2-2017

Generating swarm solution classes using the Hamiltonian Method of swarm design

M. Li
Jisan Research Institute

C. Qiu
University of California Santa Barbara

J. Park
Jisan Research Institute

D. Chan
Jisan Research Institute

J. Na
Jisan Research Institute

See next page for additional authors

Authors
M. Li, C. Qiu, J. Park, D. Chan, J. Na, C. Wong, B. Zhao, E. Chang, Sanza Kazadi, and S. Hettiarachchi

Generating swarm solution classes using the Hamiltonian Method of swarm design

M. Li[1], C. Qiu[2], J. Park[1], D. Chan[1], J. Jeon[1], J. Na[1], C. Wong[1], B. Zhao[1], E. Chang[1], S. Kazadi[1,3], S. Hettiarachchi[4]

[1]*Swarm Research Group, Jisan Research Institute, 308 S. Palm Ave., Alhambra, CA 91803, USA*

[2]*University of California Santa Barbara, Santa Barbara, CA 93106, USA*

[3]*Illinois Mathematics and Science Academy, 1500 Sullivan Rd., Aurora IL 60506, USA*

[4]*Indiana University Southeast, 4201 Grant Line Rd., New Albany IN, 47150, USA*

skazadi@imsa.edu

Keywords: Swarm Engineering, Hamiltonian Method of Swarm Design

Abstract: We utilize a swarm design methodology that enables us to develop classes of swarm solutions to specific specifications. The method utilizes metrics devised to evaluate the swarm's progress – the global variables – along with the set of available technologies in order to answer varied questions surrounding a swarm design for the task. These questions include the question of whether or not a swarm is necessary for a given task. The Jacobian matrix, here identified as *the technology matrix*, is created from the global variables. This matrix may be interpreted in a way that allows the identification of classes of technologies required to complete the task. This approach allows us to create a class of solutions that are all suitable for accomplishing the task. We demonstrate this capability for accumulation swarms, generating several configurations that can be applied to complete the task. If the technology required to complete the task either cannot be implemented on a single agent or is unavailable, it may be possible to utilize a swarm to generate the capability in a distributed way. We demonstrate this using a gradient-based search task in which a minimal swarm is designed along with two additional swarms, all of which extend the agents' capabilities and successfully accomplish the task.

1 Introduction

Swarm engineering (Kazadi, 2000) as an area of research concerns itself with the translation of a task description into a design of multiple autonomous agents whose combined effect accomplishes the desired task. One method involves utilizing differential equations derived from state equations to generate behaviors. The Hamiltonian Method of Swarm Design (HMSD) is a method for swarm design that begins with functions called *global functions* defined over the field of agent measurables (Kazadi and Lee, 2007). This method utilizes global swarm properties to generate an abstract phase space. The initial and final system positions define the task; behaviors implemented by the agents must change the global properties' numerical values so as to move the system from the initial state to the final state. Design of the swarm involves the analysis of the practical generation of a set of behaviors, techniques, and timetables required to move the system from the initial to the final state.

In this paper, we will examine how the HMSD can be used to determine the requirements for classes of swarms that accomplish specific tasks. We begin with an examination of the way in which the task encoded as a function of measurements that individual agents can accomplish. We continue to a derivation of the *technology matrix* which defines the classes of technologies necessary for the accomplishment of the task. The technology classes are encoded in terms of their ability to change the system state as opposed to a description of the specific technology used. We demonstrate that, using this approach, we can create *accumulation swarms* of varied designs that accomplish the accumulation task. Additionally, we use the *source location task* to explore the development of a technological requirement that may or may not be possible to accomplish on a single agent. In the case that a single agent-based solution doesn't exist, the problem must be solved using a swarm. We derive three different swarms that can accomplish the task, demonstrating that in this case, a swarm is absolutely

required in order to accomplish the task.

2 The Technology Matrix

The Hamiltonian Method of swarm design (Kazadi and Lee, 2007; Kazadi, 2009), involves the writing of global variables P_i which are meant to capture the current state of the swarm. These are functions of quantities that measure locally measurable aspects of the system which the individual agents can measure and manipulate. Mathematically, these are represented by vectors $\overrightarrow{s} = (s_1, \ldots, s_{N_s})$ where N_s is the number of local measurables and each s_i represents the state of one element of the system. It is clear to see that, with each agent potentially having K degrees of freedom and the system itself having more, the number N_s can be quite large. Each of the values, s_i, is assumed to come from a range S_i of real numbers, making the system configuration $\overrightarrow{s}$ an element of $S_1 \times \cdots \times S_{N_s}$ which is itself a subset of $\mathbb{R}^{N_s}$.

2.1 The Swarm Design Problem

As each of the P_i is a function of the local variables, we write this as

$$P_i = f_i\left(\overrightarrow{s}\right). \tag{2.1}$$

P_i is meant to capture a global state of the system. Many examples may be developed for potential global goals including the location of items that are being collected, the flow rate of items that are being moved, construction details, etc. Each of these may be represented as a potentially complex function of the local measurables. Each of the functions may have a unique numerical value corresponding to a specific state of the system or, at least, a unique numerical value for the desired system state.

One might imagine writing several functions P_i describing aspects of the swarm system. A vector $\overrightarrow{P} = (P_1, \ldots, P_{N_P})$ results from the combination of the global properties. The number of global properties N_P greatly determines the complexity of the design problem. We define N_P as the *swarm design dimension*. Therefore, if $N_P = 1$ the swarm design problem is referred to as a *1-d swarm design problem*. If $N_P = 2$, the swarm design problem is referred to as a *2-d swarm design problem*.

The vector $\vec{P}$ is an element of the vector space $A = A_1 \times \cdots \times A_{N_P}$ where each A_i represents the range of each P_i. A is the *systemic phase space*. Each element represents a specific set of states. That is, given a state $\overrightarrow{P}$, there is a number of system states for which the global state is $\overrightarrow{P}$. We can define the system domain $I\left(\overrightarrow{P}\right)$ in the following way $I\left(\overrightarrow{P}\right) = \left\{\overrightarrow{s} \,|\, \overrightarrow{P} = \left(f_1\left(\overrightarrow{s}\right), \ldots, f_{N_P}\left(\overrightarrow{s}\right)\right)\right\}$. If $\left|I\left(\overrightarrow{P}\right)\right| = 1$ then we say that the system state is well defined.

The goal is to change the state of a system from an initial state to some defined final state. This is defined in terms of the global properties. Therefore, the goal is to change $\overrightarrow{P_{init}}$ to some final desired state $\overrightarrow{P_{fin}}$. However, it is clear that both of these states may be extremely degenerate in the sense that many microscopic configurations may make this happen. We define the *task* as either the ordered pair of $\left(\overrightarrow{P_{init}}, \overrightarrow{P_{fin}}\right)$ or, in the case that the initial state is ambiguous, $\longrightarrow \overrightarrow{P_{fin}}$ meaning that all initial states should go to the final state. That is, the final state $\overrightarrow{P_{fin}}$ is an attractor of the system under the dynamics we will construct.

As a result, we are interested in the dynamics of the properties $\overrightarrow{P}$. Let us consider now $\frac{dP_i}{dt}$. First, it is straightforward that, using the Einstein summation convention,

$$\frac{dP_i}{dt} = \frac{\partial P_i}{\partial s_j}\frac{ds_j}{dt}. \tag{2.2}$$

this can be rewritten as

$$\frac{dP_i}{dt} = \overrightarrow{\nabla P_i} \cdot \dot{\overrightarrow{s}}. \tag{2.3}$$

Defining $\mathbf{P}$ as

$$\mathbf{P} = \begin{pmatrix} \frac{\partial P_1}{\partial s_1} & \frac{\partial P_1}{\partial s_2} & \cdots & \frac{\partial P_1}{\partial s_{N_s}} \\ \frac{\partial P_2}{\partial s_1} & \frac{\partial P_2}{\partial s_2} & \cdots & \frac{\partial P_2}{\partial s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_{N_P}}{\partial s_1} & \frac{\partial P_{N_P}}{\partial s_2} & \cdots & \frac{\partial P_{N_P}}{\partial s_{N_s}} \end{pmatrix} \tag{2.4}$$

we can write the time change of the system as

$$\dot{\overrightarrow{P}} = \mathbf{P}\dot{\overrightarrow{s}}. \tag{2.5}$$

We can then write the goal of the swarm as

$$\overrightarrow{P_{fin}} = \overrightarrow{P_{init}} + \Delta_{t_0}^{t_f} \mathbf{P}\dot{\overrightarrow{s}}\, dt \tag{2.6}$$

when the initial and final points are well defined and as

$$\overrightarrow{P_{fin}} = \overrightarrow{P_{init}} + \Delta_{t_0}^{\infty} \mathbf{P}\dot{\overrightarrow{s}}\, dt \tag{2.7}$$

if the initial point is not well defined.

2.2 Connecting to the real world

In the real world, we can describe the swarm engineering problem differently than the somewhat esoteric description given above. Generally speaking, the goal of a swarm is to solve a task. That is, given a task, we want to determine the following items:

- The technologies that one must use
- The way to deploy the technologies
- The minimal number of agents to use
- The number of teams/groups to assemble
- The number of agents that will be "consumed", or lost, during the task
- The time needed to accomplish the task
- The amount of energy required to accomplish the task
- The supplies required to accomplish the task

These items are, interestingly, either present in the equations given above in Section 2.1 or capable of being calculated as a result.

Note that the quantity $\dot{\overrightarrow{s}}$ represents the way in which the local measurables are manipulated. These local measurables are assumed to be accessible to the agents; they are things that they can directly measure and change. This quantity defines how and, ultimately, when these measurables must be changed. While the details of the method of changing the quantity must be developed by an engineer, this provides a clear design specification.

The quantities $\frac{\partial P_i}{\partial s_j}$ represent the ways in which changes in the local properties are connected to the changes in the global properties. For instance, these may indicate that the value of $\frac{ds_j}{dt}$ is coupled with a value for $\frac{\partial P_i}{\partial s_j}$ that is positive. Therefore, the system must have some way of determining $\frac{\partial P_i}{\partial s_j}$. If it is the case that P_i must decreased in order for the global goal to be achieved, this requirement determines a potential course of action for the swarm: $\frac{ds_j}{dt}$ should be the opposite sign of that of $\frac{\partial P_i}{\partial s_j}$.

The manner in which $\frac{ds_j}{dt}$ couples to changes in P_i is determined by the $\frac{\partial P_i}{\partial s_j}$. In order to couple the two quantities together correctly, it must be possible for the agents to determine what $\frac{\partial P_i}{\partial s_j}$ is. This can be achieved only if there is a mechanism or technology capable of determining this. The matrix $\mathbf{P}$, therefore, represents the superset of all of the sensory, communication, and/or computational technologies that must be deployed in order to achieve the task. That is, the non-manipulative requisite technologies or capabilities for the task exist within the matrix $\mathbf{P}$. Yet, understandably, these technologies are not represented by the names and vendors of the technologies to be used. Rather, they represent the capabilities of the requisite technology; how that is achieved is up to the discretion of the swarm designer. Note that if any technology $\frac{\partial P_i}{\partial s_j}$ does not actually exist, this technology will either not be used in the solution or will be developed in order to solve the swarm design problem. Developing these capabilities is a way of discovering needed technologies one may not have realized is necessary.

Likewise, the various elements $\frac{ds_j}{dt}$ that are nonzero in order to lead to the desired final system configuration $\overrightarrow{P_f}$ represent the agent-level capabilities that must be in place in order to change the s_j values within the system. These might be lights, grippers, methods of movement, actuators, internal computations, or any other method of changing some part of the system. Therefore the $\frac{ds_j}{dt}$ entries represent the manipulative technologies that must be in place in order to achieve the task.

Together, these two technological requirements define the superset of all of the technologies and abilities required in order for the task to be achieved. The system engineer, then, is tasked with identifying or creating them. *A swarm is appropriate in the case that these abilities can be generated through the interaction of agents when they are not available or possible with a single agent.*

Once we have a set of actuation technologies, or perhaps if we obtain a list of currently available technologies, we can determine that the application of any given technology together with the actuation mechanisms at a given point in the system configuration will lead to a change in the system configuration $\Delta\overrightarrow{s}$. This change will have a concomitant change in the position of the system in phase space, which may be approximated as $\mathbf{P}\Delta\overrightarrow{s}$. Therefore, our goal is to determine the sequence of applications of actuation technologies that leads the system from the initial point to the final point. That is, the swarm engineering design problem consists of determining a set of applications of actuation technologies and concomitant steps through phase space $\left\{\Delta\overrightarrow{s_l}\right\}_{l=1}^{N_{steps}}$ for which the resulting change in $\overrightarrow{P}$ yields

$$\overrightarrow{P_f} = \overrightarrow{P_i} + \sum_{l=1}^{N_{steps}} \mathbf{P}\Delta\overrightarrow{s_l}. \tag{2.8}$$

As $\mathbf{P}$ may be a function of the state of the system, the problem may be more generally written as

$$\overrightarrow{P_f} = \overrightarrow{P_i} + \sum_{l=1}^{N_{steps}} \mathbf{P}\left(\overrightarrow{s_l}\right)\Delta\overrightarrow{s_l}. \tag{2.9}$$

As a result of this sequence of applications, we can now determine many of the desired quantities. Clearly, we have determined the technologies to use as well as their deployment schedule. Each of the manipulations will require at least one agent. In the case that more than one agent is required, a *team* is

needed. When a manipulation requires the transition of an agent from an active to an inactive state, this "consumes" an agent. Summing those for which a reverse transition is not applied along the pathway indicates how many agents will be consumed. Each of the transitions requires an amount of energy, supplies, time, etc. Representing the consumption as γ_l, we can write out the consumed quantity as

$$\gamma_{consumed} = \sum_{l=1}^{N_{steps}} \gamma_l. \tag{2.10}$$

This expression allows us to calculate what we will need to accomplish the task along the pathway.

While the determination of this set of applications of actuation technologies provides a mathematically rigorous definition of a swarm strategy that will accomplish the task, its determination can be nontrivial. In the remainder of this paper, we will examine two relatively simple swarms that can be designed using this approach.

3 Accumulation swarms

An *accumulation swarm* is a swarm that, when initially organized with agents physically far from one another, responds by moving the agents toward one another to eventually form a tightly packed group. A swarm behaving in this way might be a precursor to a swarm that then forms a well-ordered formation for a secondary purpose.

Mathematically, accumulation swarms are defined as follows: Suppose that each member of the swarm has a position given by $\overrightarrow{x_i}$. Let the entire swarm have a set of positions given by $\overrightarrow{s} = (\overrightarrow{x_1}, \ldots, \overrightarrow{x_N})$. Then we can define a global property P as

$$P\left(\overrightarrow{s}\right) = \sum_{i<j}^{N} \left(\overrightarrow{x_i} - \overrightarrow{x_j}\right)^2. \tag{3.1}$$

P is the dispersion of the group. An accumulation swarm will have the defining property that $\frac{dP}{dt} < 0$. Note that

$$\frac{dP}{dt} = 2\sum_{i<j}^{N} \left(\overrightarrow{x_i} - \overrightarrow{x_j}\right) \cdot \left(\frac{d\overrightarrow{x_i}}{dt} - \frac{d\overrightarrow{x_j}}{dt}\right). \tag{3.2}$$

This can be simplified to

$$\frac{dP}{dt} = 2\sum_{i}^{N} \frac{d\overrightarrow{x_i}}{dt} \cdot \left(\sum_{j\neq i} \overrightarrow{x_{ij}}\right) \tag{3.3}$$

where $\overrightarrow{x_{ij}} = \overrightarrow{x_i} - \overrightarrow{x_j}$. If we let $\overrightarrow{x_{M,i}} = \left(\sum_{j\neq i} \overrightarrow{x_{ij}}\right)$ then the change in P is given as

$$\frac{dP}{dt} = 2\sum_{i}^{N} \frac{d\overrightarrow{x_i}}{dt} \cdot \overrightarrow{x_{M,i}}. \tag{3.4}$$

We may identify $\overrightarrow{x_{M,i}}$ as the center of mass of the remaining agents from the point of view of the i^{th} agent.

This swarm is defined entirely in terms of its desired behavior by a single global property. As the swarm requirements are defined in terms of only a single global function, it is ***a single dimensional swarm***.

Examining the form of (3.4), we see that there are two main terms, $\frac{d\overrightarrow{x_i}}{dt}$ and $\overrightarrow{x_{M,i}}$. These two terms define the capabilities of the agents that must be in place in order for the swarm to perform as desired. That is, in particular, the swarm must have the ability to move, and it must have the ability to determine the center of mass of the system. Therefore, these two basic capabilities define the technological requirements for the swarm.

In order for the swarm to coalesce, we must have

$$\frac{dP}{dt} < 0. \tag{3.5}$$

I.e., the sum must be negative. The minimal requirement for this to happen the magnitude of the summed negative values should exceed that of the summed positive values. There are many ways in which this can be accomplished including making each of the terms negative. Making each of the terms negative requires that the angles between $\overrightarrow{x_{M,i}}$ and $\frac{d\overrightarrow{x_i}}{dt}$ exceed 90°. If this is the case for all agents, the swarm will always coalesce.

In order to validate these theoretical results, we simulate a swarm of thirty generalized agents. Each identical agent has the ability to move and has simulated onboard sensors that report the relative positions of all other agents. The agents are capable of calculating relative positions using the coordinates of other agents. Using this data the agents also can calculate the center of mass of all agents. All agent movement is dictated by the center of mass.

The simulation initializes 30 agents with random coordinates. Each iteration, the agents obtain the relative positions of other agents. The center of mass of the set of positions of the other agents is then calculated. The agents then move towards the center of mass either directly or partially tangentially. When all agents' positions are close to or equal to the center of mass, the simulation stops.

In the first case, the agents move directly towards the center of mass as they have calculated. In Figure 3.1, all of the agents move towards the center in a direct path as indicated by the trail each agent leaves. This accomplishes the task as envisioned.

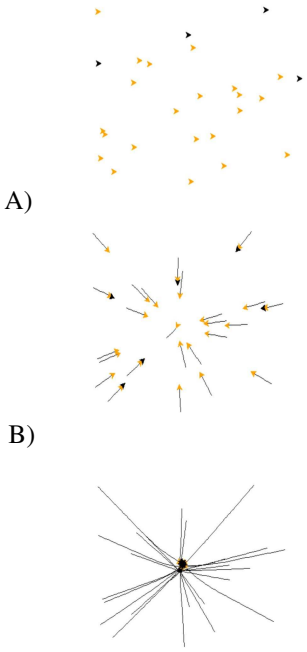
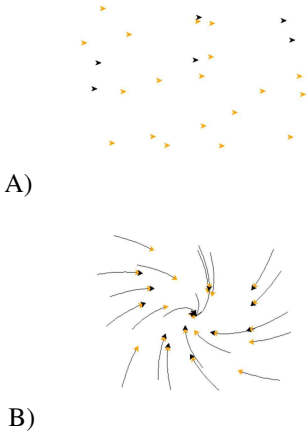


Figure 3.1: These figures illustrate the evolution of the coalescing agents when moving directly toward calculated centers of mass. In this and later figures, the agents are the triangular-shaped objects in the scene. The black lines illustrate the path taken by the agent connecting to the line.

In the second case, the agents behave identically to the previous case except that they move in at an angle of 30 degrees with respect to vector directed at the calculated center of mass. Figure 3.2 shows the agents moving in an indirect path towards the center of mass; the swarm spirals inward, ultimately still completing the task.



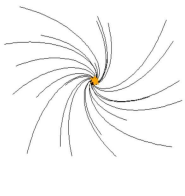


Figure 3.2: These figures illustrate the evolution of the coalescing agents when moving at an angle of 30° from directly toward calculated centers of mass.

We again utilize an indirect trajectory but in this case we assign an offset angle of 60 degrees. As illustrated in Figure 3.3, we again observe the swarm achieving the task through a longer spiral.

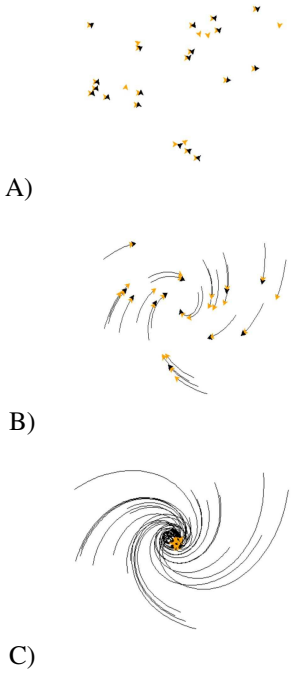


Figure 3.3: These figures illustrate the evolution of the coalescing agents when moving at an angle of 60° from directly toward calculated centers of mass.

In all cases, the agents accomplish the task by satisfying the swarm requirement given in equation (3.5). As these last two cases demonstrate, even when the agents' behaviors are significantly perturbed the swarm is still capable of achieving the global task. In fact, in this case, it is straightforward to determine the limit of the perturbation still yielding the desired global goal.

4 When to use a swarm and minimal swarm size

One of the most important goals of swarm engineering centers around simply verifying that a swarm is suitable for the given task. Interestingly, despite the myriad of studies surrounding the use of swarms, no systematic approach to the determination of the appropriateness of using a swarm for a given task has been proposed. In this section, we will examine how this question can be answered in the context of a simple swarm. As we shall see, the question is complex because it requires the consideration of the availability of *technology*, as opposed to simple swarm agent control algorithms. What makes swarms so interesting is *their ability to create, as a group, a capability that their constituent members cannot*. We shall see in this example that, when technology is not available, the swarm can make up the ability. In such a case, a swarm is *required*. Additionally, we can determine how many agents must be in this swarm in order for the task to be achieved.

We consider a task quite similar to plume tracking (Spears et al., 2009) or odor source identification swarms. We suppose that in a space there exists a source of some desirable thing (light, food, odorant, etc.), and that the dispersion of the substance is defined by some real function of position $S(\overrightarrow{x})$. The goal is to find a "good" location in the sense that it is locally optimal (Kazadi et al., 2015).

As indicated in Sections 2 and 3, we begin to design this swarm by defining a global property of the swarm.

$$P = \sum_{i=1}^{N_s} S(\vec{x}_i) \tag{4.1}$$

where S is the position-dependent average measurement of the target, $\overrightarrow{x_i}$ is the position of the i^{th} agent, and N_s is the number of agents in the swarm. The time derivative is

$$\frac{dP}{dt} = \sum_{i=1}^{N_s} \overrightarrow{\nabla} S \cdot \dot{\overrightarrow{x_i}}. \tag{4.2}$$

In order to move the swarm to an optimum, we must have that

$$\frac{dP}{dt} = \sum_{i=1}^{N_s} \overrightarrow{\nabla} S \cdot \dot{\overrightarrow{x_i}} \geq 0. \tag{4.3}$$

Equations (4.2) and (4.3) indicate both the technologies and their functional requirement. Firstly, two technologies are needed for this process to work. The swarm agents must have a method of evaluating the quantity $\overrightarrow{\nabla} S$, and the agents must have the ability to move, as indicated by $\dot{\overrightarrow{x_i}}$. Technologies that give us these capabilities can be used to generate the desired motion. Secondly, the motion given by $\dot{\overrightarrow{x_i}}$ and the quantity $\overrightarrow{\nabla} S$ must be oriented in such a way that the sum of positive products at most equals that of negative products. Given these requirements, the swarm will move toward the optimum.

As we've indicated above, swarms are indicated for a task if their capability exceeds that of the individual agents. Focusing on the technology that generates $\overrightarrow{\nabla} S$, we can ask whether or not individual agents have access to a technology with ability to determine $\overrightarrow{\nabla} S$. There are two possibilities:

1. such a technology exists and can be integrated with the agent design; or
2. such a technology does not exist.

In the first case, the gradient sensor can be integrated on a single agent. The problem will be solved by that single agent; a swarm is not needed – the agent can find its way to the optimum of S.

In the second case, there are again two possibilities:

1. A single agent, by moving around and sampling the local area can generate a local gradient, thereby making the information available; or
2. Such a movement and integration is unavailable or impractical.

Again, in the first case, the integration of this technological capability solves the problem and only a single agent is required. In the second case, a single agent has no way of obtaining this information. Therefore, if we mean to get the data needed to solve the problem, we need at least two agents. As a result, a swarm with a minimal size of two (2) is required.

We can design swarms that accomplish the overall task with the limitations imposed by the last case. We describe three strategies for accomplishing the gradient ascent and demonstrate their capabilities below.

Swarm of two agents

Two agents represent the smallest group that can accomplish this task in the event of the restrictions given above. Our swarm acquires and processes the local gradient by cooperation. One agent initially remains stationary while the second agent orbits the first in a circular orbit, sampling the value of $S(\overrightarrow{x})$ at points along its path. The second agent executes more than one full orbit, measuring the intensity of the field as it goes. After its first time around, the second agent stops at the location of the highest intensity in the first orbit when it re-encounters it during the second orbit. The direction indicated by the two positions

of the agents when the second one stops is approximately that of the local gradient. Once the second agent stops, the agents switch roles and the process restarts. After multiple iterations, the swarm finds and stays at the area with the global intensity peak.

We illustrate in Figure 4.1 the performance of the two-agent swarm below:

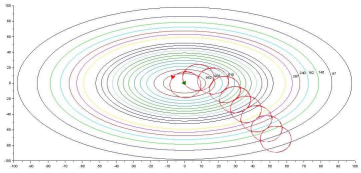


Figure 4.1 The trial of the two-agent swarm in a radiation field modeled by functions one through three.

The swarm finds the highest intensity, overcoming the limitations that would hobble a single agent.

Swarm of three agents

We construct a simulated swarm comprising three identical agents. Each of these agents is capable of making an intensity measurement at their location. Additionally they can determine the relative range and bearing of the other agents. The agents' utilize a modified physicomimetic control algorithm (Gordon et al., 1999; Spears and Gordon, 2007; Spears et al., 2004) in which the force function is given by

$$\overrightarrow{F} = 5\arctan\left(\frac{G}{r^2}\right)\hat{r} + \frac{\vec{\nabla} I}{\left|\vec{\nabla} I\right|}. \tag{4.4}$$

We assign G to 1200. When this simulation is run, three agents are randomly placed in the arena. Each iteration each agent obtains the intensity measurements from other agents and uses these to calculate the force as in (4.4). The agent then moves in the direction of the combined force. To calculate the gradient, we use a first order linear approximation defined by

$$I(x,y) = ax + by + c \tag{4.5}$$

where x and y are the coordinates of each agent, and a and b are undetermined coefficients. We can calculate the local gradient

$$\vec{\nabla}(I) = (a,b). \tag{4.6}$$

In Figure 4.2, the performance of the three-agent swarm are displayed. It is worth noting that, constrained by the method we implemented in the algorithm, the swarm does not always move perpendicularly to the contour lines. The swarm linearly approximates its local gradient vector. The further apart the agents are, the less accurate the estimation is. Nonetheless, once in the correct configuration, the swarm is able to find the peak intensity. One crucial advantage of this three-agent design, comparing to the two-agent swarm, is that the former does not require designated memory space for each agent, for the collection of all three agents is capable of computing the gradient instantaneously from their relative positions and their current intensity readings.

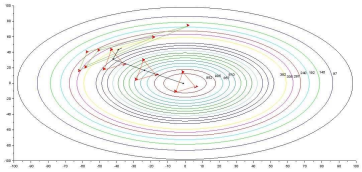


Figure 4.2 The trial of the three-agent swarm in a radiation field modeled by two functions.

Large physicomimetic swarms

We illustrate in Figure 4.3 the performance of a multi-agent swarm. As in the three-agent swarm, the agents in the multi-agent swarm do not require internal memory; the agents compute the local intensity gradient vector instantaneously which is again a linear approximation. For the multi-agent swarm to find the peaking intensity for the three functions is not extremely challenging given that swarm consists of 20 agents and the ability of the swarm controlled by physicomimetics to perform well in noisy fields (Hettiarachchi et al., 2008). It is highly improbable in these three scenarios that the swarm would get stuck in a local minimum; there are multiple neighboring agents which are capable of obtaining the intensity measurements required to compute their force vectors. Moreover, there are no other obstacles in the field. The swarm easily finds the peaks of the first two functions causing the agents to surround the peak. In the third function the swarm faces difficulty due to the flat trough like peak. Since the swarm stretches out in the trough, it becomes challenging for the agents in the back of the formation to compute intensity. Given adequate time, the swarm is capable of reaching the peak.

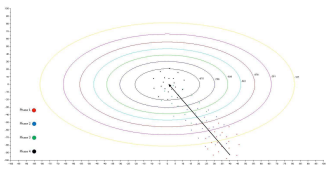


Figure 4.3 The trial of the large physicomimetic swarm in a radiation field modeled by two functions.

In all three cases, the swarm finds the highest intensity, overcoming the limitations that would hobble a single agent.

5 Conclusion and future work

This paper examined the HMSD. Global functions of the local measurables were used to gemerate a state space describing the swarm's state. Next, a set of swarm requirements was developed that generating swarm classes. This approach equates to a mathematical proof that the swarm's task will be achieved, if sometimes indirectly.

Within the equations generated from the global functions are mathematical descriptions of the varied technologies that one must have in order to achieve the task. The technology matrix is made up of expressions that describe the function of the technology on one or more aspects of the phase space describing the system. It is sufficient to enable the identification of solutions or the determination that such equipment does not yet exist.

In our estimation, the completion of a task is not dependent on a swarm if all requisite technologies can be deployed on a single agent. However, if the technologies cannot be implemented on a single agent, then a swarm is required.

We illustrated these principles using two classes of swarms: accumulation swarms and gradient ascent swarms. All developed gradient ascent swarms which achieved the task. It was demonstrated that minimalist stigmergic swarms consisting of two agents could achieve the gradient ascent task, as could larger physicomimetic swarms, as long as they implemented the technological and behavioral requirement that emerged, even when they had to do it cooperatively.

These developments enable us to approach the problem of swarm engineering from a different point of view than has generally been employed. The global properties amount to metrics on the state space of the system and their time derivatives enable the identification of technologies that are related to the achievement of the goal. The design problem, then, is reduced to developing a method of applying these technologies sequentially so that the system will move in state space from its initial state to a predetermined one. As a result, this approach reduces the swarm design problem to a multidimensional search through technology space, guided by the movement through phase space.

We have focused in this study on designing low dimensional swarms with simple design requirements. We turn in future work to more complex swarms whose design requirements are multidimensional as opposed to single dimensional.

6 Acknowledgements

The authors would like to acknowledge the efforts and assistance of several students of the Illinois Mathematics and Science Academy. These students are Katherine Bezugla, Ankit Agrawal, and Rohit Mittapalli. These students assisted in some of the simulation programming for Section 4 in this paper.

REFERENCES

Gordon, D., Spears, W., Sokolsky, O., and Lee, I. (1999). Distributed spatial control, global monitoring and steering of mobile physical agents. In IEEE Conference of Information Intelligence and Systems 1999, Washington DC., USA.

Hettiarachchi, S., Maxim, P. M., Spears, W., and Spears, D. (2008). Connectivity of collaborative robots in partially observable domains. In Proceedings of the International Conference on Control, Automation and Systems 2008, Seoul, South Korea.

Kazadi, S. (2000). Swarm Engineering. PhD thesis, California Institute of Technology.

Kazadi, S. (2009). Model independence in swarm robotics. Journal of Intelligent Computing and Cybernetics, Special Issue on Swarm Robotics, 2(4):672–694.

Kazadi, S., Jin, D., and Li, M. (2015). Utilizing abstract phase space in swarm design and validation. In Advances in Swarm and Computational Intelligence, volume 9140, pages 14–29, New York, NY, USA. Springer.

Kazadi, S. and Lee, J. (2007). Artificial physics, swarm engineering, and the hamiltonian method. In Proceedings of the World Congress on Engineering and Computer Science 2007, San Francisco, CA, USA.

Spears, D., Thayer, D., and Zarzhitsky, D. (2009). Foundations of swarm robotic chemical plume tracing from a fluid dynamics perspective. International Journal of Intelligent Computing and Cybernetics, 2(4):745–785.

Spears, W. and Gordon, D. (2007). Using artificial physics to control agents. In Proceedings of the IEEE Conference of Information, Intelligence, and Systems 1999, San Francisco, CA, USA.

Spears, W., Spears, D., Hamann, J., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. Autonomous Robots, 17(2):137–162.