CHAPTER

10

Evolving Information Processing Organizations

John H. Miller

The organization of information processing systems is a central question in economic, organizational, and computational theory. Over the past one hundred years, society has witnessed the rise of large industrial organizations—at times encompassing hundreds of thousands of workers, many of whom are devoted to directly or indirectly "managing" the various activities of the organization. Notwithstanding the economic and social importance of such organizations, with a few rare exceptions (for example, Simon 1981, Sah and Stiglitz 1986, and Radner 1992) economic theory has not been greatly concerned with questions concerning the creation, design, and operation of information processing organizations. Of course, other groups of researchers, ranging from sociologists to computer scientists[1] have focused on various complementary aspects of the basic problem under consideration in this chapter. While the focus here will be relatively narrow, interested readers can find elements of this broader organizational view contained within the many chapters of this book.

Suppose we are given a set of information processors each limited in its ability to accommodate and analyze incoming information (for example, workers in a firm or CPUs in a computer). What is the most effective way to organize these processors? One approach to answering this question, pursued by Radner (1992) and others, is to develop formal mathematical models of information processing networks. This approach provides a useful formal framework from which to derive some valuable insights into, for example, the economies of scale of optimized systems. One potential problem with the mathematical approach, however, is that the inherent complexities of organizations

may require extreme simplification before analytic techniques become tractable, and thus the scope of the subsequent analysis may be severely restricted.

In this chapter a complementary approach is pursued that relies on computational experiments.[2] Such an approach removes a variety of the current limitations of the formal mathematical approach, thus allowing us, at the very least, to foreshadow likely formal results. More importantly, the computational approach makes accessible a variety of questions to which pure mathematical answers may not be forthcoming. For example, questions about learning in organizations and behavior under a large range of environmental conditions can be easily explored via computational experiments.

The analysis proceeds in two related directions. The initial focus is on the generic properties of randomly generated organizations. I analyze the implied behavior of organizations generated by randomly connecting some simple, predefined processing elements to one another. While the analysis of random organizations may, at first, appear frivolous, in fact it offers the potential for generating some critical insights into organizational structure. As has been shown by Kauffman (1991) in numerous biological contexts, randomly generated structures often exhibit "order for free"—that is, even though structures are randomly generated, the resulting behavior that emerges from such structures is well behaved. My analysis supports this "order for free" hypothesis in the context of information processing organizations.

Such "order for free" may fundamentally alter our view of these systems. In such a world, the creation of large, complex, productive structures no longer requires either the directed insight of a rational planner or the highly unlikely congruence of fortuitous events, but rather it is the inevitable result of emergent structure. Even crude approximations to optimal structures are likely to persist, and provide the grist for survival and improvement.

The second area of analysis is on the ability of organizations to adaptively learn better structures. Could a simple process of adaptation lead to the development of superior structures? What structural characteristics are likely to emerge from such an adaptive process? Moreover, insofar as the adaptive algorithm has the ability to find good solutions to the problem of maximizing organizational performance, the structures emerging from the algorithm may provide valuable hints about optimal organizational design under conditions too difficult to solve formally.

The Model

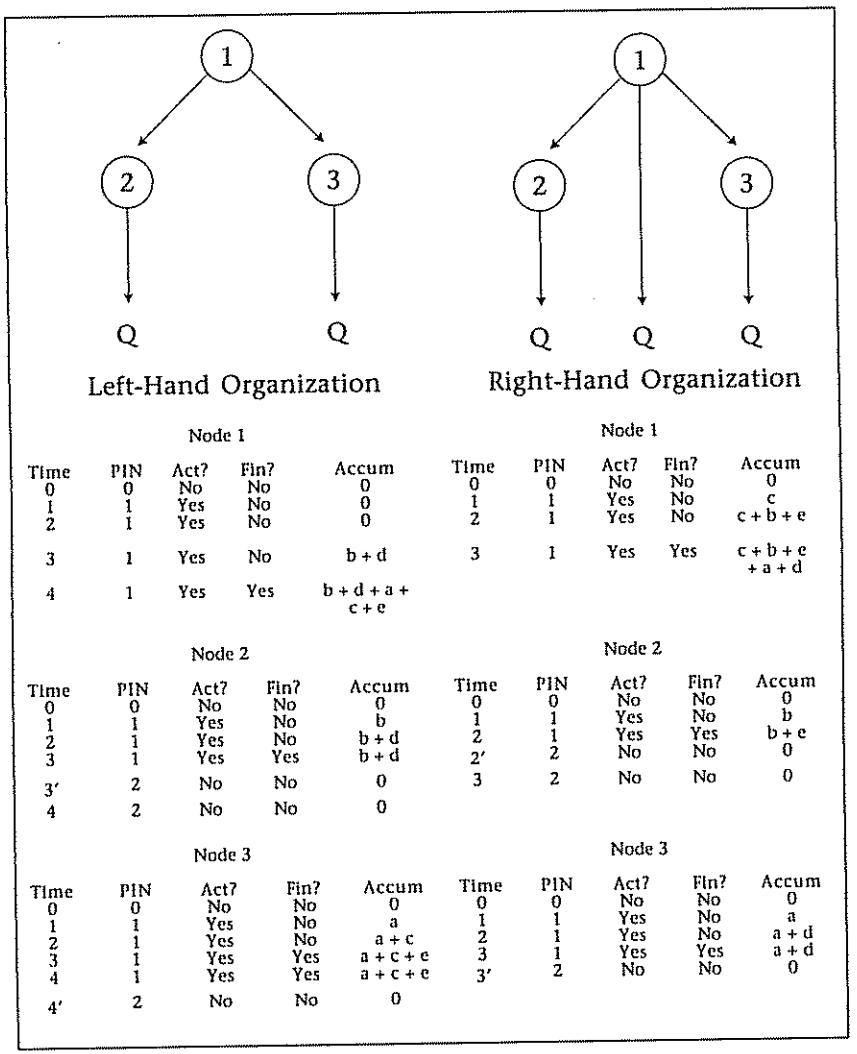
The goal of organizations in this model is to solve a series of information processing problems, the solutions of which are assumed to allow the organization to effectively respond to its environment. For example, firms may need to combine various accounting and market information to make operating decisions that affect profitability. While firms need to get correct answers to such problems, they also face a variety of other challenges. For example, ceteris paribus firms that can answer these problems faster and with fewer processing resources are likely to have a competitive advantage over other firms. While my modeling approach can easily accommodate various firm objectives, I focus only on speed: firms that can solve their problems quicker than other firms are considered to have a competitive advantage.

Following Radner (1992), in the analysis below I assume that firms face a set of fully decomposable (associative) problems.[3] The notion of decomposability implies that while every component of the problem must be incorporated into the solution, the order in which this occurs is not important. For example, the process of adding a series of numbers is fully decomposable, since the order in which the addition is done does not matter. Obviously, the advantage of decomposable problems is that they can always be solved in a decentralized manner since sub solutions (for example, sums of small groups of the numbers to be added) can be easily incorporated into the full solution (by summing up the sums of the sums). Therefore, such problems are amenable to being solved by decentralized organizations.[4]

Organizational Structure

Organizations are composed of a set of interconnected nodes, each of which represents a simple processing resource. At the top of the organization is a single (root) node that needs to accurately calculate the ultimate solutions to a set of problems that the organization faces. For example, one could view the root node as attempting to make a production forecast based on a variety of incoming information concerning consumer demand, production costs, etc. Every node in the organization may be connected to one or more child nodes; every child node, though, can have only one parent node. The connections represent communication channels for the flow of information between two nodes.

The incoming elements of each problem reside in a data queue. Each node has the possibility of being directly attached to this queue. Nodes that have no children are considered to be *terminal nodes* and



Node 1

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	0
2	1	Yes	No	0
3	1	Yes	No	b + d
4	1	Yes	Yes	b + d + a + c + e

Node 1

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	c
2	1	Yes	No	c + b + e
3	1	Yes	Yes	c + b + e + a + d

Node 2

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	b
2	1	Yes	No	b + d
3	1	Yes	Yes	b + d
3′	2	No	No	0
4	2	No	No	0

Node 2

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	b
2	1	Yes	Yes	b + e
2′	2	No	No	0
3	2	No	No	0

Node 3

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	a
2	1	Yes	No	a + c
3	1	Yes	Yes	a + c + e
4	1	Yes	Yes	a + c + e
4′	2	No	No	0

Node 3

Time	PIN	Act?	Fin?	Accum
0	0	No	No	0
1	1	Yes	No	a
2	1	Yes	No	a + d
3	1	Yes	Yes	a + d
3′	2	No	No	0

Figure 1. The operation of two simple organizations.
(Assumes a single, five-element (a, b, c, d, e) addition problem in the queue, and a node firing sequence that always works backwards from the highest numbered node during every time step.)

are always attached to the queue. Information flows "upward" through the organization, always moving in a single direction from the data queue, through any intermediate nodes, and eventually ending at the root node. The top half of figure 1 illustrates two simple organizational structures.

Every node is assumed to be able to process a *limited* amount of information during any given time step. Processing is limited to the following three tasks: deciding what problem the node should be working on, seeing if it is possible to incorporate any subsolutions to this problem from at most one of the node's downstream connections (either a child node or the data queue), and deciding if any more processing is required by the node on the current problem. Nodes must wait for their parent nodes to retrieve information before they can begin working on the next problem.

Each organization needs to "solve" a given number of problems, each composed of a fixed number of elements (pieces of information).[5] For example, an organization may need to find solutions to ten addition problems, each requiring the summation of five integer values. The information for each problem is placed in the data queue. Each element of the queue contains both a problem identification number (PIN) and the associated data. (Thus, if the first problem consists of summing up the numbers 12 and 15, and the second problem consists of summing up the numbers 21, 34, and 56, the queue would be represented as $(1_{pin}, 12_{data})$, $(1_{pin}, 15_{data})$, $(2_{pin}, 21_{data})$, $(2_{pin}, 34_{data})$, and $(2_{pin}, 56_{data})$.) The problems are queued in order, and the queue is accessed one element of the problem at a time. Any time an element of the problem is read off of the queue, the queue automatically advances to the next element. Each organization faces the same set of problems.

Organizational Operation

The global behavior of an organization is the outcome of a simple set of "standard operating procedures" governing the behavior of each individual node. The operating procedures assume limited information processing capability for each node, and confront a number of critical timing issues (such details are often trivialized in theoretical discussions, see below). The operating procedures were created so that nodes can operate in a completely decentralized environment, requiring only limited information about the activities of the other nodes.

During each cycle of the organization, every node in the organization is "fired" once. The sequence of firing is either *random* or *ordered*. Under ordered firing, lower level nodes in the organization are always fired before upper level ones, that is, children are always fired before their respective parents. Obviously, ordered firing requires a high level of synchronization of the nodes' activities. The prerequisites for such synchronization include the ability to easily communicate with each node, a general "knowledge" of overall organizational structure, etc. The impact of coordinating the firing of nodes will be of central concern to the analysis.

Each node can have one of three internal states: (1) inactive, in

which case the node is not currently working on any problem; (2) active and unfinished, in which case the node is currently working on a problem, but it has not finished processing because lower nodes either are, or potentially could be, working on the same problem;[6] and, (3) active and finished, in which case the node is done working on the problem and is ready to pass its solution to a higher node. Each node in the organization also keeps track of the PIN it is either working on or expecting. Initially, all nodes are set to inactive and are expecting PIN 0.

When an inactive node is fired, it tries to activate. It first checks the PINs of all of its active connections (the queue, if the node is attached, and any active children it may have) and its inactive children. If the lowest PIN of its inactive children is less than the lowest PIN of its active connections, then the node goes to the lowest inactive PIN and remains inactive—thus, it will not start up if there is the possibility of a lower numbered problem eventually coming in on one of the inactive connections. If, however, the lowest active PIN is at least as low as the lowest inactive one, then the node begins to calculate a new solution and activates by marking itself as unfinished on that PIN—thus, it begins processing as soon as it is clear that no earlier problems can be passed up from below. By assumption, this activation process requires no energy from the node.[7] Once activated, the node is marked as unfinished and continues to the processing step described in the next paragraph.

A node that is active will always attempt to work on the problem given by its PIN. First, it checks its own children, in random order, to see if any of them have finished working on the problem. If it finds such a child, the node will incorporate the child's solution into its own, and then mark the child as inactive and increment the child's PIN by one. The incorporation of any child's solution takes all of the node's energy for the step, and thus the node quits checking its children at this point and tests if it is finished (described below). If no children are finished and the node is attached to the queue, it checks to see if the queue has an element of its current PIN—if so, this element is incorporated into the node's solution, the queue is advanced, and all of the node's energy is used for the step.

Finally, the node checks to see if it has finished with its current PIN. The node is finished if and only if all of its subordinate connections (both active and inactive children and, if relevant, the queue) are on greater PINs. (Since a child's PIN is immediately advanced when it finishes, this rule potentially allows the parent node to finish on the same cycle that it incorporates a child's solution.) If the node is finished, it marks itself as such and remains active. The node remains in this state until a parent node incorporates the finished solution and deactivates the node.

This procedure is followed for each node during a given cycle. Once every node has been fired, the root node is checked to see if it has solved the current problem (that is, if it has finished processing the PIN of the current problem). If so, the root node is marked as inactive and its PIN is incremented by one.

The firing procedure and activation rules outlined above insure that upper nodes never get ahead of lower ones, and that the problems in the queue are always solved in order. An organization is finished when it has solved all of the problems in its queue.

The bottom table of figure 1 illustrates how this processing occurs when there is a single, five-element (a, b, c, d, e) addition problem in the queue. The nodes are fired in order from the highest to the lowest numbered node during every time step. Given this scenario, the left-hand organization behaves as follows. In the first time step, node 3 activates on PIN 1 and accumulates the first element from the queue (a). Next, node 2 activates on PIN 1 and accumulates the second element (b). Finally, node 1 activates on PIN 1, but cannot accumulate any of the solution since both of its children have not finished processing. In the second time step, nodes 2 and 3 each accumulate one more element off of the queue (leaving node 3 and 2 "knowing" respectively $a + c$ and $b + d$); node 1 remains active but still cannot begin to accumulate any of the solution. In the third time step, node 3 incorporates the last element of the queue (e) into its solution and marks itself as finished (since its only downstream connection is the queue which is now empty). When node 2 fires during this step, it finds that there is nothing in the queue, and thus it also marks itself as finished. When node 1 fires during this step it first checks with node 2, finds that it is finished, and therefore accumulates the child's solution ($b + d$). In response to node 1 taking its solution, node 2 proceeds to the next PIN, becoming inactive and unfinished (this process is designated as step 3'). Finally, on the fourth time step, nodes 3 and 2 fire and remain unchanged, node 1 fires and accumulates the solution of node 3 (giving node 1 the full solution of $a + b + c + d + e$), node 3 is advanced to PIN 2, and lastly node 1 becomes finished. At the end of this time step, the organization has successfully solved the problem in the queue. Although the right-hand organization in this diagram has the same number of nodes as the left-hand one, it also has a queue connection from node 1. This slight change in structure allows the right-hand organization to complete the identical calculation in one less time step.

The notion of a cycle in the above operating procedure differs from that used in Radner (1992). Here, nodes are allowed to use the results from lower nodes that have been fired on the same cycle. Thus, the potential exists for much more processing to be undertaken in a given cy-

cle under this system than the one used by Radner. For example, if nodes are fired from the bottom up, the potential exists for a solution to quickly move upward in a single cycle as the lower nodes finish their processing and pass their results to the upper ones. Obviously, the model could be modified to prevent nodes from using results acquired on the current cycle. Doing so, however, would make issues of synchronization moot and eliminate a central question of the analysis.

The algorithm described above represents a simple set of standard operating procedures for the organization. Obviously, many other possible procedures exist, and, comparisons of such alternatives may be an interesting area of study. Some simple experiments with variants of these procedures indicate that the qualitative results appear fairly robust to simple changes. Nonetheless, I leave a more explicit analysis of this topic for future research. Note that the computational model developed above is easily modified to incorporate new assumptions on, for example, the capabilities of individual nodes, the costs of observing data or communicating within the organization, etc.

Creating "Random" Organizations

"Random" organizations are created for the analysis of both generic properties and evolutionary behavior. Organizations are randomly generated in the following manner. First, the organization's final size (number of nodes) is randomly chosen from the integers between one and fifty. Starting with a single root node, the organization is iteratively constructed by adding one new child node per iteration. During each iteration, a parent node is randomly picked from one of the organization's existing nodes and a child node is added. Thus, during the first iteration the second node always attaches to the root, during the second iteration the third node will either attach to the root or the second node with equal probability, etc. This iterative process continues until the final size is achieved. After the organization is fully generated, any terminal nodes are automatically attached to the queue, and each interior node is attached to the queue with a 0.5 probability.

Organizational Evolution

I explore the evolution of organizations through a simple adaptive computation procedure based on Holland's (1975) genetic algorithm. This allows the analysis of issues of learning and adaptation within an organization. Can organizational structures evolve that are better at meeting processing objectives? If this is the case, what are the structural elements that promote superior behavior?

For the genetic algorithm, a population of fifty random organizations

was initially generated. An iterative procedure was then invoked that creates new populations of organizations by selectively reproducing and modifying organizations in previous populations. Each organization is given a "fitness" measure, here taken to be the time required (number of cycles) to solve the set of problems in the queue, with less time being preferred to more.[8] A new population of fifty organizations is then generated by randomly selecting (with replacement) two organizations from the old population and placing a copy of the better one into the new population—this "tournament selection" is repeated fifty times. Note that this selection procedure tends to generate a population of better performing organizations (although there is some chance that the worst performer will be included and (or) the best will be excluded).

The fifty organizations selected for reproduction are then randomly paired, and with a 0.5 probability undergo two genetic operations. The first genetic operation selects a random node from each organization (excluding the root), and "swaps" the two chosen nodes along with all of their children (that is, the corresponding subtrees) between the two organizations.[9] This swapping procedure is a "crossover" operator, and is similar to Koza's (1992) genetic programming crossover operation. Crossover allows useful building blocks (suborganizations) to be maintained and recombined in, perhaps, a more productive manner. The second genetic operator used is mutation. Each organization has an equal chance of undergoing either zero, one, or two mutations. For each mutation, either: an interior node is selected (if possible) and the queue connection status of that node is altered to the opposite condition (if it was not connected to the queue, it becomes connected and vice versa), a node is randomly chosen and a terminal node is added to it, or a node (other than the root) is randomly chosen and deleted.[10] With equal probability one of these three types of mutations is performed. The mutation operator models small random changes to an organization's structure.

After each pair of organizations is subjected to genetic operators (or not), the newly formed population replaces the old population and a generation is concluded. The system is iterated for fifty generations, following the cycle of gathering fitness measures, selection, and modification described above for each successive population. At the end of the fifty generations, the best organization existing in the final population is used in the data analysis. Since the algorithm has stochastic elements, fifty separate trials of the genetic algorithm are conducted and analyzed for every condition. Although genetic algorithms require a variety of parametric and algorithmic choices, they tend to be robust to reasonable changes in these choices. Limited experimentation with the algorithm used here support this conclusion.

The evolutionary procedures outlined above model simple notions of organizational adaptation. Organizations that perform poorly relative to others are eliminated from the system and replaced by new ones. New organizations "model" themselves after more successful old organizations—either directly copying them or creating a unique variant composed of "parts" of old organizations (crossover) and a few minor alterations (mutation).

Some Experiments

The model developed above allows the conduction of a variety of computational experiments on adaptive organizational behavior. Here, I focus on two questions. First, how important is the ability to synchronize the individual information processors (nodes) to the performance and structure of organizations? Note that in this model, the coordination issue is not whether processing units are working or shirking, but rather in what overall order they are activated. The actual ability of an organization to coordinate processing will depend on a number of factors, including communication technology, recognition of a hierarchical structure, standardization of the time required for task completion, etc. The second question I explore is the impact of the number of problems an organization faces on its structure and performance. Do organizations that face only a single problem differ in fundamental ways from those that face multiple problems? Such a question arises in organizational theory for issues such as designing organizations that can quickly handle small unusual challenges (for example, during a crisis (Carley and Lin 1995)) versus ones that need to process a continual stream of incoming data.

The two factors discussed above are implemented in the following way. The synchronization of processing nodes is either *ordered firing,* in which case child nodes are fired before parent nodes, or *random firing,* in which case nodes are fired in random order during each cycle. Intuitively, organizations that have coordinated firing should be able to support more complicated structures and perform faster than unsynchronized ones. Since the natural flow of processing in these systems is from the bottom up, synchronization may prevent "grid lock" by allowing lower nodes to finish processing and pass on their results to higher nodes on the same cycle. To analyze how the number of problems impacts an organization's structure, a *single problem* condition is studied, in which the organization must solve only a single problem, versus a *multiple problem* condition where a series of ten problems must be solved. Each individual problem is composed of five elements each.

Since problems must traverse each level in an organization, the average solution time per problem is expected to decline in the multiple problem condition as this "transmission overhead" is eliminated after the first problem is solved.[11]

The analysis relies on simple techniques designed to reveal underlying trends. I concentrate on the following structural characteristics: the size of the organization as given by its number of nodes, the number of terminal nodes and total attachments to the data queue, and measures surrounding the *level* (where the level is given by the shortest distance between a given node and the root) of various activities, including queue attachments, maximum level of any node, and the maximum number of nodes at any level (here, called breadth). An organization's mean path length is also observed— that is, the average level of its nodes.

The Generic Behavior of Organizations

A thousand organizations were randomly generated according to the procedure outlined in the section above. The average characteristics of these structures are given in table 1. These characteristics have very little meaning other than to indicate the generic structure that emerges from the random generation process. They also serve as a benchmark for the analysis of the evolving system.[12]

Table 2 shows the average processing time as a function of firing procedure and number of problems. Random firing has a big impact on processing speed, requiring about 47 percent more time in the single problem and 22 percent more time in the multiple problems condition (all of the differences between means are significantly different from 0 at the 0.01 level). Also note that although (not surprisingly) the multiple problems require more time, the time per problem is only 67 percent and 55 percent of that required in the single problem in the ordered- and random-firing conditions respectively. As previously discussed, when a single problem is solved by an organization, its components must be moved through all levels of the organization, one cycle at a time. When multiple problems are solved, additional problems do not have to "wait" until the first problem has cleared all levels before beginning, and consequently average processing time is reduced. Table 3 provides an indication of the economies for solving larger groups of problems. As the number of problems increases in the queue, the average time to solution decreases at a decreasing rate. This reflects the spreading of the fixed cost for the first problem across the other problems.

Figure 2 shows the relationship between processing speed and number of nodes for the randomly generated organizations under random

Characteristic	Average	(S. D.)	Max	Min
Number of Nodes	25.6	(14.7)	50	1
Number of Terminal Nodes	12.8	(7.5)	31	1
Mean Path Length	1.1	(0.6)	2.8	0.0
Nodes Attached to Queue	19.3	(11.3)	43	1
Highest Level Attached to Queue	0.5	(0.6)	3	0
Maximum Levels (0=root)	5.0	(2.0)	10	0
Maximum Breadth	7.7	(4.1)	18	1

Table 1. Random organizations.
n = 1000.

Single Problem	3.71 (0.67)	5.46 (1.01)
Multiple Problems	24.69 (5.28)	30.13 (4.75)

Table 2. Random organizations.
n = 1000/cell. Average (S.D.) processing speed.

	Ordered Firing	Random Firing
1 Problem	3.71	5.46
10 Problems	2.48	3.07
20 Problems	2.45	2.79
30 Problems	2.43	2.67

Table 3. Average processing speed per problem.
n = 1000/cell. Average processing speed per problem.

(top) and ordered (bottom) firing.[13] When solving multiple problems (right) the following patterns are apparent. Regardless of the firing procedure, small organizations (here, four or fewer nodes) are very dichotomous in their performance—either somewhat fast or very slow. Ordered firing dramatically reduces the variation in performance of both types of small organizations. Once the number of nodes increases beyond eight, there is remarkable consistency in the performance of generic organizations, with all processing speeds falling within about a ten cycle range. Note, however, that under ordered firing there does appear to be a gentle downward trend, while under random firing, the trend has a gentle U-shape that reaches a minimum at around eight nodes. Thus, there are indications that ordered firing allows larger organizations to be more effective, while under random firing, an intermediate number of nodes may be advantageous. Random organizations solving a single problem (left) exhibit similar patterns. Ordered firing reduces the performance variation across organizations, and actually forces performance into either three, four, or five cycles regardless of the number of nodes. Random firing results in a gentle upward trend in processing speed as a function of the number of nodes.

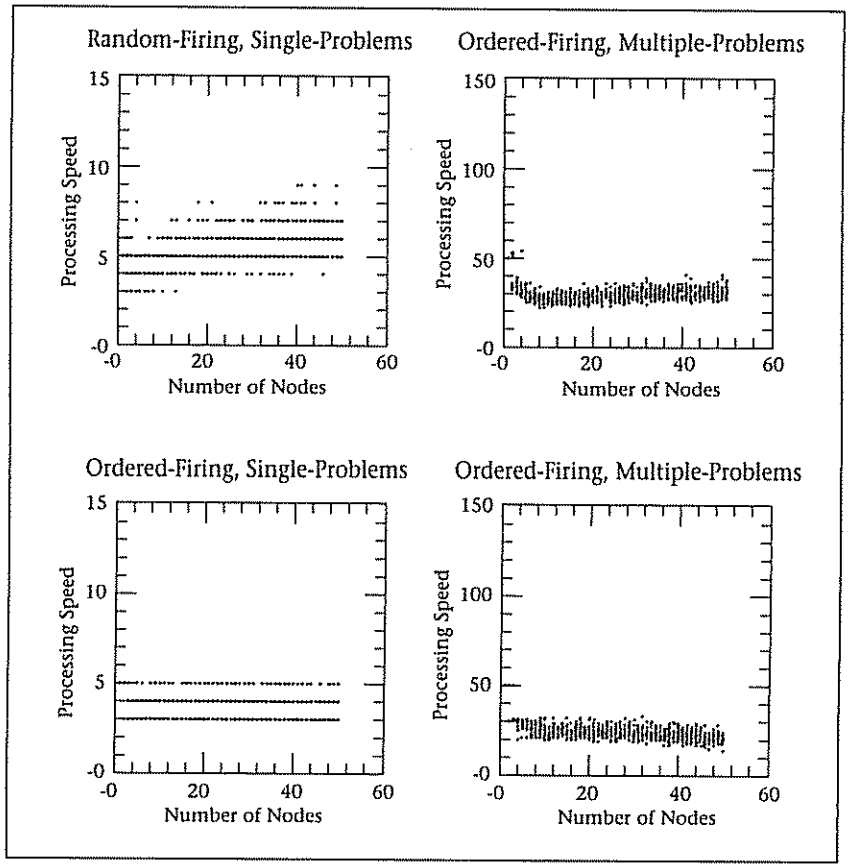


Figure 2. 1,000 randomly generated organizations.

The search for generic properties of randomly constructed organizations reveals a variety of new findings. Perhaps most fundamentally, the performance of even randomly generated organizations is subject to strong constraints. This "order for free" has many implications. For example, small organizations appear to have very discontinuous performance characteristics, and thus are much more "brittle" than large organizations. In small organizations, the connections among information processors are critical as bottlenecks can easily form. However, as the number of processors grows, the impact of local bottlenecks is

minimized by the denser web of alternative routes. Thus, large organizations experience relatively homogeneous performance regardless of structure. Such "size" results are very consistent with the available empirical evidence (see Blau and Schoenherr [1971], or for a more general review, see Kimberly [1976]). Also note that the ability to coordinate processing improves the underlying performance of organizations. Moreover, size advantages are quickly exhausted under uncoordinated firing—intuitively, larger organizations need to transmit the information through more processors, and without coordination the additional time required to do so exceeds the time saved by having the additional processors. A central issue in organizational theory is the emergence of large-scale organizational forms. While such organizations could be the result of a slow, finely-tuned, conscious process, the findings suggest that even large, spontaneously generated organizations are likely to exhibit good relative performance.

Despite the order observed in the performance of random organizations, there remains enough variance in performance to suggest a useful role for search directed to finding better organizational forms. Moreover, it is not clear whether random generation forecloses structural opportunities that might dramatically improve performance (for example, even though almost all random configurations of gears and springs result in roughly the same (very low) quality timepiece, a few such configurations become chronographs). To explore these issues more directly, we turn next to the analysis of the behavior of adaptive organizations.

Behavior of Evolved Organizations

The analysis of evolving organizational structures allows us to consider a number of questions. At the most basic level I am interested in whether simple adaptive processes can lead to better organizational structure. We know from above that innate processing relations may constrain an organization's performance—the issue here is whether these constraints hinder or help adaptation. Clearly the existence of powerful organizing principles may allow the quick development of large structures. However, these same principles may thwart local adaptive search for improved structures. We can also explore the relationship between environmental characteristics and adaptation. Do certain environmental conditions, say, coordinated processing or multiple problems, favor adaptive learning? In this analysis fifty trials of the fifty-generation genetic algorithm described above are run, and the best performing organization during the final generation of each trail is used as the unit of analysis.

	Ordered Firing	Random Firing
Single Problem	2.24 (0.43)	2.92 (0.27)
Multiple Problems	13.90 (0.84)	21.56 (0.77)

Table 4. Best evolved organizations.
(n = 50/cell). Average (S.D.) processing speed.

Table 4 lists the average processing speed of the best evolved organization for fifty separate runs of the genetic algorithm. Notice that the evolutionary process was able to evolve better performing organizations than random generation, with speeds ranging from 53 percent (single-problem, ordered-firing) to 72 percent (multiple-problems, random-firing) of the average randomly-generated organization. (All differences among the means in table 4, as well as differences with the corresponding entries in table 2, are significantly different from 0 at the 0.01 level.) The standard deviations indicate that the most variation in performance occurs in the single problem regime, with single-problem, ordered-firing having the greatest coefficient of variation.

The results from the genetic algorithms can also be compared to those obtained during the 1,000 trials discussed previously. These trials provide a rough benchmark for an algorithm based on random search with preservation of the best. Since each completed fifty generation run of the genetic algorithm is searching at most 1,275 ($50 + 25 \times 49$) new structures, a comparable number of structures is being investigated under each method. In all cases, the average evolved organization is quicker than the best observed generic organization, although with the exception of single-problem, ordered-firing (25 percent quicker) the performance is only marginally better (around 2 percent quicker) for the other three conditions. Nonetheless, given that we are comparing the average genetic algorithm performance to the single best observed random organization, it appears that the genetic algorithm is doing more than simple random search.

The above data indicate that organizations can evolve to be better information processors. Next is an analysis of whether organizations are evolving to a common structure, and, if so, how does that structure depend on environmental conditions. Tables 5-8 provide the summary statistics for organizational structure across single and multiple problems and random and ordered firing.

One of the best indicators of organizational differences is in the number of processing nodes in the evolved organization.[14] In the single problem with random firing, evolved organizations use very few nodes on average (3.0), while with ordered firing, the number of nodes

Characteristic	Average	(S. D.)	Max	Min
Number of Nodes	33.9	(14.2)	50	7
Number of Terminal Nodes	14.4	(8.4)	26	1
Mean Path Length	1.1	(0.9)	3.7	0.0
Nodes Attached to Queue	27.1	(11.9)	48	4
Highest Level Attached to Queue	0.7	(0.6)	3	0
Maximum Levels (0 = root)	8.0	(2.5)	16	4
Maximum Breadth	8.2	(4.5)	18	1

Table 5. Best evolved organizations.
$n = 50$. Ordered firing, single problem.

Characteristic	Average	(S. D.)	Max	Min
Number of Nodes	3.0	(1.1)	6	2
Number of Terminal Nodes	1.3	(0.4)	2	1
Mean Path Length	0.2	(0.3)	1.0	0.0
Nodes Attached to Queue	2.8	(1.0)	6	2
Highest Level Attached to Queue	0.1	(0.3)	1	0
Maximum Levels (0 = root)	1.7	(0.7)	3	1
Maximum Breadth	1.3	(0.5)	2	1

Table 6. Best evolved organizations.
$n = 50$. Random firing, single problem.

goes up dramatically (33.9). In the latter situation, there is a lot of variation in final size (with a range of 43 nodes). With a single problem, only a limited number of nodes can possibly be used to capture and process data. Under ordered firing, adding extra nodes to the organization without disturbing the core processing structure will not increase processing time since the coordination allows the core structure to function as if the additional nodes were not present. Under multiple problems, the average number of nodes (48.3) approaches the maximum of fifty with ordered firing, and only 12.1 under random firing. The low number of nodes used in the random-firing structures, suggests that the advantages of size may be rapidly diminished. Also note that the variance in structure size under ordered firing drops dramatically as compared to that in the single problem—as long as there is coordination, multiple problems are best solved with more processors. Figure 3 displays the performance of the evolved organizations in the four conditions versus number of nodes.[15]

While additional nodes are useful because they potentially add processing power to the organization, they may come at a high cost as they increase the number of levels through which information must flow. Random firing of the nodes seriously disrupts the ability of an organization to rapidly move information from lower to upper nodes. For example, consider an organization composed of n nodes lined up in single file (that is, each parent has exactly one child) that must solve a single problem composed of only one element. Thus, this organization must simply pass the element from the bottom node to the root. If the nodes are fired sequentially starting at the bottom, then regardless of the size of the organization, the information will arrive at the root at the end of the first cycle since each node is always able to retrieve the element and finish processing before its parent is fired during the cycle. However, suppose that the nodes are always fired from the root downward. Here, it will take n cycles since parents always fire before children, and therefore only the lowest-level node will be able to finish during the first cycle, the penultimate node during the second cycle, etc. Thus, as coordination breaks down, the time to transmit information through the levels of the organization also increases. The analysis indicates that good organizations must recognize that the benefits in processing power implied by additional nodes are mitigated by increasing transmission cost.

Characteristic	Average	(S. D.)	Max	Min
Number of Nodes	48.3	(1.9)	50	43
Number of Terminal Nodes	22.2	(2.3)	29	18
Mean Path Length	1.62	(0.7)	3.2	0.4
Nodes Attached to Queue	38.5	(2.6)	46	34
Highest Level Attached to Queue	0.7	(0.7)	2	0
Maximum Levels (0 = root)	8.3	(1.6)	13	5
Maximum Breadth	10.5	(2.3)	16	6

Table 7. Best evolved organizations.
$n = 50$. Ordered firing, multiple problems.

Characteristic	Average	(S. D.)	Max	Min
Number of Nodes	12.1	(2.4)	IS	9
Number of Terminal Nodes	5.7	(1.3)	9	4
Mean Path Length	1.1	(0.4)	2.1	0.2
Nodes Attached to Queue	10.0	(2.4)	16	6
Highest Level Attached to Queue	0.6	(0.7)	2	0
Maximum Levels (0 = root)	4.1	(0.8)	6	3
Maximum Breadth	4.3	(0.9)	7	3

Table 8. Best evolved organizations.
$n = 50$. Random firing, multiple problems.

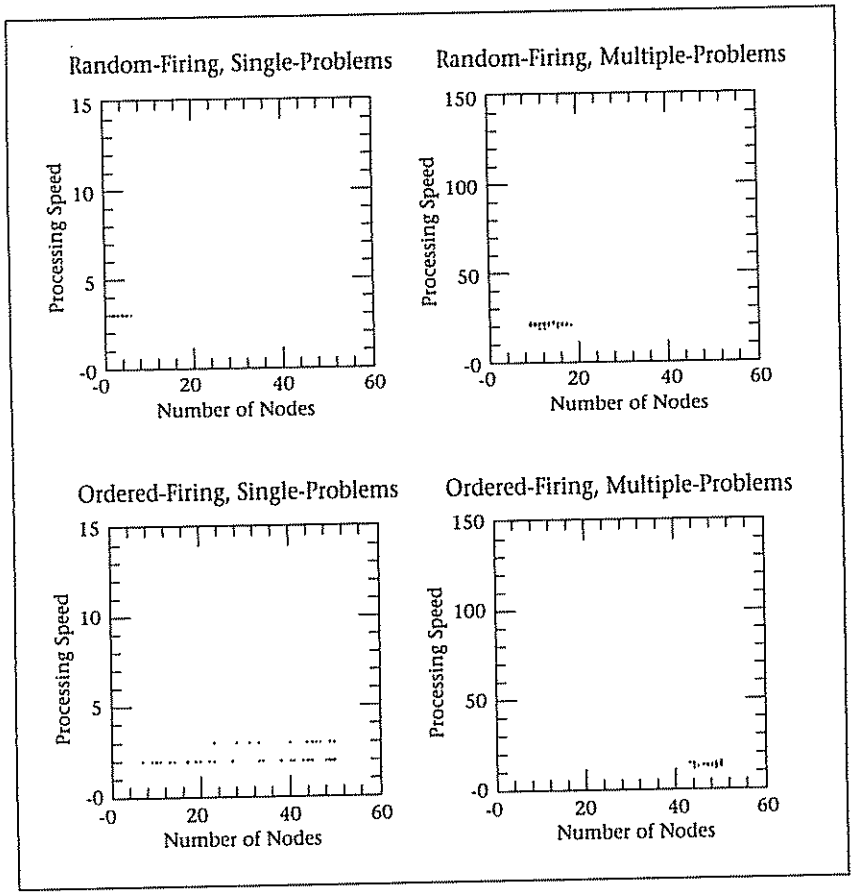


Figure 3. Fifty best-evolved organizations.

These results appear to support the following conclusions. First, simple adaptive mechanisms allow for the creation of superior organizational structures. The type of structure that emerges, is closely linked to the underlying environmental factors that the organization confronts. Random firing of the nodes, that is, low organizational coordination, reduces the evolved size of the organizations. Also, as the number of sequential problems that must be solved increases, so does the number of nodes incorporated into the structure. The results suggest that there is a trade off between the increased processing power of additional nodes and the ability to coordinate the transmission of the information through larger structures. While I do not have proofs of optimal structures across these conditions (nor, is it clear that such proofs are possible using current analytic techniques), the genetic algorithm

was designed to solve difficult, nonlinear problems, and thus the structures that emerge from the algorithm should contain valuable hints about optimal forms.

Discussion and Conclusions

Through the use of simple computational techniques I have been able to analyze some issues surrounding the development and performance of information processing organizations. At the most fundamental level, such organizations exhibit "order for free," that is, once a simple standard operating procedure and open framework for organizational structure are imposed, even randomly generated structures imply well-defined patterns of behavior.

These generic patterns suggest the following: (1) small organizations are subject to great variation in performance, while large ones tend to quickly converge to similar performance levels; (2) coordination of processing activity can significantly increase performance; (3) initially, larger organizations tend to improve performance, but only with coordination do the advantages of size persist (thus even when processing units are free goods, one may still want to limit their use if sufficient coordination mechanisms are not available); (4) even under conditions when more processing units are useful, they exhibit rapidly diminishing returns; and (5) there are economies to solving multiple problems, although at the cost of requiring additional processing units.

Beyond the immediate applications of the above results, the notion that order exists among random organizations has some simple, far-reaching implications. Foremost, it implies that the emergence of large organizational structures does not require great forethought. Depending on the ability to coordinate activities, large collections of even randomly organized processing units could achieve high levels of performance (especially, relative to smaller collections of nodes).[16] While the role of conscious (or unconscious) efforts at refining the structure and improving the performance of organizations is not denied, the above finding may alter the assumptions under which we begin to model such behavior.

To explore the issue of adaptive organizations, a genetic algorithm was employed to evolve better performing structures. The algorithm models a system in which poorly performing organizations are replaced by new ones that are derived from the more successful existing organizations. The results indicate that under such an adaptive process: (1) superior organizational forms are quickly developed; (2) with high coordination of processing units, a variety of forms will

solve single problems, but multiple problems tend to employ the maximum number of nodes possible; and (3) with low coordination of processing units, an intermediate number of nodes is used, with multiple problems requiring more nodes than single problems.

The goal of this research has been to better understand the generic properties of organizational structure and to provide insight into the dynamic process of organizations trying to adaptively improve their performance. The ability to understand, or, even ask, such questions is very dependent on the set of theoretical tools available. Here I rely on recent advances in adaptive computation to develop a simple methodology from which we can conduct numerical experiments and explore key issues. The opportunities for analysis using such techniques are immense. For example, experiments in which processing units either randomly misfire (for example, workers shirk on occasion), are absent from the organization (for example, workers call in sick), or have specialized abilities can be conducted.[17] Obviously, the basic methodology developed here also makes numerous other questions concerning the emergence, design, and operation of organizations amenable to analysis.

Acknowledgments

I am grateful to Jody Lutz and, especially, Hollis Schuler for research assistance, and to Dean Behrens, Kathleen Carley, Robyn Dawes, and Steven Klepper for useful comments. The Adaptive Computation program of the Santa Fe Institute provided partial research support.

Notes

1. In fact, one practical application of this work is in the design of large scale, decentralized computing networks (Schwartz 1980). The ready availability of large, interconnected networks of independent processors, as well as the inherent physical limits of serial processing, has provided the impetus for developing distributed (parallel) processing systems.

2. See, for example, Holland and Miller (1991) for a discussion of the broader implications of the computational approach, or Carley (1995) for a discussion more focused on organizational theory.

3. The methodological approach outlined here should also work for more complex varieties of problems, such as those discussed by Reiter (1995).

4, Note that another prerequisite for decomposability is that each processing unit has the ability to solve any given piece of the problem.

5. One extension would be to have queues with problems of variable length. Experiments with this condition indicate that this has little effect on the results reported here.

6. To avoid losing information about a given problem, a node must insure that all of its children are done working on the problem before passing it onward.

7. Obviously, assumptions about the amount of energy required during any processing step can be easily altered.

8. As previously mentioned, we define "better" organizations as those that have quicker processing abilities, however, modifications that included costs of nodes or various processing steps could easily be incorporated into the model.

9. The maximum allowable size for an organization is 50 nodes-if the chosen swaps would violate this constraint, then it is not performed.

10. When a node is deleted during mutation, if it has no children it is simply removed from its respective parent (if this causes the parent to become a terminal node, then a queue connection is forced on the parent). If children exist, then all of the connected children are placed in the deleted node's former position.

11. Intuitively, like oil in a pipeline, while it may take a long time for the first drop to be transmitted, the second drop follows immediately.

12. Since the genetic algorithm uses the same random procedure to create an initial population of structures, differences in evolved structural characteristics will give some indication of the effect of adaptation on organizational structure.

13. In all of the figures, the density of observations is difficult to depict, although an attempt was made to "fuzz" the plotting to give some indication of this property.

14 This is not too surprising, as most of the other characteristics are highly correlated with this variable; Organizations that are larger on average, also tend to have more terminal nodes, greater mean path length, more nodes attached to the queue, more levels, and greater breadth.

15. Note that in the NW panel, the distribution of final organizations is not readily apparent. In fact, 76 percent of the trials had a speed of two and size of four.

16. Please note that this statement is conditional on nodes not being a scarce resource.

17. In fact, we have begun to explore the first two of these conditions, and have found that the missing node condition results in better performance than the misfiring node condition, since eliminating a node by reconnecting its links to the parent still allows fairly efficient processing, while misfiring directly slows down each cycle.

Copublished and distributed by The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, and London, England.

Library of Congress Cataloging-in-Publication Data

Dynamics of organizations : computational modeling and organization theories / edited by Alessandro Lomi and Erik R. Larsen.
p. cm
Includes bibliographical references and index.
ISBN 0-262-62152-5 (pbk. : alk. paper)
1. Organizational behavior, Mathematical models.
I. Lomi, Alessandro, 1961– II. Larsen, Erik R.
HD58.7 .D96 2001
302.3'5–dc21
2001022724

Printed on acid-free paper in Canada.

10 9 8 7 6 5 4 3 2 1

Dynamics of Organizations

Computational Modeling and Organization Theories

Edited by
Alessandro Lomi and Erik R. Larsen

AAAI Press / The MIT Press
Menlo Park, California Cambridge, Massachusetts London, England