orientate (Spears et al., 2004b). However, these elitist features are not characteristics of the naïve robotic devices we assume.

A few control strategies embrace message passing views (Trianni and Dorigo, 2005; Rajbhupinder et al., 2010; Rodriguiz et al., 2007). In these, robotic devices may hold blocks of textual messages (Rajbhupinder et al., 2010) to share with other agents of the swarm. However, processing of textual data is equally complex. Useful data values are often lost during text conversion and interpretation. The message passing model we propose assumes robotic devices that can share geometric vectors and the confidence measures associated with using the shared vectors. These are much easier to interpret. Although vector arithmetics are popular in machine learning, network analyses, and spatial data representation, they are a fairly new angle in resolving the robotic device control problem. The emphasis we put on simplicity and specificity in the design of the message passing XSet gives this strategy a computational edge. In our view, a message passing XSet will, potentially, augment the stigmergic counterpart towards notable developments in swarm intelligence systems, particularly agent control issues.

## 3 METHODS

An XSet driven swarm simulator of message passing robotic devices was developed to solve the path finding and path following problem on a simulated environment comprising a food-like target and a nest-like starting point. The default task of the swarm is to locate the food-like target, and upon finding it, travel back to the nest-like starting point. Trips between the food-like target and the nest-like starting point are repeated over and over until a set simulation period lapses.

### 3.1 Configuration of the XSet

Control routines included in the message passing XSet are sequentially listed in the order in which they are used by robotic devices. Meta information include (a) the alias name of the XSet. Stigmergic XSets were referred to as stigXSet (Chibaya, 2014). Message passing XSets are aliased as msgXSet. This alias tells us the inspiring metaphor on which included control routines are built. Another meta data is the cardinality of the XSet. Cardinality tells us how many control routines are required in each robotic device's internal state. Stigmergic robotic devices needed, at most, four control routines in each internal state (Chibaya, 2014; Chibaya, 2015). From repeated tests, message passing robotic devices require one extra control routine in each internal state. The number of internal states and the amount of memory robotic devices need are also important meta data. In this case, four internal states are supported by both stigmergic and message passing robotic devices. A message passing XSet can sufficiently drive robotic devices towards emergent behaviour as long as memory is sufficient to keep internal state information, direction vectors to and from the target, and the confidence measures thereof.

```

msgXSet (cardinality, states, mem)
{
  switch (states)
  {
    case 0: {MsP, PtV, Nrm, MvP, StS}
    case 1: {NOp, NOp, NOp, NOp, StS}
    case 2: {MsP, PtV, Nrm, MvP, StS}
    case 3: {NOp, NOp, NOp, NOp, StS}
  }
}

```

Listing 1: Template of a message passing XSet.

We summarize the design of a message passing XSet in listing 1, where msgXSet is the alias name. The model accepts three parameters, cardinality, number of internal states supported, and amount of memory allocated to each robotic device. In this case, robotic devices can hold up to 8 memory blocks. Control routines and related parameter values are listed between the curly brackets. Each control routine and its related parameter values are listed after each switch-case entry. A control routine and its parameter values are separated by a colon. Parameter values to a control routine are separated by commas. Different control routines are, also, separated by commas. Control routines used in different internal states are demarcated by curly brackets. A code such

as: (MsP:  $V_1, V_2, C$ ) tells a robotic device to share particular direction vectors and confidence measures. The vectors shared are used to determine the next direction to follow. Robotic devices can detect their proximity to targets using the control routine aliased as (PtV:  $V, C$ ). The resultant vectors yield from any computation are normalized using the control routine (Nrm:  $X, Y, Z$ ). Once orientated, robotic devices can relocate to desired destinations using the control routine (MvP:  $X, Y, Z$ ). Should it be necessary at the time, robotic devices can flip between different internal states using the control routine (StS:  $m, n, x$ ). However, there are moments when robotic devices have to do nothing. An empty control routine aliased as (NOp:) is used. The next section describes the computational interpretation of each of these control routines.

### 3.2 Computational Design of Routines

In XSets, control routine names have three letters, e.g. MsP, PtV, Nrm. These control routines are primarily used by robotic devices for sharing vectors, detecting targets, normalizing vectors, making movements, or flipping between different internal states. This section discusses the computational interpretation of these control routines.

We indicated earlier on that message passing robotic devices require some memory in which to store internal state information, vectors, and confidence measures. Keeping internal state information in memory spells out a robotic device's tasks at the time (Panait and Luke, 2004a, 2004b, 2004c). It influences the behaviour of nearby robotic devices (Parunak, 2005). For example, a robotic device may flip to another internal state triggered by a nearby robotic device's proximity to the target (Parunak, 2005). In some cases, such flips occur as a reward for an achievement (Panait and Luke, 2004a).

Robotic devices in the search internal state communicate with counterparts in the returning internal state because those robotic devices likely know where the food-like target is. Robotic devices in the returning internal state prefer interactions with members in the searching internal state because those robotic devices would likely know better about the direction towards the nest-like starting point. Internal state changes are repeated every time a target is hit.

Listing 2 interprets the process through which robotic devices achieve the flipping between different internal states in computational terms. The mnemonic (StS:  $m, n, x$ ) summarizes the control routine for flipping between internal states. It tells a robotic device to set its internal state to mode  $m$  on condition



```

bool StS ( m , n , x )
{
    for-each robotic-device  $R_t$ 
    if ( n is true in domain x )
    {
        internal state = m
    }
}

```

Listing 2: Flipping between internal states.

that requirements  $n$  are satisfied in domain  $x$ . In our case,  $m$  ranges between 0 and 3, representing the four possible internal states message passing robotic devices support. Then,  $n$  is a set of conditions which indicates aspects of the simulation to trigger a robotic device's interests in changing from one internal state to another. Inclusion of  $x$  sets the domain in which  $n$  is satisfied. For example, (StS:1,0,0) tells a robotic device to change to internal state case 1, provided that the target indicators marked as 0 are in quantities above 0 at the robotic device's current location.

Message passing robotic devices can share message blocks (Trianni and Dorigo, 2005) of the format:  $(x_i; y_i; \vec{f}_v; f_w; \vec{n}_v; n_w)$ . In these message blocks,  $(x_i; y_i)$  is the offset of a communicating  $i^{\text{th}}$  robotic device. Communication is allowed between a path finding robotic device and those robotic devices whose offsets are within the set communication range. The path finding robotic device self-localizes relative to nearby robotic devices, placing itself at the origin of its local coordinate system. The choice of which vectors are attractive at the time depends on the path finding robotic device's internal state. When searching, robotic devices are attracted to  $\vec{f}_v$  (food vectors), weighed by  $f_w$  (confidence measures in  $\vec{f}_v$ ). The path finding robotic device accumulates the vectors read from all robotic devices around, each weighted by its related  $f_w$ . A resultant vector is calculated, which overwrites the path finding robotic device's current  $\vec{f}_v$ . Returning robotic devices are attracted to  $\vec{n}_v$ , weighed by  $n_w$ . They also find a resultant vector which points in the likely direction of the nest-like starting point. Every time a resultant vector is calculated, either  $f_w$  or  $n_w$  of the path finding robotic device is upgraded or downgraded depending on the quality of the information received. Below, we show how resultant vectors are calculated.

### 3.3 Calculation of Resultant Vectors

A similar approach where vectors are shared before resultant vectors are calculated was used in Ngo et al. (2005). Resultant vectors summarize nearby robotic

devices' past experiences (Rodriguez et al., 2007). These combined experiences upgrade or degrade a path finding robotic devices' knowledge and confidence in the vector it is following.

Upon deployment, robotic devices randomly pick  $\vec{f}_v$  and initialize  $f_w$  to a minimal value possible. The hope is that  $f_w$  would improve as robotic devices gain more knowledge of the environment through interactions with other robotic devices. The local coordinate system created by robotic devices span over three simulated grid cells in 2D spaces. Figure 1 shows a typical local coordinate system with eight possible paths a robotic device can follow. The task of the path finding robotic device is to decide on a direction to take based on the information shared from nearby robotic devices.

To find that resultant vector, a message passing robotic device calculates intersection points and points of closest approaches between all possible pairs of vectors read from neighbours. The notion is that, two vectors representing the knowledge of two independent robotic devices would intersect at a point close to the target. That vector which starts from the origin of the local coordinate system to the intersection point of the two vectors is a candidate direction to follow next. In the event of a pair of vectors not intersecting within the defined scene, points of closest approach between the selected direction vectors are determined. Once a set of all intersection points and points of closest approach is established, message passing robotic devices use least squares point estimation to pick a fair direction vector which represents the knowledge gathered, at the same time updating or downgrading its own confidence measure in the new selected direction.

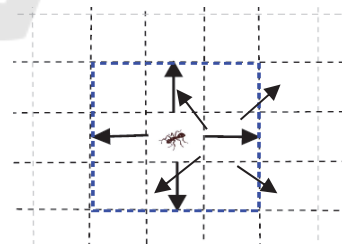


Figure 1: coordinate system for path finding robotic devices.

Figure 2 shows a set of intersection points and points of closest approach for an arbitrary case. However, how do we geometrically find those values of  $x$  and  $y$  at which two vectors intersect or where points closely approach each other? The problem of determining these coordinates is geometric.

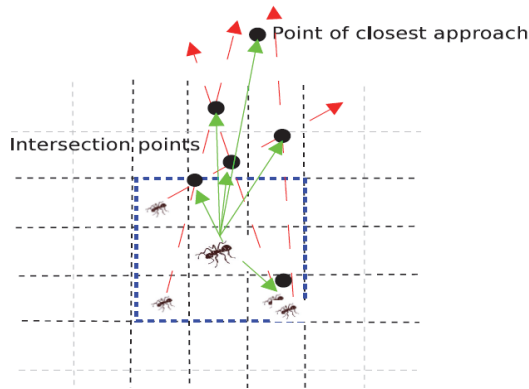


Figure 2: Intersection points and points of closest approach.

If we denote the vector followed by the  $i^{\text{th}}$  robotic device as  $\vec{d}_i$ , and that followed by the  $j^{\text{th}}$  robotic device as  $\vec{d}_j$ . Suppose the offsets of the  $i^{\text{th}}$  and  $j^{\text{th}}$  robotic devices relative to the path finding robotic device are  $(x_i; y_i)$  and  $(x_j; y_j)$  respectively.

Then the points along each line segment represented by vectors  $\vec{d}_i$  and  $\vec{d}_j$  are  $\vec{d}_i + s \times \vec{d}_i$  and  $\vec{d}_j + t \times \vec{d}_j$  respectively. Here,  $\vec{d}_i$  and  $\vec{d}_j$  are unit vectors of  $\vec{d}_i$  and  $\vec{d}_j$ . Parameters  $s$  and  $t$  are magnitudes. Let the points at  $\vec{d}_i + s \times \vec{d}_i$  and  $\vec{d}_j + t \times \vec{d}_j$  be  $(x_{i+1}; y_{i+1})$  and  $(x_{j+1}; y_{j+1})$  respectively. The intersection point of  $\vec{d}_i$  and  $\vec{d}_j$  is therefore  $(x; y)$ , where  $x$  and  $y$  are computed as shown in equations (1) and (2). With two points along each vector, we find the point at which two vectors meet (Wikipedia, 2004) by finding the matrix determinants of the coordinates of the points identified along each vector.

$$x = \frac{\begin{vmatrix} x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \\ x_j & y_j & 1 \\ x_{j+1} & y_{j+1} & 1 \end{vmatrix}}{\begin{vmatrix} x_i & 1 \\ x_{i+1} & 1 \\ x_j & 1 \\ x_{j+1} & 1 \end{vmatrix} \begin{vmatrix} y_i & 1 \\ y_{i+1} & 1 \\ y_j & 1 \\ y_{j+1} & 1 \end{vmatrix}} \quad (1)$$

$$y = \frac{\begin{vmatrix} x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \\ x_j & y_j & 1 \\ x_{j+1} & y_{j+1} & 1 \end{vmatrix}}{\begin{vmatrix} x_i & 1 \\ x_{i+1} & 1 \\ x_j & 1 \\ x_{j+1} & 1 \end{vmatrix} \begin{vmatrix} y_i & 1 \\ y_{i+1} & 1 \\ y_j & 1 \\ y_{j+1} & 1 \end{vmatrix}} \quad (2)$$

When vectors do not intersect, we get a point of closest approach. If  $L_i = \vec{d}_i + s \times \vec{d}_i$  and  $L_j = \vec{d}_j + t \times \vec{d}_j$  are equations of the line segments along vectors  $\vec{d}_i$  and  $\vec{d}_j$ , the point at which these two vectors have a minimum offset,  $w = L_i - L_j$ , is the point of closest approach. This point is when  $w$  is perpendicular to  $L_i$  and  $L_j$ . At this point,  $w \times \vec{d}_i = 0$  and  $w \times \vec{d}_j = 0$ . Algorithm 1 shows how  $w \times \vec{d}_i = 0$  and  $w \times \vec{d}_j = 0$  are used to solve for  $s$  and  $t$ , and how we find the point where  $w$  is smallest. Two vectors are parallel when  $ac - b^2 = 0$ . Robotic devices pick a midpoint along the line segment  $w$  as the wanted point of closest approach. The set of  $(x; y)$  value of intersection points and points of closest approach for all pairs of vectors taken from nearby devices is the data from which a robotic device finds its new path. Hopefully, the vector found fairly represents the consensus of nearby robotic devices.

Algorithm 1: Determining the values of  $s$  and  $t$ .

Let  $w = L_i - L_j$   
 $= (\vec{d}_i + s \times \vec{d}_i) - (\vec{d}_j + t \times \vec{d}_j)$

Therefore

$$w \times \vec{d}_i = (\vec{d}_i + s \times \vec{d}_i) - \vec{d}_j - t \times \vec{d}_j \times \vec{d}_i = 0 \quad (3)$$

$$w \times \vec{d}_j = (\vec{d}_i + s \times \vec{d}_i) - \vec{d}_j - t \times \vec{d}_j \times \vec{d}_j = 0 \quad (4)$$

From equation (1) above, after expanding the brackets

$$d_i \times \vec{d}_i + s \times \vec{d}_i \times \vec{d}_i - d_j \times \vec{d}_i - t \times \vec{d}_j \times \vec{d}_i = 0$$

$$= s \times \vec{d}_i \times \vec{d}_i - t \times \vec{d}_j \times \vec{d}_i = \vec{d}_i \times (d_j - d_i) \quad (5)$$

From equation (2) above, after expanding the brackets

$$d_i \times \vec{d}_j + s \times \vec{d}_i \times \vec{d}_j - d_j \times \vec{d}_j - t \times \vec{d}_j \times \vec{d}_j = 0$$

$$= s \times \vec{d}_i \times \vec{d}_j - t \times \vec{d}_j \times \vec{d}_j = \vec{d}_j \times (d_j - d_i) \quad (6)$$

Let  $\vec{d}_i \times \vec{d}_i = a$   
 $\vec{d}_i \times \vec{d}_j = b$   
 $\vec{d}_j \times \vec{d}_j = c$   
 $\vec{d}_i \times (d_j - d_i) = d$   
 $\vec{d}_j \times (d_j - d_i) = e$

Substituting in equations (3) and (4),  $t$  and  $s$  are:

$$t = \frac{ae - bd}{ac - b^2}$$

$$s = \frac{be - cd}{ac - b^2}$$

$$\sum y = a_1 \sum x + mb_1 \quad (7)$$

$$\sum xy = a_1 \sum x^2 + b_1 \sum x \quad (8)$$

$$\sum x = a_2 \sum y + mb_2 \quad (9)$$

$$\sum xy = a_1 \sum y^2 + b_2 \sum y \tag{10}$$

$$a_1 = \frac{m \sum xy - \sum x \sum y}{m \sum x^2 - (\sum x)^2} \tag{11}$$

$$b_1 = \frac{m \sum y \sum x^2 - \sum x \sum xy}{m \sum x^2 - (\sum x)^2} \tag{12}$$

To pick a direction, least squares point estimation (Chibaya, 2014) is used on that set of (x ; y) coordinates. Two least squares regression lines arise, one for y on x, and another for x on y. The regression line for y on x is  $y = a_1x + b_1$ , and that for x on y is  $x = a_2y + b_2$  (Chibaya, 2014). Equations (7) and (8) determine  $a_1$  and  $b_1$  in  $y = a_1x + b_1$ , while equations (9) and (10) find  $a_2$  and  $b_2$  in  $x = a_2y + b_2$ . In both cases,  $m$  is the population of robotic devices around the path finding robotic device. Equations (11) and (12) show how we simplify equations (7) and (8) for  $a_1$  and  $b_1$ . Flipping  $x$  and  $y$  in these equations gives the formula for finding  $a_2$  and  $b_2$  in equations (9) and (10). Least squares point estimator finds the point at which the two regression lines intersect. That point coincides with the centre of mass ( $\bar{x}$ ;  $\bar{y}$ ). A vector starting from the origin of the local coordinates system to the centre of mass is the resultant vector we want.

### 3.4 Updating Confidence Measures

We indicated that each vector has an associated confidence measure which indicates how well a robotic device has performed in previous orientation choices. Confidence measures indicate the robotic device’s trust in the path followed (Chibaya, 2014). They reflect the quality of the vectors previously followed. These are float values between 0 and 1, where 1 indicates awareness of the direction to the target and 0 indicates complete ignorance of the scene features. A robotic device updates its confidence measure by combining the confidence levels of nearby robotic devices with its own using equation (13).

$$w_i(t + 1) = \frac{1}{2} \left( w_i(t) + \frac{\sum_{j \in k} w_j(t)}{k} \times (1 - c) \right) + \lambda \tag{13}$$

If there exist  $k$  nearby robotic devices to a path finding robotic device, then let the vector held in each of the  $k$  robotic devices be denoted as  $j$ . Therefore  $\frac{\sum_{j \in k} w_j(t)}{k}$  is the average confidence measure of the  $k$  robotic devices. To minimize the effects of outliers, we find the spread of the views of the  $k$  robotic devices and denote it as  $c$ , which is within the range

0 to 1. To penalize larger  $c$ , we use  $(1-c)$  as the desired weight. An average of  $w_i(t)$  of the path finding robotic device and  $w_i(t)$  of the  $k$  nearby robotic devices gives the updated  $w_i(t+1)$ . Some randomness ( $\lambda$ ) is added to allow independence in robotic devices’ actions. Randomness is especially useful when robotic devices are isolated from the rest. That way, robotic devices’ confidence levels would never deplete.

### 3.5 The Message Passing Routine

Listing 3 presents the message passing routine. The routine has three parts, one where vectors are shared in order to create a set of intersection points and points of closest approach, another part where the least squares point estimator is used to pick the vector, and a part where confidence measures are updated. In the mnemonic: (MsP:  $v_c, v_j, v_j$ ),  $v_c$  are blocks in robotic devices’ memories where confidence measures are recorded. Then,  $v_j$  are blocks in robotic device memories where attractive vectors are held. The third parameter indicates blocks in memory in which the resultant vectors are recorded after they are determined. Robotic devices overwrite their old vectors in  $v_j$  by the vectors yield. The routine tells a robotic device to read vectors held in nearby devices’ memories, read related confidence measures, and find a vector to follow.

```

bool MsP (Vc , Vj , Vj)
{
    for-each robotic-device Rt
    for-each robotic-device Kt
    {
        pos(Kt) = (xk , yk , zk) - Dk
        for-every-other Kv
        {
            pos(Kv) = (xv , yv , zv) - Dv
            if (Dk ∩ Dv)
            {
                x = calculated from (1)
                y = calculated from (2)
            }
        }
    }
    else
    {
        x=random x; y=random y
    }
    Add (x , y) to array []
    c = avg (sdvev (x , y))
    path-wght=avg (w (t) , sum (w (t) k) × c+β
}
    
```

Listing 3: The message passing primitive instruction.

```

bool Nrm (x, y, z)
{
  length = x2 + y2 + z2
  if (length ≠ 0)
  {
    x=x/length; y=y/length; z = z/length
  }
  else
  {
    x = random x
    y = random y
    z = 0.0
    path-weight = 0.00001
    Nrm (x, y, z)
  }
}

```

Listing 4: Normalizing vectors.

```

bool PtV (pi, x)
{
  for-each pos-L around device Rt
  {
    if (Q(pi) @ L > x)
    {
      path (Rt+1) = (Lx, Ly, Lz)
      path-weight (Rt+1) = 1.0
    }
  }
}

```

Listing 5: Detecting target indicators.

```

bool MvP (x, y, z)
{
  for-each robotic-device Rt
  {
    if (Rt = (x, y, z))
      Rt+1 = (x+xi, y+yi, z+zi)
  }
}

```

Listing 6: Robotic device movement.

Magnitudes of vectors are variable. They dictate robotic devices' movement steps. Normalizing these vectors reset the magnitudes to 1, defining unit robotic device steps. Listing 4 summarizes the semantics for normalizing a vector. The control routine receives the  $x$ ,  $y$ , and  $z$  components of a vector and use these values to find the length of the vector. This length is non-zero when a robotic device has neighbours. Components of the vector are divided by this length. Isolated robotic devices yield vectors of magnitude 0, after which the robotic devices follows randomly picked direction vectors with depleted confidence measures. Random vectors are recursively normalized.

It is critical that robotic devices detect targets (Cavalcanti and Freitas, 2005) in order to trigger internal state changes. Points where key objects exist in the simulation environment are marked by target indicators (Cavalcanti and Freitas, 2005). Target indicators are virtual chemicals set when the environment is launched. A method is required with which message passing robotic devices can detect target indicators and appropriately interpret this into vector information. This is not the first time vectors have been used to interpret pheromone levels (Chibaya, 2014). A magnitude of zero indicate that the robotic device is not yet on target. Listing 5 describes target detection and conversion of related information to vectors. It accepts two parameters, one passing the ID of the target indicator and another setting the lowest level of target indicators to trigger changes in robotic devices' actions. If detected, the robotic device overwrites its vector by a vector pointing to the location of the target indicators. This upgrades confidence measures to the highest measure possible, indicating absolute trust in the vector to the target.

Movement is the robotic device's last task in each cycle. Listing 6 interprets the movement policies run after orientation. Three parameters indicating the offset of the chosen destination are received into the routine. In these offset values, the  $z$  component is always 0 as we operate in 2D.

Meta information stipulates each XSet's precise cardinality. However, there are moments when robotic devices require fewer control routines than stipulated. (NOp:) tells robotic devices to do nothing when it becomes necessary. This is merely an empty routine.

## 4 EXPERIMENTS SETUP

An experiment is administered in which to assess path finding and path following properties in swarms controlled by message passing and stigmergic XSets (Chibaya, 2014).

The platform on which robotic devices are assessed for path finding and path following abilities is called an environment (Chibaya, 2014). In computational terms, an environment is a grid of rows and columns which intersect to form cells which we regard as locations. A location is practically a tuple which stores information such as target indicators. For experimental purposes, we assume a  $1000 \times 1000$  grid environment. Target and the starting point are hard coded at particular locations to allow repeatable tests.



Robotic devices are allowed to run for, as a case study, 10,000 steps in which to score performances. However, simulation time can be changed without compromising the desired outcomes. Each swarm consists of, as a case study, 5000 robotic devices.

A performance metric measures the extent to which emergent behaviour is manifest as a result of robotic devices using the control routines listed in an XSet. Two performance metrics are of interest in this study. First, quality of emergence measures adherence of robotic devices to the schedule, evaluating robotic devices' engagement with the task at hand (Werfel et al., 2006). On the other hand, speed of emergence evaluates timeliness. Combined, the two metrics yield an index of merit of the XSet used.

To measure speed, the time it takes the first robotic device to find the target is determined. Time is measured in iterations. Then, the time it takes the last robotic device in the swarm to also find the target is found. The time gap between these two time intervals is the speed of emergence towards the target. Speed of emergence towards the starting point is the time gap between the times taken by the first and last robotic device to complete round trips. The average between speed of emergence towards the target and speed of emergence towards the starting point is the overall speed of emergence of the swarm. Ten repeated simulations administered, before centrally placed speed measures, are reported.

To determine quality of emergence, we set a time frame in which successful trips of robotic devices in each direction are counted and recorded. Ten repeated tests are also administered in order to achieve smoothened and centrally placed performance trends.

The index of merit is determined by scaling speed and quality measures so that they lie in the range [0; 1]. Equation (14) shows how the average of the scaled metrics is found, giving an index of merit of an XSet.

$$\text{Index of merit} = 0.5 \times ((1 - S/T) + Q/T) \quad (14)$$

The hypothesis which drove the experiment administered is: message passing XSets are a useful alternative for controlling swarms of robotic devices towards emergent behaviour. The null hypothesis thereto is: there aren't significant differences in performance between swarms coordinated using message passing and stigmergic XSets. The dependent variables in this experiment are XSets' indices of merits. Two independent variables are: the time taken in simulation and the control levels at which measures of emergence are extracted. In this case, performance measures are extracted at intervals of 1,000 iterations. All other variables are controlled,

including the population of robotic devices, size of the environment, the positions of the starting point and the food-like target, the cardinalities of XSets, number of internal states, and the sizes of the memory blocks supported by robotic devices. Listing 7 summarizes the design of the experiment, showing the procedure followed.

<b>Title</b>	Assessment of the validity of a message passing XSet
<b>H<sub>0</sub></b>	Message passing XSets are an alternative protocol to stigmergic XSets. This suggests that there are no significant performance difference between message passing and stigmergic swarms tasked to path find.
<b>H<sub>1</sub></b>	Message passing XSets closely resemble random wandering XSets. This suggests that stigmergic XSets outperform message passing XSets.
<b>Dependent variable:</b>	indices of merits (calculated using average speeds and average qualities of emergence).
<b>Independent variables:</b>	simulation time, control levels.
<b>Controlled variables:</b>	population of robotic devices, environment size, position of starting point, position of food-like target, cardinalities of XSets, number of internal states, and size of memory.
<b>Procedure</b>	Generator functions which create the environments are invoked. Robotic devices are deployed to score performances over ten repeated tests. Speeds and qualities of emergence are extracted in each repeat and used to find the index of merit of the XSet in that cycle. Average index of merit over ten iterations are reported at each control level. Final measures are compared.
<b>Algorithm</b>	<pre> for-each swarm driver [rndXSet ; stigXSet ; msgXSet]   repeat     find speed at iteration 1000, 2000, ..., 10000     find quality at iteration 1000, 2000, ..., 10000     find index of merit at 1000,2000,..., 10000     keep record of index of merit at each level   until 10 replications   Find overall indices of merit next swarm compare overall indices of merits Report relative performances                     </pre>

Listing 7: Experiment design.

## 5 RESULTS

The key results reported are speeds of emergence, qualities of emergence, and indices of merits of message passing and stigmergic XSets. In these results, measures of central tendencies and dispersion are important. Correlations extracted at a 99% level

of confidence are also key. The sample size we use is statistically small, so two-tailed correlation tests are administered because we cannot tell, in advance, the sign of the correlation coefficients we will get. The critical correlation value we use is 0.765 when alpha is 0.01. Absolute values of the correlation coefficient above 0.765 indicate sufficient evidence to accept the hypothesis that there are significant relationships between the speeds and qualities of emergence yield.

Figure 3 compares the indices of merits yielded from using the two XSets. The x-axis is marked by the control levels. The y-axis shows the average scaled indices of merits thereof. The speeds and qualities of emergence found satisfy Kolmogorov-Smirnoff tests for normality with a statistical p value above 0.05. A correlation coefficient of 0.91 is yielded when speed and quality of emergence in the stigmergic category are compared, implying a strong correlation between speed and quality of emergence in that model. The message passing counterpart yielded a correlation coefficient of 0.86 when the same metrics are compared. Chances are therefore slim that any generalized views we deduce from these results would significantly differ from the trends and relationships shown.

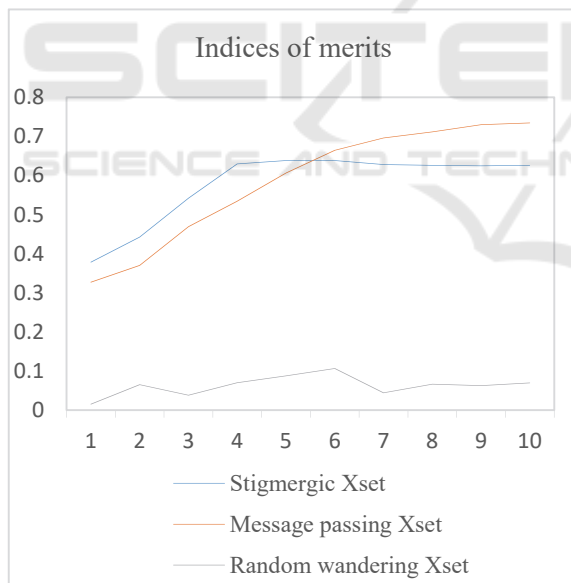


Figure 3: Comparisons between indices of merits.

We have sufficient evidence to accept the hypothesis that the two XSets are alternatives to one another, both achieving plausible outcomes. We make the following additional observations:

- The stigmergic XSet reaches a turning point in performances with time in simulation, after which swarm performances deplete. A turning

point in performances occurs when the environment gets saturated with pheromone levels which reverts robotic devices to a random wandering mode. We noted in (Chibaya, 2015) that pheromone dissipation could be, to some extent, the remedy to this flaw. On the contrary a message passing XSet improves related swarm performances in Sigmoid-like patterns. Related performances gradually improve until robotic devices converge on deterministic paths. Thus, while extended time in simulation is detrimental to stigmergic swarm performances, it allows message passing swarms to build vector fields with deterministic cues to the targets. Swarm performances are therefore a function of time in simulation in both cases.

- Stigmergic control routines come in specific sequences in order to yield good performances. Similarly, message passing control routines are executed sequentially. The configuration of the two XSets is similar, connoting possibilities of a generalized XSet.
- Message passing and stigmergic XSets both influence swarm behaviour. They are, both, useful swarm drivers. This is a milestone in the study of robotic systems.

The results provide sufficient evidence to support the view that message passing XSets are an alternative to stigmergic XSets. No significant differences are observed between their performances, with both XSets showing causal properties.

## 6 CONCLUSION

A new XSet for controlling swarms of robotic devices towards desired formations is proposed. The new XSet is inspired by the behaviours of message passing agents. Controlling swarms of robotic devices requires us to provide five meta data during the configuration of the XSet. We need to provide the alias name of the XSet, cardinality, number of internal states supported, amount of memory, and the separators used to demarcate the XSet's entries. Six routines form the vocabulary of message passing XSets as follows:

- (MsP:) – a control routine with which robotic devices share vectors, finding resultant vectors to follow next, updating their confidence measures, and orientating in each step.

- (PtV:) – a control routine for detecting target indicators, updating confidence measures, and orientating robotic devices towards the targets.
- (MvP:) – a routine for relocating robotic devices.
- (Nrm:) – a routine for normalizing vectors.
- (StS:) – a control routine which allows robotic devices to conditionally flip between different internal states.
- (NOP:) – telling robotic devices to do nothing.

An experiment to evaluate the message passing XSet for causal properties showed that this is an alternative swarm control protocol to the stigmergic version. Four contributions emanate from this work:

- The design of the message passing XSet adds to new developments towards practical use of robotic device based swarm intelligent systems.
- The control routines used are creative, adding relevant content to the robotic device control and programming problem.
- The metrics used to measure the performances of XSets are innovative. These metrics can be useful in verifying other forms of emergent behaviours, opening up new research avenues in areas related to quantification of emergency.
- The statistical tests applied during validation of the XSets and tests for normality on the results are also innovative. Similar statistical tests may inspire the development of more scientific and deductive outcomes with positivism angles.

Although the general robotic device programming problem is not resolved, this work brings us closer to such generalization. It provides a baseline upon which further investigations may arise. More so, the work strengthens the foundation set when the stigmergic XSets were identified. Importantly, the investigations undertaken may soon inspire the development of more generic control routines with which robotic devices, in general, would engineer predictable object assembly.

## ACKNOWLEDGMENTS

We acknowledge support from the department of Computer Science at Sol Plaatje University, for allowing us time to work on this article, the brotherly advices, and financial support, without which this work would not have been a success. However, professor Shaun Bangay remains our most inspiring mentor ever.

## REFERENCES

- Chibaya, C. 2019. A Message Passing XSet for Path Finding Robotic Devices. *In the proceedings of the 1<sup>st</sup> International Multidisciplinary Information Technology and Engineering Conference*.
- Chibaya, C. 2014. An investigation into XSets of primitive behaviours for emergent behaviour in stigmergic and message passing ant-like agents. *A PhD thesis submitted at Rhodes University*. South Africa.
- Chibaya, C. 2015. An XSet based protocol for coordinating the behaviour of stigmergic ant-like robotic devices. *In the ACM International Conference Proceeding Series and the SAICSIT' 2015*. South Africa.
- Geer, P, McLaughlin, H,W, Unsworth, K. 2003. Cellular lines: an introduction. *In Discrete mathematics and theoretical computer science*.
- Green, D, G.. 1994. Emergent behaviour in biological systems. *Complexity international*.
- Sanders, J, W, Smith, G. 2009. Refining emergent properties. *International Institute for software technology. Electronic Notes in Theoretical Computer Science*.
- Di Caro, G, Ducatelle, F, Gambardella, L,M. 2004. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *Technical report Dalle Molle Institute*.
- Negulescu, S,C, Barbat, B, E. 2004. Enhancing the effectiveness of simple multi-agent systems through stigmergic coordination. *In the 4<sup>th</sup> symposium on engineering of intelligent systems*.
- Bonaneau, E, Dorigo, M, Theraukaz, G. 1999. Swarm Intelligence: From natural to artificial systems. *Oxford*.
- Chibaya, C, Bangay, S. 2007. A probabilistic movement model for shortest path formation in virtual ant-like agents. *In the ACM International Conference Proceeding Series, and In SAICSIT 2007 on IT research in developing countries*.
- Dorigo, M. 1992. Optimization, learning and natural algorithms. *A PhD thesis submitted to the Dipartimento di Electronica, Politecnico di Milano*. Milan, Italy.
- Dorigo, M, Di Caro, G, Gambardella, L,M. 1999. Ant algorithms for discrete optimization. *In the Proceedings of Artificial Life*.
- Nagpal, R. 2006. Self-Organizing shape and pattern: From cells to robots. *IEEE Intelligent Systems*.
- Nagpal, R, Kondacs, A, Chang, C. 2002. Programming methodology for biologically-inspired self-assembling system. *In the Proceedings of the American Association for Artificial Intelligence*.
- Rothmund, P, W, K. 2006. Folding DNA to create nano-scale shapes and patterns. *Computation and neural systems and computer science*.
- Werfel, J. 2002. Autonomous multi-agent construction. *Amorphous projects. On-line on [http://groups.csall.mit.edu/mac/projects/amorphous.6.978/nal\\_nal.pdf](http://groups.csall.mit.edu/mac/projects/amorphous.6.978/nal_nal.pdf)*.
- Wehmer, R, Boyer, M, Loertscher, F, Sommer, S, Menzi. 2006. Ant navigation: one-way routes rather than maps. *Current biology. Elsevier Science Ltd*.

- Ngo, V, T, Nguyen, A,D, Ha, H. 2005. Integration of Planning and Control in Robotic Formations. *In the Proceedings of the 2005 Australian Conference on robotics and Automation.*
- Spears, W, M, Spears, D,F, Zarzhitsky, D, Heil, R. 2004a. Physicomimetics for mobile robot formations. *In the 3<sup>rd</sup> International conference on autonomous agents and multi agent systems.*
- Spears, W, M, Spears, D, F, Hamann, J, C, Heil, R. 2004b. Distributed, physics-based control of swarms of vehicles. *Autonomous robots.*
- Spears, W, M, Spears, D, F, Zarzhitsky, D. 2005. Physicomimetics positioning methodology for distributed autonomous systems. *In the Proceedings of the government microcircuit applications and critical technology conference, Intelligent Technologies.*
- Harris, D, M, J. 2007. Direct motion of a parallel-linkage robot through the Jacobian. *12<sup>th</sup> IFToMM World Congress, France.*
- Trianni, V, Dorigo, M. 2005, Self-Organisation and communication in groups of simulated and physical robots. *Technical report: Université Libre de Bruxelles. Biological Cybernetics.*
- Raijbhupinder, S,D, Harwinder, S, S, Amarpreet, S, G. 2010. Load Balancing of Ant Based Algorithm in MANET. *International journal of computer science and technology.*
- Rodriguez, S, Salazar, R, McMahon, T, Amato, N, M. 2007. Roadmap based group behaviour: generation and evaluation. *Technical report: Parasol Lab, Computer Science.*
- Panait, L, Luke, L. 2004a. A pheromone-based utility model for collaborative foraging. *In the 3<sup>rd</sup> International Joint Conference on Autonomous Agents and Multi-Agent Systems.*
- Panait, L, Luke, L. 2004b. Ant foraging revisited. *In the 9<sup>th</sup> International conference on simulation and synthesis of living systems.*
- Panait, L, Luke, L. 2004c. Learning Ant Foraging Behaviours. *In the 9<sup>th</sup> International Conference on the Simulation and Synthesis of Living Systems.*
- Parunak, H, V, D. 2005. A survey of environments and mechanisms for human-human stigmergy. *In the 2<sup>nd</sup> International conference on environments for Multi Agent Systems.*
- Wikipedia. 2004. Line-line intersection. *Wikipedia.*
- Cavalcanti, A, Freitas, R, A. 2005. Nanorobotics control design: A collective behaviour approach for medicine. *IEEE Transactions on NanoBioScience.*
- Werfel, J, Nagpal, N, Seung, H, S. 2006. Ant-hills built to order: automating construction with artificial swarms. *A PhD thesis submitted to the MIT.*
- Gengan, D, Schoeman, M, A, Van der Poll John, A. 2014. An ant-based mobile agent approach to resource discovery in grid computing. *In the proceedings of the SAICSIT 2014.*