# Evolving Towers in a 3-Dimensional Simulated Environment

**Gary B. Parker, Andrey S. Anev, and Dejan Duzevik**
Computer Science
Connecticut College
New London, CT 06320
{parker, asane, dduz}@conncoll.edu

**Abstract -- This paper describes a system that uses evolutionary computation to evolve tower-like structures. The construction takes place in a computer simulated gravitational environment. The evolution targets the morphology; each chromosome carries structural description of the entity. Fitness functions evaluate the structural integrity and "goodness" of each individual based on indicators such as joint tension, center of gravity, position in space, height, etc. Twelve evolution-tests were performed and all successfully reached tower solutions.**

## 1 Introduction

Extensive research in artificial intelligence in the field of robotics has been conducted on the development of more efficient ways to evolve the cognitive processes of robots with predefined morphologies. The theoretical rationale behind these approaches is that by evolving the mind of a robot to perform a certain task, human programming is reduced, while a near optimal result is achieved by using some form of evolutionary computation. However, confining the mind to a predetermined body creates unintended constraints for the efficiency of a robot. Recently, attempts to simultaneously co-evolve the mind and the body have been conducted.

O'Reilly made significant advances in incorporating evolutionary-strategy modules in CAD systems. The Generative Genetic Explorer targets partial transformations of pre-designed frames to facilitate architectural aesthetics. This is achieved without an embedded fitness function but relying on the feedback of the architect [orei98]. The Emergent Design Group has created The Rule Genetic Programmer (RGP) that comes with solutions for high-density structures to be placed within an existing urban area [orei00].

Research led by Jordan Pollack at Brandeis University has shown that evolving static structures is possible using basic laws of physics and no human influence regarding physical characteristics of the created objects [fune98, fune99, poll98]. A group of researchers led by Henrik Lund used pre-designed modules to develop a system that creates dynamic LEGO MINDSTORMS robots [lund97]. The system is initially equipped with structures that combine wheels, axles, and motors. The evolution of different structures uses these modules later to develop the best design for a given task.

Our research follows the logic of these approaches, but is to evolve locomotion without the limitations of predetermined modules and uses a general chromosome structure that allows for the use of multiple types of joints crucial for evolving complex structures. This also gives the system freedom to perform various crossover and mutation operations. In addition, our work utilizes an uninhibited evolution, free from predetermined modules. Karl Sims created a virtual environment to simultaneously evolve the morphology and controllers of creatures [sims94]. He achieved significant virtual results, whose real construction would be difficult. We use LEGO Mindstorms set to construct real structures because of their simplicity, availability, and affordability. At this phase of our research we use only the basic bricks. However, the various other pieces included in the set such as motors, wheels, axles, cogwheels and connectors provide ground for future advance in the research of evolvable morphologies. The properties of the RCX enable downloading controls on the robot itself, which creates an autonomous entity. We believe that even though pre-constructed modules speed up evolution, they inhibit optimal results. Thus, we undertake an atomic approach where the smallest particles are pieces whose function is not clearly defined until they are placed in the context of the entire unit. A list of frames that describe all available pieces (piece pool) allows easy construction of the output.

Our research has been separated into three stages: evolution of the morphology, evolution of the controller, and the combination of the two. The morphological evolution is separated into two phases: creation of stable structures and the evolution of movable robots. The goal of the first phase is development of a program that generates cohesive and stable structures according to a given set of objectives in terms of height, length, size or weight. This paper reports on the first phase of morphological evolution to obtain stable structures. In the second phase, which is still in progress, robots with locomotion will be developed to perform a specific task.

The system has to use pieces with different dimensions, shapes, functions and connections in order to have high degrees of freedom in the creation of robots capable of performing various tasks. Constraints are placed on how the bricks connect as well as on the number of parts the system uses to reflect a real world situation. These factors presented many challenges in terms of building the structures and implementing their crossover in a virtual environment governed by the laws of physics.

## 2 Environment Simulation

The need for generating numerous generations, each containing a population of 70 individuals or more, made the possibility of manually building each product and using the resultant fitness as feedback to the system too time consuming to be feasible. Therefore, a simulation environment was created that reflects the constraints that gravity has on the objects in our environment.

The program is composed of five modules (Figure 1): Builder Module, Physics Module, Stability Module, Fitness Evaluation, and Reproduction.

The Physics Module evaluates if the structures created in the Builder will break due to gravity. If that is the case the structure is returned to the Builder with the point of fracture. The Builder deletes the fractured part and sends the chromosome for reevaluation to the Physics Module. The process is reiterated until a rigid structure is obtained. The Stability Module determines whether a structure can stay on its base. Negative results lead to termination of the individual. Otherwise, its fitness is determined and the structure is included in the pool for reproduction.

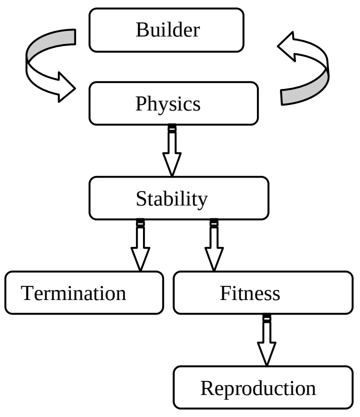


Figure 1: A diagram of the System's Modules and how they interact.

### 2.1 Brick Representation

The LEGO MindStorms set used contains 139 different parts. Each structurally different piece has a frame that acts as a physical representation of the brick. It specifies spatial dimensions, weight, and joint types and their positions on the piece. Each frame plays the role of abstract knowledge; once used in a structure, it is altered to represent reality (only free joints are left and their positions are updated according to the coordinates of the piece in the builder). The system, however, can always reference the frame to retrieve the original properties of the brick.

### 2.2 Builder Module

The Builder Module is responsible for creating individuals by connecting pieces from the piece pool in the virtual building space. The building space is a three-dimensional pixel matrix, providing for computationally efficient bonding and intersection detection. Each pixel is labeled according to its current state: *free*, *solid*, (*joint-type piece-reference*) or (*point-of-connection piece-reference*$_1$ *piece-reference*$_2$). As a piece is added the relevant pixels are checked and if a *solid* or (*point-of-connection piece-reference*$_1$ *piece-reference*$_2$) result is obtained the placement is rejected. In the case of a free joint the new piece is checked for availability of a complementary joint at this coordinate and if the piece is placed, all connections are reflected in the builder and also in the chromosome of the unit. The builder is restricted to a finite piece-pool (for instance a single LEGO set) that determines the types and number of pieces available, consequently limiting the yielded structures to be comprised solely of pieces available to the user. The products of the Builder Module are structures of interconnected pieces.

| Support | Supported |
|---|---|
| 3A | 4A |
| 4A | 2A |
| 2A | 1A |
| 3A | 1A |
| 1A | 2A |
| 2A | 4A |

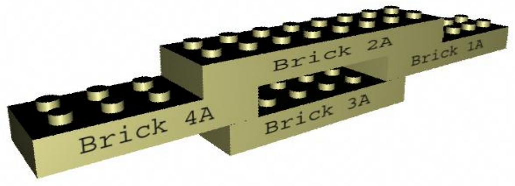


Figure 2: A diagram of a structure and the table produced to hold the support links.

### 2.3 Physics Module

The Physics Module tests the overall integrity of the created structure. It distributes torques that pieces in a structure exert on one another and evaluates the cohesion of the structure. Pollack used a greedy generalized network flow algorithm [dega95] to evaluate the stability of a structure. Our system uses an algorithm to traverse the structure and marks every piece either as support or as supported in a specific connection.

Determining which brick supports which can be straightforward only in the case when there are no loops formed by the pieces. Figure 2 gives an example of how the algorithm determines which brick is a support in a loop situation. The algorithm starts at brick 3A because it is the only connection to the ground and traverses either through brick 4A or 1A. Assume it starts with 4A. It looks at 3A and 4A and determines 3A as the support because it is on the ground. Then it observes 4A and 2A and flags 4A as support of 2A. At the next step it marks 2A as support of 1A. The algorithm stops at the comparison between 1A and 3A because it knows that this is a point of loop. Then it goes back to the last unexplored link and looks at 3A and 1A noting that 3A is the support. The process is continued until no unexplored links are left and we obtain the results in Figure 2.

This reflects the fact that 4A simultaneously supports 2A and is supported by it. However, 3A is not supported by any of the bricks because it is the place where the loop started and is on the ground. The algorithm works with more complex situations that may occur and outputs a table of the support-supported relation used to determine the force distribution.

The force that a single piece exerts is spread according to its supports, and is dispersed all the way to the ground. The force may be decreasing the strength of the joint or increasing it when it balances another one. The possibility of fracture exists when the sum of forces applied by the weights of supported pieces prevails over the sum of the binding capacities of the knobs on the pieces that supports it. The initial capacities of the knobs and the weights of all pieces were derived experimentally. The final capacities of all joints are calculated for each individual by following a network of support links that ends with the ground. Fractures do not exist in the structure if the capacities of all joints are larger than zero after the torques throughout the system have been applied to all pieces. In a case of a structural fissure, the broken part is removed and the remaining structure is reevaluated.

### 2.4 Stability Module

The system evaluates the stability as defined by the ground supports of a structure using the Stability Function. In contrast with the Physics Module, which assesses the overall integrity of the structure, the Stability Module is concerned only with the issue of balance. An algorithm finds the outer edges of the supports on the ground. The distance between a projection of the structure's center of gravity (COG) and the peripheral lines formed by the edges of pieces located on the ground level is used to determine the degree of stability of a given individual. The further the COG is from the support edges (toward the center), the more stable the structure, and hence the better the fitness score. If the COG falls out of the boundary of the ground supports, the structure is determined to be unstable. The centralization of the COG is inversely adjusted for the height; higher structures call for better ground stability. This is done to support future research when we have moveable objects. This will cover situations where the robot is on a slope, reducing the possibility that the projection of the COG will fall out of the support figure.

## 3 Learning

To find an optimal solution for a structure in a given environment evolutionary computation (a genetic algorithm [holl75]) was used. It would preserve and promote beneficial traits in the individuals and guarantee that the system be capable of overcoming local peaks in its search for solutions.

On this level of our research we target the accomplishment of a single task – to create a tower-like structure. A population of 70 individuals was used. Preliminary tests using a larger population size did not yield better results and increased the computational time.

### 3.1 Chromosome Structure

The essence of our research is to create a system capable of generating task specific entities from a pool of various pieces. The genetic module of the system had to be general to cope with various tasks and provide a good interface for future implementations.

Similar to the approaches described by de Garis, I. Kajitani, T. Hoshino, M. Iwata, and T. Higuchi the system has a variable chromosome length, which does not constrain the actual size of the final product [orei00, fune99]. A vector that represents the chromosome contains all the pieces that comprise a structure. The genotype of each entity clearly describes the phenotype by providing the comprising pieces: showing their type, positioning in space, and physical properties and displaying the interconnectivity of the pieces. The form is shown in Figure 3. An alternative approach would have included only the type and positioning in space, leading to higher computational demand and thus making the system less efficient.

```
(piece-name (space-coordinates piece-orientation)
   (free-joints
      (joint-type (coordinates) (coordinates) …)
      (joint-type (coordinates) …))
      …
   )
   (index-of-piece-connected-to
      (joint-coordinates) (joint-coordinates) …)
   (index-of-piece-connected-to
      (joint-coordinates) (joint-coordinates) …)
      …
)
```

Figure 3: The chromosome is made up of a list of pieces in the structure. This is an example chromosome in Scheme representation.

An initial population is randomly created. An initial piece is placed at the center of the building space. Random pieces are picked from the piece pool and an arbitrary orientation and connection are chosen. If the binding is unsuccessful the piece is returned to the piece pool and the process is repeated once again. Initially all individuals have the same genetic size. During a mating process the size can vary yielding smaller and bigger structures.

### 3.2 Fitness Evaluation

In the first phase of the research, the tasks of the morphological evolution require solid structures, such as towers, tables, bridges and arches. Thus, in this stage of research the fitness evaluation is designed to utilize stability as the most important characteristic of an individual. However, besides stability, the fitness function also uses tension optimality, height and weight.

The stability fitness determinant uses equation 1 after communicating with the Stability Module, obtaining the center of gravity (COG) and the stability factor. The first part of this equation is assigning higher stability fitness to structures with wide bases that centrally support the center of gravity (*a* is a constant used to bias this part's importance). The variable $x$ is the smallest distance from the ground projection of the center of gravity to a support side; $\bar{x}$ is the mean of all distances of COG to the support sides. The second part of the equation is reducing the stability fitness depending on how high the center of gravity is (*b* is a constant used to bias this part's importance). The variable *cogheight* is the height of the center of gravity from the ground.

$$a\left(\frac{x}{\bar{x}}\right)x - b \times cogheight \qquad \textbf{(1)}$$

The formula used accounts for proportional symmetry (Figure 4); bricks with identical form but different dimensions will get the same score ( $x$ / $\bar{x}$ ). However, in reality the larger support (support 1) will prove to be more stable than the second one. Therefore, we multiply by the $x$ distance and get a higher value. The first part of the function tends to place the COG in the center of the support. The height of the COG is subtracted to account for the fact that tall or top-heavy structures are less stable.

The tension optimality function looks for structures with less tension between the pieces in order to yield stronger structures. It sums the potentials of additional stress that the joints can hold, so that the structures with a higher factor are assigned a better fitness.

The height fitness function promotes higher structures, and as such it leads to construction of the tower-like structures. At this level the weight function is used to promote economical creation of structures instead of exploration of the entire piece pool.

Each individual's fitness is determined and the elite (best individuals) of the population are selected for reproduction. The percentage selected can vary; we tried 30%, 40%, and 50% (when the mutation and crossover levels are high, the smaller elite groups yielded better results). Individuals are randomly selected from the elite to mate. Once the mating process is complete the two structures are returned back to the elite population preserving the possibility of being selected for mating once again. The offspring created offset the loss in the previous population to bring it back to its original size.

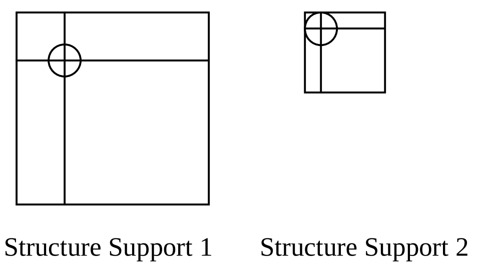
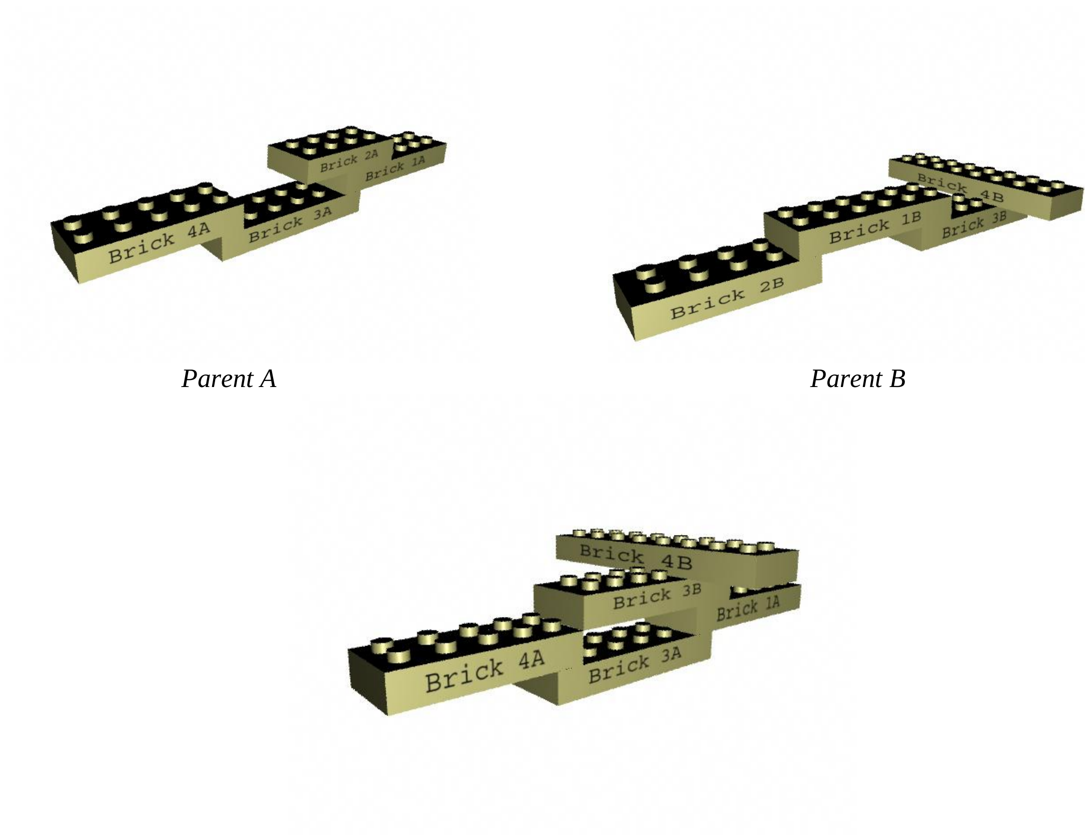


Figure 4: Example of supports for two distinct structures. The center of gravity of the structure is shown by a circle.

### 3.3 Crossover

The Crossover (Figure 5) is responsible for the reproduction of individuals. Each piece of an individual's structure is a gene of its chromosome. At the initialization phase the size of the offspring chromosome is set, but during evolution the generic algorithm creates individuals in a variety of sizes. The fitness of the two parents is obtained and the size of the offspring is biased to be closer to the one with the greater fitness. An element of randomness is introduced that can yield structures with a larger or smaller chromosome than either parent but not larger than their combined chromosome length.

Two modes of the crossover process are involved in the reproduction stage. The first one is the link follower where one of the individuals provides the genetic material. The first piece is always the one with index zero, and for future research this piece will always be the RCX. A parameter determines the bias of the next piece selection. This bias parameter varies from -1 to 1 where -1 mimics a breath-first search, 0 complete randomness and 1 depth-first search. A breath-first search yields stacked structures, while depth-first produces elongated ones. At this level the bias was set to zero and is left for future use when the system will be determining the level on its own. The link follower is interrupted at a random point (crossover point) to which a randomly chosen piece from parent B (initial piece) is attached. Next, the system tries to attach pieces that had been linked to the initial piece (from B). If pieces from parent A are in the way of the pieces to be attached from parent B, mutation will be invoked and the piece may change enough to fit. If mutation is unable to come up with a solution, the system looks for free connection points, links specified in the parent's chromosome. If none are left, forced crossover occurs and pieces from parent A are used again. When forced crossover occurs the system starts using genetic material from the other parent and connects it to a point specified by the bias. If the bias is equal to zero it will pick a random connection position. The products of the Crossover are individuals that resemble characteristics of both parents.



*Offspring*

Figure 5: Example of crossover. The top two figures portray two individuals: A and B. In the initialization of the reproduction process a size of five is chosen and a crossing point after the third element. A starts giving genetic material through the link follower: pieces 1A 3A and 4A. Crossover occurs and B gives piece 3B that takes the place of 2A. The link follower then provides 4B. The size of five is reached and the reproduction finishes.

### 3.4 Mutation

Mutation can be invoked randomly or by force. It is applied by force when an attempt to place a piece was unsuccessful and the system tries to overcome the problem with a small change instead of resorting to forced crossover. Initially the system sets a proximity table that indicates how close in structure different pieces are. This is determined by the parameters of the piece: length, width, height, and number of knobs. Even though selection of the mutated piece is random, those that are closer in structure to the original piece have higher probability of being chosen. Three types of mutation can occur. Structural mutation is invoked when the genetic operator is attempting to place a piece that is not in the piece pool (all of those pieces have been used). Mutation accesses its proximity tables and based on similarity of the pieces and availability in the piece pool returns a piece. The second type is *limited positional* and leads to small, random changes in a single piece's position and orientation. The *general positional* mutation exerts a domino effect on all subsequent pieces by shifting them according to the shift in the mutated piece, creating a global impact on the structure.

## 4 Results

We show the results of the 4 trials done with a population of 70 individuals to demonstrate the progression of learning in Figure 6. The mutation level was set to 10% but in fact it would vary due to forced-mutation. The bias in selecting the next piece (determining if the system would undertake a breath-first or depth-first approach) during crossover was set to 0, which yields random choices. An elite group of 30% of the population (21 individuals) was chosen to reproduce and persist in the next generation. Each line in Figure 6 shows the improvement of the top fitness over a total of three hundred generations. In all cases, there was fast initial improvement that decreased as the learning got to the higher generations.

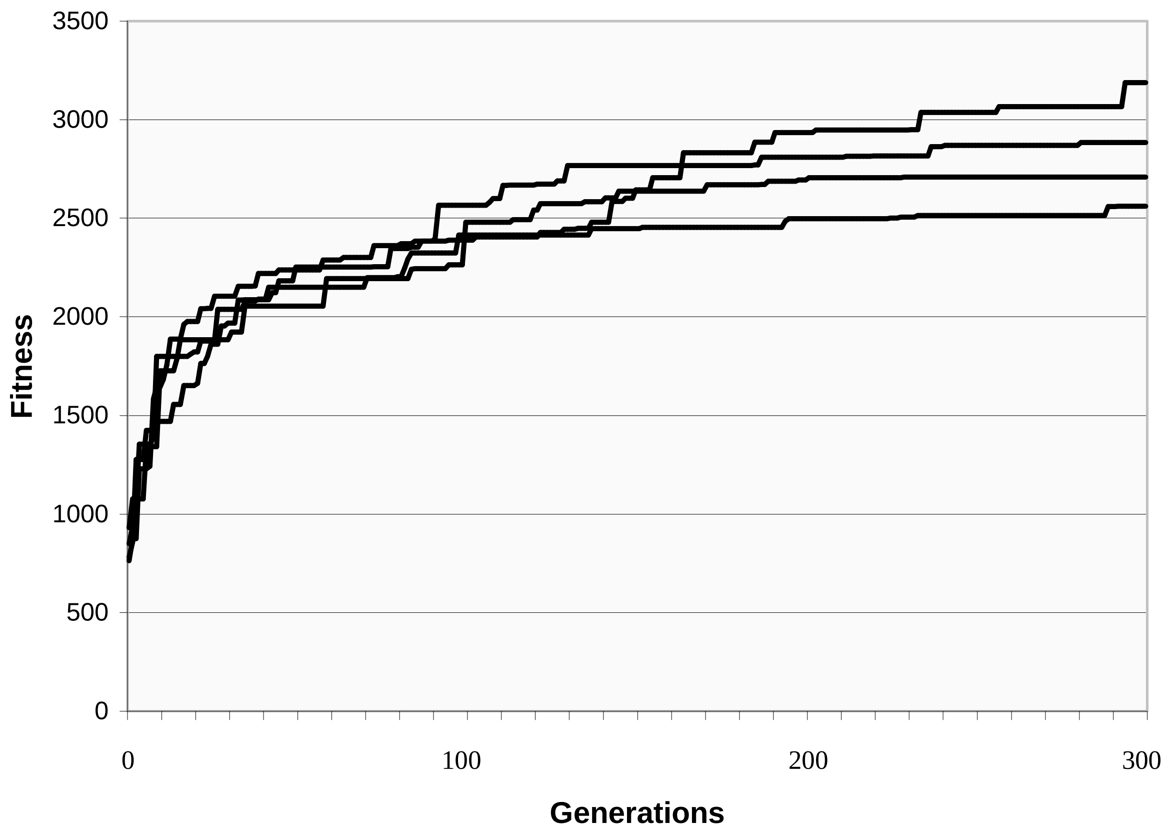
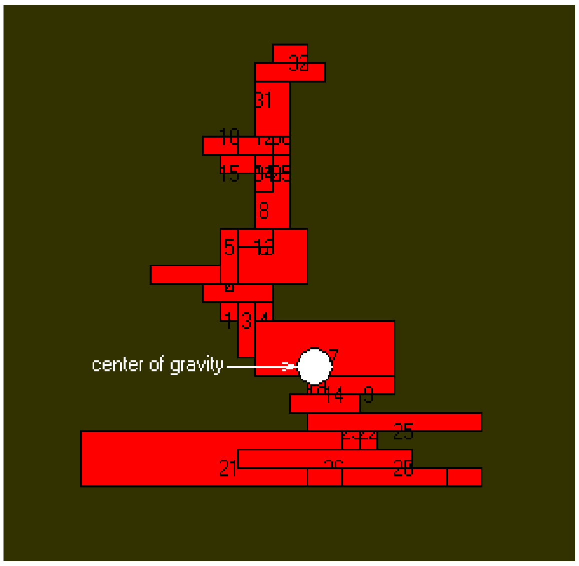


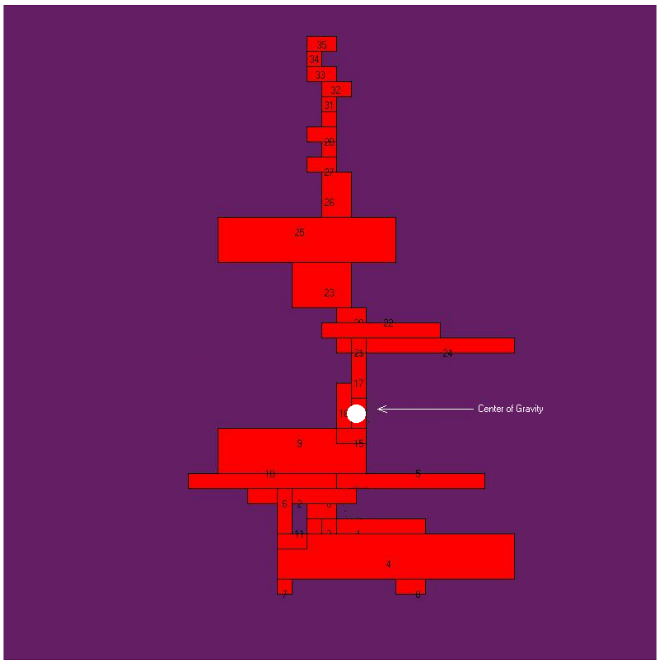
Figure 6: Graph of the evolutionary process of four tests using population sizes of 70 individuals, mutation rate of 10%, and 30% elite selection. They portray rapid initial increase in fitness that slows down as the generation count increases.

The goal of our tests was evolution of stable tower-like structures. Because of the narrow definition of our objective, the fitness function was evaluated using only two properties: the height of the structure and its stability level. Better stability and greater height were, therefore, key factors to raise the fitness of an individual. The limited piece pool results in an inverse relation between the stability and the height of evolved structures; to enhance stability the system uses more pieces on the ground and the lower levels, and less pieces are available to reach a greater height. Similarly, utilizing more pieces towards the height diminishes the stability of the individual. Our evolution provided solutions by balancing both properties.

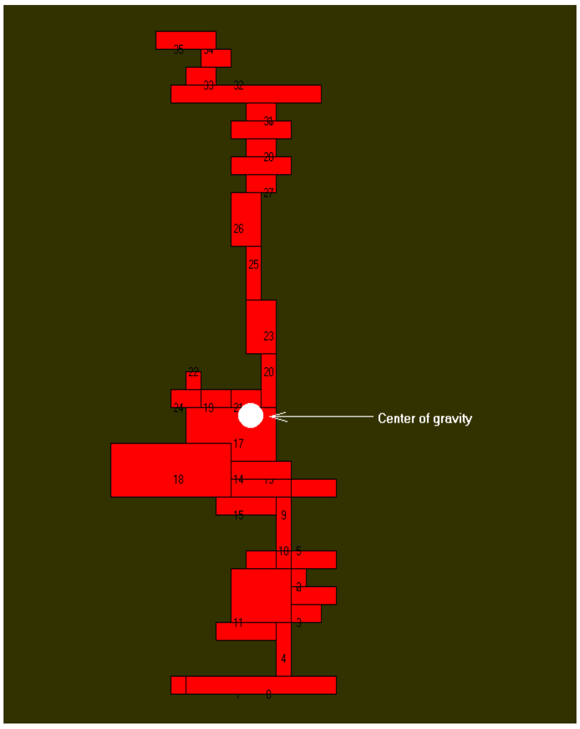
In the early stage of the evolution the system tends to maximize the individual's fitness by diminishing the tension between the parts through multiple connections and to expand on the base (stability). Further in the evolution it discovers that it can be more economical and at the same time preserve small degree of tension by placing bricks one on top of the other. In this way it creates stable and high structures. This can be seen in Figure 7 where we have a stack structure in an example of a typical 70th generation that maximizes on solidity. As the number of generations increases the systems discover that height yields better fit. The example of a typical 300th generation solution shows that the learning system preserved the longest and widest piece from the piece pool on the ground and has found better utility for the rest to contribute for the height of the structure. This can also been seen by observing the footprint of two example structures in Figure 8. Evolution utilized all pieces from the piece pool in the construction of the tower-like structures.



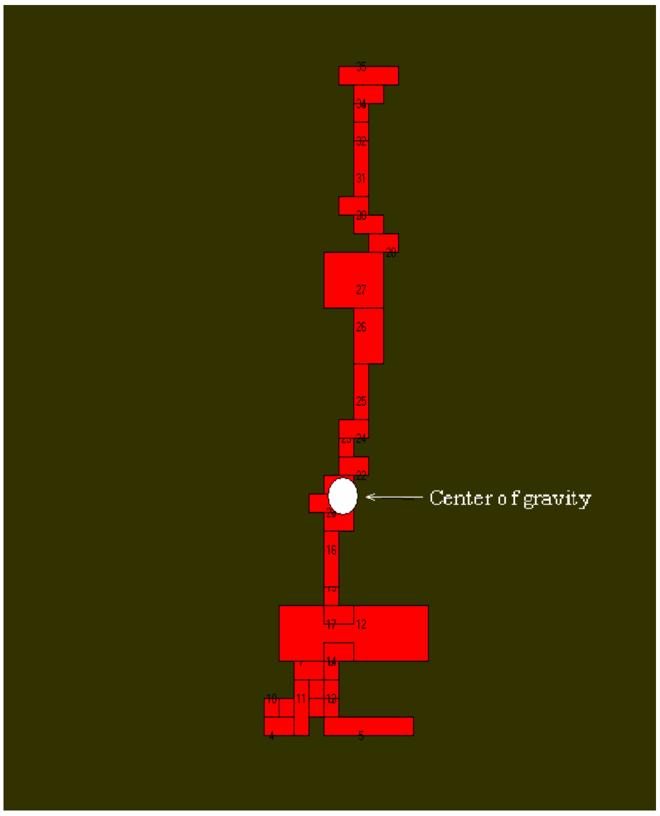
*70 generations*



*150 generations*

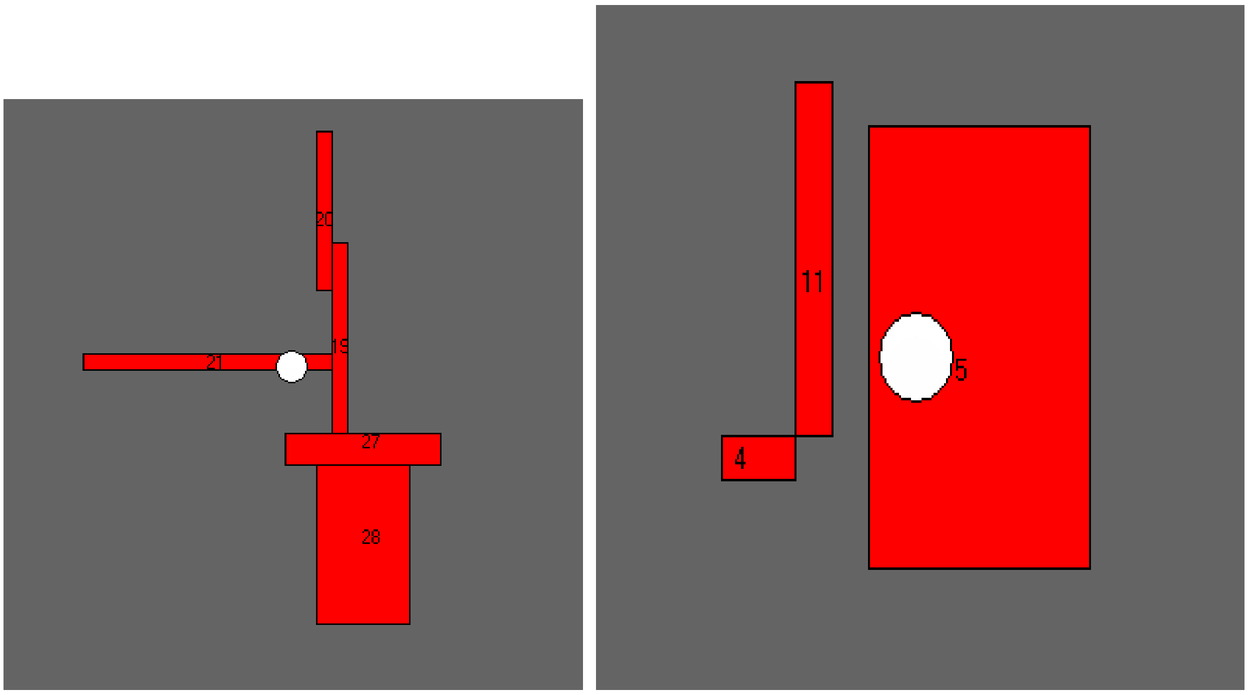


*200 generations*



*300 generations*

Figure 7: A two dimensional representation of four structures produced at different numbers of generations. The ones with higher generation level optimize height, in contrast with those on a small evolutionary step which gained better fitness among the other individuals at that generation by expanding stability.



*base at 70 generations* *base at 300 generations*

Figure 8: This figure shows the footprint of the 70 and 300 generations structures from Figure 7.

## 5 Conclusions

Our system of evolving tower-like structures in a computer simulated gravitational environment was a success. The towers that we evolved show structural integrity and stability. This work completes the first phase of stage one of our co-evolving morphology/control research – the creation of a stable structure. The next step of our research involves the addition of pieces such as axles, wheels, motors, etc., to enable the system to evolve movable structures that will be capable of performing specified tasks.

## References

[dega95] de Garis, H. "Evolvable hardware: Genetic programming of a Darwin machine." *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms.* (1995) 441-449.

[fune98] Funes, P. and Pollack, J. "Evolutionary Body Building: Adaptive physical designs for robots." *Artificial Life* 4 (1998) 337-357.

[fune99] Funes, P. and Pollack, J. "Computer Evolution of Buildable Objects." *Evolutionary Design by Computers.* (1999) 387-403.

[holl75] Holland, J. H. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI (1975).

[lund97] Lund, H. H. et al. "Evolving Robot Morphology." *Proceedings of IEEE Fourth International Conference on Evolutionary Computation.* (1997).

[orei98] O'Reilly U.M. and Ramachandran G. "A preliminary investigation of evolution as a form design strategy." *Artificial Life IV: Proceedings of the Sixth International Conference on Artificial Life.* (1998).

[orei00] O'Reilly U.M. and Testa P. "Representation in Architectural Design Tools". *Proceedings of ACDM-2000.* (2000).

[poll98] Pollack, J. and Funes, P. "Evolutionary Body Building: Adaptive Physical Design for Robots." *Artificial Life* (1998).

[sims94] Sims, K. "Evolving 3D Morphology and Behavior by Competition." *Artificial Life IV Proceedings.* (1994) 28-39.