Automated Shape Composition Based on Cell Biology and Distributed Genetic Programming

Linge Bai
lb353@cs.drexel.edu

Manolya Eyiyurekli
me52@cs.drexel.edu

David E. Breen
david@cs.drexel.edu

Department of Computer Science
Drexel University

ABSTRACT

Motivated by the ability of living cells to form specific shapes and structures, we present a computational approach using distributed genetic programming to discover cell-cell interaction rules for automated shape composition. The key concept is to evolve local rules that direct virtual cells to produce a self-organizing behavior that leads to the formation of a macroscopic, user-defined shape. The interactions of the virtual cells, called Morphogenic Primitives (MPs), are based on chemotaxis-driven aggregation behaviors exhibited by actual living cells. Cells emit a chemical into their environment. Each cell responds to the stimulus by moving in the direction of the gradient of the cumulative chemical field detected at its surface. MPs, though, do not attempt to completely mimic the behavior of real cells. The chemical fields are explicitly defined as mathematical functions and are not necessarily physically accurate. The functions are derived via a distributed genetic programming process. A fitness measure, based on the shape that emerges from the chemical-field-driven aggregation, determines which functions will be passed along to later generations. This paper describes the cell interactions of MPs and a distributed genetic programming method to discover the chemical fields needed to produce macroscopic shapes from simple aggregating primitives.

Categories and Subject Descriptors

I.3.5 [**Computing Methodologies**]: Computational Geometry and Object Modeling

General Terms

Algorithms, Experimentation, Performance

Keywords

Shape Composition, Morphogenesis, Chemotaxis, Distributed Genetic Programming, Self-organization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
Copyright 2008 ACM 978-1-60558-130-9/08/07...$5.00.

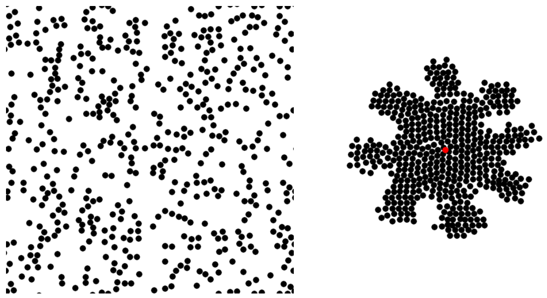


Figure 1: A set of randomly placed Morphogenic Primitives (left) aggregate to form a "gear" (right).

1. INTRODUCTION

In living things, cells aggregate and grow to create complicated structures. This process, called morphogenesis, is one of the fundamental components involved in the development of all complex organisms [17]. One of the essential processes involved in morphogenesis is chemotaxis [6]. Chemotaxis is the phenomenon where cells interact with other cells by emitting a chemical that diffuses into the surrounding environment. Neighboring cells detect the overall chemical concentration at their surfaces and respond to the chemical stimulus by moving either towards or away from the source [9]. The motions induced by chemotaxis may then produce patterns or sortings of cells [12], or even large-scale structures, e.g. cavities or vessels. These phenomena have motivated us to look to developmental biology for concepts that lead to a more organic, cell-biology-inspired approach to shape composition.

In this approach, we discover the local interaction rules that direct the self-organizing primitives, that we call morphogenic primitives (MPs) [1], to aggregate into a particular user-defined shape. Macroscopic shapes are formed automatically by the aggregation of these simple primitives responding only to local information. A collection of MPs are at first randomly placed in the modeling environment. The primitives emit a field, and then respond to the cumulative field by moving along its gradient. A macroscopic, user-defined shape then emerges from the combined actions of the individual primitives.

1.1 Morphogenic Primitives

Several principles were followed when developing morphogenic primitives. 1) MPs are autonomous "agents". Each MP is an independent entity that senses the environment, responds to it, and then modifies the environment and its internal state. There is no "master designer" directing the

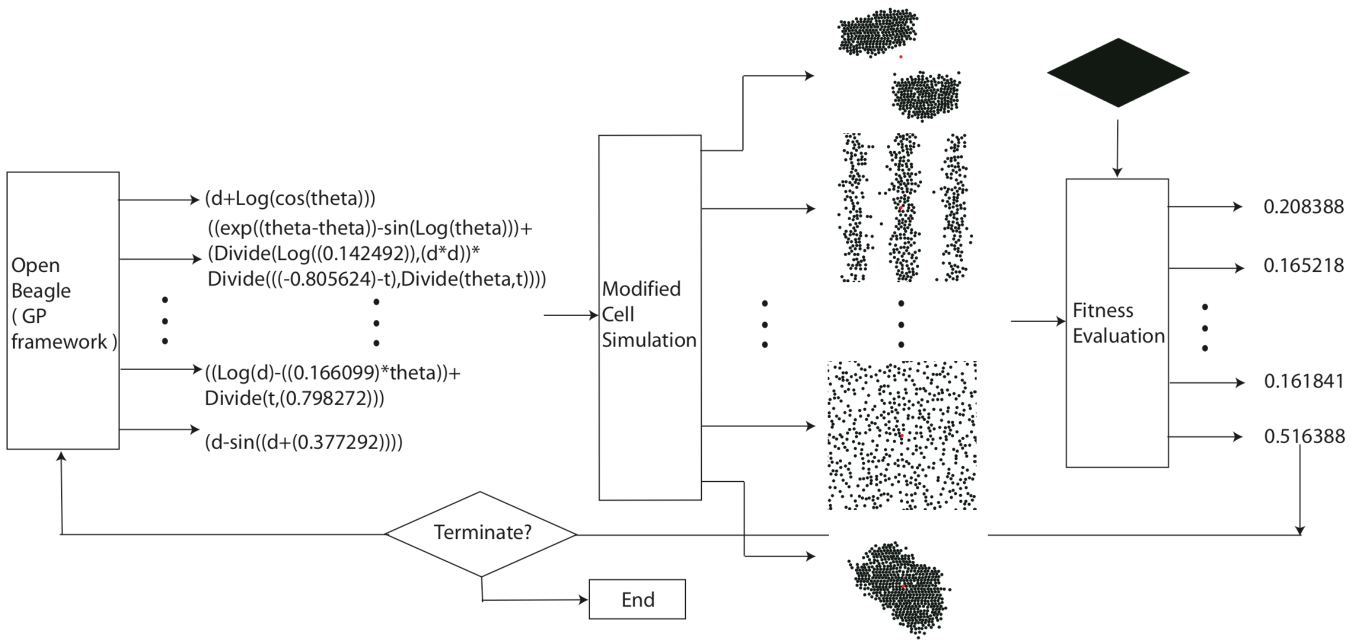

Open Beagle (GP framework)
(d+Log(cos(theta)))
((exp((theta-theta))-sin(Log(theta)))+ (Divide(Log((0.142492)),(d*d))* Divide(((-0.805624)-t),Divide(theta,t))))
((Log(d)-((0.166099)*theta))+ Divide(t,(0.798272)))
(d-sin((d+(0.377292))))
Modified Cell Simulation
Fitness Evaluation
0.208388
0.165218
0.161841
0.516388
Terminate?
End

Figure 2: Genetic programming process that produces functions for morphogenic primitive interactions.

actions/motions of the MPs. 2) Actions are based on local information. Each primitive emits a *finite* "chemical" field that can be sensed only by other primitives within a certain range. The only information received by an MP is gathered at its surface, namely the concentration of the cumulative field and contact with immediate neighboring MPs. 3) MPs respond to information with prescribed behaviors. The actions performed by each MP are the same, but the specifics of the individual actions are based on information received from the environment. 4) MPs have no representation of the final, macroscopic shape to be produced. MPs do not use information about the final shape to determine what actions to take. Their actions are pre-determined by the ultimate shape to produce, but MPs do not carry or access information about the shape. The MP's final global position relative to the shape is not known ahead of time. 5) The shape emerges from the aggregation of local interactions and behaviors. Rather than following a plan to produce the shape, MPs sense, change and respond to the cumulative field concentration. This simple behavior, when combined with somewhat complex chemical fields, will direct the MPs to take individual actions based on local information that will ultimately aggregate to produce a user-defined, macroscopic shape. The main challenge here is to determine which local chemical fields will direct the MPs to ultimately come together into the desired shape.

While MPs' fundamental interaction is based on chemotaxis, we do not limit their behaviors/properties to be physically realistic or completely consistent with biology. Instead, developmental biology provides a motivating starting point for MPs. As a way to customize chemotaxis-driven cells for shape composition, we alter the chemical concentration fields around individual cells. Instead of the chemical concentration dropping off as a function of distance (the physically accurate description), we define the concentration field with a mathematical function of distance, angle and time. Since it is extremely difficult to determine which particular local field function will direct MPs to form a specific macroscopic shape, we employ genetic programming [21] to produce the expressions that explicitly specify the field function. A fitness measure, based on the shape that emerges from the chemical-field-driven aggregation, determines which functions will be passed along to later generations. The genetic process stops once a function in the population provides the desired shape, or after a certain number of function generations have been produced and evaluated.

The evaluation of each potential chemical field function involves a complex and compute-intensive cell aggregation simulation process. Since thousands of functions are generated and evaluated, the GP process that generates the correct MP interactions has extraordinarily high computational requirements. To meet these requirements, we have implemented a distributed GP system that is capable of exploiting the computational power of a Linux cluster.

1.2 Approach Overview

The general approach, which has been implemented within the Open Beagle Framework [16], to defining the field functions that ultimately produce the user-desired shape is presented in Figure 2. We start with a population of functions, which is initially randomly generated. Each function is compiled into a chemotaxis-based cell aggregation simulation program, and defines the chemical field that surrounds the individual cells. A simulation program is executed for each field function on a node in our cluster, usually producing some kind of aggregated structure. The resulting MP configuration is compared to the user-desired shape, and a scalar fitness value is calculated that quantifies how well the computed shape matches the desired shape. A subset of the top candidates are then used to create the next generation of field functions. The process continues until a field function produces the desired shape or the maximum number of generations is reached.

Once the local field function has been identified for a specific shape, MPs may be randomly placed in the computing environment, with each MP surrounded by the GP-produced concentration field. A simulation is performed where the cumulative field is computed, and each MP moves along its gradient, until the MP population reaches an equilibrium, which is the desired shape.

We have utilized this new approach to define morphogenic primitives that aggregate to form a number of user-defined shapes, e.g. an ellipse, a diamond, a boomerang, and an hourglass. In the process of evolving the field functions for user-defined shapes, a few pleasant surprises materialized. Most of them included repeated patterns, e.g. stripes, spots and sine waves, but most interestingly a gear shape emerged from the evolutionary process. See Figures 5 through 12.

2. RELATED WORK

Sims [26] applied genetic programming to create functional representations of intricate, interesting images and solid textures. In this work, an interactive process allows a user to guide the evolution of the image functions by the selection of preferred images and results. The repeated interaction between the user and the evolutionary process leads to the definition of functions for a large number of surprising and appealing images. This approach was extended for the creation of procedural models [27] and the definition of motions and behaviors of virtual creatures [28].

Fleischer explored a cell-based developmental model for self-organizing geometric structures [13, 14]. He applied his cell interaction simulation system to produce an approach to cellular texture generation [15]. Eggenberger Hotz proposed the use of genetic regulatory networks coupled with developmental processes for use in artificial evolution and was able to evolve simple shapes [7, 18]. The combination of artificial evolutionary techniques and developmental processes provides a comprehensible framework for the analysis of evolutionary shape creation. Additional shape generation methods based on grammars and cellular automata are described in [2].

Theraulaz and Bonabeau present a modeling approach based on the swarming behavior of social insects [29, 30], a type of swarm intelligence [3]. They combine swarm techniques with 3D cellular automata to create autonomous agents that indirectly interact in order to create complex 3D structures. This indirect interaction, known as stigmergy [31], allows the agents to act cooperatively, but independently, through a stimulus-response mechanism based on modifications made to the environment. Bonabeau et al. [4] apply genetic algorithms (GA) to the stigmergic swarm-based 3D construction method in order to improve the overall process. A fitness function, chosen by human observers, is assigned to each pattern in this approach, and a GA is used to search the space of all possible patterns.

Given the heavy computation requirements of evolutionary computing and its applications, distributed evolutionary algorithms have been implemented to shorten computation times [24]. Tomassini [32] characterizes the different types of parallel evolutionary algorithms based on the granularity of the data structure and parallel synchronization. Experimentation with and analysis of different distributed genetic programming models have indicated that asynchronous models are more efficient in terms of execution time than other approaches [34]. Software frameworks for distributed genetic programming have also been developed, from tools based on MPI on a Linux cluster [33] to the open-source Java environment DGPF run over large networks [35].

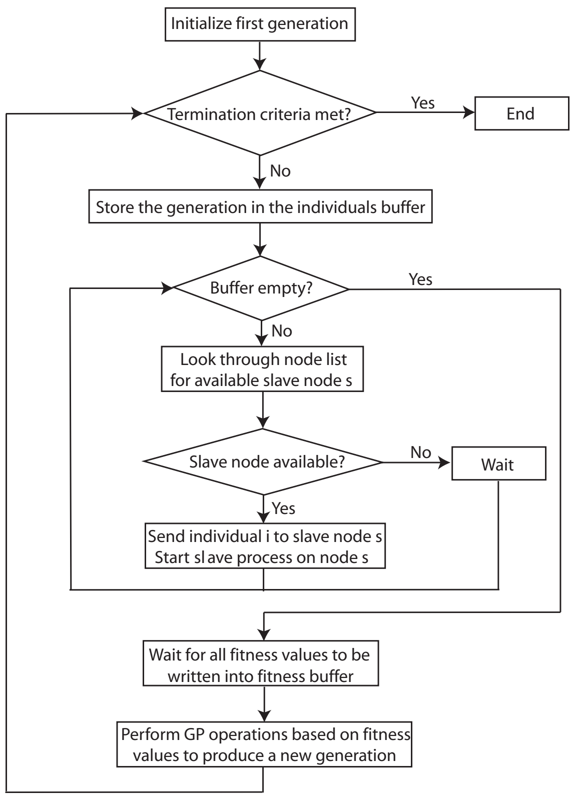
Distributed evolutionary algorithms have been utilized to solve a number of challenging computational problems. Rao and Hansdah have proposed Extended Distributed Genetic Algorithm to generate algorithms for channel routing problems [25]. Koza and Andre have implemented coarse-grained parallel genetic programming with 64 computers as processing nodes and successfully solved the 5-parity problem [22]. Implementation of fine-grained parallel genetic programming has been proposed by Juille and Pollack, in which parallel evaluation of different S-expressions was implemented on a SIMD computer [19]. Messom and Walker have applied distributed genetic programming in evolving cooperative robotic behaviors in robot soccer games [23]. Klein and Spector have proposed a system of distributed genetic programming based on JavaScript and XML, which does not require the installation of client-side software or explicit user participation [20].

Similar to some previous work, we have developed a chemotaxis-based cell aggregation simulation system. However we have extended and modified the system in order to utilize evolutionary computing to discover rules for shape composition. In our work, MPs cooperate with each other only through local interactions. The primitives have no information about the predefined global shape that is being composed. We have also developed a steady-state distributed genetic programming system, based on Unix programming [5], that provides the computational resources needed to perform numerous cell aggregation simulations. Moreover, an automated fitness evaluation process has been implemented that quantifies the similarity between the shape of the aggregated MPs produced by each cell simulation and the user-defined macroscopic shape.

3. CELL AGGREGATION SIMULATION

We have previously developed a computational model and software system that is capable of simulating chemotaxis-based cell aggregation in 2-D [10, 11]. Our 2-D model incorporated fundamental parameters involved in cell-cell aggregation, such as a cell's ability to emit and detect chemoattractant chemicals, cell motility, attachment, proliferation, aggregation and various stages of a cell's life cycle. This model and system provide the conceptual and software foundation for morphogenic primitives.

The full complexity of the original chemotaxis-based cell aggregation model was not needed for the initial implementation of MPs. The model was simplified and we focused the evolutionary process on one aspect of the possible cell interactions, the definition of the chemoattractant chemical field surrounding a single cell. Therefore we only utilize the chemoattractant emission and response portion of the cell simulation system. We also modified the system to allow for a general functional description of chemical fields. The chemical fields are defined as a function of the distance d and the angle θ between two MPs, as well as the MP's age t. Since we define the chemical fields as mathematical expressions, we utilize a number of protected operators, such as protected division and protected logarithm. These protected operators safely handle invalid input values, for example divide by zero and non-positive logarithms. Additionally, the field function is truncated at a fixed distance ($R_{Max} = 200$ units) in order to keep the MP interactions finite and local.



Initialize first generation
Termination criteria met?
Yes
End
No
Store the generation in the individuals buffer
Buffer empty?
Yes
No
Look through node list for available slave node s
Slave node available?
No
Wait
Yes
Send individual i to slave node s Start sl ave process on node s
Wait for all fitness values to be written into fitness buffer
Perform GP operations based on fitness values to produce a new generation

Figure 3: The master process.

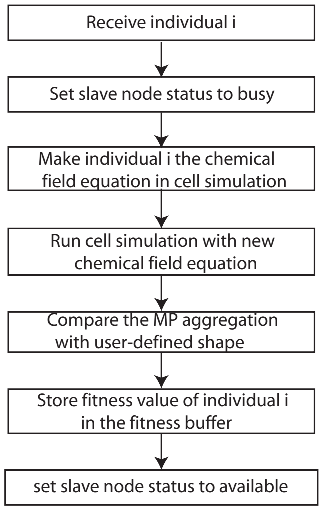
Each MP simulation process begins by randomly placing a number of MPs (500 for our examples) in the computational environment. A morphogenic primitive is represented by a small disk existing in a toroidal 2D environment. The environment is effectively infinite with no boundaries, since the top edge of our computational world is connected to the bottom edge, and the left edge is connected to the right edge. A single aggregation simulation is comprised of a series of time steps. For each time step, each cell performs a prescribed set of actions. Each MP emits a “chemical” into the environment. It then detects the cumulative field at eight receptors on its surface, and calculates the field gradient from this input. The gradient is used to determine the primitive’s velocity. We assume that MPs travel at a terminal velocity through a viscous fluid environment, therefore an MP’s velocity is directly proportional to the chemical field gradient (∇C). When an MP moves in the direction of the chemical gradient, its velocity is calculated as

$$\mathbf{Velocity} = \lambda * \nabla C, \tag{1}$$

where λ (1 for our examples) is a constant that determines the magnitude of a cell’s response to the gradient. At each simulation time step (Δt) the displacement of the MP is

$$\mathbf{\Delta x} = \mathbf{Velocity} * \Delta t. \tag{2}$$

If the displacement makes the MP collide with another MP, a small random step is taken instead. MPs do not always follow the field gradient. 10% of the time MPs take a random small step instead of following the field gradient. This randomness injects a small amount of noise into the system, helping to prevent the set of MPs from collecting into local minima configurations.



Receive individual i
Set slave node status to busy
Make individual i the chemical field equation in cell simulation
Run cell simulation with new chemical field equation
Compare the MP aggregation with user-defined shape
Store fitness value of individual i in the fitness buffer
set slave node status to available

Figure 4: The slave process.

4. DISTRIBUTED GENETIC PROGRAMMING

4.1 Steady-State Master-slave Model

Given the magnitude of the search space, i.e. all possible polynomial and trigonometric functions, we have developed an steady-state, fine-grained, master-slave, distributed computing system that parallelizes the cell simulation and fitness evaluation components of our GP-based shape composition method. We have implemented an N-slave, 1-master model, with the master process adaptively distributing individuals among the slave processes, based on Unix shell scripting [5]. Open Beagle [16], a C++ evolutionary computing framework, has been utilized and altered to function as the distributed genetic programming framework.

The master process, as outlined in Figure 3, adaptively sends out individual functions over a network to a number of slave processes, and performs most of the genetic programming steps. Each slave node hosts one slave process. As outlined in Figure 4, the slave process is responsible for incorporating the individual (chemical field function) into the cell simulation program, running the cell aggregation simulation, comparing the resulting aggregated shape with the user-defined shape and returning a fitness value to the master process. The master process then utilizes the fitness value associated with the individual function to perform genetic programming operations.

4.2 Evolutionary Process

The ramped half-and-half method [8, 21] is used to generate the initial population of functions which consists of GP trees of a predefined maximum depth. The master process first stores the whole generation into a buffer and then adaptively sends all the individuals to slave processes. Once all of the individuals have been distributed to the slave nodes, the master process waits until all of the individuals have been evaluated by the slave nodes. This is indicated when the fitness buffer contains the same number of values as individual functions.

The slave process receives an individual function and modifies the cell aggregation simulation program by setting its chemical field function to the expression. A cell aggregation

Example	Function	Generation	Fitness value
ellipse	Divide((Log(Divide(d,t))-Log(cos(theta))),Divide(d,t))	7th	0.717403
diamond	Divide(((Log(Divide(d,t))-Divide((0.924414),theta))-(0.363563)),Divide(d,t))	11th	0.659776
hourglass	Log(d)+cos(Divide((t*theta),Divide((t+Log(Divide(t,Log(d)))),(-0.356662))))	18th	0.436527
boomerang	Log((d*exp(d)))	13th	0.560553
wave	Divide(Log((Log((sin(t)+cos(theta)))*d)),exp((sin(theta)+(Log(d)+theta))))	25th	0.491477
annulus	exp(exp(theta))	25th	0.435082

Table 1: Summary of shapes, field functions, number of generations and fitness values

simulation is performed for each field function, producing an image of the aggregated shape that emerges from the local interactions defined by the function. Each aggregated result is then evaluated by the fitness function, which compares the aggregate image with a target image. A fitness value (a scalar between 0 and 1) that quantifies how closely the resulting aggregate matches the target shape is assigned to each individual function. The fitness values are stored in the master process's fitness buffer. The slave process pauses until it receives another individual and then the slave process begins again.

Once all the fitness values have been calculated, parent selection is performed on the population based on the fitness values. Variation then takes place using a swap subtree operation and single-point mutation. Generational replacement is used for survivor selection, i.e. the offspring replace the parents from the previous generation. The distributed genetic programming process repeats until the termination criteria are met, either after generating a maximum number of generations or when the fitness of an individual surpasses the threshold, i.e. the associated aggregate is approximately the same shape as the target.

4.3 Fitness function

The fitness function defines the similarity between two shapes, the aggregate generated by the MPs under the influence of the chemical field function and the user-defined target shape. Since the fitness function is based on comparing images, it is necessary to keep the total area of the MPs (the area of one MP times the total number of MPs) approximately equal to the area of the desired shape. Since MPs are not aware of a global coordinate system, we do not want the shape of the resulting aggregate to be tied to a specific fixed orientation. We therefore align the aggregated shape with the target shape before applying the fitness function. This is accomplished by calculating the centroid and major axis of the aggregated and target shapes. A translation and rotation is applied to a candidate MP that aligns its centroid and major axis with those of the target. This allows MPs to aggregate into the desired shape, but not necessarily in the same position and orientation of the target.

Once the alignment is completed, the fitness function calculates the ratio of the overlapping pixels of the aggregate and target shapes to the total pixels of the target shape.

$$f = \frac{\sum p_i, p_i \in S_a \cap S_t}{\sum p_j, p_j \in S_t}, \qquad (3)$$

where f is the fitness of the individual, p_i are the shape pixels that are present in both the aggregate and target image, p_j are the shape pixels in the target image, S_a and S_t are the aggregate and the target shape images. This floating-point number f is assigned as the fitness of the chemical field function used to generate the aggregate shape.

4.4 Parameters

The following is a detailed description of the components in the GP process.

- **Individuals:** Each individual is represented as a prefix tree with a maximum depth $D_{Max} = 17$.
- **Function and terminal set:**
  $F = \{+, -, *, /, exp, log, sin, cos\}$
  $T = \{E, d, t, \theta\}$
  In order to maintain the closure property of the GP process [8, 21], we use protected division and protected logarithm, which allows for the most general description of the functions. Here, $E \in \Re$, d is the distance between two MPs, t is the simulation time and θ is the angle between two MPs measured in the local coordinate system of each MP.
- **Selection:** Parent selection is proportional to the fitness value and uses tournament selection (tournament selection size $k = 7$). Generational replacement is used for survivor selection.
- **Variation:** A swap subtree crossover operation and single-point mutation.
- **Fitness evaluation:** We compare an image of the resulting aggregate with an image of the target shape. The comparison produces a fitness value, a floating-point number between 0 and 1, with 1 being the maximum fitness (similarity) value.
- **population size:** 100.
- **maximum number of generations:** 50.
- **crossover probability:** $P_c = 0.9$.
- **mutation probability:** $P_m = 0.1$.

5. RESULTS AND ANALYSIS

We have employed distributed genetic programming to discover the local interactions that lead to a global emergent behavior for shape composition. More specifically, we have used the evolutionary process to evolve chemical field functions that direct randomly placed MPs to aggregate into a number of user-defined shapes. The MP simulations performed during the GP process consisted of 500 primitives, where each primitive has a radius of 4.5 or 5 (depending on which shape was being composed) units[1]. The MPs move in a toroidal computational environment of 500 × 500 units on a hexagonal grid. Each simulation is run for 1000 time steps. The target shapes given to the MP aggregation framework include an ellipse (Figure 10 (left)), a diamond (Figure 10

[1]One unit of length is equal to a pixel width.

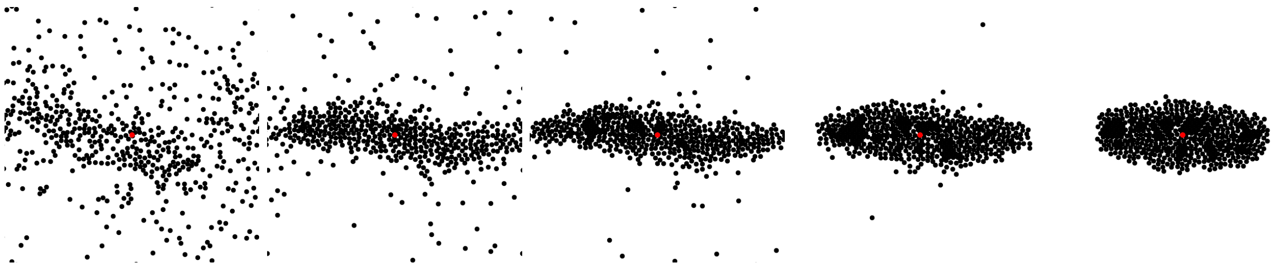


Figure 5: MPs self-organizing into an ellipse.

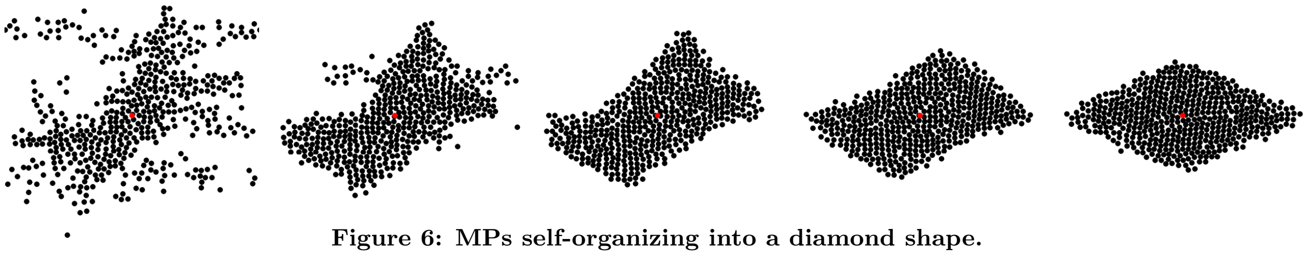


Figure 6: MPs self-organizing into a diamond shape.

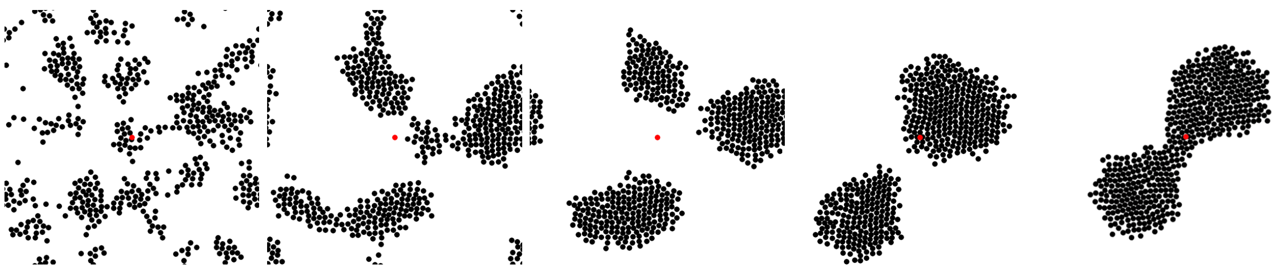


Figure 7: MPs self-organizing into a hourglass shape.

(right middle)), an hourglass (Figure 11 (left)), a boomerang (Figure 11 (right middle)), a wave-like structure (Figure 12 (left)), and a flattened annulus (Figure 12 (right middle)). The best resulting MP-generated shapes are placed next to the target shapes in these figures. The first four shapes were the most successful, creating aggregates that reasonably resemble the target shapes. The field functions that produce these shapes are listed in Table 1, along with the number of generations needed to derive them. The last two examples did not completely converge on the desired shape. The wave and annulus shapes are stand-alone objects. Recall that MPs exist in a toroidal environment. Therefore the shapes that we generated are continuous objects, where their left edge is connected to their right edge. Figures 5 through 8 present snapshots from cell simulations and demonstrate the MP's self-organizing behavior. All of these sequences began with the MPs being randomly placed in the computational environment, similar to Figure 1 (left).

There were a number of pleasant surprises produced during the evolution process. These are field functions that direct MPs to produce unwanted, but still quite interesting, shapes and patterns. A small set of these surprise results are included in Figure 9. The most remarkable of these "unwanted" shapes is the gear shape in Figure 1.

The robustness of our shape composition process was investigated by conducting 100 aggregation simulations each for the ellipse, diamond, hourglass, boomerang and gear shapes. The repeatability for the ellipse was 100% (i.e. every aggregation simulation produced an ellipse-like shape), the diamond was 99%, the hourglass shape was 10%, the boomerang shape was 4% and the gear shape was 34%.

The computation time needed to perform the genetic process that defines the chemical field functions is quite substantial. Each 1000-time-step MP simulation requires approximately 4-CPU minutes on a 1.8 GHz Opteron processor. We perform our GP calculations on an 8-node Linux cluster. Therefore each 100-individual generation requires approximately one hour of actual time to compute on our cluster. Since the overhead of distributing the computation over the nodes is quite small compared to the simulation times themselves, we found that we achieved nearly an $8\times$ speed-up by performing the calculations on an 8-node cluster. Since the GP process usually produces the desired function by the 20th generation, we can derive a shape-specific field function in approximately one day. Of course, once we have the correct function it only requires approximately 4 CPU-minutes to simulate the desired aggregation behavior.

6. CONCLUSION

We have presented a new approach to shape composition using self-organizing primitives whose behaviors are based on cell biology. The key concept is that local interactions between the 2D primitives, called Morphogenic Primitives (MPs), direct them to aggregate into a user-defined macroscopic shape. The interactions are based on the chemotaxis-driven aggregation movements exhibited by actual living cells. Here, cells emit a chemical into their environment. Each cell responds by moving in the direction of the cumu-

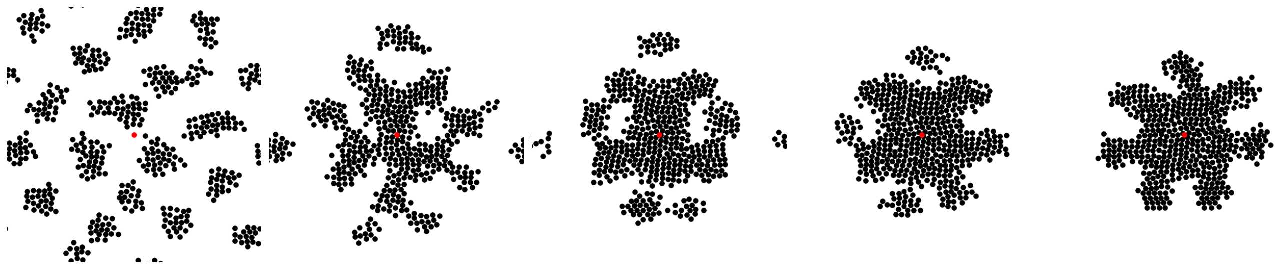


Figure 8: MPs self-organizing into a gear-like shape.

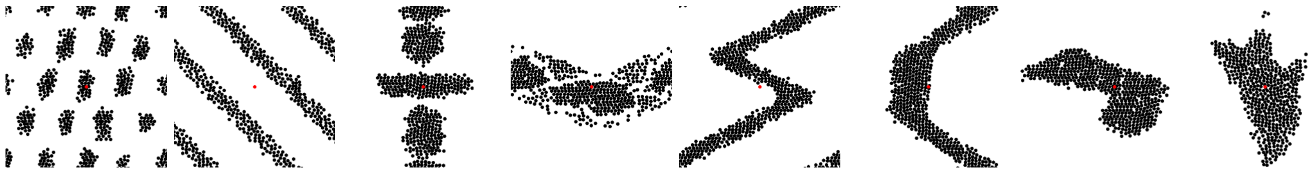


Figure 9: Unexpected patterns and shapes produced during the GP process.

lative chemical field gradient detected at its surface. MPs, though, do not completely mimic the behavior of real cells. The chemical fields are explicitly defined as mathematical functions and are derived via genetic programming. A fitness measure, based on the shape that emerges from the aggregation process, directs the evolution of the function. We have implemented a distributed GP system that parallelizes the cell simulation and fitness evaluation components of our system. We make use of an 8-node Linux cluster to implement an 8-slave, 1-master model, which provides us with nearly an 8× speed-up. We have used the technique to discover chemical field functions that lead to the creation of several 2D shapes including an ellipse, a diamond, an hourglass, a boomerang, and a gear.

7. REFERENCES

[1] L. Bai, M. Eyiyurekli, and D.E. Breen. Self-organizing primitives for automated shape composition. In *Proc. IEEE International Conference on Shape Modeling & Applications*, 2008.

[2] P.J. Bentley and D.W. Corne, editors. *Creative Evolutionary Systems.* Morgan Kaufman, San Francisco, CA, 2002.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, 1999.

[4] E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz. Three-dimensional architectures grown by simple stigmergic agents. *Biosystems*, 56(1):13–32, 2000.

[5] C. Brown. *UNIX Distributed Programming.* Prentice Hall, New York, 1994.

[6] J.A. Davies. *Mechanisms of Morphogenesis.* Elsevier, Amsterdam, 2005.

[7] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proc. 4th European Conference on Artificial Life*, pages 205–213, 1997.

[8] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003.

[9] E. Eisenbach et al. *Chemotaxis.* Imperial College Press, London, 2004.

[10] M. Eyiyurekli. A Computational Model of Chemotaxis-based Cell Aggregation. Master's thesis, Drexel University, 2006.

[11] M. Eyiyurekli, P. Lelkes, and D. Breen. A computational system for investigating chemotaxis-based cell aggregation. In *Proc. European Conference on Artificial Life*, pages 1034–1049, 2007.

[12] M. Eyiyurekli, P. Lelkes, and D. Breen. Simulation of chemotaxis-based sorting of heterotypic cell populations. In *Proc. IEEE / NIH BISTI Life Science Systems & Applications Workshop*, pages 47–50, 2007.

[13] K.W. Fleischer. *A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures.* PhD thesis, California Institute of Technology, 1995.

[14] K.W. Fleischer and A.H. Barr. A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In *Artificial Life III*, pages 389–408. 1994.

[15] K.W. Fleischer, D.H. Laidlaw, B.L. Currin, and A.H. Barr. Cellular texture generation. In *Proc. SIGGRAPH*, pages 239–248, 1995.

[16] C. Gagné and M. Parizeau. Genericity in evolutionary computation software tools: Principles and case-study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, 2006.

[17] S.F. Gilbert. *Developmental Biology.* Sinauer Associates, Inc., Sunderland, MA, 8th edition, 2006.

[18] P.E. Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In Kumar S. and P.J. Bentley, editors, *On Growth, Form and Computers*, pages 302–318. Academic Press, 2003.

[19] H. Juille and J.B. Pollack. Parallel genetic programming and fine-grained SIMD architecture. In *Working Notes for the AAAI Symposium on Genetic Programming*, pages 31–37. AAAI, 1995.

[20] J. Klein and L. Spector. Unwitting distributed genetic programming via asynchronous JavaScript and XML. In *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1628–1635, 2007.

[21] J.R. Koza. *Genetic Programming: On the*

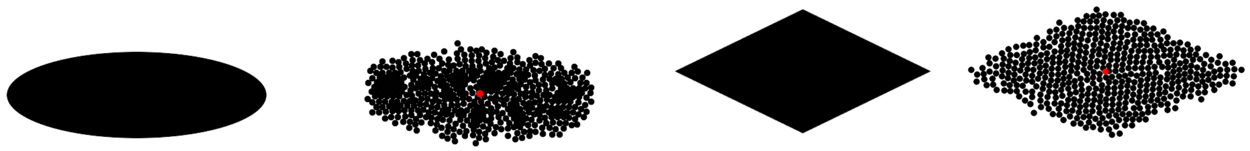


Figure 10: Ellipse: (left) Target shape. (left middle) Self-organized MPs. Diamond: (right middle) Target shape. (right) Self-organized MPs.

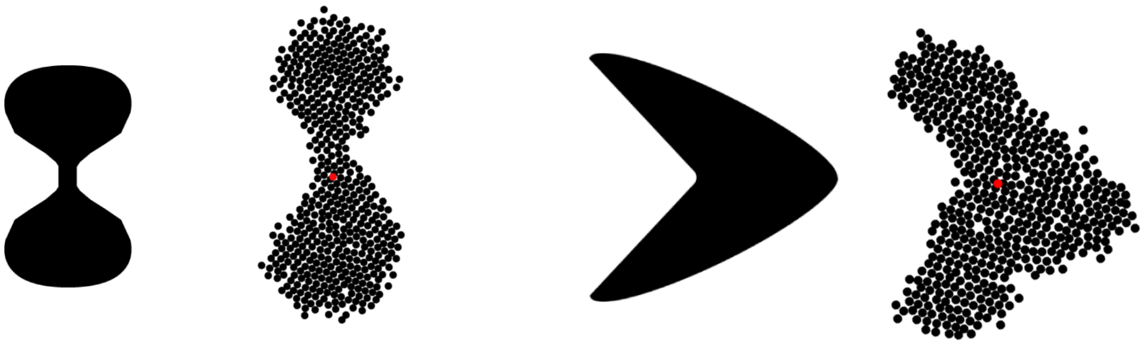


Figure 11: Hourglass: (left) Target shape. (left middle) Self-organized MPs. Boomerang: (right middle) Target shape. (right) Self-organized MPs.

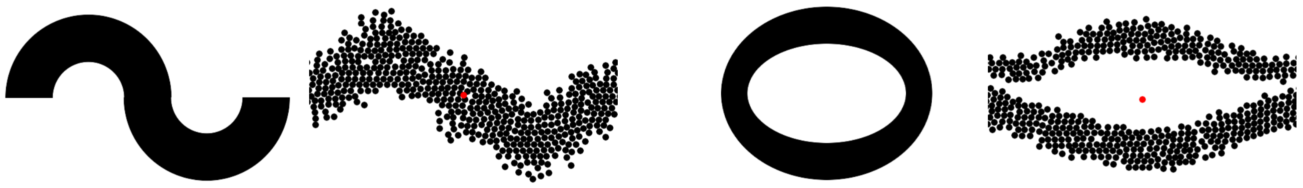


Figure 12: Approximately correct wave and annulus shapes produced by MPs.

Programming of Computers by Means of Natural Selection. MIT Press, 1992.

[22] J.R. Koza and D. Andre. *Parallel Genetic Programming on a Network of Transputers*. Dept. of Computer Science, Stanford University, 1995.

[23] C.H. Messom and G Walker. Evolving cooperative robotic behaviour using distributed genetic programming. In *Proc. 7th Int. Conference on Control, Automation, Robotics and Vision*, volume 1, 2002.

[24] R. Poli, University of Birmingham, and Cognitive Science Research Centre. *Parallel Distributed Genetic Programming.* University of Birmingham, Cognitive Science Research Centre, 1996.

[25] P.B.B. Rao and RC Hansdah. Extended Distributed Genetic Algorithm for Channel Routing. In *Proc. 5th IEEE Symposium on Parallel and Distributed Processing*, pages 726–733, 1993.

[26] K. Sims. Artificial evolution for computer graphics. In *Proc. SIGGRAPH*, pages 319–328, 1991.

[27] K. Sims. Interactive evolution of equations for procedural models. *The Visual Computer*, 9(8):466–476, 1993.

[28] K. Sims. Evolving virtual creatures. In *Proc. SIGGRAPH*, pages 15–22, 1994.

[29] G. Theraulaz and E. Bonabeau. Coordination in distributed building. *Nature*, 269:686–688, 1995.

[30] G. Theraulaz and E. Bonabeau. Modeling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 177:381–400, 1995.

[31] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5:97–116, 1999.

[32] M. Tomassini. Parallel and distributed evolutionary algorithms: A review. In *Evolutionary Algorithms in Engineering and Computer Science*, pages 113–133. John Wiley & Sons, Chichester, 1999.

[33] M. Tomassini, F. Fernández, L. Vanneschi, and L. Bucher. An MPI-Based Tool for Distributed Genetic Programming. In *Proc. IEEE Int. Conference on Cluster Computing*, pages 209–216, 2000.

[34] M. Tomassini, L. Vanneschi, F. Fernandez, and G. Galeano. Experimental investigation of three distributed genetic programming models. In *Proc. Parallel Problem Solving from Nature-PPSN VII*, volume 2439, pages 641–650. Springer, 2002.

[35] T. Weise and K. Geihs. DGPF-an adaptable framework for distributed multi-objective search algorithms applied to the genetic programming of sensor networks. In *Proc. 2nd Int. Conference on Bioinspired Optimization Methods and their Application*, pages 157–166, 2006.