EvoCAD: Evolution-Assisted Design*

Pablo Funes, Louis Lapat and Jordan B. Pollack

Brandeis University Department of Computer Science
415 South St., Waltham MA 02454 USA

Since 1996 we have been conducting research on evolutionary design using Lego[1] bricks. Our simulations are designed with buildability in mind: combining a simple evolutionary algorithm with a partial simulation of Lego bricks statics we evolve designs that come out of the computer ready to build [1-4].

Today's commercial CAD systems may add a mechanical simulator to the usual 3D manipulation tools[2]. But the new field of Evolutionary Design (ED) [5] has the potential to add a third leg to computer-aided design: A creative role. Not only designs can be drawn (as in CAD), or drawn and simulated (as in CAD+simulation), but also designed by the computer following guidelines given by the operator. Thus we envision future Evolutionary CAD systems, "EvoCADs".

An EvoCAD system has the human designer in the main role: the designer has an idea or concept for a required object. Some of the requirements can be added to the 3D canvas, creating evolutionary targets than an ED engine uses for evolving a possible design. The output of this evolutionary engine can be modified, tested and re-evolved as many times as desired (figure 1).

Methods

To demonstrate our conception of EvoCAD, we have built amini-CAD system to design 2D Lego structures. This Lego EvoCAD allows the user to manipulate Lego structures, and test their gravitational resistance using the same structural simulator we have been using in the past to do ED with Lego bricks. It also interfaces to an evolutionary algorithm that combines user-defined goals with simulation to evolve candidate solutions for the design problems. The results of evolution are sent back

*Appeared in: A*rtificial Intelligence in Design 00 Poster Abstracts.* Key Centre of Design Computing and Cognition, University of Sydney. Pages 21-24.

[1]Lego is a trademark of The Lego Group

[2]PTC's Pro/Engineer software, whose CAD tool can generate output for the mechanical simulator, Pro/Mechanica, is an example.

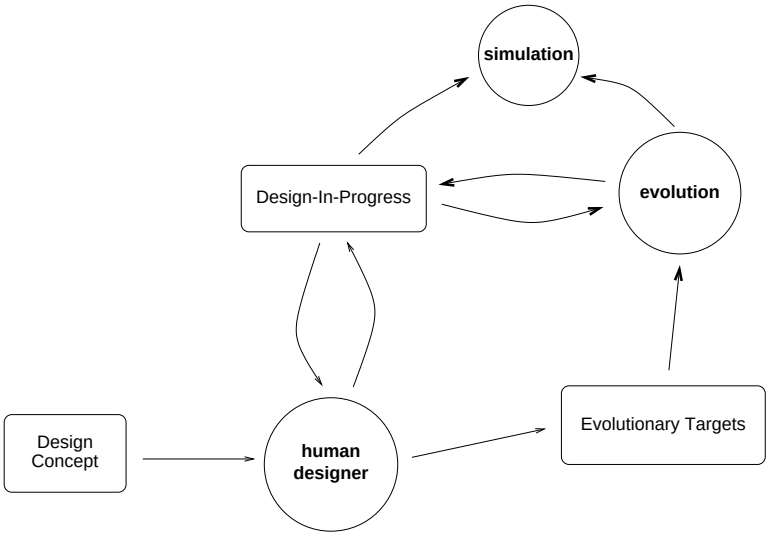

Figure 1: A conceptual EvoCAD system has two "creative minds", the human designer and the ED software. The human designer is in control, and calls upon the remaining elements as tools. A problem description language (PDL) allows CAD, evolutionary and simulation components to communicate with each other (bold arrows).

to the CAD front-end to allow for further re-design until a satisfactory solution is obtained.

Simulation and Representation

The simulation algorithm [6] is based upon the concept of maximum torque capacity at brick-brick joints. A Lego structure with loads becomes a torque flow network where each edge has a maximum capacity. A network flow algorithm accepts a structure only if there is a valid global flow without any overstressed joints. To allow for recombination and crossover, Lego structures are represented as Lisp-like functions [2] that have one or more "root" bricks with zero or more descendants attached at diverse points. Mutations can modify a brick's size or position, whereas recombinations allow for interchange of entire subparts between two parents.

Evolutionary Algorithm

To begin an evolutionary run, a starting structure is first received, consisting of one or more bricks, and "reverse-compiled" into a genetic representation that will seed the population. Mutation and crossover operators are applied iteratively to grow and evolve a population of structures [2]. The simulator is run on each new structure to test for stability and load support, needed to calculate a fitness value. Evolution stops when all objectives are satisfied or when a timeout occurs.

Brick Problem Description Language

We designed a "brick problem description language" (BDL), as an interface between the evolutionary algorithm, simulator, and the CAD front-end. When the user clicks the "evolve" button, a BDL description is sent over the internet to our evolution server which evolves a solution for the problem. The result of the evolution is sent back to the CAD using the same language. The simulator receives BDL-encoded structures for testing, both from the CAD (when the human wants to test a structure) and from the evolutionary engine, which tests every mutated or recombined structure.

Target Points and Target Loads

The goals for the ED engine are deducted from user-defined *Restrictions, Target Points* and *Target Loads*. A structure will be fully satisfactory if it touches all the target points and target load points, whereas avoiding all the restricted points, and supports all the specified loads at each target load point.

Fitness Function

Instead of writing code in some computer language for a fitness function, here we need to compute a generic fitness value for any BDL structure. Although optimized fitness functions will require further study, the fitness of a structure S has been defined as:

$$\sum_{t \in \text{targets}} \frac{1}{1+d(S,t)} + \sum_{l \in \text{loads}} \frac{1}{1+d(S,l)} + \sum_{\substack{l \in \text{ loads} \\ d(S,l)=0}} \text{supp}(S,l)$$

(where d computes the distance between a point and the nearest brick in the structure, and supp uses the simulator to compute the fraction of a certain load that the structure supports)

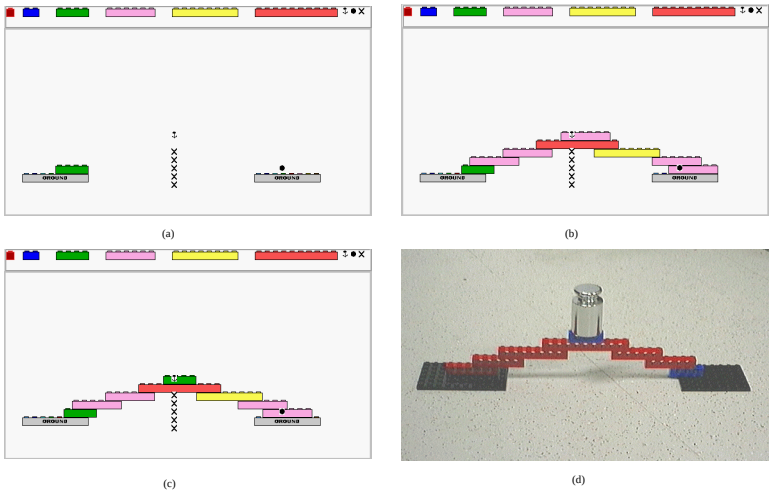

Figure 2: Sample working session with the EvoCAD program: (a) The user has defined two grounds and several evolutionary hints: restrictions (x), target (dot) and load (arrow). An initial brick was laid down. (b) Evolution designed a structure that fulfills all requirements. (c) The user made cosmetic corrections (d) The structure has been built with Lego bricks.

Results

Our simplistic Lego EvoCAD system (figure 2) demonstrates how this new kind of application could employ ED techniques to let the computer no only be a canvas and a simulation tool, but also to create its own designs following the users' specifications. Our system allows human and computer to create a design collaboratively, greatly reducing the human effort needed to create and optimize a design.

References

[1] Funes P. and Pollack. J. B. (1997), Computer evolution of buildable objects. in P. Husbands and I. Harvey, (eds), *Fourth European Conference on Artificial Life*, MIT Press, Cambridge, pp. 358-367.

[2] Funes P. and Pollack. J. B. (1998), Evolutionary body building: Adaptive physical designs for robots. *Artificial Life* **4**(4):337-357,

[3] Funes P. and Pollack. J. B. (1999), Computer evolution of buildable objects in P. Bentley (ed.), *Evolutionary Design by Computers*, Morgan-Kaufmann, San Francisco, pp. 387-403.

[4] Pollack, J. B., Lipson, H., Funes, P., Ficici, S. G. and Hornby. G. (1999), Coevolutionary robotics, in Koza, et. al. (eds), T*he First NASA/DoD Workshop on Evolvable Hardware*. IEEE Press.

[5] Bentley, P. (1999) (ed.), *Evolutionary Design by Computers.* Morgan-Kaufmann, San Francisco.

[6] Funes, P. and Pollack. J. B. (1998), *Componential structural simulator,* Technical Report CS-98-198, Department of Computer Science, Brandeis University.