ELSEVIER

BioSystems 56 (2000) 13–32

BioSystems

www.elsevier.com/locate/biosystems

Three-dimensional architectures grown by simple 'stigmergic' agents

Eric Bonabeau [a,*], Sylvain Guérin [b], Dominique Snyers [b], Pascale Kuntz [b], Guy Theraulaz [c]

[a] *Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA*
[b] *Ecole Nationale Supérieure des Télécommunications de Bretagne, BP 832, 29285 Brest Cédex, France*
[c] *Laboratoire d'Ethologie et de Psychologie Animale, CNRS-UMR 5550, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cédex, France*

Received 21 December 1998; received in revised form 21 December 1999; accepted 4 January 2000

Abstract

A simple model of multi-agent three-dimensional construction is presented. The properties of this model are investigated. Based on these properties, a fitness function is defined to characterize the structured patterns that can be generated by the model. The fitness function assigns a value to each pattern. The choice of the fitness function is validated by the fact that human observers tend to view patterns with high (resp. low) fitness as structured (resp. unstructured). A genetic algorithm based on this fitness function is used to explore the space of possible patterns. The genetic algorithm is able to make use of sub-modules of existing patterns and recombine them to produce novel patterns, but strong epistatic interactions among genes make the fitness landscape rugged and prevent more complex patterns from being produced.

Keywords: Collective behavior; Social insects; Nest construction; Self-assembly; Genetic algorithm

1. Nest bulding in social insects

1.1. Introduction

Numerous animal species produce complex architectures, which fulfill numerous functional and adaptive requirements, including protection from predators, facilitation of social life and reproductive activities, thermal regulation, etc (Hansell, 1984). Among them, social insects are capable of generating extremely intricate patterns (Hansell, 1984; Jeanne, 1975; Wilson, 1971). The production of these patterns certainly requires some form of coordination or integration of individual activities. Although this question is of great interest, there have been few model-based attempts at explaining the nature of this coordination (Gallais-Hamonno and Chauvin, 1972; Deneubourg, 1977; Skarka et al., 1990; Deneubourg et al., 1992; Franks et al., 1992; Karsai and Penzes, 1993).

* Corresponding author.
E-mail address: bonabeau@santafe.edu (E. Bonabeau)

0303-2647/00/$ - see front matter
PII: S0303-2647(00)00067-8

1.2. Stigmergy

A possible organizational principle that could be the underlying mechanism of coordination is stigmergy, which was introduced by Grassé (1959, 1984) to describe the indirect communications taking place among individuals through the dynamically changing shape in construction. Stigmergy was originally defined by Grassé in his studies on the reconstruction of the termite nest of *Bellicositermes natalensis* (Grassé, 1959). In order to explain the coordination of individuals' tasks, Grassé showed that the regulation of the building activity does not depend on direct interactions among workers but on the nest structure:

1. A stimulating configuration triggers a response from a termite worker;
2. this response transforms the current configuration into another configuration,
3. which may trigger in turn another, possibly different, action performed by the same termite or any other worker in the colony.

Although this principle is attractive, it has remained unused for a long time because Grassé had not shown how this mechanism could canalize building and lead to specific structures.

Gallais-Hamonno and Chauvin (1972) and Deneubourg (1977) were the first to show how stigmergy could be made operational. Gallais-Hamonno and Chauvin (1972) developed a computer model of mound building in ants. Deneubourg (1977) proposed a more formal model for the emergence of pillars in termites, in which chemical cues organize a significant portion of the building activity. A termite picks up a soil pellet, impregnates it with an oral secretion which diffuses away from the pellet once the pellet is deposited; this diffusing substance, the cement pheromone, attracts other termites and stimulates them into depositing more pellets in the vicinity of the initial pellet. The existence of an initial deposit of soil pellets stimulates workers to accumulate more material through a positive feedback mechanism, since the accumulation of material reinforces the attractiveness of deposits through the diffusing pheromone emitted by the pellets. In this process, communication between termites relies on pellet deposits and the resulting pheromonal field. Building on these seminal contributions, other researchers have presented models, primarily based on stigmergy, to explain the coordination, regulation and integration of individual activities (Skarka et al., 1990; Karsai and Penzes, 1993; Theraulaz and Bonabeau, 1995a,b).

1.3. Evolutionary exploration of building mechanisms

The focus of this paper is the simple model of construction of Theraulaz and Bonabeau (1995a,b) (thereafter: TB), described in Section 2, which we explore by means of a genetic algorithm. Beyond the insight that such an exploration gives us about the properties of the model, and possibly of building algorithms in insects, the process of designing a fitness function for the genetic algorithm forces us to formalize the very notion of structure: what is a structured pattern? What is so specific about the coordination of building activities in social insects that allows them to produce structured patterns?

One could argue that function, rather than structure, is the relevant property of social insect architectures. However, the level of functionality or the adaptive value of a model architecture is particularly difficult to test. Krink and Vollrath (1997), in a remarkable work, have been able to do just that with their model of web construction behavior in orb web-building spiders: they measured the fitness associated with a given artificial web by computing costs and benefits. Construction time and the amount of silk used in the web, subdivided into sticky and non-sticky silk, make up the cost function. Benefits were determined by measuring the response of the web to artificial prey items (total amount of prey caught, weighted by the impact position, size and value of each single item). Their model lends itself well to this type of fitness evaluation. TB's (1995a, 1995b) model is too abstract to allow functional features to be examined directly. In addition, it is difficult to define, for example, what the anti-predator function exactly is, or how thermal regulation is achieved, all the more as ethologists themselves often do not know with certainty how such functions are actually implemented in insect colonies.

Therefore, the only tractable way of exploring the properties of this model is through quantifying the (subjective) notion of structure. Focusing on structure rather than function also extends the scope of the model to morphogenesis and multi-agent-based pattern formation in general.

2. Model

2.1. Model

Although TB's model (1995a, 1995b) was originally inspired by the building behavior of wasps, it can be viewed as a general model of multi-agent building motivated by the following question: what is the simplest possible agent-based model that can generate complex patterns similar to those observed in nature? A very simple model consists of automata that perform random walks in a three-dimensional discrete space and take actions asynchronously on a purely local stimulus-response basis. The deposit of an elementary building block, thereafter called a brick, by an agent depends on the local configuration of bricks in the cells surrounding the cell occupied by the agent. Two types of bricks can be deposited: our experience is that no 'interesting' pattern can be generated using a single type of brick. No brick can be removed once it has been deposited. All simulations start with a single initial brick. We call micro-rule the association of a stimulating configuration with a brick to be deposited, and we call algorithm any collection of compatible micro-rules. Two micro-rules are not compatible if they correspond to the same stimulating configuration but lead to the deposition of different bricks. An algorithm is characterized by its micro-rule table, which is basically a lookup table.

A random exploration of the space of such algorithms yields no interesting result, only random or space-filling patterns. It is, however, possible to produce complex patterns, some of which are strikingly similar to those observed in nature (Bonabeau et al., 1994), by working 'backwards' from some patterns to be generated to their corresponding algorithms. TB (1995a, 1995b) found a posteriori that structured shapes can be built only with special algorithms, coordinated algorithms, characterized by emergent coordination: stimulating configurations corresponding to different building stages must not overlap, so as to avoid the disorganization of the building activity. This feature creates implicit handshakes and interlocks at every stage, that is, constraints upon which structures can develop consistently. This approach, which is described at length in (Bonabeau et al., 1994; TB, 1995a, 1995b), therefore shows how the nest itself can provide the constraints that canalize a stigmergic building behavior into producing specific shapes. However, the important notions of structure and building stage were never made clear: they will be clarified when we define the fitness function in Section 3.

2.2. Patterns produced

In Fig. 1, all architectures, except 1g and 1h, have been obtained with coordinated algorithms. The difference between (1g, 1h) and all other architectures is striking, though hard to formalize. One given coordinated algorithm always converges towards architectures that possess similar features. On the other hand, algorithms that produce unstructured patterns may sometimes diverge: the same algorithm leads to different global architectures in different simulations. As an illustration of this fact, architectures 1d and 1e result from two successive simulations using the same coordinated algorithm, and architectures 1g and 1h result from the same non-coordinated algorithm. We see that there is a high degree of structural similarity between 1e and 1d, in sharp contrast with 1g and 1h. This tendency to diverge results from the fact that stimulating configurations are not organized in time and space and many of them overlap, so that the architecture grows in space without any coherence. It is not true, however, that every unstructured architecture is generated with a non-convergent algorithm: some of them can be produced consistently in all simulations. Moreover, even in architectures built with coordinated algorithms, there may be some degree of variation, which is higher in cases where the number of different choices within one given building state is large.

For example, we see that 1d and 1e are similar but not perfectly identical: this is because there are several possible options at many different steps in the building process, or, in other words, the process is not fully constrained. Architectures 1k, 1l, 1m, and 1n provide examples of fully constrained architectures: two successive runs produce exactly the same shapes.

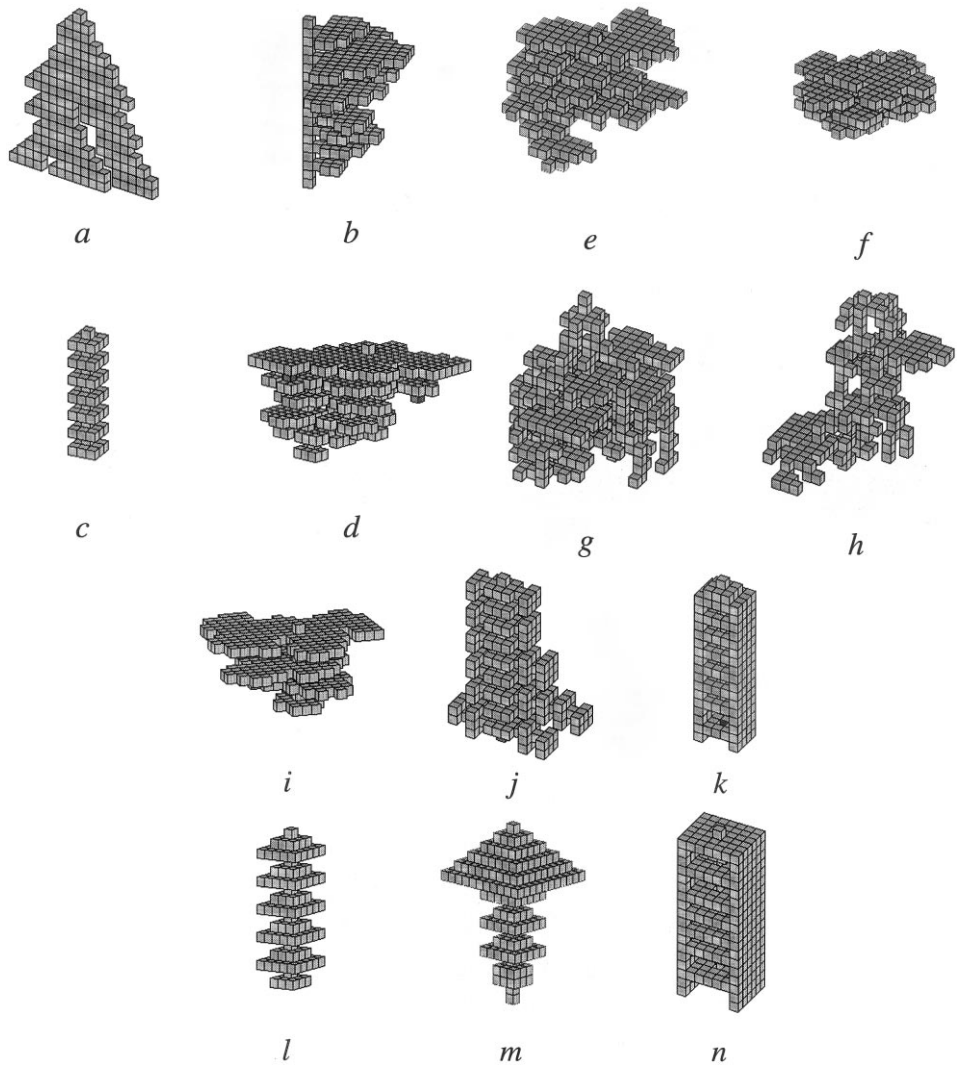


Fig. 1. Patterns produced (cubic bricks). Simulations of collective building on a 3D cubic lattice. Simulations were made on a $20 \times 20 \times 20$ lattice with 10 wasps. For those architectures which are reminiscent of natural wasp nests, we give the name of the genera exhibiting a similar design. (a) Nest-like architecture (*Parapolybia*) obtained after 20 000 steps; (b) nest-like architecture (*Parachartergus*) obtained after 20 000 steps; (c) architecture obtained after 20 000 steps; (d) nest-like Architecture (*Stelopolybia*) obtained after 20 000 steps; (e) nest-like Architecture (*Stelopolybia*) obtained after 20 000 steps; (f) nest-like architecture (*Stelopolybia*) obtained after 20 000 steps; (g) architecture obtained after 15 000 steps using a non-coordinated algorithm; (h) architecture obtained after 15 000 steps using the same non-coordinated algorithm as in (g); (i) nest-like architecture (*Vespa*) obtained after 20 000 steps; (j) architecture obtained after 80 000 steps; (k) nest-like architecture (*Chatergus*) obtained after 185 000 steps; (l) architecture obtained after 125 000 steps; (m) architecture obtained after 85 000 steps; (n) nest-like architecture (*Chatergus*) obtained after 100 000 steps.

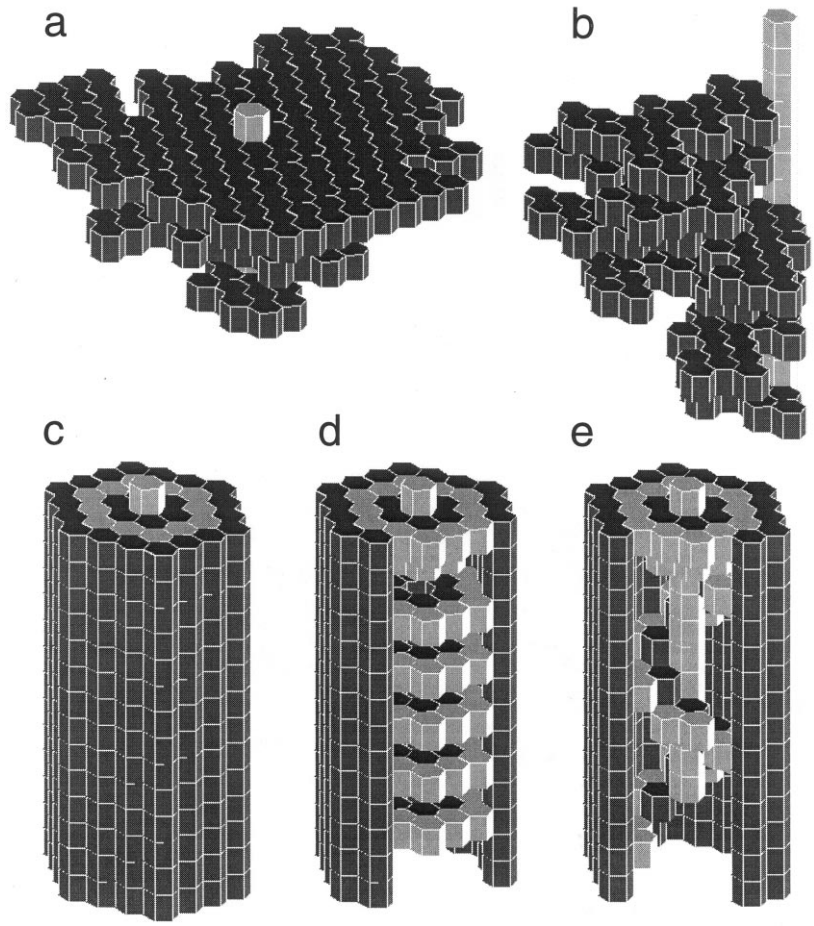


Fig. 2. Patterns produced (hexagonal bricks) Simulations of collective building on a 3D hexagonal lattice. Simulations were made on a 202020 lattice with 10 wasps. Some architectures are reminiscent of natural wasp nests and exhibit a similar design. (a) Nest-like architecture (*Vespa*) obtained after 20 000 steps; (b) nest-like architecture (*Parachartergus*) obtained after 20 000 steps; (c, d) nest-like architecture (*Chatergus*) obtained after 100 000 steps. A portion of the front enveloppe has been cut away in (d); (e) Lattice architecture including an external enveloppe and a long-range internal helix. A portion of the front envelope has been cut away.

Simulations were also performed with bricks that are hexagonal in the $x-y$ plane. In other words, the horizontal plane is a triangular lattice. Fig. 2 presents a few of the 'interesting' architectures produced with this particular topology.

2.3. Analysis

TB (1995a,b) showed:

(1) The smoothness, or more precisely the consistency, of the mapping from algorithms to architectures in the structured subspace: two close structured architectures appear to be generated by two close coordinated algorithms. This result extends to unstructured patterns generated by coordinated algorithms. That two close algorithms generate two close architectures could not be fully tested but is certainly not generally true: for example, removing a micro-rule from a coordinated algorithm's micro-rule table may result in a disorganization of the building process. Moreover, outside the subspace of coordinated algorithms, one given algorithm may produce quite different patterns in different runs, so that comparing non-co-

ordinated algorithms among themselves is irrelevant. Therefore, we expect the algorithmic landscape to be generally rugged but somewhat smoother in the coordinated subspace.

(2) The compactness of the subspace of structured shapes (that is, a factorial correspondence analysis (Benzecri, 1973; Lefebvre, 1980) showed that not only are such shapes rare, but they also all lie in the same, small region). This property suggests that the range of patterns that can be produced with this family of algorithms is very limited.

In order to understand what a coordinated algorithm is, let us take an example, depicted in Fig. 3. This figure represents the successive steps in the construction of a nest which looks like nests built by *Epipona* wasps. The transition between two successive building steps is shown to depend on a given number of local configurations that stimulate the deposit of a brick. Once all the bricks in the current step have been deposited, the building process goes to the next step. Steps 1–5 correspond to the enlargement of the top of the

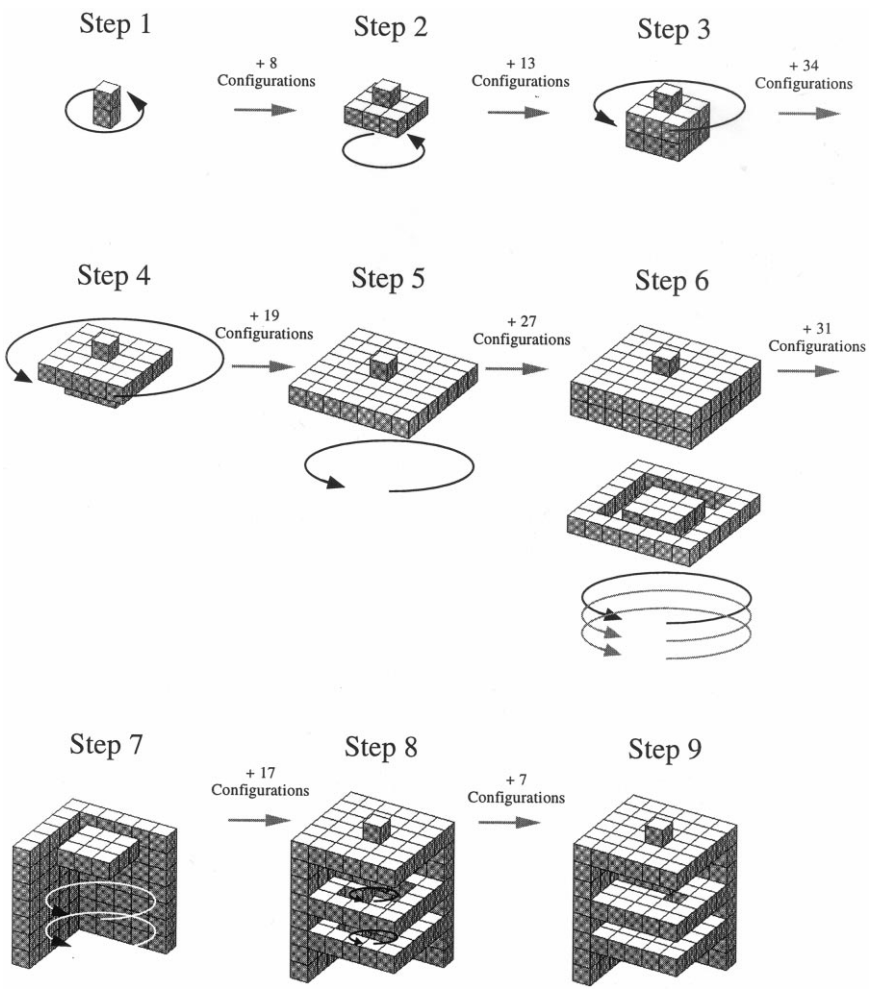


Fig. 3. Building stages and modularity. Successive building steps in the construction of an *Epipona* nest with a lattice swarm. The completion of each step gives rise to stimulating configurations belonging to the next step. All stimulating configurations are organized so as to ensure a regular building process. In particular, within a given step, the stimulating configurations do not have to be spatially connected and can occur simultaneously at different locations. In steps 7–9, the front and right portions of the external envelope have been cut away.

nest, including the first comb of cells (step 3). Step 6 represents the construction and lengthening of the external envelope, from which parallel combs will be built (steps 7 and 8). These steps determine the distinction between this nest, where the entrance and access holes at the different levels lie in the periphery of the comb, from the *Chartergus* nest represented in Fig. 1k and m, where this hole lies in the middle of the nest.

Fig. 3 also illustrates the concept of modularity. Recurrent states may appear, inducing periodicity in the group's behavior. This cyclical behavior results in a modular pattern, where each module is built during a cycle. All modules are qualitatively similar.

3. Genetic algorithm

3.1. Introduction

Despite the analysis summarized in Section 2.3, our understanding of how 'structured' patterns are produced remains intuitive and incomplete. In an attempt to better characterize those algorithms that produce 'structure', we have been looking for ways of exploring the enormous space of possible patterns grown by artificial agents on a lattice and capable of depositing two types of bricks according to the local state of their environment.

Using a genetic algorithm (GA) sounds like a good idea, except that it requires prior knowledge of how to characterize structure in order to define a fitness function! Therefore we used the GA as a way of experimenting with different fitness functions. In the rest of Section 3, we describe only the most 'successful' fitness function resulting from a trial-and-error process. Let us emphasize that the fitness function is based on a set of heuristic criteria defined ad hoc for the sole purpose of understanding structure-producing mechanisms. Our goal is to characterize, with the model described in this paper, what makes an algorithm produce a structured pattern.

In order to check the relevance of the fitness function, it was computed on previously obtained patterns, as well as on patterns discovered by the GA itself, and then tested against ratings by an ensemble of human observers. Although this 'relevance' check is far from perfect as it relies on subjective criteria, it is consistent with the fact that 'structure' is in the eye of the beholder. Within this context, a successful fitness function is one that exhibits a strong positive correlation with the observers' ratings.

3.2. Definition of the fitness function

3.2.1. Observations

The fitness function is based on several observations:

(1) In algorithms that generate coherent architectures, many micro-rules are used, whereas in algorithms that generate unstructured patterns, only one micro-rule or a few micro-rules are actually used in the course of the simulation. A given algorithm is defined by its micro-rule table, which assigns an action to be taken to each configuration of bricks. Some of the micro-rules may never be used, because the configurations of bricks they represent never occur. Only algorithms many micro-rules of which are actually applied in the course of the simulation can produce coherent architectures, because such architectures require a subtle and complex sequence of correlated deposits. Obviously, this condition is necessary but not sufficient, but a population of algorithms containing a large proportion of algorithms that satisfy this criterion is certainly a much better point to start from.

In relation to this observation, it is also interesting to mention that it usually takes more time to generate a complex architecture because of the constraints that are generated in the course of construction. Each constraint acts as a bottleneck preventing any progress from being made until the constraint is satisfied. This result relates to the notion of logical depth, which is defined as the time is takes for a program to generate a given pattern (Bennett, 1988). Interesting architectures are logically deep.

(2) Most structured architectures are compact in the following sense: bricks have adjacent faces with many of their neighbors. Building compact patterns requires collections of complementary, correlated micro-rules.

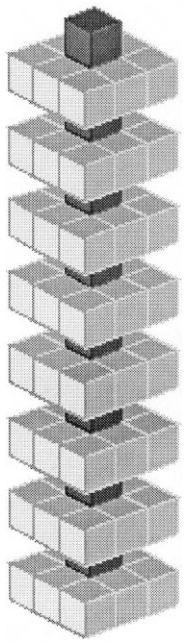


Fig. 4. Example architecture. Simple structured architecture.

(3) Structured architectures are most often characterized by modular patterns, that is, patterns that repeat themselves (see Figs. 1 and 2). This modularity requires coordination. More complex patterns are characterized by more complex and larger modules. One has therefore to look for (1) large patterns that (2) repeat themselves.

3.2.2. The construction graph

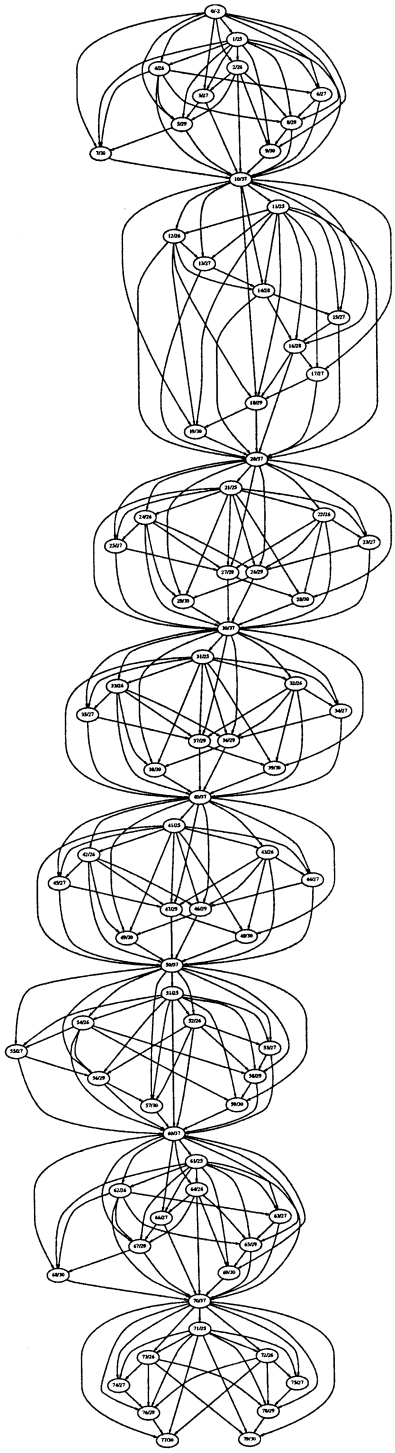
In order to take advantage of these three observations, a graph is constructed for each pattern, in such a way that it is possible to reconstruct the pattern from the graph, but not the graph from the sole pattern. The whole dynamics of the building process is contained in the graph. A fitness function can then be defined directly on the graph. The graph is constructed as follows:

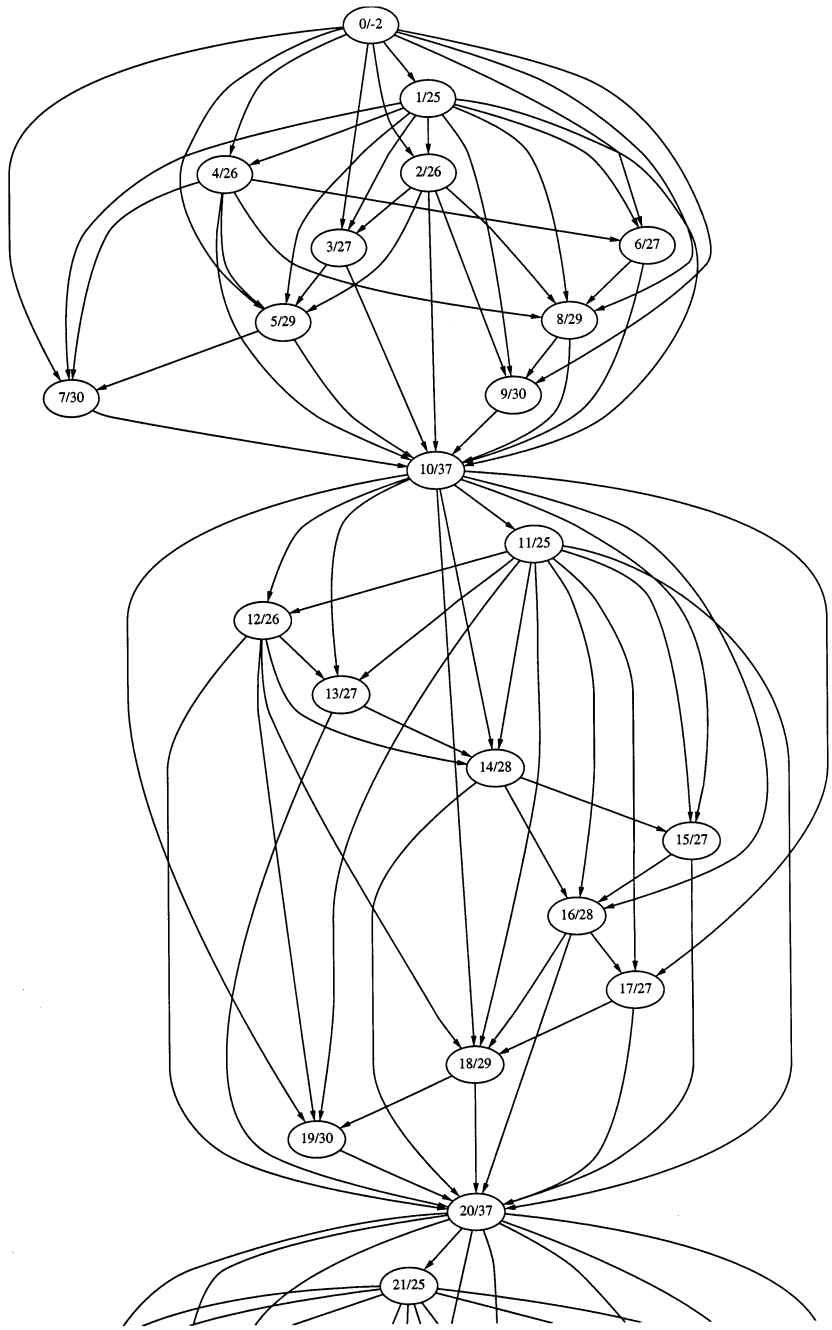
(1) The graph is initialized with one node that corresponds to the initial brick. (2) Each time a brick is deposited, a node is added to the graph. (3) The new node receives incoming connections (edges) from the nodes that represent bricks previously deposited in the neighborhood of the newly deposited brick. Such neighboring bricks obviously had an influence on the deposition of the new brick, since the micro-rule that has been activated for this new deposition to occur contains the neighboring bricks of the new brick. (4) Another way of viewing point (3) is to consider that outgoing connections to the new node are added to nodes which represent bricks that are neighbors of the newly deposited brick. (5) The new node is assigned two numbers: (1) its number in the sequence of brick deposits (for example, number n if the node corresponds to the nth deposited brick), denoted by X; and (2) the number of the micro-rule that has been activated to deposit the brick, denoted by Y.

It is useful at this point to give a concrete example of how the graph is created. In this perspective, we found it convenient to use a relatively simple structured architecture, which is represented in Fig. 4. The micro-rule table contains 59 micro-rules, not all of which are used in every simulation. The architecture (Fig. 4) looks like a tower, with each floor a simple module attached to the module below it by a single brick.

This structure is reflected in the graph shown in Fig. 5, where eight clusters of nodes, which correspond to the eight floors of the architecture, can be clearly observed. The first node is located in the uppermost part of the graph, and is labeled $X/Y = 0/-2$ by convention: the first node corresponds to the first brick, which is always present at the beginning of the simulation. Other node labels represent the order in which the corresponding bricks have been deposited and the micro-rules that have been activated to deposit

Fig. 5. Construction graph. (a) Graph associated with architecture of Fig. 4. The graph is constructed as follows: (1) The graph is initialized with one node that corresponds to the initial brick; (2) each time a brick is deposited, a node is added to the graph; (3) the new node receives incoming connections (edges) from the nodes that represent bricks previously deposited in the neighborhood of the newly deposited brick; (4) The new node is assigned two numbers: (i) its number in the sequence of brick deposits (for example, number n if the node corresponds to the nth deposited brick), denoted by X, and (ii) the number of the micro-rule that has been activated to deposit the brick, denoted by Y. (b) Zoom on the first two building stages (= floors).




0/-2
1/25
4/26
2/26
3/27
6/27
5/29
8/29
7/30
9/30
10/37
11/25
12/26
13/27
14/28
15/27
16/28
17/27
18/29
19/30
20/37
21/25

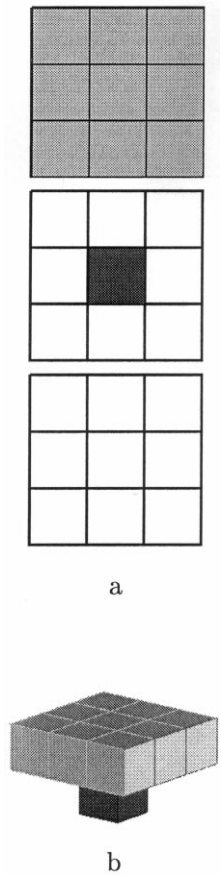


Fig. 6. Bottleneck micro-rule. (a) Representation of rule 37 in the rule table used to generate the architecture of Fig. 9. This figure tells that micro-rule 37 consists of depositing a type I brick (in the center cell of the intermediate level, which is where the agent is located) when the nine cells above are filled with type II bricks; and (b) local configuration of bricks after rule 37 has been triggered.

them: for example, the node labeled $X/Y = 10/37$ corresponds to the 10th deposited brick, which has been deposited after activation of micro-rule 37. The two-dimensional display of the graph has been obtained with a simple graph drawing algorithm. The graph contains 80 nodes, eight floors of nine bricks each plus one brick connecting two successive floors, and 295 edges. Two successive clusters of nodes are separated by an isolated node, which receives all incoming connections from the previous cluster and emits all outgoing connections the next cluster. The label of isolated nodes tells us that micro-rule 37 is used to produce the links between successive clusters. We see that bricks 10, 20, 30, 40, 50, 60 and 70 have all been deposited after activation of micro-rule 37.

Fig. 6 shows micro-rule 37: this rule constitutes a bottleneck to go from one floor to the next, because it is the only one that allows to proceed to the next floor, and it can be applied only when the previous floor has been completed. The notion of a building stage can also be more clearly understood with this graph: each 'floor' corresponds to a building stage. More formally, but still not unambiguously, one can define a building stage as a cluster of nodes in the graph

3.2.3. Definition of the fitness function

From the graph, a fitness function F is computed, which is the product of four components:

$$F = k \times F_1^{1/3} \times F_2 \times F_3 \times F_4^{1/3}, \quad (1)$$

where F_1 is fraction of micro-rules actually used in the course of a simulation, F_2 is the compacity of the architecture (measured here by the ratio (# of egdes)/(# of nodes), which is the average connectivity per node), F_3 is proportional to the mean size $\bar{s}$ ($F_3 = 0.2\bar{s}$) of the patterns detected in the graph, F_4 results from the evaluation of a pattern matching scheme, and k is a multiplicative constant.

In order to detect patterns in the graph (that correspond to actual patterns in the architecture), the following procedure is used: one starts from a node in the graph and construct a subgraph by adding nodes and edges until (1) the same micro-rule as the micro-rule associated with the starting node is applied again, or (2) exploration depth exceeds some threshold S (here, $S = 4$, from root to leaf). Once all patterns, up to a maximum depth S, have been determined, a pattern matching procedure is applied as follows: 100 pairs of detected patterns are randomly selected. Patterns in a pair are compared, taking account of translations along x, y and z, and rotations around the z-axis. Let n_1 and n_2 be the respective sizes of patterns 1 and 2, d_1 the number of cells which are filled with a brick (irrespective of brick type) in the second pattern and not in the first pattern, d_2

the number of cells which are filled with a brick (irrespective of brick type) in the first pattern and not in the second pattern, and d_3 the number of cells which are filled with a brick in both patterns but not with the same brick type. $d_1 + d_2 + d_3 = N_m$ is the number of sites where there is a mismatch between patterns 1 and 2. The smallest value of N_m obtained over all translations and rotations of the patterns is selected. The correlation x_i between the two patterns of the ith pair is defined by

$$x_i = 1 - (N_m^i/(n_1^i + n_2^i)), \quad (2)$$

where N_m^i, n_1^i and n_2^i are the respective values of N_m, n_1 and n_2 for the ith pair. F_4 is then given by

$$F_4 = 0.01 \sum_{i=1}^{100} x_i^{100/(n_1^i + n_2^i)}, \quad (3)$$

where the $(n_1^i + n_2^i)$ term allows to favor patterns with a lot of bricks, characterized by relatively large values of $(n_1^i + n_2^i)$, because $x_i \in [0, 1]$. Whereas F_3 measures the complexity of the patterns present in the architecture, F_4 is more a measure of the modularity of the architecture, although it also depends on how complex the modules are: it measures complex modularity because it is large when large patterns are detected many times in a given architecture. Fig. 7 shows examples of patterns detected in the graph associated with the architecture of Fig. 4. The fitness components of this architecture are: $F_1 = 0.1186$, $F_2 = 3.6875$, $F_3 = 0.3800$, $F_4 = 0.725346$, and $F = 0.271$.

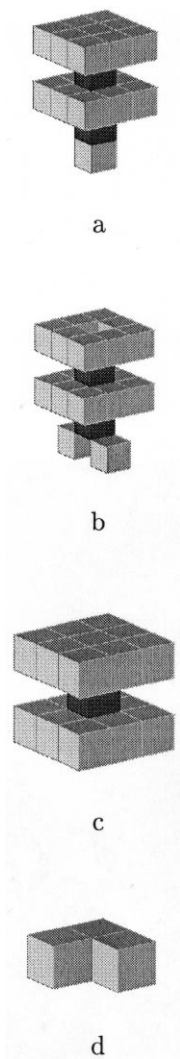


Fig. 7. Patterns detected. Four examples of patterns found in the architecture of Fig. 4. The procedure to detect these patterns is described in Section 3.2.3.

3.3. Encoding, initialization, mutation, crossover and learning

3.3.1. Genetic algorithm

Each generation is composed of 80 individuals. A straightforward fitness-proportionate selection GA is used: the probability for an individual to be selected into the reproductive pool is proportional to its fitness (roulette-wheel sampling (Goldberg, 1989)). New individuals are generated until a new generation of 80 individuals is formed.

3.3.2. Coding

Algorithms are encoded with a variable-length genotype. A gene is the equivalent of a micro-rule. Let us remind that a micro-rule is the association of a stimulating configuration with the non-empty action to be taken (that is, the brick to be deposited). An algorithm that has n micro-rules in its rule table is encoded into n genes, which are elementary units for crossover. Fig. 8a illustrates this encoding.

3.3.3. Initialization and the introduction of new building algorithms

The number of micro-rules of a newly generated algorithm in the initial population is a random variable. Its probability distribution is Gaussian with mean $m = 50$ and standard deviation $\sigma = 10$. The choice of micro-rules in the initial population, and when new random micro-rules are injected into subsequent generations, must be biased to allow construction to start:

(1) Stimulating configurations containing a large number of bricks must be avoided because they may never be encountered in the course of one simulation. We implement this bias by using probabilistic templates that, in addition to probabilistically ensuring that all directions are evenly covered, implement a rapidly decaying probability distribution for the number of bricks in stimulating configurations. A template simply defines the probability of putting a brick in all of the 26 cells that make a stimulating configuration. We use about 10 templates: for each stimulating configuration to be generated, one template is randomly selected.

(2) Simulations start with one and only one brick in space, so that there must exist one micro-rule the stimulating configuration of which contains only one brick.

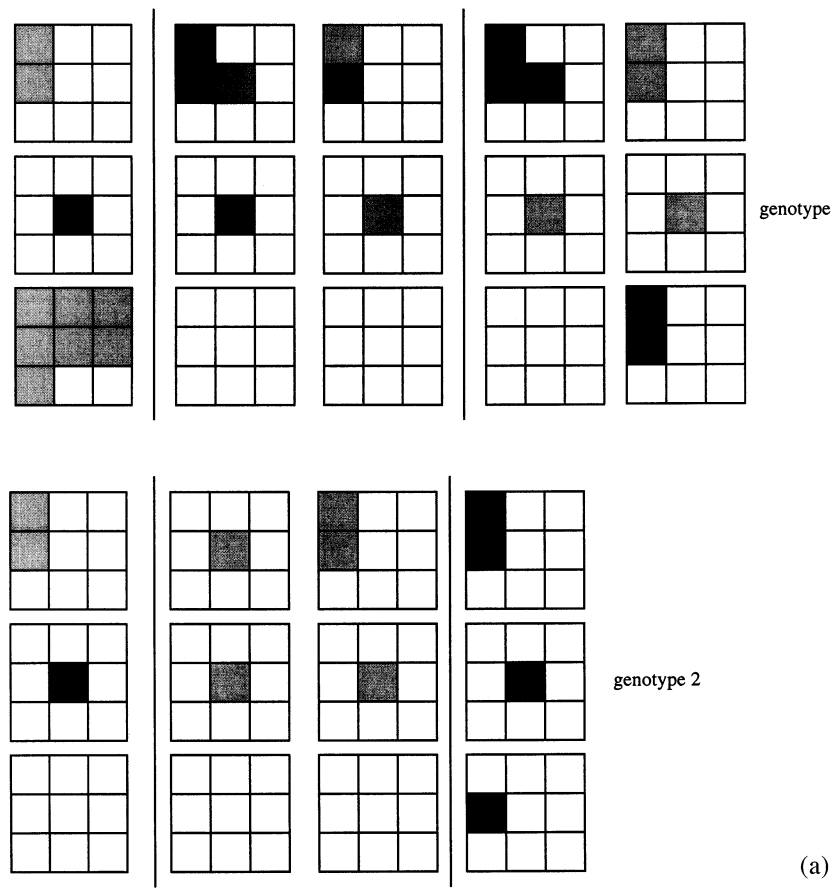


Fig. 8. Encoding of genotype. An n-gene individual implements an algorithm with n micro-rules with two types of bricks, grey and black. Each gene corresponds to a micro-rule: the central site of the central 9×9 square is occupied by the brick to be deposited, where the agent is located; the remaining 26 sites are either empty or occupied by a neighboring brick. (a). Genotypes of a 5-gene individual and a 4-gene individual selected for crossover. Vertical bars indicate locations of crossover application. (b). Genotypes obtained after application of the two point crossover operator.

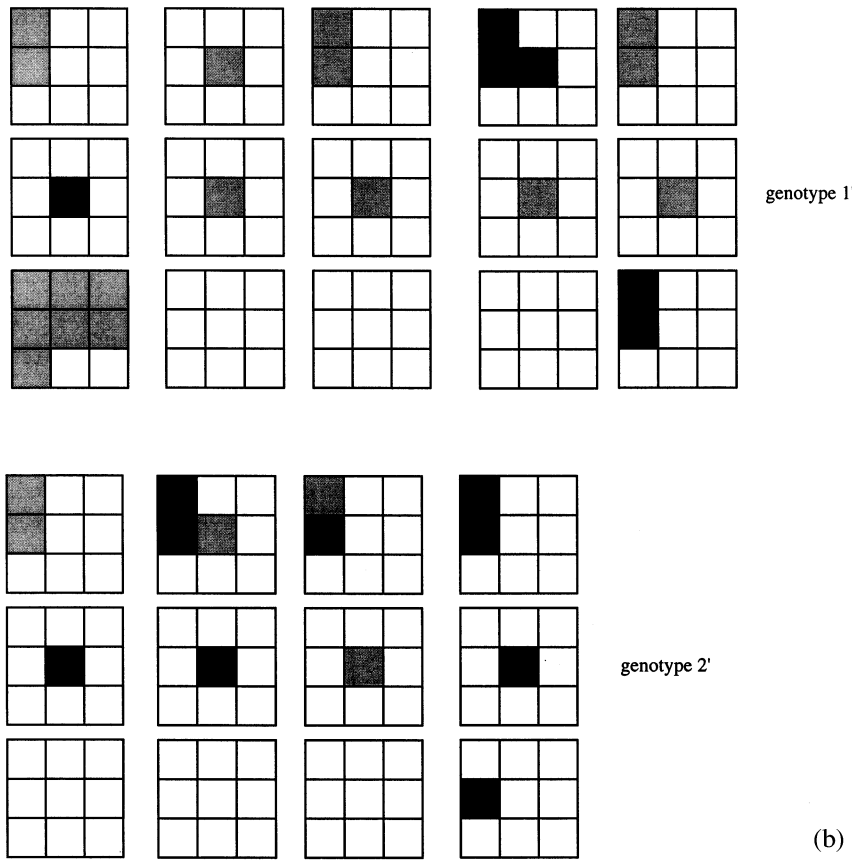


Fig. 8. (*Continued*)

(3) Diagonal brick deposits are not allowed, because they often lead to quick random space filling and are never necessary to obtain interesting architectures. By diagonal brick deposit, we mean a deposit in which the deposited brick has only adjacent edges (as opposed to adjacent faces) with already present bricks. Only stimulating configurations allowing the deposit of bricks which have at least one face in common with at least one other brick are used.

(4) The set of micro-rules is checked for compatibility.

In addition to randomly generated algorithms, the initial population also contains a small number of algorithms that are known to produce structured patterns. Such algorithms allow to focus the evolutionary exploration in the vicinity of coordinated algorithms.

3.3.4. Mutations

Mutations are applied to every gene of every algorithm that results from the combination of two algorithms in the reproductive pool. Mutations are applied as follows: if a micro-rule has not been used during a simulation, its corresponding gene is replaced with a randomly generated micro-rule with probability p_1; if the micro-rule has been used, the same procedure is applied, but with probability p_2, with $p_2 << p_1$, in order to favor applicable micro-rules. In the simulations, $p_1 = 0.9$ and $p_2 = 0.01$.

3.3.5. Crossover

Crossing-over is applied to randomly selected pairs of genotypes with probability $p_c = 0.2$. In the simplest version of the GA, a two-point crossover is performed at two randomly selected

points, as shown in Fig. 8. The crossover operator can be made more intelligent by preventing micro-rules that have been activated during a simulation run from being dissociated by the crossover operator: if two micro-rules have been activated in a simulation, there is some probability that they depend (either directly or indirectly) on each other. High-order dependencies between micro-rules occur frequently in structured architectures.

However, such a crossover mechanism would not be useful because it would result in exchanges of micro-rules that have not been activated. One way of overcoming this problem is to further subdivide active micro-rules into two sets of rules: micro-rules within a given set tend to depend on each other much more than micro-rules from different sets. In order to generate these two sets of micro-rules, we need to define a simple notion of co-dependency. The simplest approximation is to consider the number of times n_{ij} that micro-rule j has been activated right after micro-rule i. This results in a matrix $[n_{ij}]_j$, most entries of which are equal to 0.

This matrix is transformed into an undirected, weighted graph G, which has as many nodes as there are micro-rules. Nodes that represent inactive micro-rules form isolated sub-graphs: they are not connected to any other node in G. Let i and j be two rules that have been activated in the course of the simulation. The weight of the edge that connects the node that corresponds to i to the node that corresponds to j is $n_{ij} + n_{ji}$. If $n_{ij} + n_{ji} = 0$, no edge connects i and j. G is constructed in such a way that only micro-rules are activated next to each other in the simulation are connected, with a weight that reflects the number of their co-occurrences.

A graph bi-partitioning algorithm (Fiduccia and Mattheyses, 1982) is then applied to the set of micro-rules to yield two sets of micro-rules, with the same number of micro-rules in each set, so as to minimize total edge weight across sets, that is, here, co-dependencies between micro-rules that belong to different sets. The crossover mechanism is then applied using the two partitions of G, so as to keep together the micro-rules that are in the same set. With this procedure, we expect the GA to take advantage of existing architectural modules, which result from sets of strongly co-dependent micro-rules. Indeed, better results are obtained with this method.

3.3.6. Learning

We have investigated the effects of including a simplified Lamarckian learning algorithm into the GA: if only a small number of micro-rules have been activated in a simulation, and if the fitness of the current algorithm is less than the average fitness of the previous generation, some of the recently encountered non-stimulating configurations are arbitrarily selected and are made stimulating, and the simulation continues for some time. The learned micro-rules are added to the genotype and inherited without mutation. Such a procedure forces the agents to take actions because they will deposit bricks when they meet configurations that actually exist, thereby boosting brick deposits, but it has not produced significantly better results so far.

3.3.7. Runs

A given simulation stops either when a predefined number of bricks have been deposited in the $16 \times 16 \times 16$ grid, or after a predefined number of steps. In all simulations, the maximum allowed number of bricks is 500 and the maximum number of iterations is 30 000, where one iteration corresponds to all agents taking one action (10 agents are used in the runs). Most algorithms generate random space filling quite rapidly: limiting the number of bricks allows to proceed to the next simulation without spending too much time waiting for the end of a run that would not produce anything interesting. On the other hand, simulations have to be long enough because it takes time to produce coherent architectures, owing to the numerous constraints they induce.

3.4. Results

Our first major result is related to the fitness function itself. In order to test the relevance of the fitness function F, we have measured F on a collection of 29 architectures comprised of structured and unstructured patterns. Fig. 9 shows

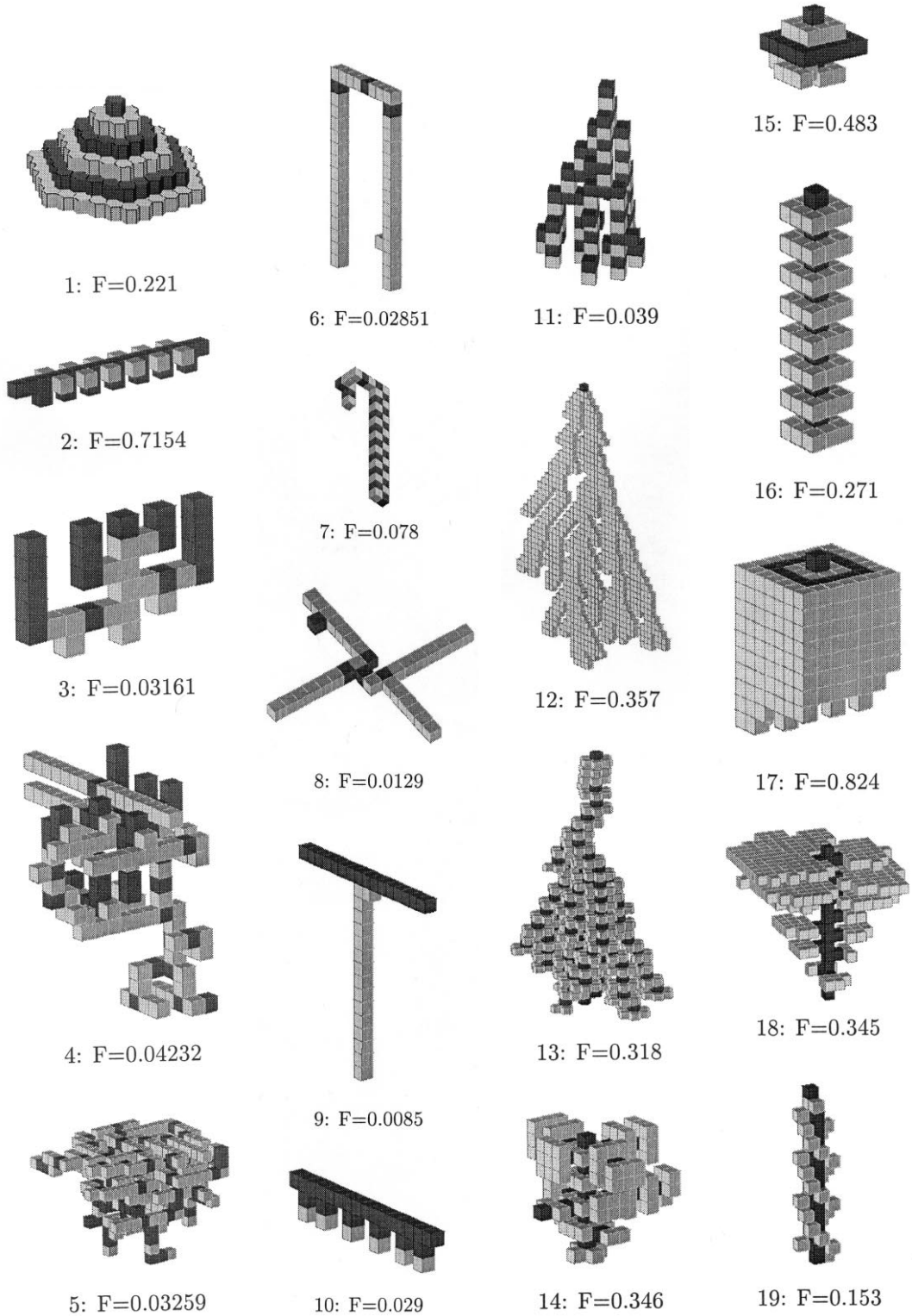
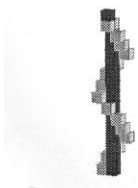
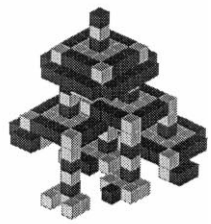


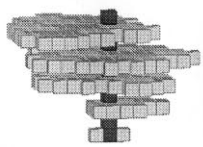
Fig. 9. Sample of patterns used for the test of the fitness function. This figure shows 29 patterns or architectures and their associated fitness (F). Some of the patterns have been generated by hand-tuned algorithm (1, 12, 16, 17, 18, 20, 21, 22, 24, 29) and all others have been discovered by the genetic algorithm.



20: F=0.097

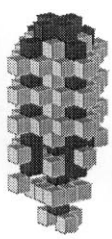


25: F=0.2911

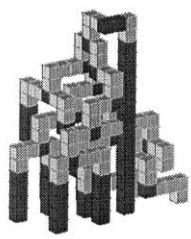


21: F=0.335

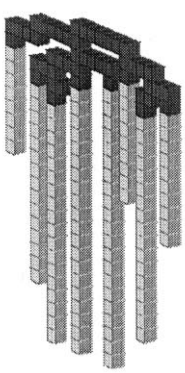
26: F=0.43205



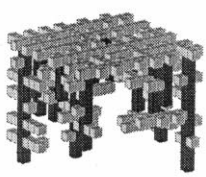
22: F=0.583



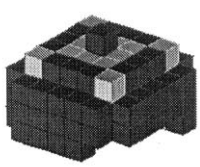
27: F=0.0185



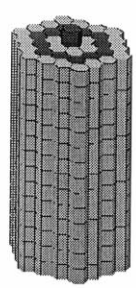
23: F=0.015



28: F=0.38095



24: F=1.278



29: F=0.453

Fig. 9. (*Continued*)

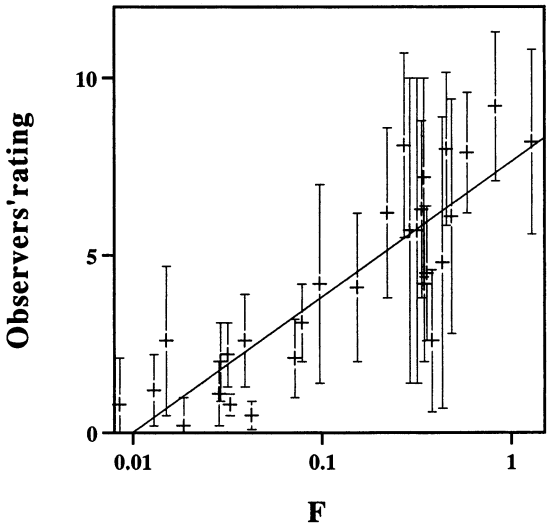


Fig. 10. Computed fitness versus observer-based fitness. Observers' fitness as a function of computed fitness, represented on logarithmic scale, for the 29 architectures represented in 14. Error bars correspond to averaging over 11 observers. Best logarithmic fit is shown (straight line): $y = 3.816 \log F + 7.663$, $r = 0.868$, $df = 317$, $P < 0.001$.

these 29 patterns with their associated fitnesses. Fig. 10 shows the calculated fitness F as a function of a rating obtained from 11 human observers. There is a significant positive correlation between the rating by human observers and the fitness function (best logarithmic fit: $y = 3.816 \log F + 7.663$, $r = 0.868$, $df = 317$, $P < 0.001$). Observers were asked to rapidly scan the collection of architectures, and then take architectures one by one and rate them from 0 to 10 according to the perceived amount of structure, 10 being for the most structured patterns. Observers were asked to use the full spectrum [0, 10] as much as possible. Structure was not predefined but the observers were provided with six examples of patterns that we considered either structured or unstructured (two structured and four random). These training examples were not included in the set of 29 test architectures.

Our second major result is related to the genetic algorithm runs that were based on the fitness function F. Fig. 11 shows a typical example of how the fitness function evolves with generations. The GA was run with 'intelligent' crossover and no learning. All three curves represent averages over 20 runs, each of 400 generations, with the same initial conditions in each run. Error bars represent the standard deviation over the same set of simulations. The average and best fitnesses increase, while the lowest fitness always remains close to 0, most often because of algorithms for which no micro-rule gets activated. As illustrations of patterns found by the algorithm, low-fitness patterns such as those presented in Fig. 9.7, 9.10, 9.11 or 9.27 have been found by the GA; high-fitness patterns such as those shown in Figs. 9.13, 9.14, 9.15, and 9.25 have also been found with the GA. These latter patterns are particularly interesting in that they make use of existing sub-modules.

For example:

- Pattern 9.13 makes use of sub-modules of pattern 9.1.
- Pattern 9.14 makes use of sub-modules of pattern 9.16.
- Pattern 9.15 borrows of sub-modules from patterns 9.16, 1k and 1l.
- Pattern 9.25 makes use of sub-modules of pattern 1l or 1m.

The slow increase of fitness suggests that coherent associations of micro-rules may be destroyed by recombinations and/or mutations. In particular, the application of some micro-rules requires the prior application of other micro-rules. This problem is equivalent to the problem of dealing with epistatic interactions among genes: the fitness of a gene depends on the presence or other genes, making the fitness landscape rugged. But the proportion of interesting shapes among the highest-ranking algorithms is definitely much larger than in random populations. Convergence, however slow, is not unlikely: the fact that the algorithm is capable of making use of existing structures to combine them is encouraging. The mixed success of the GA in producing complex novel patterns is not a 'failure': in the process of designing the fitness function we learned a great deal about what makes an algorithm produce structure, which was our main goal.

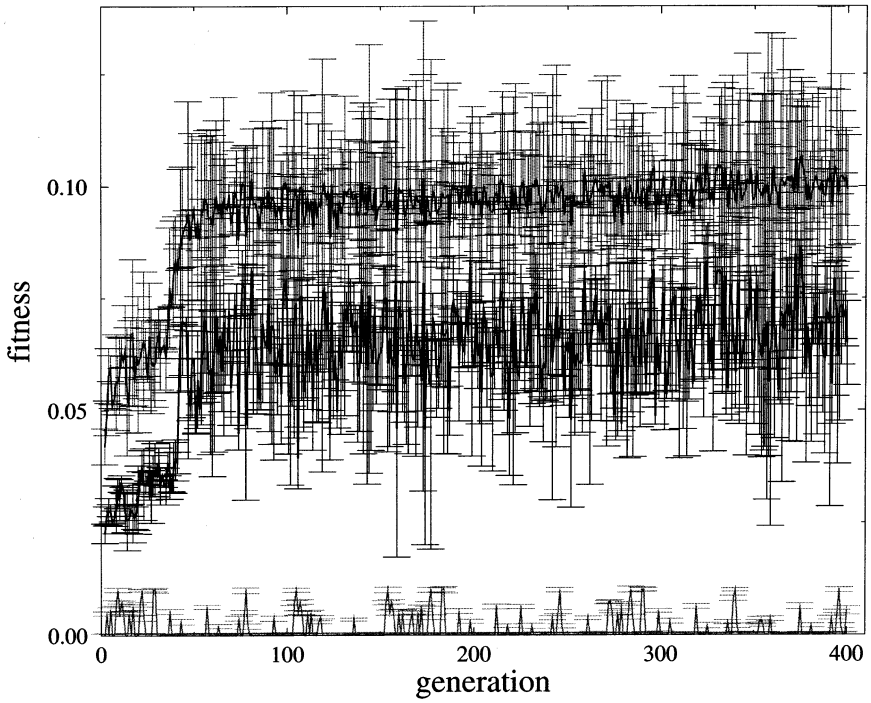


Fig. 11. Fitness evolution. Best, average and worst fitnesses as a function of generation number. Average over 20 runs, each of 400 generations, with the same initial conditions. Error bars represent the standard deviation over the set of simulations.

4. Discussion

4.1. Summary

A simple model of multi-agent building in a three-dimensional discrete lattice has been presented that can produce structured patterns without any direct communication between agents. Some of the properties of the space of algorithms which can be defined within the framework of this model have been described, and an attempt has been made to formally characterize the properties of those architectures that appear structured to a human observer. In order to define a fitness function to be used in a genetic algorithm, a graph-based approach has been developed: a graph is associated to a run that produces an architecture, and fully characterizes that run. A fitness function computed on this graph was then defined, that takes into account the observed properties of structured architectures. The consistency of this function was checked against the ratings of a set of human observers and shown to correlate positively and significantly with these ratings. A genetic algorithm has been used to explore the space of algorithms and architectures using this fitness function. Although the increase of the fitness function is slow, owing to epistatic interactions among genes, some interesting structured patterns have been found: some of these patterns make use of sub-modules. One of the problems was to define a semi-automated procedure to undertake a systematic exploration. Our use of a GA combined with a heuristic criterion is one step in the right direction. The graph-based approach initiated here looks very promising and future work will be aimed at determining a set of graph-theoretic properties that would well characterize structure.

4.2. Implications for morphogenesis

The model presented in this paper is a model of morphogenesis with epistatic interactions among

genes: we have seen that the application of some micro-rules requires the prior application of other micro-rules, indicating that the architectural landscape is more rugged than the factorial correspondence analysis suggests. The patterns generated by our developmental model are inherently polygenic (Nijhout and Paulsen, 1997): the model can therefore contribute to our understanding of how development mediates between the genetic and phenotypic levels of evolution. Whether a particular gene is perceived to be a major gene, a modifier gene or a neutral gene depends entirely on the genetic background in which it occurs, and this apparent attribute of a gene can change rapidly in the course of evolution (Nijhout and Paulsen, 1997). Nijhout and Paulsen (1997) reached similar conclusions with their model of development, which uses a diffusion gradient and threshold mechanism. The parameters of the gradient and threshold mechanisms are affected by genes and in turn affect the phenotype. Eggenberger (1997) also found the same phenomenon in his model of multicellular growth using cell division and differential gene expression in identical cells. In both cases, the fitness function is extremely simple. For example, the fitness function of Eggenberger (1997) is simply a function of the number of cells in the organism and of the symmetry of the organism with respect to a predefined axis. Another interesting conclusion that we share with Eggenberger (1997) is that epistatic processes reduce the need for a long genome: in our model, interactions among micro-rules generate constraints and lead to the precise unfolding of the architecture without any need for the architecture to be encoded in its entirety. Finally, bricks in our model can be viewed as randomly walking cells that all have the same genotype: cellular differentiation (into either one of the brick types) and positioning are not predefined but rather emergent.

4.3. From biology to engineering

This computer model can have interesting consequences for biologists as well as for engineers. Biologists hope to gain some understanding of the algorithmic processes involved in, and required by, building behavior through simulating minimal algorithms followed by simple agents. Such simulations can provide engineers with valuable insight into the design of programs or machines that exhibit collective problem-solving abilities, such as self-assembling machines or robots. Areas of current potential applications for self-assembly include microelectronics, optics, micro-electro-mechanical systems and displays (Hosokawa et al., 1995, 1997; Bowden et al., 1997). In both cases, biologists and engineers are faced with the same 'inverse problem': given a desired shape, or more generally a desired state, find a simple behavioral algorithm that can generate it. Within this context, we believe it is worth developing correspondence tables between simple algorithms and the shapes that they can produce. With our approach one can deal in principle with even more complicated cases, where the objects have more local, but still not a lot of, intelligence. One can hope to be able to implement these more intelligent units using small-scale simple microprocessors, that would connect to one another if neighborhood conditions were satisfied.

The work of Funes and Pollack (1997), which consists of evolving Lego structures to perform specific tasks (crane arm, rotating crane, or bridge) points to what is missing in our work: a functional perspective. Our fitness function is based on rather abstract attributes of graphs, while the fitness function of Funes and Pollack (1997) is based on such concrete attributes as the length of the structure in one direction, the normalized distance to a target point, or the maximum external weight that the structure can support. Our lack of functional perspective was justified by the fact that we don't know what architectural attributes make nests adaptive in nature; but when it comes to designing tools that solve problems, the fitness function is directly derivable from the problem to be solved. On the other hand, despite this limitation, our model has the great advantage of being self-organizing: structures emerge once all elements are put together. The next step will be to combine the functional perspective of Funes and Pollack (1997) with a self-organizing design algorithm, or to combine our self-organizing design algorithm

with a functional fitness. The work that comes closest to achieving this goal is one that combines emergent, collective properties with a functional perspective: Crutchfield and Mitchell (1995) studied the emergence of collective computation in cellular automata models by means of an evolutionary algorithm. The function to be performed was input classification, and allowed the fitness function to be stated quite clearly.

Acknowledgements

This work was supported in part by a grant from the GIS (Groupe d'Intérêt Scientifique) Sciences de la Cognition. During this work E.B. was supported by the Interval Research Fellowship at the Santa Fe Institute.

References

Bennett, C.H., 1988. Logical depth and physical complexity. In: Haken, R. (Ed.), The Universal Turing Machine: A Half-Century Survey. Oxford University Press, Oxford.

Benzecri, J.P., 1973. Analyse des Données. II. L'Analyse des Correspondances. Dunod, Paris.

Bonabeau, E., Theraulaz, G., Arpin, E., Sardet, E., 1994. The building behavior of lattice swarms. In: Brooks, R., Maes, P. (Eds.), Artificial Life IV. MIT Press, Cambridge, MA, pp. 307–312.

Bowden, N., Terfort, A., Carbeck, J., Whitesides, G.M., 1997. Self-assembly of mesoscale objects into ordered two-dimensional arrays. Science 276, 233–235.

Crutchfield, J.P., Mitchell, M., 1995. The evolution of emergent computation. Proc. Natl. Acad. Sci. USA 92, 10742–10746.

Deneubourg, J.-L., 1977. Application de l'ordre par fluctuations à la description de certaines étapes de la construction du nid chez les termites. Insectes Sociaux 24, 117–130.

Deneubourg, J.-L., Theraulaz, G., Beckers, R., 1992. Swarm-made architectures. In: Varela, F.J., Bourgine, P. (Eds.), Toward a Practice of Autonomous Systems, Proceedings of The First European Conference on Artificial Life. MIT Press, Cambridge, MA, pp. 123–133.

Eggenberger, P., 1997. Evolving morphologies of simulated 3D organisms based on differential gene expression. In: Husbands, P., Harvey, I. (Eds.), Proceedings of the 4th European Conference on Artificial Life. MIT Press, Cambridge, MA, pp. 206–213.

Fiduccia, C.M., Mattheyses, R.M. 1982. A linear-time heuristic for improving network partitions. In: Proceedings of the ACM/IEEE Design Automation Conference, pp. 175–181.

Franks, N.R., Wilby, A., Silverman, V.W., Tofts, C., 1992. Self-organizing nest construction in ants: sophisticated building by blind buldozing. Animal Behav. 44, 357–375.

Funes, P., Pollack, J., 1997. Computer evolution of buildable objects. In: Husbands, P., Harvey, I. (Eds.), Proceedings of the 4th European Conference on Artificial Life. MIT Press, Cambridge, MA, pp. 358–367.

Gallais-Hamonno, F.G., Chauvin, R., 1972. Simulations sur ordinateur de la construction du dôme et du ramassage des brindilles chez une fourmi (*Formica Polyctena*). Compte-Rendus de l'Académie des Sciences, Paris 275, 1275–1278.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Grassé, P.-P., 1959. La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux 6, 41–81.

Grassé, P.-P., 1984. Termitologia, Tome II — Fondation des Sociétés — Construction. Masson, Paris.

Hansell, M.H., 1984. Animal Architecture and Building Behavior. Longman, London.

Hosokawa, K., Shimoyama, I., Miura, H., 1995. Dynamics of self-assembling systems. Analogy with chemical kinetics. Art. Life 1, 413–427.

Hosokawa, K., Shimoyama, I., Miura, H., 1997. Self-assembling microstructures. In: Langton, C.G., Shimohara, K. (Eds.), Artificial Life V. MIT Press, Cambridge, MA, pp. 362–369.

Jeanne, R.L., 1975. The adaptativeness of social wasp nest architecture. Q. Rev. Biol. 50, 267–287.

Karsai, I., Penzes, Z., 1993. Comb building in social wasps: self-organization and stigmergic script. J. Theor. Biol. 161, 505–525.

Krink, T., Vollrath, F., 1997. Analysing spider web-building behaviour with rule-based simulations, and genetic algorithms. J. Theor. Biol. 185, 321–331.

Lefebvre, J., 1980. Introduction aux Analyses Statistiques Multi-Dimensionnelles. Masson, Paris.

Nijhout, H.F., Paulsen, S.M., 1997. Developmental models, and polygenic characters. Am. Natural. 149, 394–405.

Skarka, V., Deneubourg, J.-L., Belic, M.R., 1990. Mathematical model of building behavior of *Apis mellifera*. J. Theor. Biol. 147, 1–16.

Theraulaz, G., Bonabeau, E., 1995a. Coordination in distributed building. Science 269, 686–688.

Theraulaz, G., Bonabeau, E., 1995b. Modeling the collective building of complex architectures in social insects with lattice swarms. J. Theor. Biol. 177, 381.

Wilson, E.O., 1971. The Insect Societies. Harvard University Press, Cambridge, MA.