1

Introduction of Swarm Intelligence

Abstract

Because biology is swarming intelligence, for millions of years, many biological processes have addressed complex issues through information exchange with groups. By thoroughly examining the behavioral factors aspects of individuals and integrating compartmental observations with mathematical or simulated modeling, the processes of the collective conduct in biological systems are now understood. We use insect-world examples to demonstrate how structures are developed, collective choices are taken, and how enormous groups of insects may move as one. This first chapter encourages computer programmers to look more carefully at the biomedical domain.

***Keywords*:** Swarm behavior, collective behavior, particle swarm optimization (PSO), swarm agents, optimization method, global behavior, fish schooling

1.1 Introduction to Swarm Behavior

- A swarm may be described as an organized grouping of organizations that interact (or agents).
- In a swarm, individuals cooperate to achieve a global goal more efficiently than a single person could.
- Although an individual's conduct is straightforward, group activities can become very difficult.
- Computer scientists use birds (for flocks), ants, poultry (for schools), bees, and wasps in swarm intellectual ability studies [1].

1.1.1 Individual vs. Collective Behaviors

- Swarming and individual behaviors are closely interconnected.
- Individuals form and determine the behaviors of the swarm. On the other hand, the swarm's behavior affects the

Kuldeep Singh Kaswan, Jagjit Singh Dhatterwal and Avadhesh Kumar. Swarm Intelligence: An Approach from Natural to Artificial, (1–36) © 2023 Scrivener Publishing LLC

environment under which a person acts. Individuals specialize in one specific duty in a colony of ants. Taken together, activities and behaviors of ants guarantee the construction of optimum nesting systems, the protection of the queen and larva, the purification of nests, the search for the optimum sources of food, the improvement of assault methods, etc. Global behavior arises non-linearly from the activity and interconnections of members in the swarm [2].

The connection between refining aid interactions between people boosts experienced environmental knowledge and optimizes swarm development.

- The collaboration between people is determined genetically or through social interaction.
- Social relationships may be direct or indirect.
- Visual, auditory, or chemical contacts are immediate interactions [3].
- Indirect relationships happen when some people alter the environment, and others react to a different world.
- Social networking provides lines of communication to communicate knowledge.

1.2 Concepts of Swarm Intelligence

Swarm intelligence (SI) is a system that produces globally integrated functioning patterns due to the combined conduct of (unsupported) agents communicating locally with their environments [4].

- On the basis of SI, communal (or spread) issue solving may be explored without central authority or global modeling for function improvement, optimization of paths, timings, optimization of structures, design, and processing, and video analysis, productive implementations; and
- PSO (particle swarm optimization) and ACO (ant colony optimization).

1.3 Particle Swarm Optimization (PSO)

- The intent of the design is to visually recreate the beautiful and unexpected dance of a flock of birds.

- The objective of identifying patterns governs birds' capacity to fly simultaneously and alter their course quickly, with an ideal grouping.

This PSO technique has become a convenient and straightforward optimization method [5].

1.3.1 Main Concept of PSO

- PSO is a population-based search technique where people are sorted into a pool, called particles. The possible answer to the optimization problem is every particle in the swarm.
- In the PSO system, each particle is "flown" in a multifunctional search area, adapting its search space to its own and adjacent particles' experience or knowledge.

Therefore, a particle uses its best position and its neighbors' most vital position to the situation itself towards the optimal solution.

- The consequence is that, while still exploring a region surrounding the perfect option, nanoparticles are flying towards the global minimum.
- Every particle's effectiveness is measured based on a specified fitness function linked to the challenge.

1.4 Meaning of Swarm Intelligence

Pattern recognition is the systematic gathering of naturally occurring dispersed, self-organized systems. The idea is used in machine intelligence development. In 1989, it was presented in the methods used in cellular cyborgs by Gerardo Beni and Jing Wang.

Swarm intelligence (SI) systems generally consist of a community of essential agents or boots that interact independently and with each other. Nature, in particular living organisms, frequently inspires. The agents obey elementary principles. While no command and control structure determines how the agents behave, local exchanges lead to the formation of "intelligent" universal behaviors, which are unknown to the agents. Examples of natural SI include organisms, flocking of birds, hawk chasing, herding of animals, and growth of bacteria, fish schooling, and microbiological information.

Swarming concepts are called swarm cyborgs, but swarm intelligence refers to a broader range of algorithms. In the framework of forecasting issues, swarm prediction has been utilized. Similar techniques in synthetic intellectual capacity for genetic engineering to those suggested for swarm cyborgs must be studied [6].

1.5 What Is Swarm Intelligence?

The study of decentralized, self-organized networks is cognitive swarming that can rush in a coordinated way. Swarms exist naturally, and evolutionary habitats, such as ant colonies, flocks of birds, and animal husbandry, have been studied by scientists to discover how distinct bioproducts cooperate with their environments to achieve a shared objective.

In cyborgs, swarm knowledge involves the observation of nature and the use of concepts by scientists in machines. A cyborg swarm, for instance, may consist of small, identical appliances, each with sense. If data is exchanged with the other gadgets in the group acquired by one cyborg agent, it allows the users to act as a unified group. A cyborg swarm is typically straightforward, and agents frequently use sonar, radar, or camera to acquire additional data [7].

1.5.1 Types of Communication Between Swarm Agents

Isolated bots or swarming agents can interact in several ways, including:

- Point to point: Information is transmitted immediately from the agent to warn the swarm of places, impediments, or objectives.
- Broadcasting: One agent in the swarm immediately broadcasts information to the rest of the swarming via sound, light, or wireless media.
- Contextual information exchange: An agent leaves a message inside the swarm that can transmit information to affect the behavior of other members. The way insects leave behind a trail of pheromones to take their equivalents to a particular area is comparable.

1.5.2 Examples of Swarm Intelligence

Pattern recognition has a lot of uses. Small, drone-like cameras for risky search and recovery operations can show optimization techniques.

The cyborgs may execute an extremely light duty in destructed regions, such as looking for survivors if scheduled for working together as a single unit. Pattern recognition is also used to mimic crowds, such as augmented reality games in films and the importance of technology.

"Smart dust" is a term used to characterize a microelectromechanical system (MEMS) that is tiny enough to remain hanging in the air. Researcher think that smart dust should analyze environmental details in distant worlds.

1.6 History of Swarm Intelligence

Independent creatures in naturalistic cloud computing usually have no idea of a high-level purpose but may mimic complicated real-world systems. When satisfied, numerous low-level objectives make this feasible. This enables significant collective activity resulting from these stupid and non-influent single individuals. A reintroduction to the modeling of natural things such as fire, wind, and liquid in computer-based animation tracks back to the early efforts of William "Bill" Reeves in 1983, culminating in the Pixar film *Luca* in 2021. During the development process, agents or "droplets" were created. In the virtual simulation, they undertook modifications, wandered around, and were finally rejected or "died." Reeves found that such a pattern could represent the dynamism and shape of natural surroundings, which had been unfeasible with conventional surface depictions. The Boid model (1986) created basic principles that enhance the independence of particle behaviors and set simple low-level norms that may lead to emerging behavior by boids (bird-oid objects) and particles. Therefore, the sophistication of the Boid model is a direct result of the fundamental interconnections between each component. Craig Reynolds established three different swarming regulations for the following particles: segregation, alignment, and cohesion. While the concept of separating enables molecules to move away from one another to prevent crossover, the harmonization and cohesive ideas need directional upgrades to advance towards the aggregate direction and location of the adjacent troops. The intrinsic intermodulation distortion of books makes the group dynamically chaotic, but the negative criticism produced by the essential low-level rule influence makes the behavior orderly. If each book knows the true identity of each other book, it's computerized by an $O(n^2)$ complexity. Reynolds offered a neighborhood model with an interchange of information between boxes, thereby lowering the complexity to $O(n)$ and accelerating the use of algorithms. In 1995, an expansion of Reynold's

work officially established the Support Vector Machine method. In this method, the flock or swarm has unexpectedly converged unanimously, including local information sharing to the closest neighbor speed comparison. Therefore, the velocity of particles has been randomly disturbed, resulting in sufficient fluctuation and consequent lifelike swarm behavior. These characteristics were removed because the flock looked to convergence without attracting them. This scenario concluded with a community of agents that more closely followed a swarming than a flock's dynamics [7].

1.7 Taxonomy of Swarm Intelligence

The evaluation of taxonomy is provided in this part of the PSO algorithm.

- Continuity: The PSO is split into two sections, continuous and discontinuous, from the consistency in the area where these particles are situated. As a change in the locations of the nanoparticles in the same dimension, the travel route of the nanoparticles continues. However, this motion route changes the likelihood of the orientation coordinate's value being zero or one in a discontinuous state.
- Fuzzification: The PSO is examined from two perspectives from a defuzzification point of view. The fuzzy model of the method is evaluated in a few PSO applications, such as multi-target quadratic allocation issues. Accelerating and positional representation in vector form are translated to fuzzy matrices from real vectors.
- Accordance: At times, the swarming development process is almost stopped and stagnant during the PSO. This is sometimes because certain particles are inactive; in other words, they cannot be searched locally and globally; therefore, their current or previous locations don't move much, and their speed is almost nil. One option is to substitute these passive nanoparticles with new elements to retain the present rotation of the nanoparticles based on PSO particles. The accelerated PSO (APSO) technique is used. But this halt is occasionally caused by a swarming propensity to reach a balance condition that prohibits other regions from being searched and caught at a local minimum. An energy dissipation system that introduces negative mobility and causes

chaos among particles is developed to tackle this problem utilizing the dissipative PSO (DPSO) algorithm. By using this method, the above-stated stagnant condition is almost averted. The two techniques described above are used concurrently in this regard. In other words, on the contrary, an adam optimizer is utilized to enhance swarm variety when the algorithm finds equilibriums in the final few runs, and the dissipative algorithm is designed to introduce negative volatility in the PSO. Therefore, an adaptive method for the weight update tricks is devised to maintain domestic and international optimization balance. The dynamically adaptive dissipative PSO (ADPSO) algorithm was used. Both static and dynamic conceptions may be evaluated in the printed images.

- Attraction: There are three techniques, namely attraction, repulsion, and emotional connection, to tackle difficulties such as premature convergence. The additional operator is used in the attracting phase to upgrade the speed equations, while the subtracting operator is used during the repelling phase. The electrons are transported in the attraction stage and in the repelling step they get away. In the attraction/repulsion stage, the swarming development is carried out through attracting and repelling periods.
- Topology: From the standpoint of availability of particle knowledge, the PSO method is split into several topologies. All particles are linked to one another by "guest" particles. Indeed, all the electrons are mutually influenced. But each component is in the list topology attached to neighboring particles, and a network is established. Pyramidal morphology is another type of morphology that is like a three-dimensional triangle that displays the three-dimensional connection of the components. A virtual network in the star network impacts and influences the whole community. Smaller network topology consists of isolated sub-spans and particles, a homogenous example. The up/down and each neighborhood side are placed on a loop in a two-dimensional space in the von Neumann topology. The Vis-best topology proposed in this work for the first time is an averaging state of common lbest and guest topologies. In these topologies, the knowledge may be split between particles in a discreet immediate neighbor and between all particles in a particle

monitoring zone. In reality, the particles in each other's observing area are conscious of the excellent spot and can draw a closed condition to the facts.

Many more topologies have been spontaneously developed in addition to the above-stated topologies [8].

- Activity: There are two kinds of activities. Each particle attracts the other swarm in active mode to represent sexual interactions in a whole multitude. But with passive movement, a sociological phenomenon is not displayed in the entire hive, although there is an interest for each component from other nanoparticles.
- Group: There are two types of clustering. 1) The first type is an inactive (passive) agglomeration with a passive swarm and a physical property; e.g., as plankton is filled with liquid, the water flow maintains it. 2) The second type is a grouping that divides itself into two groups; i) A proactive accumulation in which an absorbency source is aggregated. Food or water may be this resource; ii) A gathering separate from the assembly. There are no environmental and physical variables in the absorption source. It is also split into two kinds: a) the passive kind, which gravitates from one molecule to another but does not show a social activity; and b) the social kind in which the particles have social interactions and are severely linked.
- Mobility: The PSO efficiency increases by updating the particle locations to use a clear illustration and dynamic methods occasionally. For example, the DAPSO algorithm was suggested to balance exploitation and exploration within PSO and preserve the various particles. The distance of each particle is computed for the optimal location to modify the particle velocity. However, conventional static methods are used in contrast.
- Divisibility: The PSO is classified into indivisible and non-divisible kinds from particle division. The main swarm is separated into sub-swarms to enhance the efficient algorithm or improve the swarm variation or multi-objectivity.
- Particle types: Particles in PSO sometimes are permitted to follow qualitative behavior rather than Newton's conventional dynamics. In other words, the particles utilize a

qualitative motion rather than a Newtonian one in the solution space. The findings are better than the classical state in high dimensions. In particular, the necessary parameters for setup are reduced.

In IEC standardization, FF is substituted with the concept of the user. That is, the user opines on each particle, considering current criteria, since the transferring of information between the particles of this iteration is not limited to the PSO, unlike EC and IEC. Since it also applies particle information for former repetition, the IPSO and IEC mechanisms are distinct. The IPSO is the same as the PSO since the user and not the FF user identifies the optimal particles.

- Particle trajectory sign: There are two viewpoints when determining the traveling route of the nanoparticles. From a positive point of view, the nanoparticles change positions from their best prior locations to the top international site in the swarming, which is the same as the traditional perspective. In the pessimistic view, the particles adapt to the worst places, i.e., they strive to avoid taking the worst classes.
- Recursivity: The PSO process presents two methods from a repetitive perspective factor. In the first view, information is utilized to adjust the process to prevailing circumstances, and we have a recursive PSO during the process. In the following perspective, however, the process has no control mechanism.
- Hierarchy: The hierarchical method of PSO attempts to position particles in a dynamic hierarchical structure in a way that is equivalent to the quality of the solutions given in a higher degree of hierarchical structure. The particles at higher levels influence the entire swarm more.
- Limitation: From the point of view of restrictions, the PSO is split into restricted and unrestrained kinds. The method is limited to a normal state, the same as the classic algorithm. The speed and position update formulae are the same in both situations. The exception is that in the traditional (constrained) case, there have been up and down limits for the location and speed when the limitations are exceeded. However, there is no such restriction in the UPSO situation [9].

- Synchronicity: This algorithm has been retrieved from the PSPSO algorithm. The distinction between synchronicity and a synchronicity in PSO is in positions and speed update calculations. The PAPSO performs continuously and gains opportunities to update the velocity and location of the particles. To decrease the imbalance, the algorithm dynamically balances the load with a chain-centered approach
- Combinatorial: The PSO variant, dubbed CPSO, is used with continuum and integer parameters to optimize the combining issues. The conventional PSO algorithm is its counterpart; it is just constant.
- Cooperation: Various stars can be utilized jointly to optimize distinct components of the case to enhance the productivity of classical PSO. It's known as CPSO. An unpleasant situation, however, will occur with a distinct swarm.
- Objective: In light of multi-target and single techniques addressing such issues, development models are classified into mobile and non-problem from the perspective of objective numbers. In the multi-target method, many targets with one swarm, but in accordance with the priority of the goals, are to be optimized.
- Another optimal combination: This technique has been coupled with other optimization techniques such as simulated annealing (SA), ant colony optimization (ACO) or genetic algorithm (GA) to enhance performance and resolve issues, such as entrapment in an optimal location, and promote variety to discover improved alternatives to PSO.
- Speed type: The speed parameter is key in the PSO to indicate the direction of the particle's motion. By altering this parameter utilizing several heuristics, numerous instances are shown in the taxonomies; and better results may be achieved.
- Uncertainty: In stochastic cases, information about stochastic models are utilized rather than utilizing guest information from the standpoint of the information source shared across swarms.

1.8 Properties of Swarm Intelligence

The typical swarm intelligence system has the following properties:

- It is composed of many individuals.
- The individuals are relatively homogeneous.
- The interactions among the individuals are based on simple behavioral rules that exploit only local information that the individuals exchange directly or via the environment.
- The system's overall behavior results from individuals' interactions with each other and their environment; that is, the group behavior self-organizes.

1.8.1 Models of Swarm Behavior

Craig Reynolds has created a 1986 artificial life software, Boids, which replicates bird flocking. His papers were published at the ACM SIGGRAPH Conference in 1987 on this topic. An abbreviated form of "bird-oid object" is referred to as "boid," referring to a bird-like item [10].

Boids are an example of emerging behavior, just like with most artificial lifetime simulations; which means the complexity of boids arises because of the interaction between the agents (in this instance, boids) and a series of basic rules. In the primary universe of boids, the following laws apply:

- Separation: steering to avoid crowding local flockmates.
- Alignment: steering towards the local flockmates' average heading.
- Cohesion: steering to move toward the average position of local flockmates.

More complicated rules such as preventing obstacles and searching for objectives can be implemented.

1.8.2 Self-Propelled Particles

In 1995, Vicsek *et al.* developed self-propelled particles (SPP), often known as the Vicsek model, based on a particular boid model previously established by Reynolds in 1986. In SPP, a swarm is modeled by a collection of objects that move at a steady rate but respond to a random disturbance by increasing the average intercultural experience of the other nanoparticles in their local area. The SPP models indicate that independent of the swarm animal species, the swarmed creatures share certain features at the governmental level. Jamming systems produce emergent behaviors, some of which are universal and stable at many different sizes. It has become a

problem in theoretical physics to create minimum statistical methods to represent such conduct.

1.9 Design Patterns in Cyborg Swarm

It was demonstrated that the performance of a swarm depends on several environmental and swarm characteristics. Consequently, the relative success of one cyborg control strategy over another control strategy is a function of these characteristics. Cyborg swarm designers should thus decide what cyborg control algorithms to implement based on the available mission facts. To this end, design patterns can provide a valuable set of guidelines.

A single design pattern can be understood as a "template" for a particular part of a cyborg control algorithm. For example, a design pattern might suggest how information about worksites is exchanged between cyborgs. Multiple design patterns can be combined to create a cyborg control algorithm suitable for a given mission, which can be implemented on all or a subgroup of cyborgs in a swarm because they include a description of cyborg behavior and discuss consequences of that behavior on macro-level swarm characteristics and performance; design patterns aid the decision-making of developers by providing solutions that work well in particular missions. In this research study, only design patterns for homogeneous cyborg swarms are considered. Therefore, any description of their consequences on swarm-level behavior is written assuming that all cyborgs execute the same control algorithm. However, in general, design patterns could also be applied in heterogeneous swarms, where only a sub-group of cyborgs would behave according to a particular design pattern.

The following essential mission characteristics that need to be considered when selecting appropriate design patterns have been identified throughout this research study:

- Worksite density, i.e., how probable it is that a worksite can be found by a cyborg, given its current location?
- Worksite volume, i.e., how quickly a worksite gets depleted when cyborgs perform work on it.
- Misplacement of reward from worksites, i.e., whether cyborgs need to travel away from worksites to obtain compensation, for example, to drop off resources in the base during the collection task.

- Dynamics of the environment, i.e., whether the worksite characteristics, such as their location, change over time. In dynamic environments, worksite characteristics are temporary, and reward needs to be extracted from worksites as quickly as possible.

Furthermore, there are four characteristics of a swarm that result from its control strategy:

- Scouting efficiency, related to how quickly the swarm can discover new worksites.
- Information gain rate, related to how quickly information enters into the swarm and spreads within it as a result of scouting and communication between cyborgs.
- Tendency to incur misplacement cost, CM, related to how much time is spent by cyborgs approaching worksites that they have become subscribed to, as well as the number of such cyborgs.
- Tendency to incur opportunity cost, CO, related to how long it takes cyborgs to discover that the worksites that they are subscribed to have disappeared from the environment, as well as the number of such cyborgs.

The first two swarm characteristics describe how cyborgs obtain information and share it. The last two characteristics define how efficiently a swarm can transform knowledge gained. The introduced Information-Cost-Reward (ICR) framework expresses the relationship between the hive, environmental factors, and swarm characteristics. Under this framework, a swarm is understood as a single entity capable of decentralized cognition at the collective level that emerges from individual cyborgs' information processing, actions, and interactions.

When the environment is static and worksites are challenging, control strategies with a high information gain rate are suitable. However, since high information gain rate is usually achieved by communication between cyborgs, it can be associated with an increase in the tendency of swarms to incur misplacement and opportunity costs. Consequently, when worksites are relatively easy to discover, hives a high propensity to perform than swarms with a lower. Furthermore, when the environment is dynamic, i.e., when new information is generated over time, a good scouting efficiency and a common tendency to incur costs are more critical. Finally, prices can be facilitated by the environment. For example, the negative effect of

misplacement cost during the collection task is more minor in swarms where cyborgs exchange information where resources need to be dropped off.

In this chapter, general lessons learned from simulation experiments are formalized as design patterns. Each pattern captures a specific aspect of cyborg behavior and describes environmental conditions for a suitable design choice. The methodology for design pattern creation is first introduced, followed by a catalog of seven design patterns. Next, the rules for combining design patterns with control strategies are presented, and examples are given. The chapter ends with a discussion of the relevant aspects of existing swarm cyborg literature. It is shown that the design patterns introduced here can be found in cyborg control algorithms used in a broad range of automated experiments and that the essential characteristics of these design patterns remain the same across different implementations. Finally, it is demonstrated that the methodology for design pattern creation established here can be applied to extend the design patterns catalog. The new design patterns can easily be combined with the existing ones.

In the swarm work cycle within the Information-Cost-Reward framework, worksite distribution in the environment is characterized by the probability, p(W), of a worksite being located at a given point in space. Scouts search for and spread information about worksites. This process is affected by the swarm's scouting and communication strategies and produces the expected reward, R′, while reducing the swarm's uncertainty cost, CU. After paying certain misplacement costs, workers turn information into reward, R, CM. They also alter the environment by depleting worksites, decreasing p(W), and potentially increasing opportunity cost, CO, paid by the swarm.

Design methodologies, such as probabilistic finite state machine models and evolutionary algorithms, and relevant future research directions are identified.

1.9.1 Design Pattern Creation

A total of nine cyborg control strategies were explored throughout the primary control strategies; solitary, local broadcaster, and bee were experimented with separately. Two add-on strategies, opportunism and anticipation, were analyzed with the three primary methods to determine whether they improved swarm performance in dynamic environments. To translate these strategies into design patterns, logic modules, each describing a

unique aspect of a cyborg control algorithm, need to be identified. Inspired by the object-oriented design pattern principles, it is proposed here that a swarm cyborg design pattern should:

- Describe a particular stand-alone module of a cyborg control algorithm regarding cyborg behaviors, relevant internal and external data structures, and relationships between them. Such a module should satisfy a particular functional requirement, and its description should be independent of other modules that deal with different needs.
- Describe suitable environments and swarm tasks where the pattern is understood to be an appropriate design choice.
- Be possible to combine with other design patterns.
- Be implementation non-specific, i.e., only describe high-level behavior, rather than a particular algorithm or implementation.

The last point is critical. To generate general knowledge from specific experiments, we need to dispose of implementation details, such as particular control algorithms or parameter values, and instead identify available patterns of cyborg behavior and their implications on swarm performance. The Information-Cost-Reward framework is helpful in this endeavor, as it describes cyborg behavior in terms of "information flow" instead of "control code" and the result of the behavior in terms of individual "costs" rather than simply "performance improvement" or "performance degradation." Descriptions of swarm behavior that use the ICR framework are thus not dependent on particular cyborg hardware or software. The behavior results can be understood in detail concerning environmental characteristics.

An essential step in creating a design pattern is identifying what category it should belong to. Categorizing the set of processes that need to be formalized as a design pattern can assist in choosing what cyborg behaviors and data structures are relevant. Based on the performed experiments with the consumption and collection tasks, it is proposed here that a swarm cyborg design pattern should belong to one of three categories, each answering a particular question:

- Transmitter patterns: What entity transmits information?
- Exchange patterns: Where and when is information shared?
- Update patterns: How is information updated?

Each design pattern description should include the following:

- Design pattern name and category;
- A list of suitable applications;
- Description of the cyborg behaviors;
- Dependencies on other behaviors of the cyborg, including recommendations about which other design patterns it works effectively with;
- A list of parameters associated with the cyborg behaviors, as well as, if possible, their impact on swarm performance;
- A list of consequences that the design pattern has on swarm characteristics, expressed in the terminology of the ICR framework. Since it is often not immediately apparent by looking at cyborg behavior descriptions, the design pattern consequences should identify how particular micro-level routines affect the emergence of macro-level outcomes.

The design pattern name, category, suitable applications, behavior description, and consequences are compulsory. On the other hand, some patterns might not have dependencies or parameters.

1.9.2 Design Pattern Primitives and Their Representation

An essential part of a design pattern is an explicit description of the cyborg behaviors that the mark represents. A visual description, i.e., a diagram, is often handy when a design pattern needs to be understood quickly. A textual description that follows a well-specified syntax is more suitable when a design pattern needs to be translated into program code.

A visual and a textual description require a well-specified modeling language with unambiguous syntax and semantics to clearly express relevant entities and processes. Various modeling languages for object-oriented design patterns exist, for example, UML. While these languages are helpful in traditional software development, their utility for modeling multi-agent embodied systems, such as cyborg swarms, is limited for two main reasons. Firstly, data in these languages is not defined explicitly, making it challenging to express where information is stored or how operations are done. As the ICR framework has demonstrated, information flow is as essential as cyborg behavior when it comes to understanding and designing cyborg control algorithms, meaning that an adequate representation of data is required. Secondly, swarm control algorithms often rely on cooperation or communication between cyborgs. A way of representing relationships

between behaviors and data of two different cyborgs is needed to account for mechanisms that lead to the emergence of desired macro-level outcomes.

Therefore, due to the shortcomings of the existing modeling languages, a new Behavior-Data Relations Modeling Language (BDRML) is proposed here. Inspired by other modeling languages, such as UML and DisCo, BDRML defines a set of primitives that represent behaviors and data, a set of relationships, i.e., relations, between these primitives and a bunch of operations allowed on the primitives. All these language elements have their visual and textual representations in BDRML.

- Behavior, i.e., a set of processes that deal with a particular situation a cyborg finds itself in, for example, "Scout" or "Rest."
- Internal data, i.e., information that represents a particular aspect of the environment or a cyborg's internal state and is stored in the cyborg's memory.
- External data, i.e., information that is stored by a non-cyborg entity in the environment, for example, by an RFID tag or by the presence of a chemical substance in an ant-inspired swarm.

Note that "behaviors" in BDRML, such as "work" or "scout," can refer to "states" or "sets of states" in finite state machines. In neural network controllers, "behaviors" would not be programmed explicitly but manifest through the network dynamics.

Also, a crucial difference is noted between internal and external data. Internal data is readily available to a cyborg at any point, while external data must be found in the environment. Moreover, when information between cyborgs needs to be exchanged, data stored internally can only be passed from one cyborg to another when they meet. On the other hand, one cyborg can deposit external data into the environment and read by another cyborg later.

Since both behaviors and data are primitives, BDRML allows the relations between cyborg actions and information to be formulated. There are sevens types of connections possible:

1) Transition: a behavior-behavior relation, where the cyborg transitions from one behavioral mode to another.
2) Read: a behavior-data relation, where internal data, stored in the cyborg's memory, is used by the cyborg when it is engaged in a particular behavioral mode.

3) Write: a behavior-data relation, where internal data is stored into the cyborg's memory when it is engaged in a particular behavioral mode.
4) Receive: a behavior-data relation, where external data, stored in the environment, is used by the cyborg when it is engaged in a particular behavioral mode.
5) Send: a behavior-data relation is sent, where external data is stored by the cyborg into the environment when it is engaged in a particular behavioral mode. Alternatively, internal data of another cyborg is written in when a cyborg is in a specific behavioral model.
6) Copy: a data-data relation, where information is copied from one data primitive to another (for example, from an external to an internal data structure that represents the same information).
7) Delete: a data operation, where data is disposed of.

It is also necessary to define a set of conditions under which a particular relation or operation occurs. A state is visually represented as an annotated triangle at the beginning of a relation or operation arrow. A condition set follows a relation or operation signature in a textual representation and is separated by a colon. A condition may be annotated as a name of a Boolean function or a probability, as existence or non-existence of a data structure, or as a simple and unambiguous textual description. A particular type of condition is an "always" condition, represented by a star (*). Visually, a relation or an operation with an "always" condition may be described without the condition triangle symbol. Multiple conditions can affect a single link or operation. The "or" logical operator is assumed when requirements are combined unless otherwise specified.

Note that there are three types of lines used in BDRML. Single solid lines represent transitions between behaviors and read/write relations between behaviors and internal data structures. Double solid lines represent some form of communication and link external data structures with behaviors (in the case of the "receive" and the "send" relations) and with internal data structures (in the case of the "copy" operation). Double solid lines can also link behavior with an internal data structure, i.e., during the "send" operation, signifying that a cyborg engaged in a particular behavioral mode sends information to another cyborg that stores the data in its memory. Finally, dashed lines are used for annotating relation and operation conditions.

A design pattern representation in BDRML consists of both a visual and a textual specification. A set of primitives (V) is defined, followed by a list of their relations and operations. Each box and arrow in the visual representation must have a corresponding line in the textual representation and vice versa.

It is essential to point out that while the design patterns are not implementation-specific, by default, it is assumed that:

- The swarm is homogeneous.
- The cyborgs can sense their environment locally. In particular, they can feel the presence of worksites. They can also sense the presence of other cyborgs and obstacles nearby and resolve collision conflicts.
- The cyborgs have a read/write internal memory.
- In the case of some design patterns, it is expected that cyborgs are capable of communicating with other entities.

These requirements are satisfied by the majority of cyborgs currently being used in swarm cyborg experiments, such as the e-pucks, the eSwarBots, the kilobits, the s-bots, and the marXbots. Future extensions to design patterns could accommodate heterogeneous swarms or swarms with other non-standard properties.

1.10 Design Patterns Updating in Cyborg

This section presents seven design patterns belonging to one of the design pattern categories: transmitter, exchange, and update. The design is an example of a design pattern specified in BDRML. The design pattern consists of three primitives: behaviors B1, B2, and an internal data structure D1. A cyborg changes its mode from B1 to B2 when a Boolean function f returns true. The cyborg transitions from B2 to B1 with a probability p(G). While engaged in behavior, B2 writes to and reads from D1. Patterns are primarily based on the control strategies used in experiments throughout, and relevant chapters are mentioned in the design pattern descriptions where appropriate. In some cases, the author's previously published work is also referenced.

The patterns are defined in the following format. First, the pattern's name and category are given, followed by a list of applications for which the design is suitable. The cyborg behaviors and data structures that the

mark represents are described in plain text and BDRML. A list of the pattern's dependencies, behavior parameters, and consequences follows.

It is important to remember that a design pattern is not equivalent to a complete cyborg control strategy. Depending on the pattern's category, only a particular aspect of the cyborg control algorithm is described, such as how information is obtained, transmitted, or updated. How multiple design patterns can be combined into cyborg control strategies is formalized and demonstrated.

1.10.1 Behaviors and Data Structures

- Information about worksites is easily obtainable; for example, when worksite density is high.
- It is strongly recommended if, in addition, new information is generated in the environment over time and continuous exploration is thus important.

A cyborg scouts the environment and can find a worksite with a probability p(F). Upon seeing a worksite, the cyborg begins work and stores the information about the worksite, such as a local vector towards it, in an internal data structure. The data structure may be updated periodically while the cyborg works.

The cyborg ignores any information and actions of other members of the swarm.

An overview map of design patterns: Design pattern categories are indicated on the left. Design pattern parameters are shown in italics below each print. Daines joins design patterns that can be combined.

- Leads to a low information gain rate, which is the reason why information needs to be readily available to cyborgs.
- The spread of cyborgs across worksites only depends on their movement pattern. If the movement of cyborgs is random, an even spread across the environment is achieved.
- Prevents spread of erroneous information.
- Minimizes any misplacement and opportunity costs.

1.10.2 Basics of Cyborg Swarming

A cyborg scouts the environment and can find a worksite with a probability p(F). Additionally, it can receive information about a worksite from other cyborgs while engaged in the "Work" behavior. When an informed and an

uninformed cyborg meet, they form a temporary peer-to-peer connection. The uninformed cyborg gets recruited to the worksite, stores information about it in its internal data structure, and begins work. Similarly, a scout remembers and starts working on a worksite that it discovers on its own. The cyborg's internal data structure may be updated periodically while the cyborg works.

- Cyborg communication range: A more extensive communication range causes a higher information gain rate and increases misplacement and opportunity costs. Consequently, performance can decrease due to congestion and overcommitment to worksites.
- Information about worksites is more easily accessible by uninformed cyborgs.
- Information is carried and transmitted by cyborgs, meaning that the information gain rate depends on the probability of cyborgs meeting, i.e., on their movement algorithm and the structure of the environment.
- Causes the cyborgs to incur misplacement costs associated with traveling to worksites after being recruited.
- Increases the probability of cyborgs incurring opportunity costs as a result of outdated information being spread across the swarm.
- Can lead to spread of erroneous information.

1.10.3 Information Exchange at Worksites

Uninformed cyborgs are likely to encounter information transmitters, i.e., other cyborgs or non-cyborg information storage devices near worksites, for example:

- In the consumption task, during which cyborgs remain at worksites until they are depleted.
- When a combination of worksite density, the cyborg scouting strategy and the communication range of the transmitters and the uninformed cyborgs leads to the probability of information exchange that is likely to be higher than with alternative exchange design patterns.

Cyborgs only exchange information near a worksite that an informed cyborg is working on. Note that in the BDRML syntax, the conditions of

the two relations that connect the "Work" behavior with the "Worksite data int." and "Worksite data ext." data structures have an "and" operator. This notation assures that both "at a worksite" conditions always have to be met when this design pattern is combined with other ways.

- The information gain rate depends on the structure of the environment and the communication range of transmitters.

Parameters

- Proximity threshold: a maximum distance at which a cyborg is considered to be "at a worksite."
- After an initial worksite discovery by a cyborg, the range at which other cyborgs can find the worksite is enlarged, increasing the swarm's scouting success.

1.10.4 Information Exchange Center

- Static and dynamic environments where worksite density is very low.
- Environments with mediocre worksite density, where the swarm task requires cyborgs to become misplaced from worksites (e.g., the collection task), provided that the Information Exchange Center is identical to the place where cyborgs need to travel to periodically as a part of their task.

Cyborgs meet at the Information Exchange Center (IEC) to exchange information. There are two types of cyborgs found at the IEC: informed cyborgs that provide information and uninformed cyborgs that search for information.

An informed cyborg pauses its work and returns to the IEC when its boolean transmission initiation function, t, returns true, to begin providing information at the IEC. The cyborg leaves the IEC based on a transmission expiry function, r, and resumes work.

An uninformed cyborg outside the IEC, i.e., a scout, periodically returns to the IEC based on a scouting expiry function, to check whether new information is available. If the cyborg finds information about where work is located, it transitions to the "Work" behavior and leaves the IEC. If no information is available in the IEC, the uninformed cyborg goes to the IEC and resumes scouting according to a scouting initiation functions.

- The swarm's scouting efficiency decreases because scouts return to the IEC based on a scouting expiry function. This function must fit the nature of the environment (for example, enough time must be given to scouts to explore a large or a highly dynamic working arena); otherwise, the swarm might be unable to discover worksites.
- Transmission initiation function, t: a rule that causes informed cyborgs to return to the IEC. For example, the need to drop off resources in the base during the collection task.
- Transmission expiry function, r: a rule that causes informed cyborgs to leave the IEC. For example, if the IEC pattern is combined with the Broadcaster pattern, the expiry of a recruitment time can trigger the cyborgs to resume work.
- Scouting expiry function, u: a rule that causes scouts to return to the IEC. For example, the expiry of a maximum scouting time.
- Scouting initiation function, s: a rule that causes uninformed cyborgs in the IEC to become scouts. For example, each second, cyborgs might become scouts with a certain scouting probability.
- Information gain rate depends less on the environment's structure than on the cyborgs' communication range and on the cyborg movement algorithm. The variance in information gain rate is slight across different environment types.
- Promotes spatio-temporal coordination between cyborgs. This is advantageous when a single worksite exists in a static or dynamic environment. On the other hand, the swarm performance is poor when the swarm needs to concentrate on multiple worksites simultaneously.
- Incurs high misplacement and opportunity costs relative to the other exchange patterns. The amount of incurred costs depends on the structure of the environment, especially on the worksite distance from the IEC. A more considerable worksite distance generally leads to higher prices.

1.10.5 Working Features of Cyborg

A cyborg continues to work from a worksite that it discovers and does not abandon the worksite until it is depleted.

- The swarm behavior is relatively easy to design and predict.
- Opportunities for a better swarm performance might be missed.

1.10.6 Highest Utility of Cyborg

Environments where it is essential to extract rewards from worksites with the highest utility:

- Static environments, where there is a time limit on how long a swarm can work.
- Dynamic environments.

A cyborg continuously evaluates the utility of its worksite. It compares it to the utilities of all other worksites that it can obtain information about (as a result of scouting, communication with other cyborgs, or discovery of a lead in external data storage devices). The cyborg abandons its worksite and subscribes to a new one when a Boolean switch function, w, returns true.

- Unregulated information exchange can lead to a high increase of the displacement and opportunity costs incurred by the cyborgs and poor sampling of the environment. For example, if Opportunism is combined with the IEC pattern, that promotes a high information gain rate, overcommitment of the majority of the swarm to a single worksite can occur, significantly decreasing the swarm performance
- It is recommended to combine the Opportunism pattern with a transmitter or an exchange pattern where information flow is regulated to some extent, for example, the Information Exchange at Worksites pattern.
- Switch function, w: a rule that a cyborg switches from working on a worksite W1 with a specific utility U1 to a worksite W2 with a "better" utility U2. For example, a real-value threshold can specify how much U2 should be greater than U1.
- Promotes preferential exploitation of high-reward worksites.
- Requires sampling of the utility of all worksites that a cyborg encounters while working. This can imply additional energy costs to the cyborg.

- If Opportunism is combined with another design pattern involving communication between cyborgs, additional data packets about worksite utilities must be exchanged during communication. In addition, more frequent communication between cyborgs is required, as worksite utilities of other swarm members need to be evaluated whenever possible. This can imply additional energy costs and data error accumulation and propagation.

1.10.7 Gain Extra Reward

Environments where it is essential to extract reward from worksites with the highest utility:

- Static environments, where there is a time limit on how long a swarm has to work.
- Dynamic environments.

– A cyborg continuously evaluates the utility of its worksite and abandons the worksite when a Boolean abandonment function, a, returns true.
– Worksite abandonment leads to information loss and thus to a higher amount of the swarm's higher uncertainty cost, therefore abandoned worksites must be able to discover new information relatively quickly.
– It is recommended to combine the Anticipation design pattern with another pattern that leads to a high information gain rate, e.g., the Information Exchange Center pattern.
– Abandonment function, a: a rule according to which a cyborg abandons its worksite. For example, the worksite might be left when its utility falls under a specified threshold.
– Promotes frequent sampling of the environment.
– Prevents overcommitment to worksites and congestion.

1.11 Property of Design Cyborg

An essential property of design patterns is that they can be combined into a specific cyborg control algorithm, i.e., a control strategy. A BDRML representation of a control strategy can be created by following six design pattern combination rules:

- Copy all sets of behaviors, B, from all patterns into a new behavior set, B′, i.e., B′ = {B1 ∪ B2 ∪ ... ∪ Bn}.
- Copy common data structures from design pattern data structure sets D into a new data structure set, D′, i.e., D′ = {D1 ∩ D2 ∩ ... ∩ Dn}.
- Add additional data structures that have a genuine relationship with behavior and do not already belong to the set D′ into D′. This allows one pattern to extend the information processing routines of another.
- Copy all relations between the primitives that belong to sets B′ or D′, including their conditions. If otherwise specified by a relation condition, assume the "or" operator when combining shapes.
- Copy all operations on the primitives that belong to sets B′ or D′, including their conditions. If otherwise specified by an operation condition, assume the "or" operator when combining shapes.
- Delete all relations that belong to shorter relation paths between behaviors and data structures (but not between behaviors).

A relation path specifies a set of links that lead from a primitive V1 to a primitive V2, including those relations that pass through other primitives and create an indirect connection between V1 and V2. If multiple relation paths exist between a behavior and a data structure after multiple design patterns have been combined, removing relations that belong to shorter relation paths allows one design pattern to redefine communication routines of another. For example, imagine that a direct-write link between "Work" and "Worksite data" exists in design pattern P1. Another design pattern, P2, defines that there is a transition between the "Work" and a "Stay home" behavior and a write relation between "Stay home" and "Worksite data," but that "Work" and "Worksite data" are not directly related. When P1 and P2 are combined, the relation between "Work" and "Worksite data" should be deleted so that P2 can redefine the communication routine suggested by P1.

Broadcaster and Information Exchange at Worksites can be combined to create the "local broadcaster" control strategy. First, a set of behaviors that belong to both patterns is found. This set includes the "Scout" and the "Work" behaviors. Next, the "Worksite data int." data structure that belongs to both patterns is included in the control strategy. The "Worksite data ext." data structure and its relations are not copied since the system does not

have a read relation to any behavior. The ties between all primitives that belong to the control strategy and their conditions are also included. The requirements of the connection between "Work" and "Worksite data int" are combined using the "and" operator, as specified by the Information Exchange at Worksite pattern.

The "local broadcaster" control strategy results from combining the Broadcaster and the Information Exchange at Worksites design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and links drawn from the Information Exchange at Worksites pattern are green. Primitives and relations not copied are established as rough text, but they are not shown visually.

The "bee swarm" control strategy combines the Broadcaster and the Information Exchange Center design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and relations, drawn from the Information Exchange Center pattern, are green. Primitives and relations that were not copied or deleted are shown as strikethrough text, but they are not shown visually.

Combining the Broadcaster and the Information Exchange Center design patterns results in the "bee swarm" control strategy. The control strategy has four behaviors and one data structure, "Worksite data int.". Similarly, as was the case with the "local broadcaster" control strategy, "Worksite data ext." is not copied from the Information Exchange Center pattern since it does not have a read relation to any behavior. The send relation between "Work" and "Worksite data int.", defined in the Broadcaster pattern, is deleted since a longer relation path that passes through the "Provide data in the IEC" behavior, inherited from the Information Exchange Center pattern, exists.

Three design patterns can be combined by following the same combination rules. It shows how the Broadcaster, Information Exchange at Worksites, and the Opportunism patterns form a "local broadcaster with opportunism" control strategy. The "Scout," "Work," and "Work data int." primitives are shared among the patterns and are a part of the control strategy. The write relations between "Worksite data int." and the two behaviors inherit conditions from all three design patterns. Note the condition that belongs to the read relation between "Work" and "Worksite data int": *, w. According to the Broadcaster pattern, the "Worksite data int." is continuously (*) updated while the cyborg is working. Additionally, the Opportunism pattern suggests that a new worksite should be adopted while a cyborg is in the "Work" behavior and finds a "better" worksite, based on the switching function, w.

Apart from a BDRML representation of the cyborg behavior, other characteristics of design patterns should be considered together when design patterns are combined. The "local broadcaster with opportunism" control strategy results from combining the Broadcaster, the Information Exchange at Worksites, and the Opportunism design patterns. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives and links are drawn from the Information Exchange at Worksites pattern shown in additions drawn from the Opportunism pattern, which are shown in magenta. Primitives and references not copied are established as strikethrough text, but they are not shown visually.

The list of suitable applications becomes more specific when multiple patterns form a control strategy. Or, from a developer's point of view, a more detailed specification of the swarm's environment and task allows for a higher number of design patterns to be combined with greater confidence. For example, suppose a swarm is required to collect easy-to-find rubbish from a city square but has no specific constraints for its operation. In that case, the design patterns catalog suggests implementing a cyborg control strategy by combining the Individualist and Blind Commitment patterns. On the other hand, an application may be more specific; for example, a swarm may be required to collect minerals that are difficult to find and appear in mineral veins of varying richness, while the cyborgs can only operate with enough sunshine provided for their solar batteries. In this case, the design patterns catalog would suggest combining the Broadcaster, the Information Exchange Center, and the Opportunism patterns.

Secondly, the list of control strategy parameters grows when multiple patterns are combined. To avoid creating a cyborg control algorithm with an ample parameter space that needs extensive optimization, it is advisable to prefer more straightforward control strategies based on several design patterns that are as small as possible. Similarly, design patterns with smaller parameters should be selected unless there is a reason for using a more complicated way. The problem with parameters is that they require decisions made by cyborg designers when the swarm is being built. Unless an exhaustive list of situations is considered during the optimization phase or a suitable online parameter learning algorithm is implemented, each new parameter can lead to undesirable results. For example, the Information Exchange Center pattern requires a parameter set for the "scouting expiry function" to specify when a scout should return to the IEC. Setting this parameter to an inappropriate value can prevent the swarm from discovering worksites when not enough time is given to scouts to explore the environment or communication between cyborgs when scouts spend too much time outside of the base do not meet with recruiters on the ground.

Before discussing how the design patterns proposed here have been used in the existing literature, it is essential to consider the level at which they describe cyborg behavior and its categorization. It has been suggested that design patterns for swarm cyborgs should represent multiple levels of behavior. For example, "local-level" or "basic" patterns should describe how cyborgs interact, while "global-level" or "composed" patterns should describe swarm-level behavior, such as "labor division." On the contrary, all design patterns presented here represent the same cyborg behavior level, equivalent to the "local-level primitives" of the "basic design patterns." The control strategies, for example, "solitary swarm with anticipation," represent a combination of design patterns and are similar to the "global-level primitives" or the "high-level patterns." The control strategies, however, do not design patterns themselves. Instead, they are particular design pattern realizations in swarm applications that fit specific mission requirements.

It is proposed here that information and cost-based description of the individual, "local level," cyborg behavior is an appropriate level at which design pattern should be defined. A detailed, lower-level description that deals with a particular object-oriented or functional implementation on a cyborg would have to include details about a specific experiment or special cyborg hardware that would potentially not be useful to a developer with slightly different hardware or application. Similarly, a description of macroscopic, "global-level" swarm behavior, for example, a "flocking pattern," would be a re-description of a combination of cyborg behaviors that fit a particular swarm mission. Such a global-level description would also potentially contain a lot of parameters. On the other hand, describing parts of cyborg behavior that deal with particular problems without providing too many implementation details allows for modularity and reusability.

While they describe the same level of behavior, the design patterns presented here are categorized based on what particular aspect of cyborg behavior they represent concerning obtaining, sharing, and updating cyborg information. They thus follow the categorization methodology of object-oriented design patterns and multi-agent systems design patterns. Each design pattern belongs to three categories: transmitter, exchange, and update.

Various combinations of these patterns can be found throughout the literature. The Individualist and Blind Commitment patterns are often used when simple foraging algorithms are needed for cyborg behavior. In contrast, swarm behaviors, such as self-regulation or task allocation, are explored.

The Individualist pattern is also often used when the performance of swarms without and with communication is compared. It is usually the

case that hives in such experiments forage from difficult deposits, resulting in better swarms that utilize communication.

A combination of the Broadcaster and Information Exchange at Worksites patterns has been explored. Confirming the design pattern characteristics presented here, the authors showed that increasing the strength of interaction between cyborgs (e.g., due to a giant swarm size or an extensive communication range of cyborgs) leads to sub-linear performance improvement. In other words, in the ICR framework terminology, a high information gain rate often leads to high misplacement and opportunity costs that prevent the performance from improving linearly with the amount of information that the cyborgs can get. Similarly, it is argued that recruitment in swarms that used the Broadcaster and the Information Exchange at Worksites patterns leads to increased congestion (i.e., a higher misplacement cost) and the propagation of old information through the swarm (i.e., a higher opportunity).

The work of swarms that used a combination of the Broadcaster and Information Exchange at Worksites patterns did not outperform swarms that used the Individualist pattern. The authors proposed that the relatively poor performance of swarms that utilized communication resulted from communication noise. However, the characteristics of the Broadcaster design pattern point to two additional possible explanations. Firstly, only four cyborgs were used in the experiments, and it is possible that they did not meet often enough for communication to make a positive difference to their performance. Secondly, the cyborgs were collecting pucks that were pretty far apart from one another, considering the size of the cyborg body. A cyborg recruited to a puck thus searched to locate another puck nearby. In such a setup, it is possible that the negative effect of misplacement and opportunity costs outweighed the positive impact of recruitment.

The Broadcaster pattern has also been combined with the Information Exchange Center pattern, often in bee-inspired cyborg swarms and agent-based simulations. In these experiments, cyborgs collect items from the environment and return them to the base, where they also recruit in a peer-to-peer fashion. Since the Information Exchange Center pattern is the most suitable when items of interest are difficult to find but need to be collected into a central place, swarms that use it outperform other, non-communicating, swarms in foraging experiments where items of interest are clumped in a small number of patches. Interestingly, in their simulations, contrary to the characteristics of both of these patterns, they did not find any adverse effects of communication, such as the increase of congestion or fast depletion of resource deposits. They thus claimed that the swarm performance increases linearly with swarm size. A closer inspection

of their algorithm reveals that their agents were allowed to occupy the same space, meaning that the physical aspect of agents was not fully modeled, preventing misplacement costs from being incurred due to congestion. Furthermore, their simulations could not incur opportunity costs since resource deposits had unlimited volumes.

Cyborg control algorithms that contain the update patterns can also be found throughout the literature. The Opportunism pattern has been used in foraging simulation experiments where agents preferred to head towards resource deposits closer to the base, i.e., residues that allowed a faster collection of resources. It showed that Opportunism causes the majority of a swarm to concentrate on a single resource deposit when the information spread in the swarm is not regulated.

Opportunistic behavior, where a swarm prefers to forage from more profitable resource deposits, can also be achieved when cyborgs that utilize the Information Exchange Center pattern and recruit in the base, recruit for a more extended amount of time when their deposits are more profitable. Even though the mechanism of achieving opportunism is different than when unemployed cyborgs prefer to be recruited to better promises, the results of such behavior are similar. In line with the Opportunism design pattern characteristics, Schmickl *et al.* demonstrated that a sufficient amount of scouting in a swarm where cyborgs behave opportunistically is significant for the ability of the swarm to react to environmental changes appropriately.

Finally, the Anticipation design pattern has been used in a control algorithm that allowed cyborgs to decide which type of puck they should search for to maintain the desired density of puck types in a drop-off location. The abandonment function, which caused a cyborg to stop foraging for a particular puck type, was related to the locally perceived behavior of other swarm members.

Design patterns allow us to consider a broad range of experiments with different cyborg hardware and identify building blocks of cyborg behavior that fit specific swarm mission requirements. For example, other design methodologies exist, probabilistic finite state machine models and evolutionary algorithms. Unlike design patterns, these methodologies are more suitable for parameter optimization than behavior selection. Therefore, design patterns complement these methodologies when developing cyborg control algorithms.

1.12 Extending the Design of Cyborg

The transmitter design patterns presented above did not communicate or rely on local, peer-to-peer communication between cyborgs. Another

type of communication, called stigmergy, involves exchanging information between agents through the environment. Algorithms that utilize stigmergy are often inspired by the pheromone-based touch characteristic of ant colonies. To help their nestmates search for food, ants leave chemicals called pheromones on the ground. An extended overview map of design patterns shows the new ways in green. Design pattern categories are indicated on the left. Design pattern parameters are shown in italics below each print. Lines join design patterns that can be combined, and any extra parameters required for the pattern combination are shown next to the bars.

They travel back and forth between the nest and the food, forming trails in the environment. Other ants can sense pheromones and thus use the pheromone trails to navigate foraging. The evaporation rate of pheromones assures that a path no longer being used, for example, because the food source has been depleted, eventually disappears and does not recruit more workers.

There are two aspects of stigmergy that are interesting from the design pattern perspective. Firstly, it involves a stationary and external medium to the cyborgs and holds information relevant to the swarm's work. Secondly, it is the fact that information is available in many locations across the work arena, rather than only being exchanged in the base or near worksites. Two design patterns can be created to capture these aspects of cyborg behavior A transmitter pattern, called Information Storage, according to which information is stored in data storage devices, and an exchange pattern, Information Exchange Any Time, according to which data can be exchanged anywhere in the work arena. This section formalizes these two patterns using information from experiments found in the swarm cyborgics literature. Their description is not as detailed as the patterns presented since no experiments that could thoroughly test the suggested cyborg behaviors have been performed yet. Nevertheless, it is demonstrated here that the new design patterns can easily be combined with the other design patterns according to the design pattern combination rules defined, which shows unique ways on an extended design pattern map.

1.12.1 Information Storage in Cyborg

A cyborg scouts the environment and can find a worksite with a probability p(F). Additionally, it can receive information about a worksite if it finds a data storage device located in the environment. Once a cyborg discovers information about a worksite, either as a result of scouting or when seeing a data storage device, it stores data about it in its memory and begins

work. The cyborg's internal data structure is updated periodically while the cyborg works.

An informed cyborg stores information about its worksite into a data storage device(s) when appropriate. For example, when the design pattern is combined with the Information Exchange Any Time pattern, special data storage devices, such as RFID tags, may be dropped into the environment and updated by the cyborg. Chemicals that mimic ant pheromones, such as alcohol, can also exist. Alternatively, stationary cyborgs that do not directly participate in work can store information. On the other hand, when the Information Storage and the Information Exchange Center design patterns are combined, data is stored in a central location, for example, in the cyborg base.

The information is deleted from the storage device(s). According to an evaporation function, the data deleted from the storage device(s) is how long the information about worksites remains available in each storage device, i.e., the life span of the stored data. The function must consider the dynamics of the environment. If the information life span is too long, cyborgs follow information to depleted worksites and incur a high opportunity cost. On the other hand, a brief information life span prevents cyborgs from finding and utilizing the stored data.

- Evaporation function, e: a rule according to which information in the data storage device(s) is deleted or considered too old. This function plays a similar role as the evaporation rate of ant pheromones. For example, information might have a pre-defined life span. Upon life span expiration, the storage device deletes the information if such an ability has been programmed into it. Alternatively, cyborgs that read the report also evaluate its age and death there and decide whether it should exist.
- Detection range: a range at which a cyborg can find a storage device.
- Information about worksites is more easily accessible by uninformed cyborgs.
- Information is stored in the environment, meaning that the information gain rate depends on the probability of cyborgs detecting the information storage devices, but not on the likelihood of cyborgs meeting each other.
- Causes cyborgs to incur misplacement and opportunity costs due to recruitment to remote worksites. The extent of these costs increases with an increasing swarm size due to

congestion. However, the expenses paid may be smaller than when the Information Exchange Center pattern is used since data storage devices may be closer to worksites.

1.12.2 Information Exchange Any Time

Maximum storage device density: When the Information Exchange Any Time pattern is combined with the Information Storage pattern, the maximum allowed information storage device density must be specified to prevent the environment from being cluttered with storage devices. For example, the minimum distance between two RFID tags should be set. This could be related to the frequency at which the chemical is deposited into the environment in chemical trials.

1.12.3 The New Design Pattern Rules in Cyborg

The new design patterns can be combined with other ways from the catalog by following the design pattern combination rules.

Using the Information Storage and the Information Exchange Any Time patterns together results in an ant-inspired cyborg control strategy, resulting from combining the Information Storage and the Information Exchange Any Time design patterns. Primitives, relations, and operations of the Information Storage pattern are black. Other relations, drawn from the Information Exchange Any Time pattern, are green. Relations that were deleted are shown as strikethrough text, but they are not shown visually.

Cyborgs utilize their energy to find information about worksites in the environment. Cyborgs are designed by combining the Information Storage and the Information Exchange Center patterns to store information about where worksites are located in the base. Unsuccessful scouts arrive at the bottom to read the news and begin work.

Combining the Information Storage and the Information Exchange Center design patterns is a control strategy. Primitives, relations, and operations of the Information Storage pattern are black. Additional primitives and references drawn from the Information Exchange Center pattern are green. Links that were deleted are established as strikethrough text, but they are not shown visually.

Combining the Broadcaster and the Information Exchange Any Time design patterns is a control strategy. Primitives and relations of the Broadcaster pattern are shown in black. Additional primitives drawn from the Information Exchange Any Time pattern are green. Primitives and

deleted links are established as strikethrough text, but they are not shown visually.

The Information Exchange Any Time pattern can also be combined with the Broadcaster pattern, leading to behavior where cyborgs exchange information when they meet anywhere in the work arena.

1.13 Bee-Inspired Cyborg

A design pattern represents a particular aspect of cyborg behavior that addresses a specific swarm mission requirement, such as finding worksites given a particular density of worksites dealing with specific environmental dynamics. A design pattern is created by considering the results of experiments with a particular behavior of cyborg, for example, a bee-inspired information exchange in a central "base" location, and by generalizing knowledge learned during the experiments using the Information-Cost-Reward framework.

Each design pattern presented here belongs to three categories: transmitter, exchange, and update. Transmitter patterns identify entities that should transmit information. For example, a transmitter pattern might suggest that cyborgs share information or exchange information via RFID tags placed in the environment. Exchange patterns dictate where data should be exchanged, such as whether cyborgs can communicate when they meet each other or whether a specific meeting place should be designated. Update patterns deal with how individual cyborgs update their information, such as whether they continuously search for "better" worksites or decide to abandon and forget them when certain conditions are satisfied.

A description of a design pattern includes its name, category, a list of suitable applications, definition of cyborg behaviors, including their parameters, dependencies on other behaviors of the cyborg, and the consequences of the design pattern on the swarm's scouting efficiency, information gain rate and tendency to incur the misplacement and the opportunity costs. Cyborg behaviors are described using the Behavior-Data Relations Modeling Language (BDRML). A BDRML-based report consists of the visual and textual representation of cyborg behaviors and data structures used by the behaviors and conditional relations and operations between and on them.

Using BDRML, multiple design patterns can be unambiguously combined into a control strategy by following the design pattern combination rules. The design patterns catalog introduced here provides cyborg designers with knowledge about suitable applications, parameters, and consequences

of various cyborg behaviors. It thus allows them to devise practical cyborg control algorithms based on known mission characteristics.

1.14 Conclusion

Characterization of movements provide remarkable similarities: groups of people seem to remain nearly fixed from their neighbors; they are aligned with their closest neighbor, showing a definite inclination to stay with their neighbors. Indeed, this definition may readily be extended to many other species, such as salmon and songbirds, which travel in groups. The motives for their united activity, however, differ significantly. It is essential to stay with the swarming for solitary bees in the swarm as a single bee cannot live. The intricate structure of the microenvironment induces cohesive locomotion in locusts.