Breeding Better Buildings

Civil engineers may be able to design more innovative and improved structures by borrowing from genetics

Rafal Kicinger and Tomasz Arciszewski

Civil engineers are often very conservative in how they pursue their profession—and for good reason: If their designs don't work, things can fall down and kill people (as the recent bridge collapse in Minnesota so clearly demonstrates). Following novel approaches increases the probability of making such disastrous mistakes. It can therefore be prudent to build on designs that are known to function well. But global society is changing rapidly, and new challenges often call for engineers to innovate while simultaneously remaining vigilant about safety. Costs, competition, energy savings, climate change and security issues are several of the burgeoning concerns that new buildings must be designed to address.

To create dramatically new kinds of structures, engineers can mine many different sources of inspiration, but the natural world offers perhaps the richest lode. Indeed, engineers have probably drawn ideas from nature for millennia: A fallen log may have inspired the first bridge; a cave entrance, the first archway. And as scientists gradually began to understand the mechanisms governing various biological processes, engineers of all stripes were able to apply this knowledge to building complex devices. (The most famous example of such "biomimicry" may be Velcro fasteners, the idea for which came from the observation of sticky burdock seeds.)

Rafal Kicinger is senior analyst at Metron Aviation in Virginia. In 2005 he received his Ph.D. from George Mason University, where he was formerly a research assistant professor. Tomasz Arciszewski is a professor in the department of civil, environmental and infrastructure engineering at George Mason University. He earned his doctorate from Warsaw University of Technology in Poland. Address for Kicinger: Metron Aviation, Inc., 131 Elden Street Suite 200, Herndon, VA 20170. Internet: kicinger@metronaviation.com

We hope to heighten civil engineers' appreciation for biology by suggesting that they go one step further, not only imitating natural shapes and forms but simulating nature's evolutionary and developmental processes to arrive at their designs. Below we describe some computational methods that we have developed to design the support structures for buildings by imitating the action of genes and DNA.

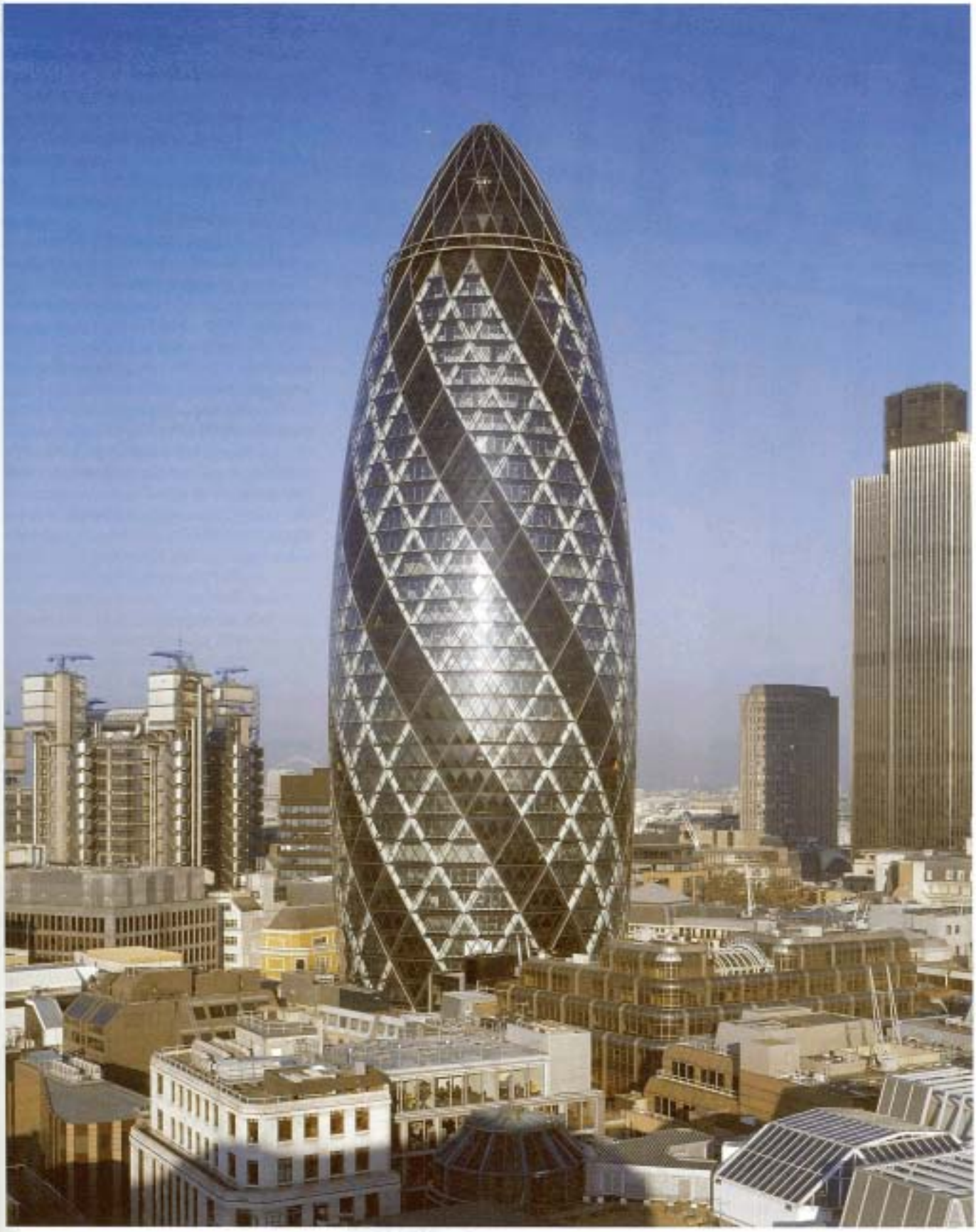
Evolving a Blueprint

Living things adapt and advance in the natural world through natural selection, which acts on the genetic variation present in the population. In this way, nature can over time produce organisms that are optimized to meet the challenges of their environments. Simulated evolutionary processes may likewise provide pragmatic solutions to difficult problems. Such "genetic algorithms" were investigated in the 1960s and have since been successfully used in a variety of spheres.

The basic idea behind an evolutionary algorithm is that a computer program uses a large population of different inputs that each specify one way to perform some task—it could be anything from scheduling events to designing an airfoil. However, no effort is made to optimize these inputs at the beginning; the computer usually generates them randomly. The program transforms these inputs into a set of corresponding outputs, which it evaluates based on some predetermined criteria. Most often in the first round these outputs perform the task poorly, but the program can take the inputs that give the best results and combine them in some fashion to create a second generation. With these hybrid inputs, the computer then creates a new set of outputs, which it once more evaluates so that it can generate a third generation of inputs. The program repeats this process many times until it comes up with a satisfactory output. In this way, the process emulates "survival of the fittest," in that only those individuals that perform best go on to have "offspring."

In engineering, practitioners have traditionally used evolutionary techniques only to make incremental improvements to their handiwork. But recently, some more powerful evolutionary systems have emerged. With these schemes, computer-simulated evolution doesn't just hone existing designs; rather, it hammers out wholly new and innovative concepts. Examples include novel designs of electric circuits, robots and many others. Around the turn of the millennium, a group led by one of us (Arciszewski) in the Volgenau School of Information Technology and Engineering at George Mason University developed such a system, called *Inventor*. It utilizes evolutionary algorithms to identify optimal configurations for the steel structures in tall buildings. Although the highly schematic designs *Inventor* produces are usually too rudimentary to possibly be used for the construction of a real building, they are extremely useful in stimulating new and creative thinking about how to put structures together.

The engineering designs *Inventor* works with are fully specified by a collection of "genes." Individually, each gene determines a specific structural element. Genes in four separate sections of a building's "genome" correspond to four different kinds of elements: bracings, beams, columns and support footings. Each gene has a numeric value that encodes the type of structural element to be used. For example, if the value of a gene describing a bracing is 0, nothing is to be put in that position; the value of 1 indicates a diagonal bracing; 6 calls for



VIEW Pictures Ltd / Alamy

Figure 1. Modern-day architects and engineers can gain much insight and inspiration by studying living things. This building, the Swiss Re Tower in London, resembles a microorganism called a glass sponge. By looking even deeper into biology, at the level of genes and DNA, civil engineers may be able to develop a completely new approach to their work. Using so-called genetic algorithms, they may be able to imitate the biological processes of genetic crossover, mutation and evolution in computer simulations to create optimized designs.

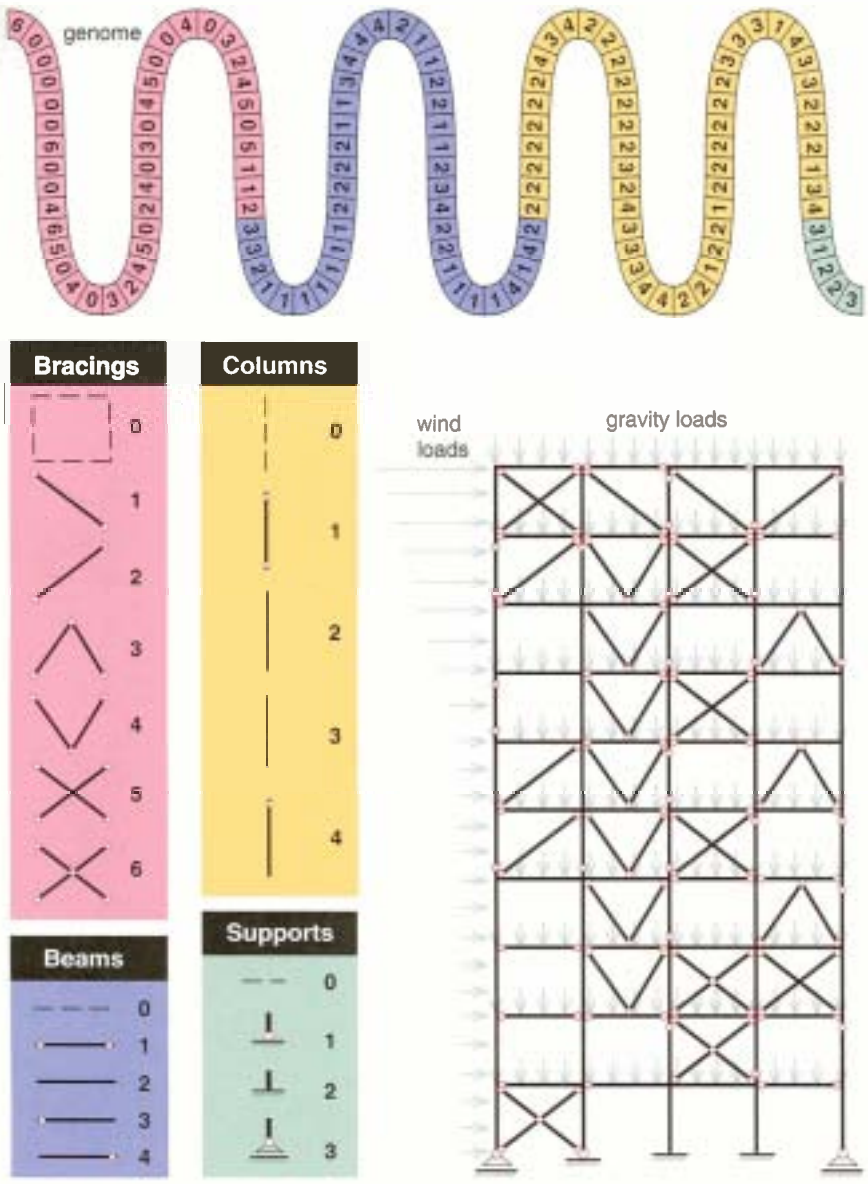


Figure 2. To simulate the internal structure of a building, the authors' genetic algorithm first produces a random "genome" containing the numerals 1 through 6 *(top)*. Different sections of the genome encode bracings *(pink)*, beams *(purple)*, columns *(yellow)* and supports *(green)*, with the numbers specifying the type of element to be used *(left)*. The computer program allows as many parameters as are needed to describe each of the segments across a given floor and each of the stories in the finished building. The leftmost position in any one section of the genome corresponds to the bottom left of the building, with the remaining sequence describing other structural units, left to right and bottom to top *(right)*. After the program translates a genome sequence to a structure in this way, it can test how well the model building holds up to standard wind and gravity loads *(gray arrows)*.

an X-bracing; and so forth. Thus there is a one-to-one mapping of the genes to the components to be used in the building.

Inventor generates a random set of genomes (which are just sequences of numbers) and from them forms an initial population of candidate designs, called "individuals" to keep with the biological analogy. Alternatively, the user can specify a set of designs to make up the initial population. It is normally better to allow the computer to start with a random set, however, because otherwise, there may not be adequate variation in the starting population, and the program will end up merely polishing some established design rather than striking out boldly to create something entirely new. In most cases, the program uses a population of genomes (of a predetermined number) and maintains this number through successive generations. To assess the individuals, *Inventor* translates the genes into structural elements to form a virtual building (or rather, a virtual steel skeleton for a virtual building). *Inventor* can then calculate the internal forces and stresses in the structure from wind and the weight of contents or occupants that the equivalent real structure would experience using another program called SODA (for Structural Optimization Design and Analysis), a commercial system for evaluating steel structures, which we integrated with *Inventor*. SODA checks the feasibility of each design by testing whether it satisfies all requirements regarding the strength, stability and usability specified by the relevant building codes. *Inventor* then calculates a "fitness" value for each design based on whether it passes the SODA test and on the total weight of all the elements (a good approximation of the cost). Low-weight designs receive high fitness marks, and high-weight designs (or ones that don't satisfy SODA's tests) receive low ones.

After the program runs through all genomes in the initial population and assigns each of their corresponding designs a fitness score, the selection of pairs of "parents" begins. All individuals can reproduce (some multiple times), but the chances of their doing so are highest for those with the greatest fitness values.

Inventor copies both parents' genomes and then swaps all the genes that lie between two randomly-chosen points. In this way, it forms two "child" genomes, which share some traits with both parents. The program then further alters the children with "point mutations." That is, it makes random changes to the value of a few genes. The computer repeats this process until it generates as many offspring as there were parents.

Because the software allows fitter individuals to reproduce more often, on average the offspring encode better structures than did their parents. If we let the computer repeat this process for hundreds or thousands of generations, we end up with designs that are vastly better than the ones we started with. Thus this simulated evolution is able to generate novel designs as well as gradually refine them. The degree of improvement, however, tends to lessen with each generation. So after a while, with this diminishing rate of return, there is little point allowing the simulated evolution to continue.

Bringing Up Buildings

We investigated the capabilities of *Inventor* and were pleased to see that it could produce interesting designs that were at least as good as conventional ones in terms of structural integrity and weight. However, the better configurations were oddly irregular, with seemingly haphazard arrangements of elements. We thought we might be able to improve on the aesthetics, and perhaps even the performance, by mimicking not only genetics, but an additional feature of living things: the processes of development.

Development in biology is the emergence of organized structures from an initially very simple group of cells. The power and creativity of such processes can be harnessed to find solutions to complex engineering problems. How can we emulate development in our models? Easy: Rather than having a direct mapping between a gene and each element, we start with a genome that represents an arbitrary recipe for assembling the structure. We then use our same genetic algorithm to improve the recipe over many generations.

The recipe we've devised has two parts: a design "embryo," which describes the elements of the first story of the building, and a design rule, which determines the arrangement of the next floor based on the one below. The design embryo corresponds to an undifferentiated group of cells in a nascent organism, whereas the design rule simulates the developmental processes encoded by these cells' DNA. The design rule is like a so-called *cellular automaton*, a computational model for the evolution of some entity on a regular grid of cells (often this is nothing more complicated than a two-dimensional pattern drawn on an otherwise featureless background grid), one that is completely defined by its iterative rule and by its initial configuration.

To keep things simple, here we consider just the wind bracings in a tall building. In addition, we allow only two possibilities for each bracing position (or "cell"): an X bracing or no bracing, represented by values of 1 and 0, respectively. For illustrative purposes, our building is just four bracing bays wide, which allows us to specify the embryo with just four numbers, such as 1, 0, 0, 0, which correspond to (from left to right) X bracing, no bracing, no bracing, no bracing.

The design rule we have adopted decides whether or not to put a bracing in a given spot by examining the cell

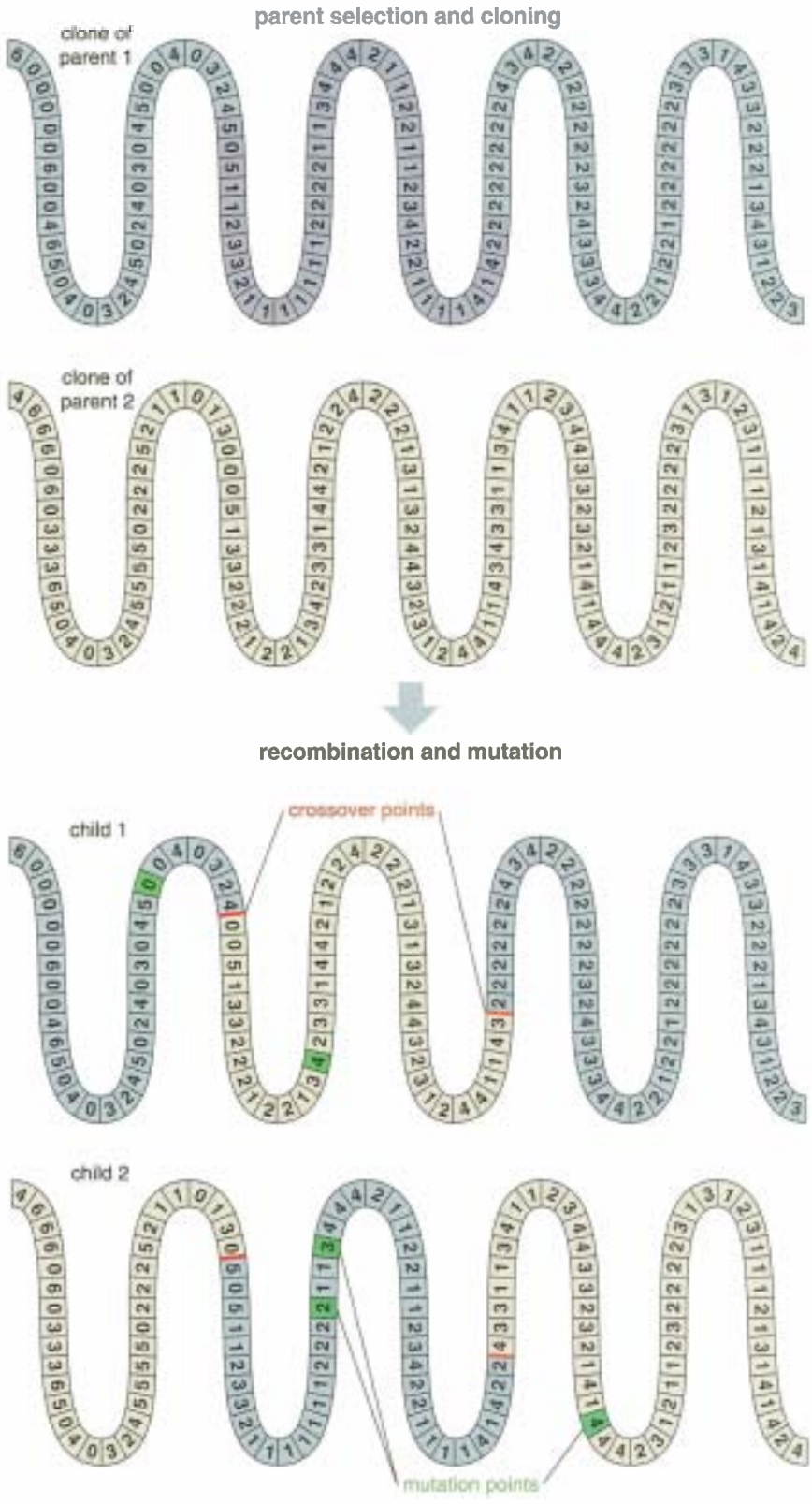


Figure 3. As the computer program tests an entire generation of building genomes, it assigns each a fitness score. These scores are used to select pairs of genomes for breeding the next generation. Once two genomes are chosen, they are copied, or "cloned" *(top)*. The program then does something similar to the genetic mixing that takes place, for example, between paired chromosomes that experience crossover during meiosis *(bottom)*: The computer switches the section of genes between two crossover points *(red lines)* on the first parent's genome with those on the second, a phenomenon biologists call recombination. The program then simulates mutation by giving a few random genes new values *(green squares)* so as to increase the variation between generations further.

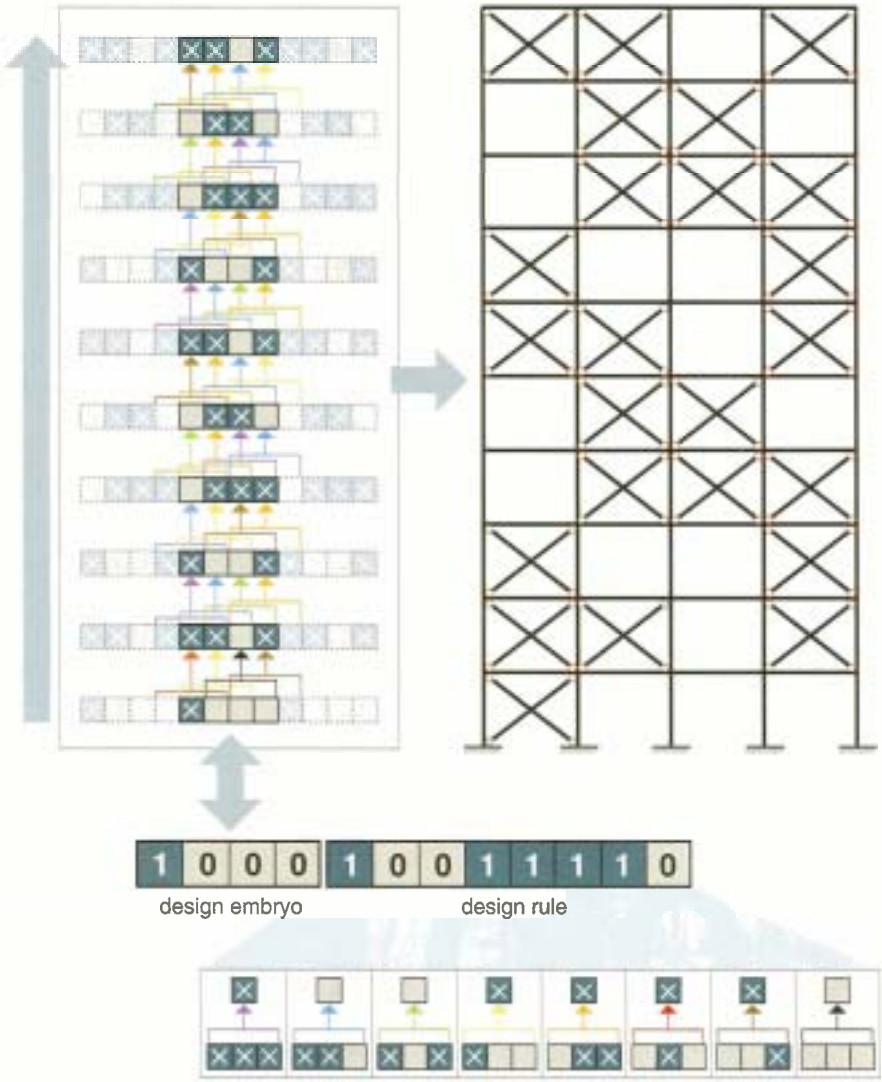


Figure 4. Rather than specifying the elements of a building independently, a genome can determine the configuration of structural elements by emulating the process of development in biology. Here the program starts with a design "embryo," which represents just the first story. The genome also contains a design rule, which defines how the embryo "grows" the next story of the building. The process is then repeated from the second to the third story, and so on until the building reaches a predetermined height. In this simple example, there are only two types of genes, a 0 and a 1, which represent, respectively, either no bracing or an X bracing. The design embryo (here, the binary number 1000) is translated into its corresponding structure (X bracing, no bracing, no bracing, no bracing). Each element and its neighbor to the left and right form a three-element local neighborhood. To form local neighborhoods for the elements at each end, the embryo is conceptually duplicated to the left and right of itself, as indicated by light-colored boxes *(upper left)*. The design rule then defines how each of the eight possible local neighborhoods should be translated into the elements of the next-higher story *(bottom right)*. For example, the first element of the design embryo is a 1, specifying an X bracing. The local neighborhood for this element contains no bracing at the left, an X bracing in the middle and no bracing at the right. The relevant part of the design rule dictates that the resulting first element in the second story should be an X bracing *(red bracket and arrow)*.

directly below and to the left and right of that location. If the cell below is on an end of the row, the value from the cell at the other end serves to fill the missing-neighbor slot. Thus the rule always applies to three cells in a row, which we call a local neighborhood.

With two bracing types (an X bracing or no bracing) and three cells, there are eight possible combinations for the local neighborhoods, corresponding to the eight binary numbers between 000 and 111. Our design rule is simply a key that tells the program whether there should be a bracing in the position above, based on the configuration of the local neighborhood below. Thus it requires one bit (0 or 1) for each of the eight possibilities.

To create a whole structure, the program works as follows: For the first floor, it simply puts X braces in the cells that correspond to the 1s in the design embryo. For the second floor, the program looks at each three-cell local neighborhood in the first floor and refers to the design rule to determine whether there should be a brace above. This process repeats until it creates a complete set of wind bracings for all floors in the building. Then the software analyzes the structure using the same engineering criteria as before. That is, it judges the building based on both its strength and weight.

Using a developmental scheme provides a more compact representation of the structure than does direct one-to-one mapping. For example, a direct mapping of a wind bracing subsystem that is 10 stories tall with four bracing slots per story would require 40 attributes (or genes), whereas the developmental representation uses only 12 (four for the design embryo and eight for the design rule). Compactness is a desirable property because real engineering structures typically have hundreds or even thousands of attributes.

Another advantage of these sorts of developmental processes is that they frequently produce unusual results. In particular, cellular automata are known to exhibit self-organizing behavior—that is, on its own, complexity just emerges. Indeed, our experiments confirmed this tendency and revealed intriguing and often quite surprising combinations of structural elements.

Piecing It Together

Engineers equipped with even very powerful computing tools cannot exhaustively evaluate all possible patterns. They need a way of focusing their searches for novel arrangements that potentially correspond to high-quality designs. One way to do so can be found by again looking to nature, which has used the combination of development and evolution to produce organisms that have enormous complexity and are capable of surviving in a changing environment. With this biological inspiration, our group implemented *Emergent Designer*, a computer program that allows us to simulate structures that can be developed from a design embryo, as described above, and that can also be evolved through crossovers and mutations, as we did earlier with *Inventor*. The new program, like the old one, al-

lows the entire genome to evolve. That is, the program modifies both the design embryo and the design rule over many generations.

Whereas the designs produced by *Inventor*'s evolutionary algorithms showed no obvious patterns, those *Emergent Designer* created exhibited patterns that are more creative, appearing as if they were formed by a human mind. Some reflected a mixture of random localized patterns embedded in a larger, more regular geometry.

Clearly, the creations of *Inventor* and *Emergent Designer* were very different in nature. But which were better? To answer that question, we made multiple runs of each type and performed a statistical analysis of the results. Such a procedure avoided the possibility that a "lucky run" of one program or the other might skew the comparison. Our conclusion was that, although both approaches work, *Emergent Designer's* evolutionary-developmental algorithm performs better in terms of producing the lightest configurations.

The use of *Emergent Designer* may thus help with a crucial balance that the designer of any tall building must strike: resistance to sway versus weight. In a traditional building, any reduction of wind-induced sway (a measure of a structure's stiffness) normally requires significant increase in the weight. One innovative method to keep weight down is the use of *macro-diagonals*, external cross-bracings that span large areas of a building, such as those that were built into the John Hancock Center in Chicago in the late 1960s and employed as recently as several years ago in the Bank of China Tower in Hong Kong. Engineers have long recognized that macro-diagonals redistribute internal forces in a way that reduces both sway and weight. So we were pleased to find that macro-diagonals spontaneously emerged in our experiments with an evolutionary-developmental algorithm. Some designs showed structural patterns very similar to ones that the late Fazlur Khan, one of the most creative designers of tall buildings, had arrived at (such as the scheme he used in the John Hancock Center). This resemblance shows the creative promise for bio-inspired computational approaches in engineering.

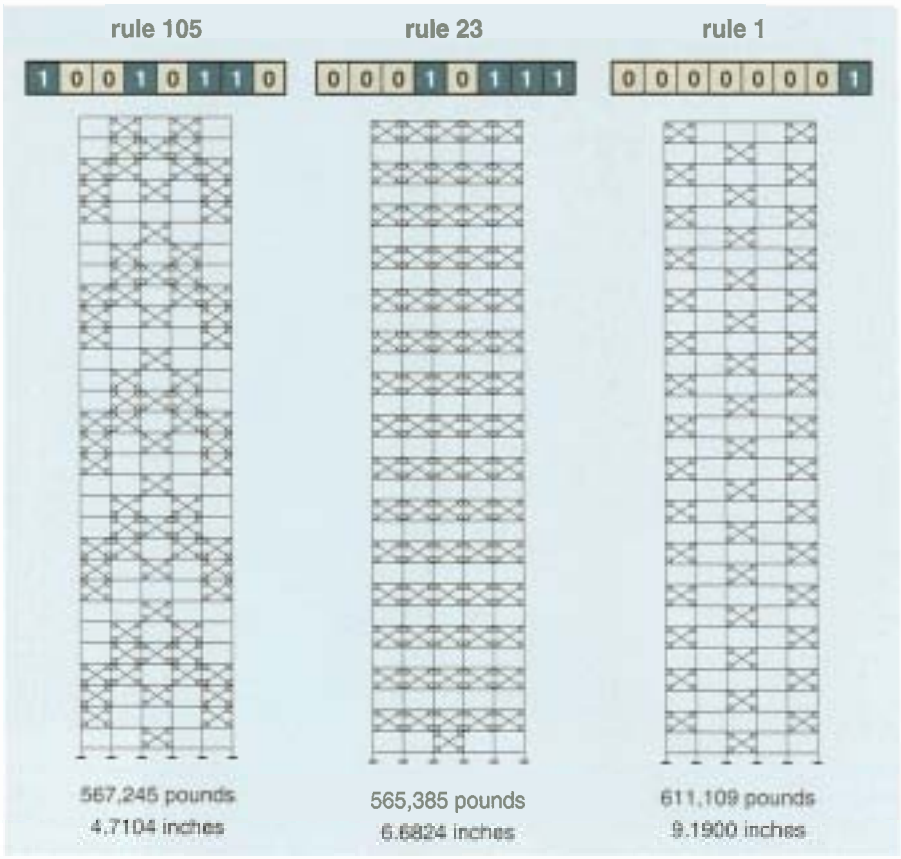


Figure 5. Some design rules and embryos produce better structures than others. The computer judges the results by their overall weight and by how far actual buildings with such skeletons would sway under standard weight and wind loads. These examples of rigid frames with bracings show the results of three different design rules. The left and middle designs are relatively good, with lower weights and less sway. Although the building on the right does not appear to be overly unstable, it has significantly higher amounts of both weight and sway. In the authors' simulated breeding process, structures such as this one would gradually "die out."

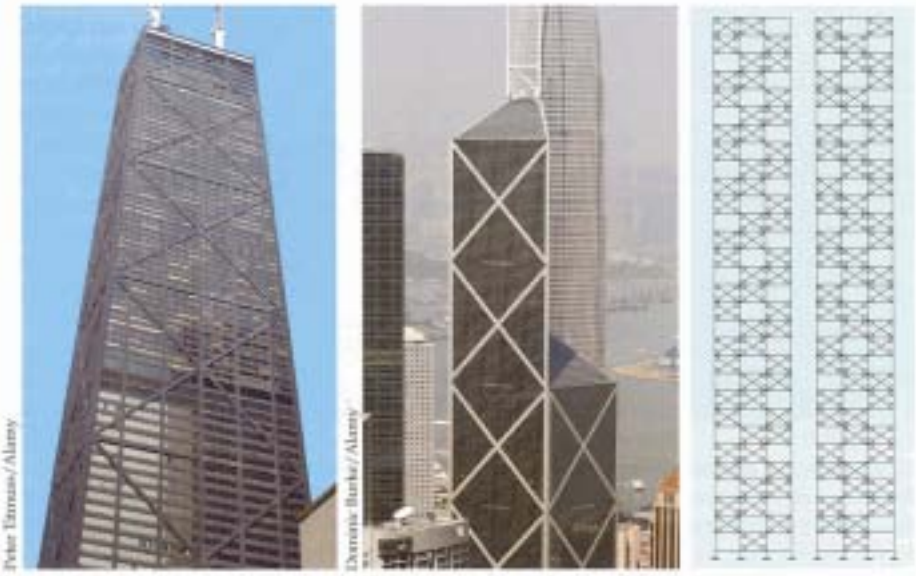


Figure 6. To reduce both the weight and the degrees of sway of a building, architects sometimes employ elements called macro-diagonals, external cross-bracings that span large areas of a structure. These elements were incorporated into the John Hancock Tower in Chicago in the late 1960s *(left)* and in the last few years in the Bank of China Tower in Hong Kong *(middle)*. The authors' program *Emergent Designer* has spontaneously created, over numerous generations, several optimized designs that show similar macro-diagonal patterns in the resulting structures *(right)*.

Capping It Off

Modern-day civilization advances at an ever-increasing pace, creating ever more difficult challenges, especially

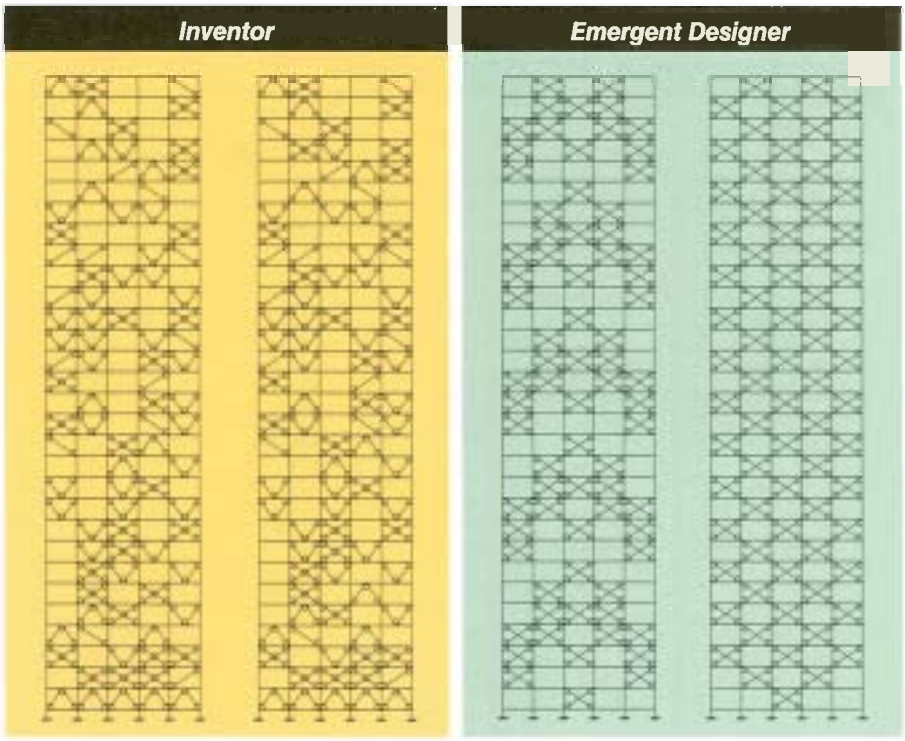


Figure 7. The buildings generated by the *Inventor* computer program (which uses only evolutionary mechanisms such as recombination and mutation) show no obvious patterns *(left)*. Those produced by *Emergent Designer* (which simulates both evolution and development—design embryos and design rules) show more apparent creativity, in some cases containing a mixture of random, local patterns embedded in a larger, more regular geometry *(right)*.

for engineers. We hope to convince our professional colleagues to look for ideas outside of engineering and to keep in mind that nature has always been one of the most fruitful sources of such insight.

Of course, a human designer must make the final decisions and actually create the structure. This person, however, must be well prepared for the challenge and have state-of-the-art tools at his or her disposal. The patterns created by simulation programs such as ours are valuable because they may provide engineers with ideas for novel configurations and shapes, ones that a person would never have arrived at from the application of systematic reasoning. Although these structural patterns may not always be feasible to build, they can illustrate new ways of approaching a problem. In this sense, computer algorithms that produce previously unknown patterns are helpful to creative engineers by providing them with inspiration, whether or not the specific designs are translated directly into beams, trusses and columns.

The software we have developed is currently limited in scope and complexity. It deals with highly idealized structures and lacks any consideration of the countless details that go into the design of a real building. But like the virtual populations subjected to our genetic algorithms, we expect that these programs themselves will continue to evolve. They will, we hope, be infused with beneficial traits from other areas of engineering, computer science and biology, and over time they will surely be subjected to natural selection. So we are excited to see what the next generation of genetic algorithms for building design will be able to accomplish.

Acknowledgments

The authors wish to thank Kenneth DeJong, a professor of computer science at George Mason University, and Krzysztof Murawski, a professor of computer science at the Military University of Technology in Warsaw, for contributions to the design and implementation of software in our reported research.

References

Arciszewski, T., and J. Cornell. 2006. Bio-inspiration: Learning creative design principia. In *Intelligent Computing in Engineering and Architecture*, ed. I. F. C. Smith. Berlin/Heidelberg: Springer.

Arciszewski, T., and R. Kicinger. 2005. Structural design inspired by nature. In *Innovation in Civil and Structural Engineering Computing*, ed. B. H. V. Topping. Stirling, Scotland: Saxe-Coburg Publications.

Bentley, P. J., and D. W. Corne. 2002. *Creative Evolutionary Systems*. San Francisco: Morgan Kaufmann Publishers.

Burks, A. W. 1970. *Essays on Cellular Automata*. Urbana, Ill.: University of Illinois Press.

Coello, C. A., M. Rudnick and A. D. Christiansen. 1994. Using genetic algorithms for optimal design of trusses. *Sixth International Conference on Tools with Artificial Intelligence (ICTAI '94)*, New Orleans, 88–94.

Fogel, L. J., A. J. Owens and M. J. Walsh. 1966. *Artificial Intelligence Through Simulated Evolution*. Chichester, U.K.: John Wiley.

Goldberg, D. E. 2002. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Boston: Kluwer Academic Publishers.

Goldberg, D. E., and M. Samtani. 1986. Engineering optimization via genetic algorithm. *Ninth Conference on Electronic Computation*. Birmingham, Al.: University of Alabama Press.

Grierson, D. E., and S. Khajehpour. 2002. Method for conceptual design applied to office buildings. *Journal of Computing in Civil Engineering* 16(2).

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Mich.: University of Michigan Press.

Kicinger, R., T. Arciszewski and K. A. De Jong. 2005. Emergent Designer: An integrated research and design support tool based on models of complex systems. *International Journal of Information Technology in Construction* 10:329–347.

Kicinger, R., T. Arciszewski and K. A. De Jong. 2005. Evolutionary computation and structural design: A survey of the state of the art. *Computers & Structures* 83:23–24.

Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Mass.: MIT Press.

Kumar, S., and P. J. Bentley. 2003. *On Growth, Form and Computers*. London: Academic Press.

Murawski, K., T. Arciszewski and K. A. De Jong. 2001. Evolutionary computation in structural design. *Engineering with Computers* 16(3-4):275–286.

Parmee, I. C. 1998. Evolutionary computing for conceptual and detailed design. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advances and Industrial Applications*, eds. D. Quagliarella, J. Periaux, C. Poloni and G. Winter. Chichester, U.K.: John Wiley & Sons.

Wolfram, S. 1983. Statistical mechanics of cellular automata. *Reviews of Modern Physics* 55:601–644.

Wolpert, L. 2002. *Principles of Development*. New York: Oxford University Press.

For relevant Web links, consult this issue of *American Scientist Online:*

http://www.americanscientist.org/Issue TOC/issue/1021