

SWARMINESS

S. Kazadi
Jisan Research Institute
email: skazadi@jisan.org

A. Lee
Jisan Research Institute
email: ahyoung@jisan.org

Abstract

This paper examines the concept of swarminess, or the level of appropriateness of a swarm for a given task. We begin by defining the concept of a task and developing dependence maps including uni- and bidirectional dependences. We then define a vector swarminess measure. We examine two broad classes of tasks known as stigmergic and command tasks. We show that the swarminess of stigmergic tasks is infinite, while that of command tasks is finite. We use this to examine the swarm-based construction task and determine that this task, depending on available technology, may not be swarmy.

KEY WORDS

swarm engineering, swarminess

1 Introduction

Over the past few years, swarm engineering [8] has enjoyed a growing interest as a field of inquiry. Swarms, or bidirectionally communicating groups of individuals, have a number of interesting characteristics, the most intriguing being that some swarms seem to have the ability to accomplish as a group much more than an individual agent of the swarm could accomplish working alone, even with an infinite amount of time. This capability is closely tied in to another property of swarms known as *emergence*. In layman's terms, emergence means that the swarm has unexpected characteristics, not deliberately part of the direct programming of its constituent parts. Together, these two properties form the core of the intense interest in swarms.

Recent work [5, 6] has demonstrated that emergent swarms can be designed so that the emergent properties are natural and predictable results of the interactions. Having this kind of predictability and design capability begins to solve one of the most important questions in swarm engineering - that of how to design a desired swarm once the goal has been clearly stated. However, it does not solve another important question which has largely been ignored by the swarm community.

When faced with a difficult problem to solve, engineers typically choose from many different potential solutions to problems. One solution might be to use a complex device that can be produced in some factory. The cost might be high or reliability somewhat low, prompting us to search for alternatives. On the other hand, the machine might be useful only for this project, which makes its reusability limited. The search might bring us to swarms, which are flexible, reconfigurable, robust, and typically made up of cheap parts. But, is this really the best technology for the job? What part of the problem lends itself to use by swarms?

Tacit assumptions that search is well approached with a swarm abound. Many researchers have worked on swarm based construction [1, 9, 10]. Similarly, plume tracking has been studied with swarms [4], though to date, the authors are not aware of any work justifying this approach. Another well-studied problem has been the various ant colony optimization algorithms [2, 3] yet little literature exists which indicates a specific reason why swarms are appropriate for these kinds of problems. As a result, all of these approaches to swarm engineering have depended on an assumption that this was the proper approach. However, these approaches have not yet been implemented on many of the commercial systems that have dealt with these problems. It is not clear that the benefits of the use of swarms justify their use over other existing technologies.

The present work concerns itself with the motivation for using a swarm. We begin by making rigorous definitions about tasks, task maps, and the types of interconnections that exist between them. We use these concepts to define a measure of *swarminess*. This is a measure of how much a swarm is required and how much of the problem is this swarmy. We examine different types of tasks including two broad classes known as *command tasks* and *stigmergic tasks*. Stigmergic tasks are those in which the task involves changing an environmental variable, which in turn affects the way in which the task continues to be accomplished. Command tasks are tasks which are not stigmergic. We use these definitions to show that flocks and herds accomplish tasks that are command while ants and bees accomplish stigmergic tasks. All of these types of swarms have maximal (which for stigmergic tasks is infinite) swarminess despite their obvious differences.

2 Basic Definitions

When considering using a swarm to complete a task, we must first establish that the task requires a swarm. We must then estimate the size of the swarm that the task can utilize without interference between the various elements of the swarm or waste in the form of idle agents waiting for other tasks to be completed. We start by considering the nature of a task specification from the point of view of the agents who are going to accomplish it.

Our starting point is that, given a specific technological level, the agents will have a well-defined set of capabilities. These capabilities define the minimal task an agent might be expected to accomplish. We assume that this is well defined. That is, if a task is given along with an available technological specification, the task can be broken into pieces, each of which can be accomplished in one step.

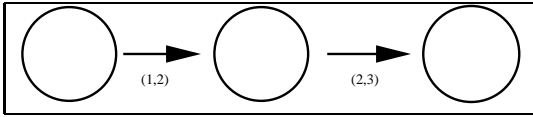


Figure 1. This figure illustrates independent and dependent atomic subtasks. The independent task is the left-most task, while the others are dependent.

2.1 Problems

Swarms that employ one technology will have one set of tasks while those that employ other technologies will have others. Moreover, these technologies must be able to be put on an autonomous platform in order to be considered as part of a swarm's design.

Tasks that can be accomplished in this way are important to the overall job, so they have a specific definition:

Definition 1 *Atomic subtasks* (N_a) are subtasks that can be accomplished by a single agent in a single step.

We consider atomic subtasks to be the smallest element of a task, and so they cannot be broken down into simpler tasks. They can also be completed by a single agent. For this reason, there is no communication involved within a single atomic subtask. There may, however, be communication among atomic subtasks.

Atomic subtasks are individual parts of a larger job that can be accomplished by a single agent. The task itself may require communication or prerequisite tasks to be accomplished. As a result, there are several different classifications of atomic subtasks.

Definition 2: *Dependent atomic subtasks* are atomic subtasks that depend on the completion of other subtasks.

Definition 3: An atomic subtask's *Dependence ranking* is defined as

1. 0 if it is an independent atomic subtask.
2. 1 plus the greatest dependence ranking of all atomic subtasks upon which it is dependent.

A dependent subtask can only begin once the subtask(s) preceding it has (have) been completed. For this reason, a dependent atomic subtask can be accomplished with the same agent that completed its preceding, completed atomic subtask(s).

Definition 4: *Independent atomic subtasks* do not depend on the progress of other subtasks.

Unlike dependent atomic subtasks, independent atomic subtasks do not require other subtasks to be completed prior to the atomic subtask in question. Often times, an independent atomic subtask starts a chain of dependent atomic subtasks. In many cases, independent atomic subtasks give information to dependent or codependent subtasks.

Definition 5: *Codependent atomic subtasks* are atomic subtasks that communicate and must happen at least partially simultaneously.

The actions of each of the codependent atomic subtasks affect the actions of all others. This relationship between codependent atomic subtasks is called *communication*. In communication, subtasks convey information to each other that help them accomplish their tasks. This allows codependent atomic subtasks to be coordinated, even though they do not necessarily start or end concurrently.

Definition 6: *Tasks* are groups of atomic subtasks that are connected to each other by some kind of dependency. A specific kind of task is a *linear task* in which the group of atomic subtasks are well-ordered by dependency.

Tasks may consist of independent subtasks, dependent subtasks, and/or codependent subtasks. At least one subtask is necessary for a task, but no one type of subtask is necessary for a task. There may be more than one task in a problem.

Definition 7: A *linear step* is the set of all atomic subtasks within one task that have the same dependence ranking.

Definition 8: A *microtask* is defined as a set of codependent atomic microtasks. These may be connected in any way including through a single common atomic microtask.

Note that a microtask is a subset of the linear step that contains it.

2.2 Graphs

Problems may be represented by graphs, which themselves can yield remarkable insight into the way in which problems are accomplished. In this section we examine how a problem may be expressed as a graph. We then use the graphs to generate theoretical results concerning problems.

Definition 9: *Graphs, or task maps,* are pictorial representations of problems.

There are two main components of graphs: nodes and ordered pairs. Each node represents one atomic subtask. We assume that there are finitely many nodes. As a result, we may uniquely label each node. Two atomic subtasks related to each other by some dependency are designated on the graph through ordered pairs. Each ordered pair represents one unidirectional dependency. The number of the node of the antecedent atomic subtask in a dependency serves as the domain value in the ordered pair; the number of the node of the dependent atomic subtask serves as the range value. Since each ordered pair represents only a unidirectional dependency, two ordered pairs must be designated for codependent atomic subtasks. In the case of codependent nodes, one ordered pair for each direction must be stated. Thus, with a list of ordered pairs and list of nodes, the graph is well-defined.

Definition 10: *Equivalent graphs* are graphs that, after neces-

sary re-numbering, contain the same set of nodes and identical ordered pairs.

Swarminess, as it will be defined below, creates an equivalence relation on the set of problem graphs. All equivalent graphs belong to the same equivalence classes, and therefore will have the same swarminess; however, two graphs that produce the same swarminess are not necessarily equivalent. Identical sets of ordered pairs imply that those graphs have the same number of atomic subtasks and that each corresponding atomic subtask has the same dependencies. In such cases, the graphs can be called equivalent.

3 Calculating Swarminess

In this section, we will define the swarminess measure. This is based on three main criterion. The first, which is the minimum simultaneity, is the smallest number of agents that can be used to complete a problem, assuming that no one agent may work on atomic subtasks simultaneously¹. Consider, for instance, that the problem in question has a specific number of atomic subtasks which must be carried out simultaneously in order for the problem to be accomplished. Then, assuming that each of these must be done by a single agent, this means that the problem must have at least this number of agents to accomplish the problem.

3.1 Simultaneity

Definition 11: *Minimum simultaneity of a task (S_t)* is the smallest number of agents that must be present for a single task to be accomplished.

Recall that a task is a track of atomic subtasks that are connected to each other by some kind of dependency. The minimum simultaneity of a task can be found directly from the graph: it is simply the number of atomic subtasks in the largest microtask.

Theorem 1 *Linear tasks have a minimum simultaneity of one.*

Proof: A linear task is an ordered list of atomic subtask which does not contain any microtasks. Therefore it cannot have a minimum simultaneity greater than one. \square

Lemma 1 *A single agent may be used to accomplish a linear task.* **Proof:** Since a linear task has a minimum simultaneity of one, this means that a single agent may accomplish each atomic subtask. As a result, this single agent, properly equipped, may accomplish all atomic subtasks sequentially. \square

As indicated by Lemma 1, any atomic subtasks connected to each other by unidirectional dependencies can be accomplished using the same one agent. On the other hand, atomic subtasks that are codependent with each other must occur simultaneously. As a result, more than one agent is required. Therefore, minimum simultaneity of a problem is determined solely

¹This is, of course, not the case for people. As an example, if the task is to put down four rocks in the shape of a square, a single person can accomplish this, but the positions of each of the rocks will affect adjustments of the positions of each of the other rocks.

from the microtasks in the different tasks. This is more rigorously proven in the next theorem.

Definition 12: *Minimum simultaneity of a problem (S_m)* is the minimum number of agents that must be present for a problem to be accomplished.

Theorem 2

$$S_m = \max_t S_t \quad (1)$$

Proof: The minimum simultaneity is the number of agents required to accomplish the entire problem. Since each task may be done sequentially, the number of agents required is equal to that required by the task with the greatest number of agents. This is the maximum among S_t . \square

Minimum simultaneity of a problem is the greatest number among all of the minimum simultaneities of tasks in that problem. This number signifies the greatest minimum simultaneity of all tasks in a problem, and therefore the smallest number of agents necessary to accomplish the problem. With this number of agents, every task is capable of being finished. Since all problems are made up of one or more tasks, accomplishing every task in a problem would accomplish the entire problem.

Definition 14: *Maximum simultaneity (S_M)* is the maximum number of agents that can be working on a given problem at the same time without interrupting each other.

Maximum simultaneity can be calculated from the minimum simultaneity of all tasks in that problem. It is the sum of all of the minimum simultaneities of tasks. This is significant because it potentially allows for all tasks of the problem to be solved at the same time. By adding the minimum simultaneities of all tasks together, we are ensuring that all of the tasks, and the problem, can be accomplished at the same time. In essence, the maximum simultaneity is the number of agents that will be working when the largest microtask of all tasks are being accomplished at the same time. This is expressed in the following theorem.

Theorem 3

$$S_M = \sum_t S_t \quad (2)$$

3.2 Saturation

Saturation (S_a) is a measure of the variance between the number of agents required for different parts of a problem. This is important because it gives an idea of how *badly* a problem requires a swarm. If the minimum simultaneity is large, and most of the microtasks have this large simultaneity, then the use of a swarm is very strongly justified. However, if most microtasks use a very small number of agents, then the use of a swarm is not strongly justified, and alternative methods of achieving the task without a swarm reasonably might be sought.

Maximum simultaneity measures the number of agents required at the peak of simultaneous work. While that number of

agents may be beneficial in solving one portion of the problem, it is possible that the remainder of the problem does not require as many agents. In this case, we say that the saturation variance is high, meaning that communication and simultaneous work do not benefit the problem, and the need for a swarm decreases.

Definition 15: *Saturation* of a problem is defined as

$$S_a = \frac{\sum_t \sum_i (N_{a_{i,t}})^2}{(N_a)(S_m)} \quad (3)$$

where $N_{a_{i,t}}$ represents the number of atomic subtasks in linear step i , N_a represents the number of atomic subtasks in the entire task t , and S_m represents minimum simultaneity of the problem.

$\frac{N_{a_i}}{S_m}$ accounts for the relative number of resources used by each linear step, by comparing the number of agents required for each linear step (N_{a_i}) to the minimum number of agents required for the entire problem (S_m). $\frac{N_{a_i}}{N_a}$ gives a value to the significance of that task, i , as compared to the entire problem. Therefore, our saturation value takes into account the resources that are being efficiently used during specific times of the problem.

In the most saturated problem, all parts of the problem require the same number of agents, and saturation is 1. Therefore, the closer the saturation value of a problem is to 1, the more justifiable it is to utilize a swarm of the given size – there is little waste in terms of idle time or interference. If the saturation value is too low, then a swarm is not an acceptable medium for solving that problem; too many agents are expected to be idle for much of the problem. One part of the problem will require a large number of agents, while other parts of it will require very few. The remaining agents that are never used again become wastes in resources and money. Other, more conventional ways of solving the problem are likely to be preferable.

3.3 Swarminess

Using the values from the previous sections, we now can define the term swarminess.

Definition 16: *Swarminess* can be defined as a vector function of 3 components.

$$\vec{S} = (S_m, S_M, S_a) \quad (4)$$

The first component accounts for minimum simultaneity of a problem (S_m), the smallest number of agents required to achieve a global goal.

The second component of the swarminess vector represents maximum simultaneity of a problem (S_M). S_M is the maximum number of agents that can be applied to a problem and can be acting on separate atomic subtasks simultaneously. It is possible for this number to act only on one specific group of microtasks, or to be active continually. In the latter case, the swarm is least wasteful. A high S_M indicates that a large number of agents can act at the same time, pushing forward the global goal. On the other hand, a low S_M indicates that only a few may act simultaneously. The smaller number provides little incentive to

use a swarm. As we shall see, there are cases in which S_M is infinite. In this case, there is no upper bound on the number of agents.

The third component of this vector is the saturation, S_a . S_a represents the variance in the number of agents required for each step of the problem. The saturation of a problem ranges from 0 to 1, with 1 being a state of complete saturation. The greater the saturation, the smaller the variance is, and the more similar different parts of the problem are in how many agents they require. Therefore, as saturation of a problem increases, the swarminess of the problem should also increase, since fewer agents and resources are being wasted.

The use of all three measures provides important information about the nature of the problem. If the problem has a very small S_m , then this indicates that the problem does not necessarily need to be solved with a swarm. On the other hand, a high S_M indicates that a swarm is potentially an efficient choice. Finally, a low S_a indicates that the use of a swarm might be inefficient for most of the problem, while a high value (close to 1) indicates that a swarm would be well utilized.

In the next section, we will examine some of the different types of problems. We shall see that there are significant differences between the two main classes of problems and that these differences lead to significant differences in functional approaches.

4 Classes of Problems – Stigmergy and Command

In this section we examine the concept of stigmergy. Stigmergy is an important part of swarm design of swarms from pucker clustering systems to ant colony optimization. It is so important that, as we shall see, there are two different broad classes of problems. One of these classes has a component that is stigmergic in nature, and the other does not. The swarminess of these two classes of problems works out quite differently.

4.1 Command tasks

We begin by defining non-stigmergic tasks.

Definition 17: An *environmental measurable* is a measurable quantity which is not directly controlled by an autonomous agent, but may be affected by behaviors of the agent.

An example of an environmental measurable is the amount of CO_2 in the atmosphere. This is measurable, but not controllable directly by any agent. As a result, it isn't a characteristic of the agent, but rather everything but the agent.

Definition 18: The *environment* is the set of all environmental measurables.

Definition 19: *Command tasks* are those that do not communicate using the environment as a communication medium.

Command tasks are made up of atomic tasks which communicate through their actions, as in construction swarms, or

through their agents, as in natural swarms. The number of command tasks in a swarm is definite and predetermined. The number of agents in each command task is also definite and predetermined.

Theorem 4 *Swarminess is well-defined for command tasks.* Since S_a , S_m , and S_M are all well-defined, swarminess is well-defined. \square **Proof:** A linear task is an ordered list of atomic subtasks which does not contain any microtasks. Therefore it cannot have a minimum simultaneity greater than one. \square

4.2 Stigmergic Swarms

Definition 20: *Stigmergic communication* is a type of communication that occurs indirectly among tasks through modifications in the environment.

All previously considered communication has been assumed to be between the atomic subtasks of a problem. In stigmergic communication, however, information is exchanged among the tasks of the problem. By definition, tasks are entirely separate jobs that do not directly communicate. Any form of communication between tasks, therefore, occurs indirectly and through the environment. In stigmergic communication, the actions of one task modifies the environment, which in turn affects the actions of the other tasks.

Definition 21: *Stigmergic tasks* are tasks that utilize stigmergic communication.

Lemma 2 *All swarms are either command or stigmergic.*

Command swarms are those that have a definitely established number of tasks. Stigmergic swarms do not have a definitely established number of tasks; the number of tasks in a stigmergic swarm depends on the environment. Swarms either depend or do not depend on the environment. Therefore, swarms are either command or stigmergic.

Definition 22: *Stigmergic graphs* are mappings of problems that utilize stigmergic communication.

Stigmergic graphs contain nodes that depict environmental variables. Since stigmergic communication occurs between a task variable and an environmental variable, the graphs must be updated to allow for this type of communication. We enhance the task graphs by giving the environmental variable its own node. Stigmergic communication is depicted using communication arrows between the interdependent tasks and environmental variable node. Since the environment feeds back into the stigmergic task, the stigmergic part of the graph is cyclic. As a result, there is no specific number of atomic subtasks, but rather a specific number per cycle. We call each cycle an *iteration* and indicate the number of iterations on the graph. We also note that each cycle might be accomplished by a single agent, but these agents need not be the same during each cycle.

There are two classes of stigmergic problems: definite stigmergic problems and stochastic stigmergic problems. *Definite stigmergic problems* are composed of subtasks that have exactly one potential outcome. This means that when a subtask is accomplished, the specific outcome occurs. *Stochastic stigmergic problems* are problems in which one of multiple outcomes occur.

There are two ways that a swarm can update the environmental variables. One of the ways is using a *synchronous update* and the second is using *asynchronous update*. In the first mode, environmental variables are updated at once from several agents. In the second mode, environmental variables are updated individually via the different agents. The first might happen with a computational swarm using a single server housing all the environmental measurables, while the second might be done in practice during construction work by, for example, ants.

When a problem contains a stigmergic portion, it may be the case that the tasks are stochastic in the sense that a task may have several different possible outcomes. In this case, each agent can carry out one of many different atomic subtasks. This means that the effect of each agent's behavior can be random. We call such problems *stochastic stigmergic problems*. The ant colony optimization algorithms are examples of stochastic stigmergic problems.

In what follows, we assume that all agents work in a state of having no memory of previous states. This means that the system evolution is dictated by the behaviors of the agents which are themselves dependent only on the system's current state. This is not unreasonable, as it can be demonstrated that even people have a limited memory of previous encounters, drawing much of their behavior from their interaction with their environments.

In general, for stigmergic tasks, the behaviors of the agents are dictated in whole or in part by the state of the environmental variables $\{e_i\}_{i=1}^{N_e}$. If we assume, for simplicity, that $N_e = 1$ then the behaviors of the agents may be parametrized as follows:

$$b = \frac{de}{dt} = f_e(e, x_1, \dots, x_N) \quad (5)$$

where x_i represents any other variable upon which f_e is dependent.

Given that a task is stochastic, we can write the various outcomes of the behavior has many possibilities. I.e.

$$b \in \{\gamma_i(e)\}_{i=1}^N. \quad (6)$$

Each of the outcomes may have a probability of occurring of

$$p(b_i) = p_i. \quad (7)$$

In the event that the system updates the environmental variables synchronously using the behaviors of several agents, the environmental variables will update according to the equation

$$\frac{de}{dt} = \sum_{i=1}^N f_e(\gamma_i(e), x_1, \dots, x_N) p_i(\gamma_i(e)). \quad (8)$$

In the opposite event, that the system does not do synchronous updating, the system will take on differing states according to the probabilities given above. The system will then become defined by Markov statistics.

This discussion clarifies the following theorem.

Theorem 5 *Suppose a problem is a stochastic stigmergic problem. Given that the behaviors of the agents is a function of the environmental variable e , the large ensemble solution to the stochastic stigmergic swarm is the solution to the differential equation given in (8).*

This is an extremely important result, as it indicates that the overall behavior of the swarm can be calculated, perhaps necessarily via numerical solution, by solving the differential equation given in (8). In contrast, no such predictive power exists without the ensemble. In that case the system evolves randomly. Though one may still make meaningful statements about what might happen because of the relative probabilities, very little is certain, as the system may move in any one of many directions randomly.

The main importance of this theorem derives from systems in which there are multiple attractors. In such systems, the behavior of the system may be predicted only if there is a way of describing how its constituent parts will effect the overall system behavior, something that is very difficult to do in a Markovian system. While the overall dynamic might be to bring the system to a desired system attractor, what might actually happen is that the randomness in the system might bring the system to a second attractor.

An ensemble system is quite reproducible. The solution to the differential equation is well defined, and thus the system tends to approximate the desired behavior during each run. This would seem to be a desirable property of an engineered swarm system, and would tend to indicate one of the strengths of the swarm as opposed to a single agent; a single agent might make mistakes that influence the overall behavior of the system while an ensemble will tend to produce more similar outcomes each time.

In the following theorem, we ignore the physical size and structure of the swarm components, and focus simply on the accomplishment of the task.

Theorem 6 *The maximum number of subtasks in a stigmergic problem cannot be determined a priori. Proof:* This theorem means that knowledge of the behavior of the interaction between the environmental property P being altered and the subtasks alone is not sufficient to predict the number of subtasks in a stigmergic problem.

Let the environment have some measurable property, P . The global goal of each problem is to modify the property so that it takes on a desired state. Each definite subtask, once completed, affects a change in the property through the action of some behavior, b . ΔP_i , the change in property P , represents the change in the environment affected by a task, i . Note that ΔP_i is a function of P and b . The behavior carried out by the subtask is a function of the property that is affected by that task. For example, ΔP_3 is a function of P_3 and the behavior of the next subtask. ΔP_T is the total change in the environment property P for the entire problem. It is the sum of the change in property over time for all subtasks. However, the number of subtasks required to accomplish this change in property P cannot be known without calculation of the change in the property P at each step using the

value of property P and the behavior generated by the subtask.

$$\Delta P_T = \sum_{i=0}^N \Delta P_i (P_{i-1}, b) \quad (9)$$

In the case of the definite stigmergic problem, each of the P_i is well-defined, though it cannot be determined without calculating, sequentially, each P_i from b and each P_{i-1} . In the case of the stochastic stigmergic swarm, equation (9) does not have a well-defined number N . P_T cannot any longer be calculated. However, given a maximal number of steps, the average value of P_T can be calculated using the distributions of values that could result from the various outcomes at each step.

These arguments complete the proof. \square

This represents a stark break with the command tasks. Command tasks can be defined and predicted *a priori*, but stigmergic tasks cannot. Moreover, if the stigmergic task is made up of stochastic subtasks, then the required time to complete the task cannot be accurately determined. This is a problem because this means that the stigmergic task cannot be planned for exactly, and termination of this task must occur via some criterion most likely related to the value of P . This also means that prediction of the final value of P , if it is not previously known, cannot be done without accomplishing the tasks.

Stochastic stigmergic problems are significantly more troublesome because, in addition to the lack of predictability, these problems have more unpredictable global outcomes. The system is likely to take a random walk through state space related to the stochastic nature of each subtask. Thus, utilizing such a system for anything critical is difficult to justify.

In order to improve the predictability of the stochastic stigmergic problems, one might utilize a large number of identical subtasks whose interactions with the property P are averaged before P is updated. Such a problem is called an *ensemble stigmergic problem*. As a result, equation (9) becomes

$$\langle \Delta P_T \rangle = \sum_{i=0}^N \Delta \langle P_i (P_{i-1}, b) \rangle \quad (10)$$

Stigmergic problems have interesting properties, particularly if they contain multiple stigmergic parts. One such property is expressed in the next theorem.

Theorem 7 *The saturation of a stigmergic problem is the weighted mean of the saturations of the stigmergic tasks. Proof:* Recall that the saturation of a task can be calculated using the formula

$$S_a = \frac{1}{S_m N_a} \sum_{i=1}^{N_{a_i}} N_{a_i}^2 \quad (11)$$

where i represents linear steps, N_{a_i} represents the number of atomic subtasks per linear step, S_m represents the minimal simultaneity, and N_a represents the total number of atomic subtasks in the problem. Thus this means that if we have K tasks, the saturation may be calculated as

$$S_a = \frac{1}{N_a} \sum_{k=1}^K \frac{1}{S_m^k} \sum_{i=1}^{N_{a_i}^k} (N_{a_i}^k)^2 \quad (12)$$

where the sum runs over tasks. Suppose that several tasks are stigmergic. Then this sum may be separated into two parts, a stigmergic part and a non-stigmergic part.

$$S_a = \frac{1}{\left(\sum_{k=1}^K N_a^k \right)} \left(\sum_{k=1}^{K_1} \frac{1}{S_m^k} \sum_{i=1}^{N_{a_i}^k} (N_{a_i}^k)^2 + \sum_{K_1+1}^K \frac{1}{S_m^k} \sum_{i=1}^{N_{a_i}^k} (N_{a_i}^k)^2 \right) \quad (13)$$

$$\leq \frac{1}{\left(\sum_{k=1}^K N_a^k\right)} \left(\sum_{k=1}^{K_1} \sum_{i=1}^K N_{a_i}^k + \sum_{K_1=1}^K \frac{1}{S_m^k} \sum_{i=1}^K \left(N_{a_i}^k\right)^2 \right). \quad (14)$$

Let us assume, for simplicity that there are two tasks in the total problem. Beyond some K_1 , the tasks are assumed to be copies of the stigmergic task, as the same task is re-run some large number of times. Let us assume that this number is N_c . Let us also assume, for simplicity that there are two tasks in the overall task. Then, $K_1 = 1$. The equation then becomes

$$S_a = \frac{1}{(N_{1,a} + N_{2,a})} \left(\frac{1}{S_{1,m}} \sum_{i=1}^K (N_{1,a_i})^2 + \frac{1}{S_{2,m}} \sum_{i=1}^K (N_{2,a_i})^2 \right) \quad (15)$$

$$= \frac{1}{(N_{1,a} + N_c N_{s,a})} \left(\frac{1}{S_{1,m}} \sum_{i=1}^K (N_{1,a_i})^2 + \frac{1}{S_{2,m}} \sum_{i=1}^K (N_c N_{s,a_i})^2 \right) \quad (16)$$

where N_s is the number of linear steps in each stigmergic cycle. Assume that $N_c \gg N_{1,a}$ as is the case for most stigmergic tasks. Then the first term drops off to zero and

$$S_a \approx \frac{N_c \sum_{i=1}^K (N_{s,a_i})^2}{S_{2,m} (N_{1,a} + N_c N_{s,a})}$$

$$= \frac{\sum_{i=1}^K (N_{s,a_i})^2}{S_{2,m} \left(\frac{N_{1,a}}{N_c} + N_{s,a} \right)} \approx \frac{\sum_{i=1}^K (N_{s,a_i})^2}{S_{2,m} N_{s,a}}. \quad (17)$$

Similar reasoning prevails for problems with stigmergic and non-stigmergic parts, yielding

$$S_a \approx \frac{\sum_{l=1}^L \left(\frac{1}{S_{l,m}} \sum_{i=1}^K (N_{s_l,a_i})^2 \right)}{\sum_{l=1}^L N_{s_l,a}} = \frac{\sum_{l=1}^L N_{s_l,a} S_l}{\sum_{l=1}^L N_{s_l,a}}. \quad (18)$$

Therefore, the arithmetic mean of the saturations of the stigmergic tasks in a problem, weighted in accordance to the size of the tasks, is the saturation of the entire swarm. \square

Another property of stigmergic tasks is that one of the vector components of the swarminess measure is infinite. In this case, we consider the swarminess measure to be infinite.

Theorem 8 *Stigmergic tasks are “infinitely swarmy.”* **Proof:** To prove this theorem, we consider the three components that we have established in our swarminess vector.

Maximum simultaneity, S_M , of a stigmergic task, given no external barriers, is infinity. We have established that the maximum number of agents that can be applied to a stigmergic task is determined by the environment. Since there is no predetermined point at which the stigmergic task is known to be completed, we cannot set an upper limit on the number of subtasks which must be accomplished to complete the task, in general. As a result, we are free to utilize an unbounded number of subtasks. Hence, we can say that the maximum simultaneity of a stigmergic task, the greatest number of agents that can be applied to a problem without interfering with each other, is unbounded.

The minimum simultaneity of a problem, S_m , of a stigmergic task is finite. Therefore, the lower bound for minimum simultaneity of a problem is finite.

Because of this, we cannot predict the exact saturation (S_a) of a stigmergic swarm. We can, however, estimate S_a of a stigmergic swarm to be close enough to 1 that the difference is

negligible. A problem becomes increasingly saturated as more parts of the problem require the same number of agents. Although there may be non-stigmergic tasks in a stigmergic swarm, the number of cycling stigmergic tasks will greatly outnumber the number of non-stigmergic tasks. Because of the nature of stigmergic tasks to repeat, we can assume that the variance in saturation due to the non-stigmergic tasks is negligible. The swarminess vector for stigmergic swarms, then, is

$$\vec{S} = (k, \infty, 1) \quad (19)$$

where k is some finite value of S_m . \square

4.3 Command Tasks

Command tasks are tasks that do not contain stigmergic components. As a result, there is no bidirectional interaction between the task and the environment. Despite this, many command tasks have emergent properties and are therefore of interest to the greater swarm community. Among swarms that carry out command tasks are flocks, herds, shoals, formations and the like. While it is understood that these are swarms, their nature and difference from stigmergic swarms is still largely misunderstood. This section will elucidate some of the properties of command tasks that are carried out by flocks and the like.

A *spatially constrained swarm* is a swarm in which the center of the swarm (or average position of the agents) and the most distant agent are maintained below a particular distance R . A swarm maintaining a *formation* is a swarm in which the neighbors of each agent do not change frequently; the time-averaged set of positions of neighbors of agents remains constant, even if the specific agents taking the positions do change.

It is clear that the task undertaken by an agent in a spatially constrained swarm is to maintain its position according to a number of different criterion. These may be extremely restricted, as in a rigid formation, or very loose, as in a fluid and dynamically changing swarm. However, as long as the agent is constrained within the swarm, it is undertaking this one task. The task is in continual communication with other positional tasks undertaken by other agents. As a result, if N agents are in the swarm, there are N mutually communicating tasks. The task map, then, consists of a fully connected network of N nodes. From this map, it is clear that $S_m = S_M = N$ and that $S_a = 1$. As a result, the swarminess vector is $(N, N, 1)$, which is maximal. This proves the following theorem.

Theorem 9 *Flocks, herds, schools (of fish), and the like, are command swarms but maximal swarminess.*

An easy corollary of this theorem concerns the saturation of spatially constrained swarms.

Corollary 1 *S_a for all swarms that are flocks, herds, schools, and the like, therefore, is always 1.0.*

This is a direct result of the theorem.

5 Discussion and concluding remarks

Previous work on swarms have concentrated heavily on the engineering of swarms and swarm emergence. While the field of swarm engineering has seen much progress in methods of swarm applications, the question of the usefulness of swarms in specific situations had not been asked until this paper. In this paper, we studied the justification for the use of swarms and formulated a vector equation that addresses the question.

Our paper does not attempt to create a metric by which to measure swarminess. It does not establish bounds on the swarminess vector to indicate whether or not a problem should be attacked with a swarm. Instead, we evaluate the properties most essential to the definition and characteristic of a swarm. We recognize that each problem-solver has a unique and specific situation according to which he or she must decide whether to apply a swarm. Such factors may include limitations on time, technology, monetary resources, etc. By providing equations by which to calculate minimum simultaneity, maximum simultaneity, and saturation, we give the user the choice to decide what properties are the most important in their decision to apply a swarm.

Overall, we have established that the connections between the tasks and subtasks within the problem are the primary indicators of the usefulness of a swarm for that problem. The connections between subtasks indicate communication, and therefore, minimum simultaneity, maximum simultaneity, and saturation.

Our method of analyzing problems is unique in that we consider the tasks of problems to determine the appropriateness of a swarm. We establish two types of tasks, command tasks and stigmergic tasks. If the task is command, the outcome of the task is generally predictable, but it may or may not lend itself conveniently to swarms (i.e., construction). If the task is stigmergic, the outcome of the task is generally a random walk (i.e., ant TSP), but it is naturally swarmy. Moreover, this random walk can be mitigated through careful application of ensemble updating in the place of asynchronous updating.

Command tasks have varying levels of swarminess, depending on the specific task. The only exception to that generalization is a command task in which one or more microtasks make up the majority of the task. A command task with a heavy concentration of codependent atomic subtasks requires communicating and simultaneously working agents, which is characteristic of a swarm. This realization led us to a conclusion about flocks, herds, schools, and the like.

Using the swarminess vector, we showed that problems such as the coordinated movement of flocks, herds, and schools of fish, have maximal swarminess. The individual animals themselves act as both agents and subtasks in these problems. Furthermore, all of the animals in a flock, herd, or school must communicate with each other; the problem, therefore, is one large microtask. Since those subtasks in the one microtask are the only subtasks in the problem, $S_m = S_M = N$, the number of agents, and $S_a = 1$.

The second type of tasks, stigmergic tasks, are infinitely swarmy, as proved in the paper. The nature of stigmergic tasks naturally lend these tasks to use by swarms. This paper also makes specific reference to the Traveling Salesman Problem for ants (ant TSP) in terms of swarminess of stigmergic tasks. The ant TSP clearly utilizes stochastic stigmergic communication;

the tasks communicate via the environment, which is updated asynchronously, after every iteration. We have proven that the ant TSP, like any other stochastic stigmergic problem, inevitably results in a random walk. The only way to solve a stigmergic problem and produce accurate and reproducible results is to update the environment synchronously. In synchronous updating, the environment is updated as a result of the outcome of several iterations. The update to the environment is an average of the result of the tasks completed by several agents. Since the results are averaged before making an effect on the environment, that effect is less likely to produce a random walk on the environment. Essentially, the more synchronously the environment is updated in a stigmergic problem, the more likely the problem is to produce a reliable result.

Stochastic stigmergic problems such as the ant TSP are common in swarm engineering. This paper has proved, however, that results from stochastic stigmergic problems are unreliable. In order to ensure that problems such as the ant TSP will produce more reliable results, stigmergic problems must be solved with synchronous updates.

This paper indicates that ant colony optimization of TSP will produce more reproducible behavior if the environment is updated synchronously. Further studies will more thoroughly investigate this by simulating and colony optimization with both asynchronous and synchronous updates. The results of such a study might have wide ranging implications for the swarm engineering community.

Recent swarm literature has put much focus on applying swarms to construction problems. Since it is widely assumed that construction readily lends itself to swarms, it will also be interesting to investigate construction problems with our swarminess vector. Future studies will further explore construction problems and determine which kind of construction problems, if any, are readily swarmy.

References

- [1] R. Beckers, O. Holland, and J. Deneubourg. *From local actions to global tasks: stigmergy and collective robotics*. **Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems**, MIT Press, 1994.
- [2] M. Dorigo, E. Bonabeau, G. Therauluz. *Ant algorithms and stigmergy*. **Future Generation Computer Systems** (16), 851-871, 2000.
- [3] M. Dorigo, L.M. Gambardella. *Ant Colonies for the Traveling Salesman Problem*. Technical Report TR/IRIDIA/1996-3, IRIDIA, Université Libre de Bruxelles, BioSystems.
- [4] A. Hayes. *Self-Organized Robotic System Design and Autonomous Odor Localization*. Ph. D. Thesis, California Institute of Technology, 2000.
- [5] S. Kazadi, P. Kim, J.S. Lee, J. Lee. *Swarm Economics*. **Proceedings of the World Congress on Engineering and Computer Science 2007**, San Francisco, California, USA, October 24-26, pp. 602-610, 2007.
- [6] S. Kazadi, J. R. Lee, and J. Lee. *Artificial Physics, Swarm Engineering, and the Hamiltonian Method*. **Proceedings of**

World Congress on Engineering and Computer Science 2007, San Francisco, California, USA, October 24-26, p. 623-632, 2007.

- [7] S. Kazadi, A. Abdul-Khaliq, and R. Goodman. *On the convergence of a puck clustering system*. **Autonomous Robots**, 38 (2), pp. 93-117, 2002.
- [8] S. Kazadi. *Swarm Engineering*. Ph.D. Thesis, California Institute of Technology, 2000.
- [9] Y. Terada and S. Murata. *Automatic assembly system for a large-scale modular structure: hardware design of module and assembler robot*. **Proceedings of IROS 2004**, Sendai, Japan, pp. 2349-2355, 2004.
- [10] J. Wawerla, G. Sukhatme, and M. Matic. *Collective construction with multiple robots*. **Proceedings of IROS 2002**. Lausanne, Switzerland, 2002.