EMERGENT DESIGNER: AN INTEGRATED RESEARCH AND DESIGN SUPPORT TOOL BASED ON MODELS OF COMPLEX SYSTEMS

SUBMITTED: August 2005
REVISED: November 2005
PUBLISHED: December 2005 at http://www.itcon.org/2005/22/
EDITOR: C. Anumba

Rafal Kicinger, Research Assistant Professor
Department of Civil, Environmental, and Infrastructure Engineering, George Mason University, USA
email: rkicinge@gmu.edu

Tomasz Arciszewski, Professor and Chair
Department of Civil, Environmental, and Infrastructure Engineering, George Mason University, USA
email: tarcisze@gmu.edu

Kenneth De Jong, Professor
Computer Science Department, George Mason University, USA
email: kdejong @gmu.edu

SUMMARY: *This paper introduces an integrated research and design support tool, called Emergent Designer, developed at George Mason University. It is a tool that implements models of various complex systems, including cellular automata and evolutionary algorithms, to represent engineering systems and their related design processes. The system is intended for conducting design experiments in the area of structural design and for the analysis of their results. It implements state-of-the-art representations supporting generation of novel design concepts and efficient mechanisms for their subsequent optimization at the topological and sizing levels. The first part of this paper describes the overall system's architecture and the flow of information among its components. The actual system's implementation is discussed next and illustrated with several screen shots of the system's graphical user interface. Emergent Designer's novel approach to representing steel structural systems in tall buildings is also presented. It is based on the use of generative representations which utilize cellular automata to generate design concepts. Several design experiments are briefly described to demonstrate the feasibility of Emergent Designer in conceptual design as well as of design processes modeled by complex systems.*

KEYWORDS: *evolutionary design, design support tools, emergence, cellular automata, structural design*

1. INTRODUCTION

With the emergence of Information Technology new design methods based on various computational models of engineering design processes are being developed. However, up until very recently, computers in structural design were used mostly and merely for drawing and visualization (CAD) and for various analytical design activities conducted in the detailed design stage (Arciszewski and De Jong, 2001). Today, we are witnessing the emergence of a new class of design methods applicable both in conceptual and detailed design stages. In order to fully benefit from this progress, these new design methods require, however, new computer tools.

At the same time, we are witnessing another paradigm change in engineering design. Traditional methods, which are focused strictly on design optimization tasks, are being replaced by new methods emphasizing not only design optimality issues but also generation of novel design concepts (Kicinger, 2004). Representations of engineering systems are one of the key issues to achieve this goal. When the focus is restricted to design optimization, the attention is usually limited to a particular concept or at most to several known design concepts (Arciszewski et al., 1995). In this case, design representations usually take a form of parameterizations of an engineering system, or its parts, i.e. the system is represented by a collection of parameters and their values which together form its complete description. Traditional representations frequently used in engineering optimization problems can be included in this category, e.g., representations in which parameters have binary values representing presence or absence of a feature (binary representations), or real values representing numerical dimensions (real-valued representations).

Inventive design requires, however, more general and usually more complex representations. Representations that are appropriate for inventive design are diverse but they nevertheless share some similarities. Typically, they are quite general and thus capable of representing large numbers of alternative shapes, forms, or morphologies (Bentley, 1999). They range from direct representations, as in voxel-based representations (Baron et al., 1997) or array-based representations (Kane and Schoenauer, 1995), to highly indirect and generative representations, i.e. representations that do not encode complete solutions but rather rules on how to build/generate these solutions. The most popular examples of indirect representations in engineering design include trees (Funes and Pollack, 1999), shape grammars (Stiny, 1980), cellular automata (CAs) (Frazer, 1995), L-systems (O'Reilly et al., 2000), and embryogenies (Bentley and Kumar, 1999).

In this paper, we introduce *Emergent Designer*, an integrated research and design support tool, developed at George Mason University. It represents, together with systems like iSIGHT (Koch et al., 2002) or modeFRONTIER™, a new generation of computer tools which automate and optimize design processes by integrating state-of-the-art models, algorithms, and analysis and visualization methods. The distinguishing feature of *Emergent Designer* is the fact that it addresses both important objectives of engineering design, namely creativity and optimality. It does that by implementing new types of generative representations based on CAs (Kicinger et al., 2004; Kicinger et al., 2005) which support and enhance generation of novel design concepts. It subsequently optimizes these designs using modern search and optimization algorithms.

A general overview of the system describing its overall architecture and major components is presented in section 2. Next, in section 3 the actual implementation of *Emergent Designer* is presented. Furthermore, several types of representations of steel structural systems based on CAs are introduced and results of design experiments conducted using *Emergent Designer* are reported. Finally, initial conclusions and recommendations for future work are presented.

2. SYSTEM OVERVIEW

Emergent Designer is intended for conducting design experiments in the area of structural design and for the analysis of their results using models, methods, and tools from statistics and time series analysis. It can be used as a design support tool equipped with state-of-the-art mechanisms for the generation of novel design concepts and for conducting their optimization. At the same time, it is a versatile research tool that implements advanced methods and tools for the analysis of design processes and of obtained experimental results.

2.1 Architecture

A high-level overview of the architecture of *Emergent Designer* is presented in Fig. 1.

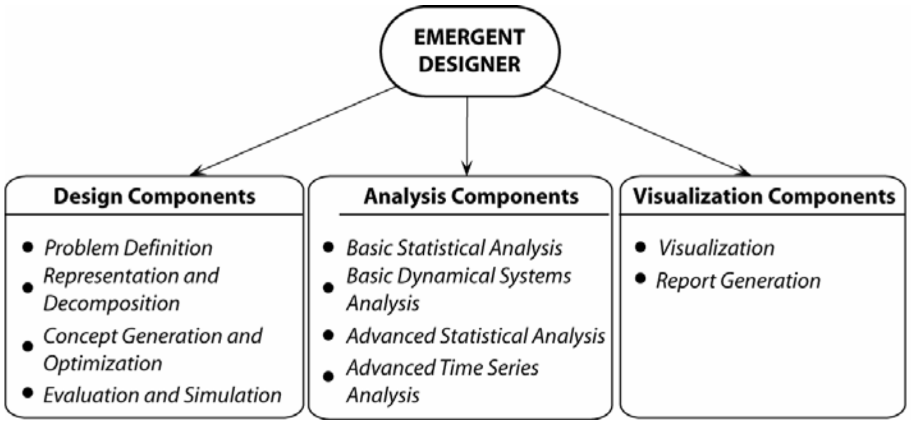

EMERGENT DESIGNER
Design Components
Problem Definition
Representation and Decomposition
Concept Generation and Optimization
Evaluation and Simulation
Analysis Components
Basic Statistical Analysis
Basic Dynamical Systems Analysis
Advanced Statistical Analysis
Advanced Time Series Analysis
Visualization Components
Visualization
Report Generation

FIG. 1: Architecture of Emergent Designer

The system consists of 10 major components/modules which can be divided into three major groups:

- **Design components**

They implement ‘Emergent Engineering Design’ (Kicinger, 2004), a design method using models of complex systems to represent engineering systems and design processes. Design components form the core of the system and conduct actual design processes.

- **Analysis components**
  They implement tools and methods for the analysis of experimental results and design processes. The components included in this group are aimed to provide quantitative information about design processes as well as statistical estimates of the performance of the tool in which the method is implemented. In this way, an indirect (but only feasible) evaluation of the design method can be conducted. These components are also intended to provide deeper understanding of the underlying dynamics of design processes and the structure of design spaces from a global/holistic perspective.
- **Visualization components**
  These components implement various visualization methods and report generation mechanisms. They include tools which support visualization of results from various analyses, e.g. statistical or time series analyses, conducted by other system’s components. Also, automated tools for generation of experimental reports are implemented, which include detailed descriptions of experimental parameters and obtained results.

2.2 Information flow

The flow of information in *Emergent Designer* is presented in Fig. 2. It provides an overview of relationships among the components discussed in section 2.1 and shows where user input/decisions are expected.

Once *Emergent Designer* has been started, the user has a choice to conduct a new design experiment or to use advanced statistical and time series analysis tools in order to analyze experimental data saved from previous experiments. By default, a new design experiment is selected and *Problem Definition Component* is called to define a design problem. *Problem Definition Component* is intended to select a domain of interest (e.g. steel skeleton structures in tall buildings) from the list of implemented problem domains, and a specific design problem that will be solved (e.g. design of a wind bracing system). This component allows for the specification of values of parameters defining the considered design problem, e.g. the number of stories in a tall building, or the height of a story. It also implements mechanisms for saving the system’s parameters and their values in a file, and retrieving previously saved values from a file.

When the design problem is completely defined, the user has to decide whether or not to decompose the problem into several sub-problems. *Representation and Decomposition Component* is used for this purpose. If the design problem is to be decomposed, then the user selects a decomposed representation of the engineering system. On the other hand, if the design problem is considered as a whole, then one of the implemented representations of the entire engineering system can be chosen. In this case, the spectrum of possible representations includes parameterized representations and generative representations. *Representation and Decomposition Component* also supports the specification of values of representation specific parameters, e.g. the shape of the local neighborhood in generative representations based on CAs (see section 4.1). When all decomposition and encoding parameters have been defined, the representation of an engineering system (artifact) is completely specified.

At that point, *Concept Generation and Optimization Component* is utilized to specify the type of a concept generation mechanism and to determine whether or not the topology optimization and/or the sizing optimization should be conducted (see Fig. 3). In this way, it defines a representation of an engineering design process.

If only a concept generation mechanism is selected, i.e. no optimization is performed, then design concepts are produced by design concept generators based on generative representations, e.g. 1D or 2D CAs (see section 4.2). On the other hand, when the optimization of engineering systems is to be performed, then the user has two possible choices:

1. If the focus is restricted to design optimization issues, then only an optimization mechanism, e.g. an evolutionary algorithm (EA), is used together with traditional parameterized representations of engineering systems.
2. If both the generation of novel design concepts and their subsequent optimization are considered as equally important objectives, then an optimization mechanism is combined with a concept generation mechanism utilizing a generative representation.

Design concepts produced by the concept generation and/or optimization mechanisms are transferred to the *Evaluation and Simulation Component* which evaluates them and assigns fitness value(s) (multiple fitness measures are used in the case of multiobjective optimization). This component is used to select an evaluation model for a given design experiment and to determine values of several parameters specifying the evaluation process, e.g. methods for the determination of wind loads acting on a structural system, or magnitudes of dead and live loads, etc. Also, general simulation parameters (e.g., the number of runs, the termination criteria) need to be defined by this component in order to run a design experiment.

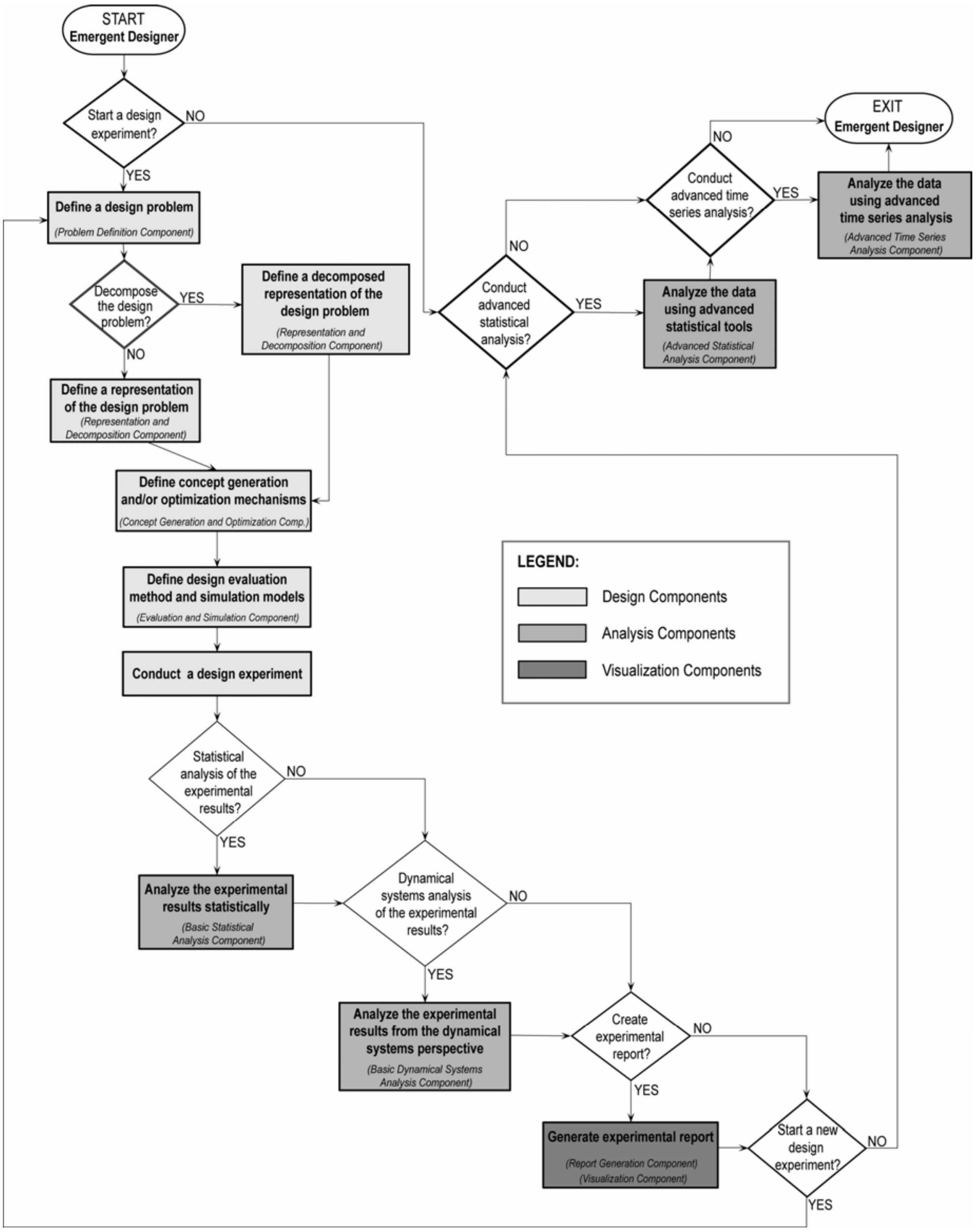

START Emergent Designer
Start a design experiment?
NO
YES
Define a design problem
(Problem Definition Component)
Decompose the design problem?
YES
NO
Define a decomposed representation of the design problem
(Representation and Decomposition Component)
Define a representation of the design problem
(Representation and Decomposition Component)
Define concept generation and/or optimization mechanisms
(Concept Generation and Optimization Comp.)
Define design evaluation method and simulation models
(Evaluation and Simulation Component)
Conduct a design experiment
Statistical analysis of the experimental results?
NO
YES
Analyze the experimental results statistically
(Basic Statistical Analysis Component)
Dynamical systems analysis of the experimental results?
NO
YES
Analyze the experimental results from the dynamical systems perspective
(Basic Dynamical Systems Analysis Component)
Create experimental report?
NO
YES
Generate experimental report
(Report Generation Component)
(Visualization Component)
Start a new design experiment?
NO
YES
Conduct advanced statistical analysis?
YES
NO
Analyze the data using advanced statistical tools
(Advanced Statistical Analysis Component)
Conduct advanced time series analysis?
YES
NO
Analyze the data using advanced time series analysis
(Advanced Time Series Analysis Component)
EXIT Emergent Designer
LEGEND:
Design Components
Analysis Components
Visualization Components

FIG. 2: Information flow in Emergent Designer

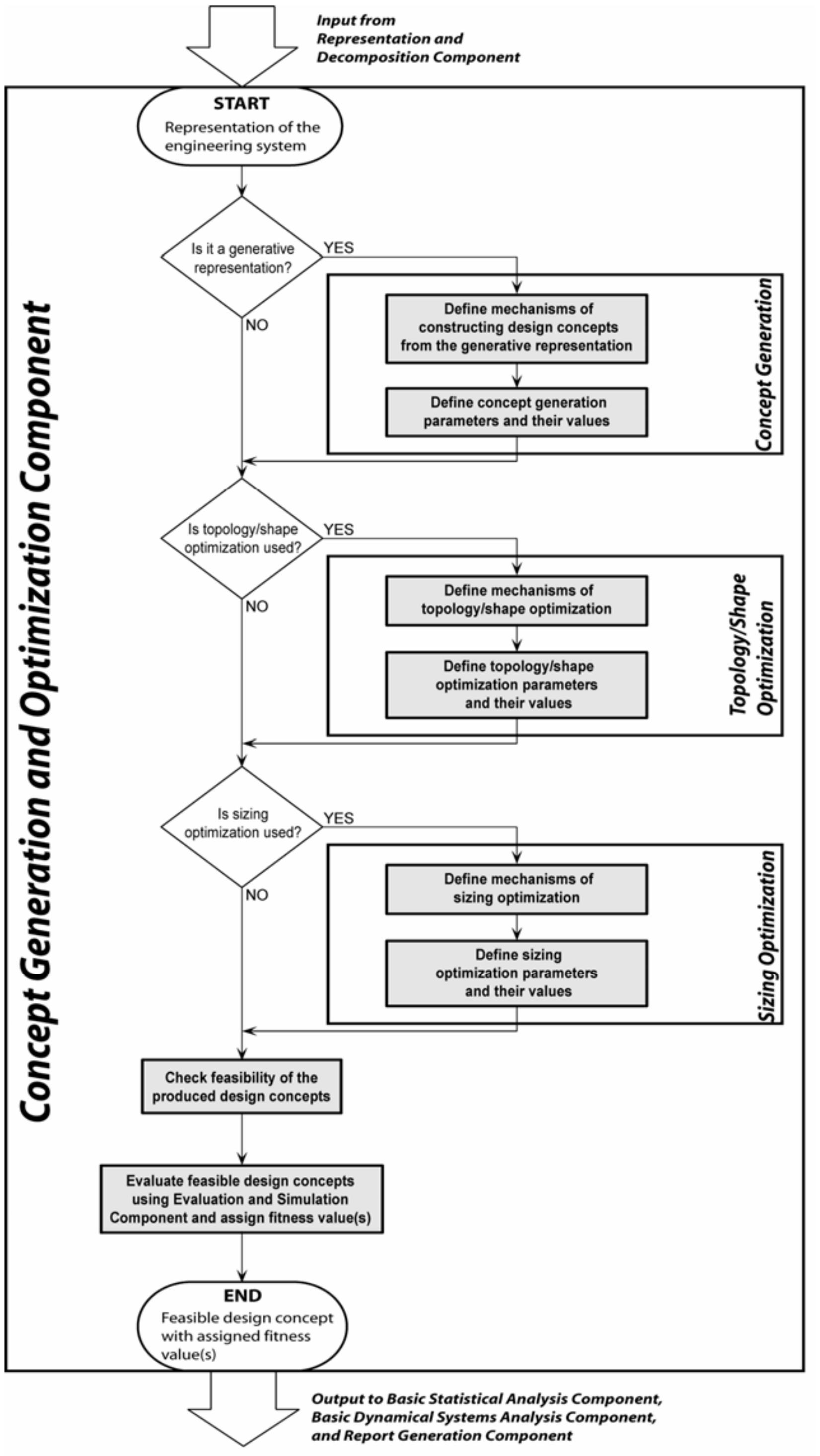

Input from Representation and Decomposition Component
Concept Generation and Optimization Component
START
Representation of the engineering system
Is it a generative representation?
YES
NO
Define mechanisms of constructing design concepts from the generative representation
Define concept generation parameters and their values
Concept Generation
Is topology/shape optimization used?
YES
NO
Define mechanisms of topology/shape optimization
Define topology/shape optimization parameters and their values
Topology/Shape Optimization
Is sizing optimization used?
YES
NO
Define mechanisms of sizing optimization
Define sizing optimization parameters and their values
Sizing Optimization
Check feasibility of the produced design concepts
Evaluate feasible design concepts using Evaluation and Simulation Component and assign fitness value(s)
END
Feasible design concept with assigned fitness value(s)
Output to Basic Statistical Analysis Component, Basic Dynamical Systems Analysis Component, and Report Generation Component

FIG..3: Information flow within the Concept Generation and Optimization Component

As described earlier, the four components, i.e. *Problem Definition Component*, *Representation and Decomposition Component*, *Concept Generation and Optimization Component*, and *Evaluation and Simulation Component* form a group of design components implementing the actual design method. Relationships among these components and major phases of the design method are shown schematically in Fig. 4.

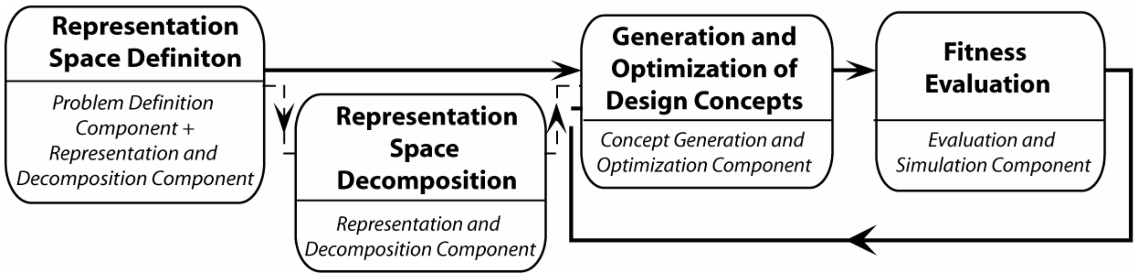

Representation Space Definiton
Problem Definition Component + Representation and Decomposition Component
Representation Space Decomposition
Representation and Decomposition Component
Generation and Optimization of Design Concepts
Concept Generation and Optimization Component
Fitness Evaluation
Evaluation and Simulation Component

FIG. 4: Phases of the design method implemented in Emergent Designer

When values of all parameters implemented in the group of design components have been determined (default values are also used where possible), the actual design experiment can be initiated. *Basic Statistical Analysis Component* and *Basic Dynamical Systems Analysis Component* support online monitoring of design processes by providing best-so-far fitness values and trajectories (sequences of points) of solutions (designs) in design spaces. *Basic Statistical Analysis Component* also provides mechanisms for extracting and collecting relevant experimental data and saving them in files.

When a given design experiment is completed, *Basic Statistical Analysis Component* is used to calculate and display average best-so-far fitness values with corresponding 95% confidence intervals. At that point, the user can also generate a complete experimental report listing all design parameters and their values used in the experiment as well as produced results. *Report Generation Component* and *Visualization Component* are employed during the process of automatic generation of an experimental report. *Report Generation Component* collects names and values of design parameters used in the experiment and extracts relevant experimental results. It also gathers important statistical data calculated by *Basic Statistical Analysis Component*. *Visualization Component* can be used to produce various visualization graphs and charts representing the progress of individual runs in design experiments. When all textual, numerical, and graphical data are available, *Report Generation Component* compiles them together into a single document that is subsequently displayed as an experimental report.

At this point, the user can choose to start a new design experiment or to analyze the experimental data using advanced statistical and time series analysis tools, or simply exit the system. If a new design experiment is selected, *Problem Definition Component* is called again and the entire process described above is repeated. On the other hand, if advanced statistical analysis, or advanced time series analysis, is chosen then *Advanced Statistical Analysis Component* or *Advanced Time Series Analysis Component* is utilized, respectively.

Advanced Statistical Analysis Component implements advanced statistical methods to more accurately analyze experimental data. It is particularly useful for analyzing results of design experiments in which only a small number of runs (repetitions) was conducted (e.g., when evaluations of generated designs are computationally expensive). In such cases, advanced statistical methods (e.g., analyses of sample distributions, histograms, normal scores plots, skewness and kurtosis estimates) should produce more reliable estimates of the performance of the conducted design processes.

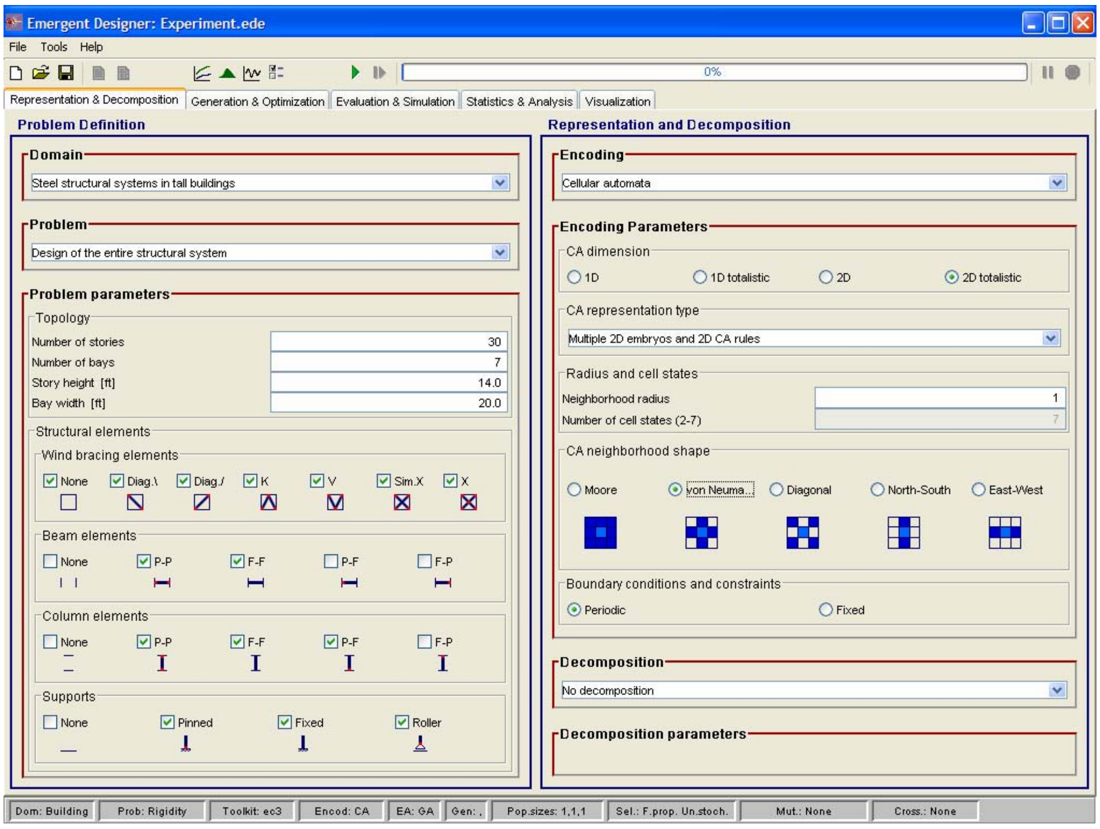
3. IMPLEMENTATION

Emergent Designer has been implemented in Java with a fully functional graphical user interface (GUI). The decision to use this particular programming language was made because several of the system’s components were built upon existing packages written in Java. Moreover, *Emergent Designer* integrates several commercially available systems (e.g., Mathematica© (Wolfram, 2003) and OpenOffice.org©) and communicates with them by using available Java APIs.

Another important reason why this programming language was selected is Java’s portability and network-orientation. Portability offers the flexibility of running the system on various platforms while built-in networking capabilities open the possibility of using distributed architectures. Both of these features are particularly important in structural design in which the process of evaluation of generated design concepts is usually computationally expensive and conducted using specialized structural analysis software.

3.1 Design Components

As of now, two problem domains have been implemented in *Problem Definition Component*: the domain of steel structural systems in tall buildings and the domain of real-valued functions (added for testing purposes and for the behavior analysis of various components of the system). The domain of steel structural systems includes two major classes of structural design problems: design of a wind bracing system in a tall building and design of an entire steel structural system in a tall building. Four types of representations are supported by *Representation and Decomposition Component*, including binary, real-valued, integer-valued, and CAs. The latter two are used to encode designs concepts of steel structural systems in tall buildings. A simple example which introduces and compares these two types of design representations is described in section 4.2. The GUI of *Emergent Designer* displaying *Problem Definition Component* and *Representation and Decomposition Component* is presented in Fig. 5.


Emergent Designer: Experiment.ede
File
Tools
Help
0%
Representation & Decomposition
Generation & Optimization
Evaluation & Simulation
Statistics & Analysis
Visualization

Problem Definition

Domain: Steel structural systems in tall buildings

Problem: Design of the entire structural system

Problem parameters

Topology

Number of stories	30
Number of bays	7
Story height [ft]	14.0
Bay width [ft]	20.0

Structural elements

Wind bracing elements: [✓ None] [✓ Diag.\\] [✓ Diag./] [✓ K] [✓ V] [✓ Sim.X] [✓ X]

Beam elements: None [✓ P-P] [✓ F-F] P-F F-P

Column elements: None [✓ P-P] [✓ F-F] [✓ P-F] F-P

Supports: None [✓ Pinned] [✓ Fixed] [✓ Roller]

Representation and Decomposition

Encoding: Cellular automata

Encoding Parameters

CA dimension: 1D 1D totalistic 2D [✓ 2D totalistic]

CA representation type: Multiple 2D embryos and 2D CA rules

Radius and cell states

Neighborhood radius	1
Number of cell states (2-7)	7

CA neighborhood shape: Moore [✓ von Neuma...] Diagonal North-South East-West

Boundary conditions and constraints: [✓ Periodic] Fixed

Decomposition: No decomposition

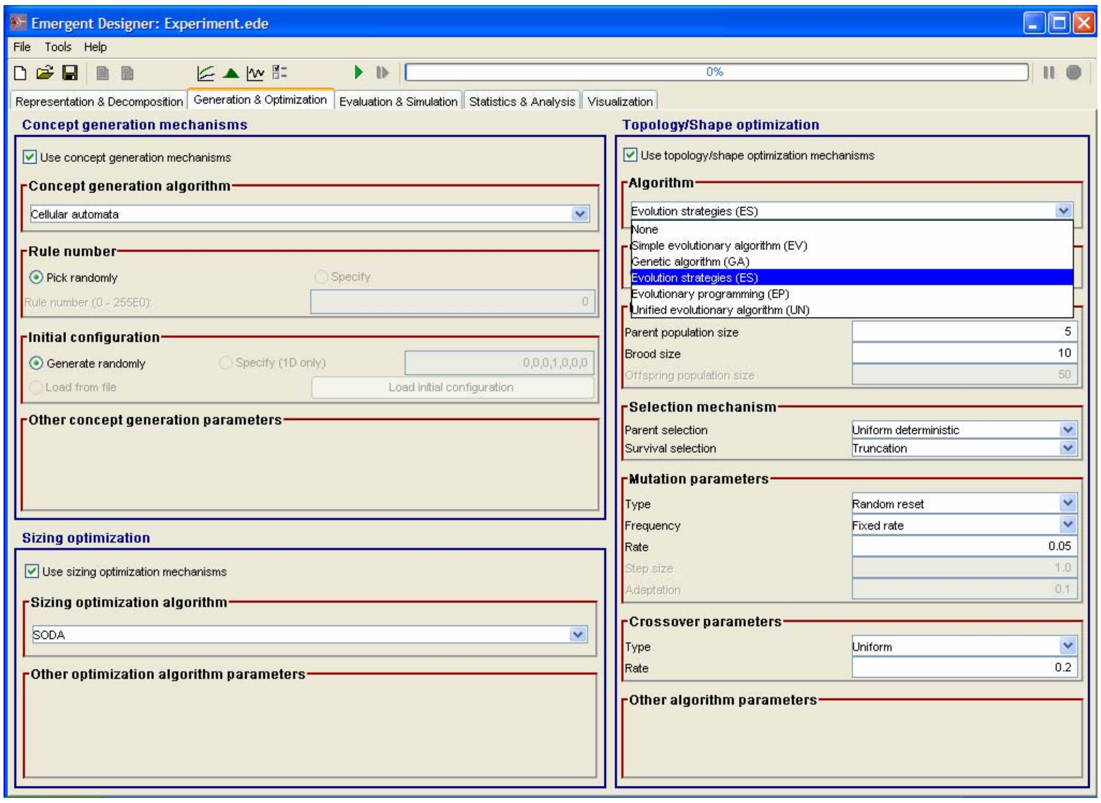
Decomposition parameters

Dom: Building | Prob: Rigidity | Toolkit: ec3 | Encod: CA | EA: GA | Gen: , | Pop.sizes: 1,1,1 | Sel.: F.prop. Un.stoch. | Mut.: None | Cross.: None

FIG. 5: Graphical user interface of Emergent Designer showing Problem Definition Component and Representation and Decomposition Component

Concept Generation and Optimization Component has been built upon four major existing packages and commercially available systems. Generation of design concepts utilizing various types of generative representations utilizing CAs is conducted by Mathematica's kernel which was integrated with *Emergent Designer* via JLink™. All major types of CAs are supported, including 1D and 2D CAs (see section 4.1).

The topology/shape optimization using EAs is supported by ec3 package, a Java-based evolutionary computation toolkit developed by Kenneth De Jong at George Mason University. Here, all canonical EAs can be utilized, including genetic algorithms (GAs), evolutionary programming (EP), and evolution strategies (ES). The system also offers a possibility of employing a unified EA (De Jong, to appear) in which all major components (e.g., parent and offspring selection mechanisms, genetic operators.) can be individually adjusted to a particular problem. The sizing optimization is conducted using an optimization algorithm based on traditional mathematical programming method and implemented in a commercially available structural analysis, design and optimization system called SODA (Grierson, 1989). Implementation of *Concept Generation and Optimization Component* is presented in Fig 6.


Emergent Designer: Experiment.ede

File Tools Help

0%

Representation & Decomposition | Generation & Optimization | Evaluation & Simulation | Statistics & Analysis | Visualization

Concept generation mechanisms

[✓] Use concept generation mechanisms

Concept generation algorithm: Cellular automata

Rule number: [✓ Pick randomly] Specify

Rule number (0 - 255E0): 0

Initial configuration: [✓ Generate randomly] Specify (1D only) 0,0,0,1,0,0,0

Load from file — Load initial configuration

Other concept generation parameters

Sizing optimization

[✓] Use sizing optimization mechanisms

Sizing optimization algorithm: SODA

Other optimization algorithm parameters

Topology/Shape optimization

[✓] Use topology/shape optimization mechanisms

Algorithm: Evolution strategies (ES)

- None
- Simple evolutionary algorithm (EV)
- Genetic algorithm (GA)
- Evolution strategies (ES)
- Evolutionary programming (EP)
- Unified evolutionary algorithm (UN)

Parent population size	5
Brood size	10
Offspring population size	50

Selection mechanism

Parent selection	Uniform deterministic
Survival selection	Truncation

Mutation parameters

Type	Random reset
Frequency	Fixed rate
Rate	0.05
Step size	1.0
Adaptation	0.1

Crossover parameters

Type	Uniform
Rate	0.2

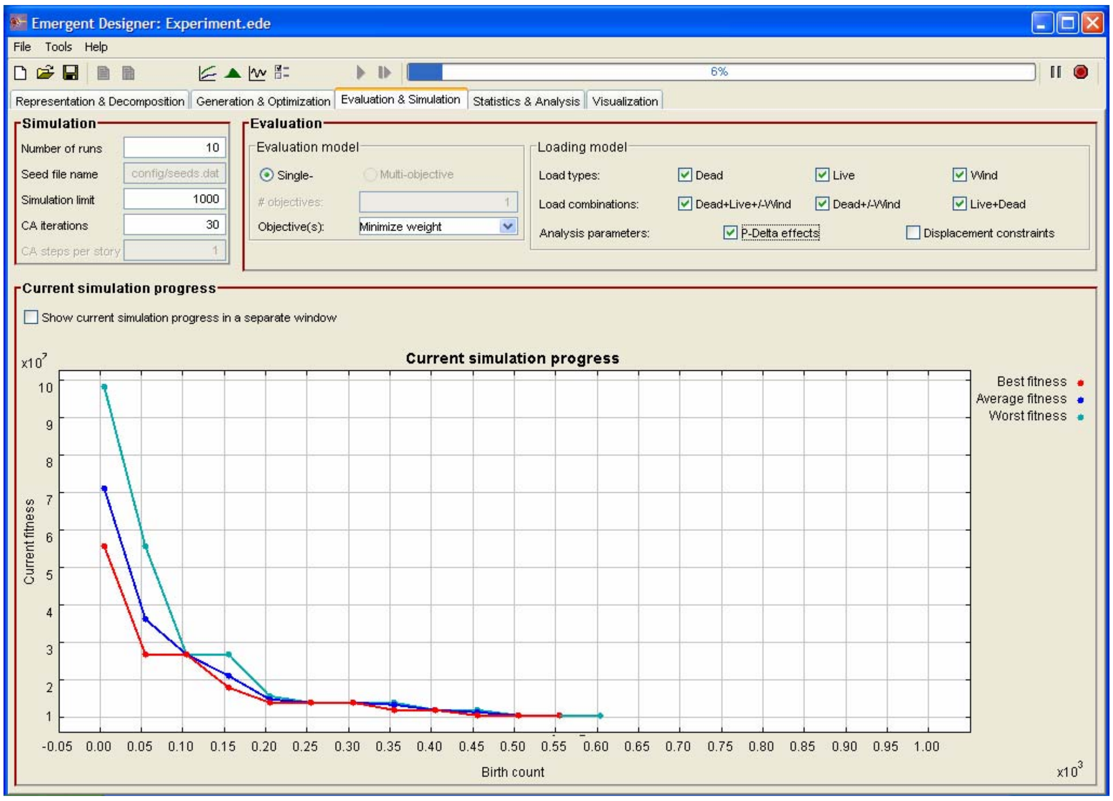
Other algorithm parameters

FIG. 6: Implementation of Concept Generation and Optimization Component

Evaluation and Simulation Component implements evaluation models used to determine fitness of generated designs. The system supports both single- and multiobjective evaluation of designs using the following evaluation criteria: the total weight (an estimate of the cost) and the maximal horizontal displacement (an estimate of the stiffness) of a steel structure. The determination of the least-weight structure is performed by SODA and conducted in conformance with the strength (stability) and stiffness (displacement) provisions of several commonly used steel codes, including AISC-ASD-89, AISC-LRFD-86, AISC-LRFD-93, CSA-Sl6.1-M89, or CSA-Sl6.1-94. The loading model required for the evaluation of generated design concepts includes dead, live, and wind loads. Wind forces are calculated for a given design situation using a modified version of a commercial system Wind Load© V2.2.S developed by Novel CyberSpace Tools. Fig. 7 shows the implementation of *Evaluation and Simulation Component* as well as *Visualization Component* displaying the current progress of a design process.

3.2 Analysis Components

Methods and models of basic statistical and dynamical systems analysis have been implemented directly in Java. These analytical processes are conducted online, i.e., during the actual design processes. The basic statistical analysis involves best-so-far fitness statistics calculated for individual runs (see Fig. 8 top left) and average best-so-far fitness statistics and 95% confidence intervals computed for an entire design experiment (see Fig. 8 bottom left). These analyses are also automatically saved in files. Implemented methods of simple dynamical systems analysis include trajectory analysis which shows the dynamics of the processes in design spaces (see Fig. 8FIG. 8 top right) as well as delay coordinates analysis (see Fig. 8 bottom right). The trajectory analysis dynamically shows the dynamics of conducted design processes and region(s) of the search space to which they converge. On the other hand, the delay coordinates analysis attempts to reconstruct attractors from the time series produced by design processes.



Simulation

Number of runs	10
Seed file name	config/seeds.dat
Simulation limit	1000
CA iterations	30
CA steps per story	1

Evaluation

Evaluation model: [✓ Single-] Multi-objective
# objectives: 1
Objective(s): Minimize weight

Loading model

Load types: [✓ Dead] [✓ Live] [✓ Wind]
Load combinations: [✓ Dead+Live+/-Wind] [✓ Dead+/-Wind] [✓ Live+Dead]
Analysis parameters: [✓ P-Delta effects] Displacement constraints

Current simulation progress

Show current simulation progress in a separate window

FIG. 7: Implementation of Evaluation and Simulation Component

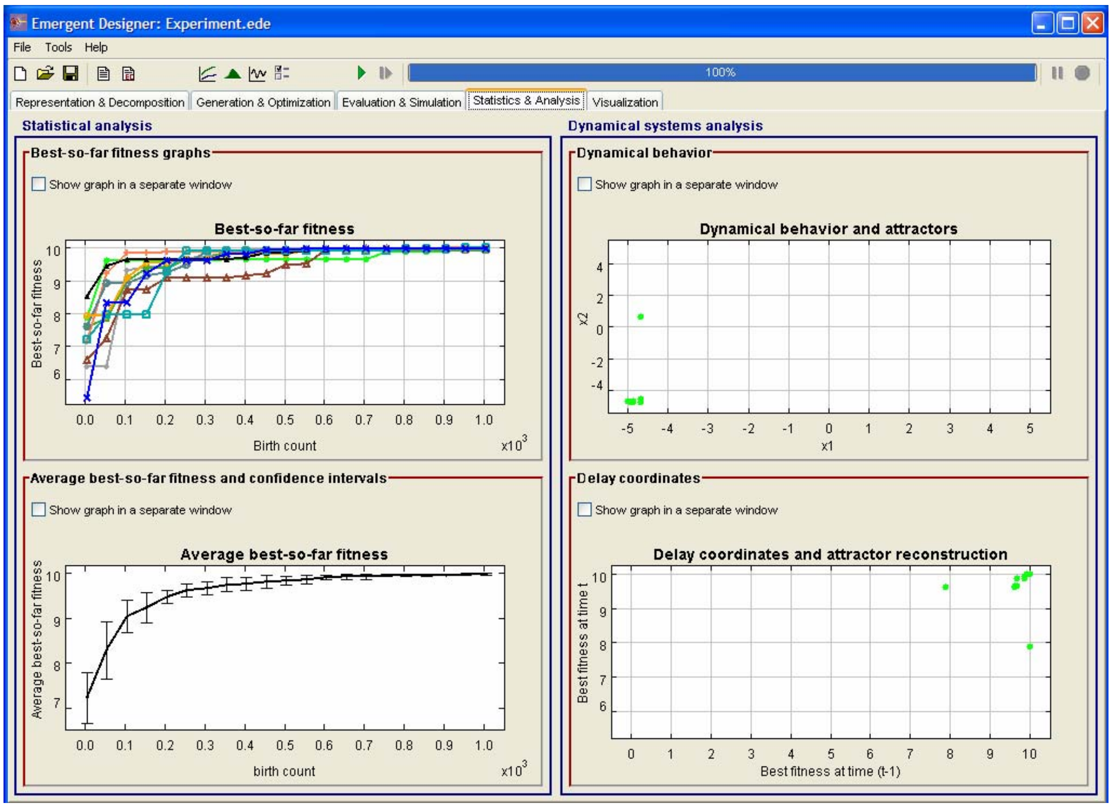
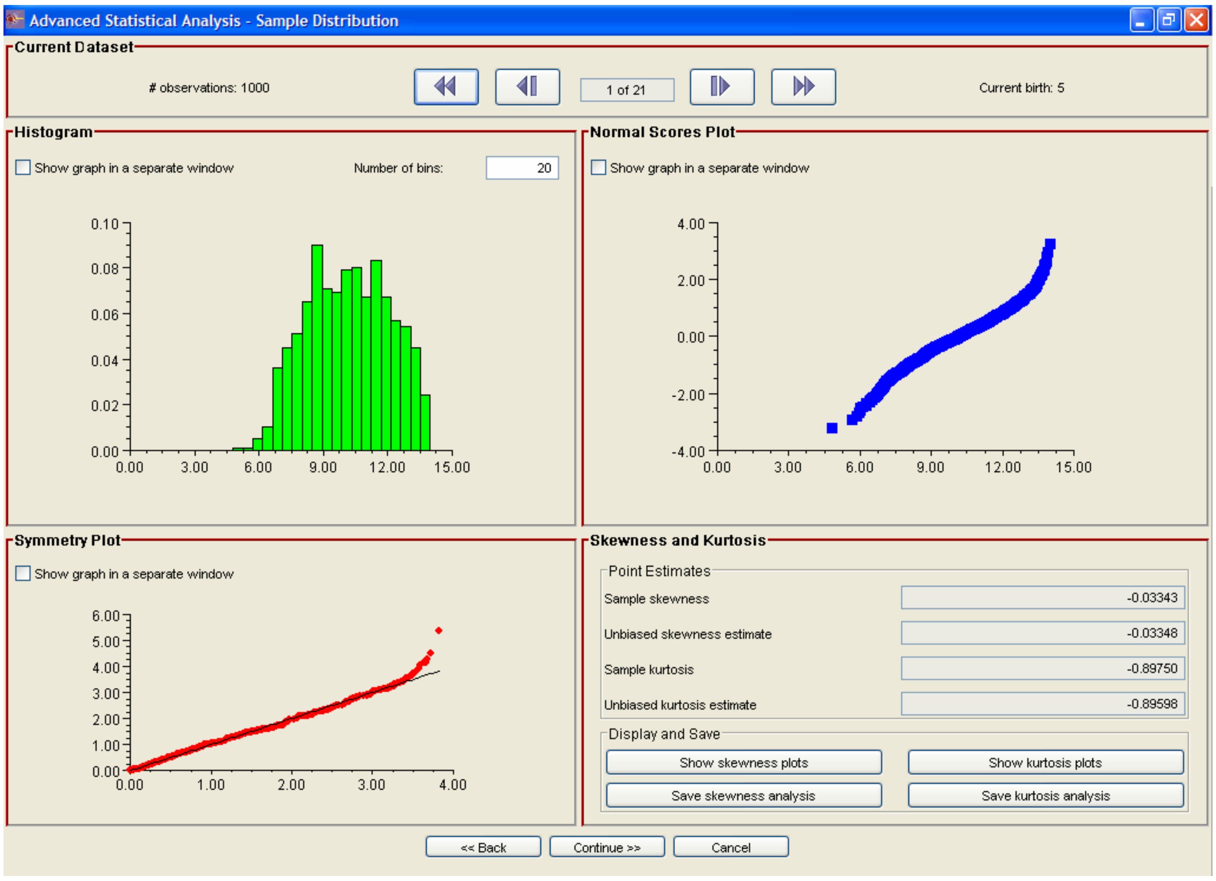

Emergent Designer: Experiment.ede
File Tools Help
100%
Representation & Decomposition
Generation & Optimization
Evaluation & Simulation
Statistics & Analysis
Visualization
Statistical analysis
Best-so-far fitness graphs
Show graph in a separate window
Best-so-far fitness
Best-so-far fitness
Birth count
x10^3
Average best-so-far fitness and confidence intervals
Show graph in a separate window
Average best-so-far fitness
Average best-so-far fitness
birth count
x10^3
Dynamical systems analysis
Dynamical behavior
Show graph in a separate window
Dynamical behavior and attractors
x2
x1
Delay coordinates
Show graph in a separate window
Delay coordinates and attractor reconstruction
Best fitness at time t
Best fitness at time (t-1)

FIG. 8: Implementation of Basic Statistical Analysis Component and Basic Dynamical System Analysis Component

Contrary to the basic analyses described above, advanced statistical and time series analyses are performed offline, i.e. when a given design experiment has been completed. Fig. 9 shows several types of advanced statistical analyses implemented in *Advanced Statistical Analysis Component*. They were either implemented directly in Java or borrowed from JMSL© Numerical Library which was integrated with *Emergent Designer*.


Advanced Statistical Analysis - Sample Distribution

Current Dataset

observations: 1000

1 of 21

Current birth: 5

Histogram

Show graph in a separate window

Number of bins: 20

Normal Scores Plot

Show graph in a separate window

Symmetry Plot

Show graph in a separate window

Skewness and Kurtosis

Point Estimates

Sample skewness	-0.03343
Unbiased skewness estimate	-0.03348
Sample kurtosis	-0.89750
Unbiased kurtosis estimate	-0.89598

Display and Save

FIG. 9: Advanced statistical analysis tools implemented in Emergent Designer

Advanced Time Series Analysis Component implements more advanced methods of linear and nonlinear time series analysis, including delay coordinates plots with adjustable parameters, power spectrum analysis, autocorrelation analysis, and two types of recurrence plots. As was the case with the Advanced Statistical Analysis Component, several tools and methods of advanced time series analysis were directly implemented in the system while some of them have been used from JMSL© Numerical Library.

3.3 Visualization Components

There are three major methods of visualizing experimental data in *Emergent Designer*. First, line plots and scatter plots (or more generally signal plots) are used to visualize experimental data transferred from *Basic Statistical Analysis Component* and *Basic Dynamical Systems Analysis Component* (see Fig. 8). The plots are produced by a Java-based signal plotter called PtPlot developed at UC Berkeley. They are embedded in *Emergent Designer*'s GUI and can be subsequently saved as bitmap files. Second, histograms are employed to visualize sample distributions produced by *Advanced Statistical Analysis Component* (see Fig. 9). These types of graphs are created using JMSL© Graphical Library integrated with *Emergent Designer*. They are also embedded in the system's GUI and provide functionality to save the produced graphs as bitmap files. Finally, renderings of fitness landscapes can be produced (see Fig. 10) for simple real-valued functions only. More advanced methods supporting visualization of design landscapes will be added in the future versions of the system. Renderings of simple real-valued fitness landscapes are produced using Mathematica's advanced graphical capabilities and their display in *Emergent Designer*'s GUI is supported by JLink.

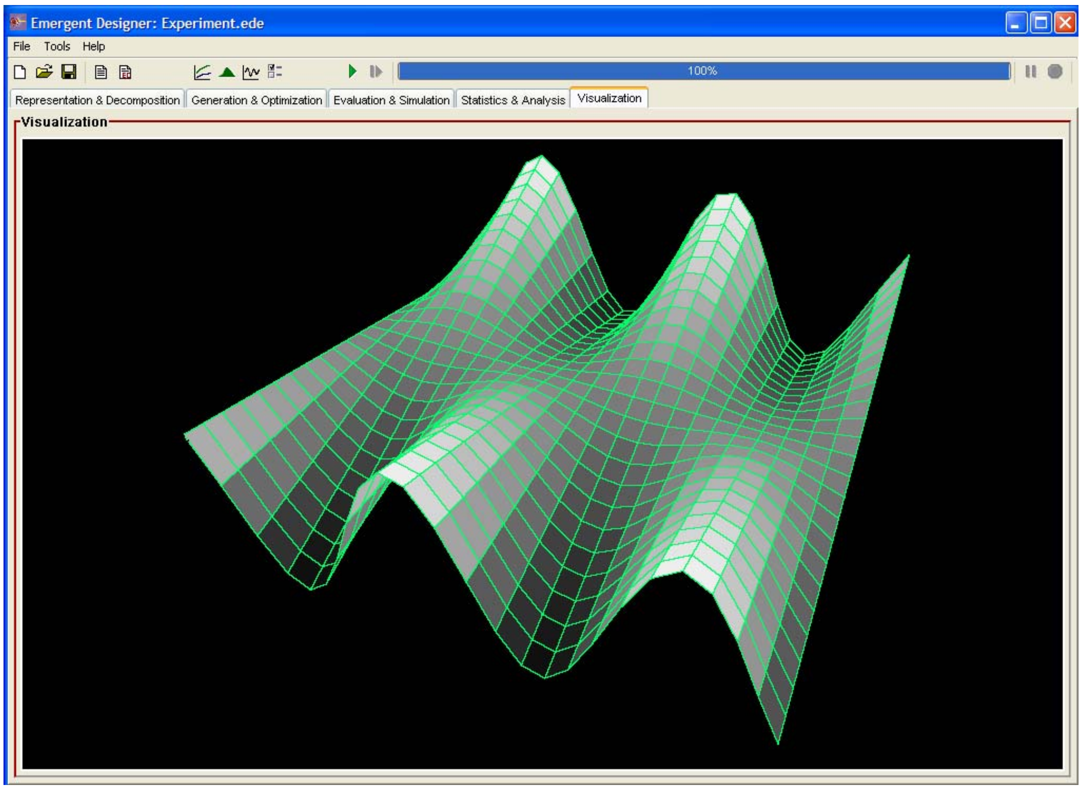


FIG. 10: Renderings of fitness landscapes produced by the Visualization Component

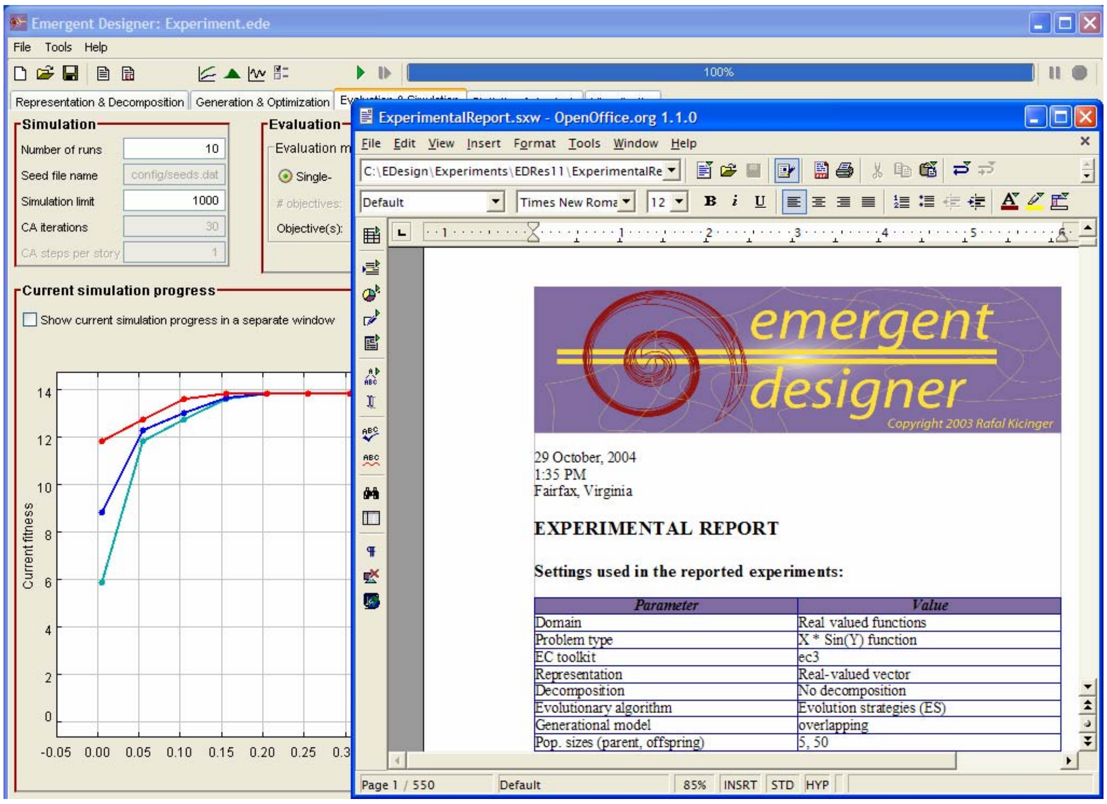
Automatic report generation capabilities have been achieved through the integration of *Emergent Designer* with OpenOffice.org© via its Java API. *Report Generation Component* collects and organizes textual, numerical and graphical data produced during design processes and includes them in an experimental report. This report is subsequently displayed as an OpenOffice.org document as shown in Fig. 11. The report can be later saved in a file in any of the supported formats. In this way, a complete summary including design parameters and obtained experimental results is automatically created.

4. EXAMPLES OF DESIGN EXPERIMENTS

Having described the system's architecture and components, this section shows how *Emergent Designer* can be used as a state-of-the-art design support system and an advanced analysis tool. First, however, CAs as well as novel design concept generators based on them are briefly introduced. Also, these concept generators are compared to traditional parameterized representations used in structural design. Next, experimental results produced using these two types of representations in the domain of steel structural systems in tall buildings are reported. Finally, it is shown how advanced analysis tools implemented in *Emergent Designer* can be utilized to obtain better understanding of design processes.

4.1 Cellular Automata

Cellular automata are one of the simplest models of complex systems (Wolfram, 1994). They are used to model many complex systems and processes consisting of a large number of identical, simple, and locally interacting components, including fluid and chemical turbulence (d'Humieres and Lallemand, 1986; Gerhadrt and Schuster, 1989), growth of crystals (Kessler et al., 1990), social dynamics (Axtell and Epstein, 1996), patterns of electrical activity in neural networks (Franceschetti et al., 1992), and others. CAs have also been the subject of significant research interests in structural design. Inou et al. (1998) used cellular automata to investigate self-organization of topologies in mechanical structures. Kundu et al. (1997) applied CAs to shape optimization of structural plates. Kita and Toyoda (2000) used CAs to optimize shape and topology of two-dimensional elastic structures and to optimize cross-sections of members in truss structures (Kita and Toyoda, 2001). Hajela and Kim (2001) applied GAs to search the space of CA rules in the structural analysis of 2D elastic structures.


Emergent Designer: Experiment.ede
File Tools Help
100%
Representation & Decomposition
Generation & Optimization
ExperimentalReport.sxw - OpenOffice.org 1.1.0
File Edit View Insert Format Tools Window Help
C:\EDesign\Experiments\EDRes11\ExperimentalRe
Default
Times New Roma
12
Current fitness
Page 1 / 550
Default
85%
INSRT
STD
HYP

Simulation

Number of runs	10
Seed file name	config/seeds.dat
Simulation limit	1000
CA iterations	30
CA steps per story	1

Evaluation

Evaluation m

Single-

# objectives:

Objective(s):

Current simulation progress

☐ Show current simulation progress in a separate window

emergent designer

Copyright 2003 Rafal Kicinger

29 October, 2004
1:35 PM
Fairfax, Virginia

EXPERIMENTAL REPORT

Settings used in the reported experiments:

Parameter	*Value*
Domain	Real valued functions
Problem type	X * Sin(Y) function
EC toolkit	ec3
Representation	Real-valued vector
Decomposition	No decomposition
Evolutionary algorithm	Evolution strategies (ES)
Generational model	overlapping
Pop. sizes (parent, offspring)	5, 50

FIG. 11: Experimental report produced by the Report Generation Component and displayed as an OpenOffice.org document

A cellular automaton is a deterministic system which is fully defined by its update rule (called here a CA rule) and an initial configuration of cell values. Fig. 12 shows how simple cellular automata work. The process of iteration of an elementary CA is presented in Fig. 12a. In this case, the individual cells have only binary values (0 and 1) and local neighborhoods affecting the iteration of the considered cell are formed by this cell and its immediate left and right neighbors. Therefore, three cells are considered in each local neighborhood and such situation is called a 'local neighborhood of size 3'. The bottom row of Fig. 12a consists of 6 squares (cells) denoting an initial configuration of cells (at *t=0*). In this particular case, the initial configuration consists of cell values 0 0 0 1 1 0. White squares in Fig. 12 denote cell values equal to 0 while black squares represent cell values equal to 1. The particular CA rule used to iterate this initial configuration for 15 time steps is presented in Fig 12b. A CA rule can be understood as a complete set of decision rules whose conditions incorporate all possible combinations of cell values in the given local neighborhoods (here of size 3) and outcomes determine the values of the considered cells (usually central cells in the local neighborhood) at the next time step. Graphical representation of the same rule is presented in Fig. 12c. The CA rule shown in Fig. 12b-c is applied to the current configuration of cells and determines the new configuration at the next time step (*t=1, 2, 3,…*). The details of the process of determining the new configuration of cells at the next time step are presented graphically in Fig. 12d.

Bottom part of Fig. 12d shows the same initial configuration (*t=0*) as in Fig. 12FIG. 12a. First, a set of local neighborhoods of size 3 is constructed by taking each cell from the initial configuration together with its left and right neighbors and placing them respectively in the middle, left, and right of the lattice defining each local neighborhood (see the set of 6 local neighborhoods of size 3 placed above the initial configuration in Fig. 12d). In this particular example, so-called cyclic boundary conditions are used, meaning that the rightmost cell in the initial configuration becomes the left neighbor of the leftmost cell in the initial configuration, and vice versa (denoted by dashed lines in Fig. 12d). Next, the local neighborhoods created that way are compared to the local neighborhoods shown in the bottom row of Fig. 12b-c. When the two match, the value shown in the top row of Fig. 12b-c defines the new value of the central cell at the next time step. This process is repeated for each local neighborhood and the values obtained are placed in appropriate slots in the new configuration of cells at time *t=1* thus completely defining this configuration. The process is repeated for an arbitrary number of time steps. Fig. 12FIG. 12a shows the results of the iteration process for the first 15 steps.

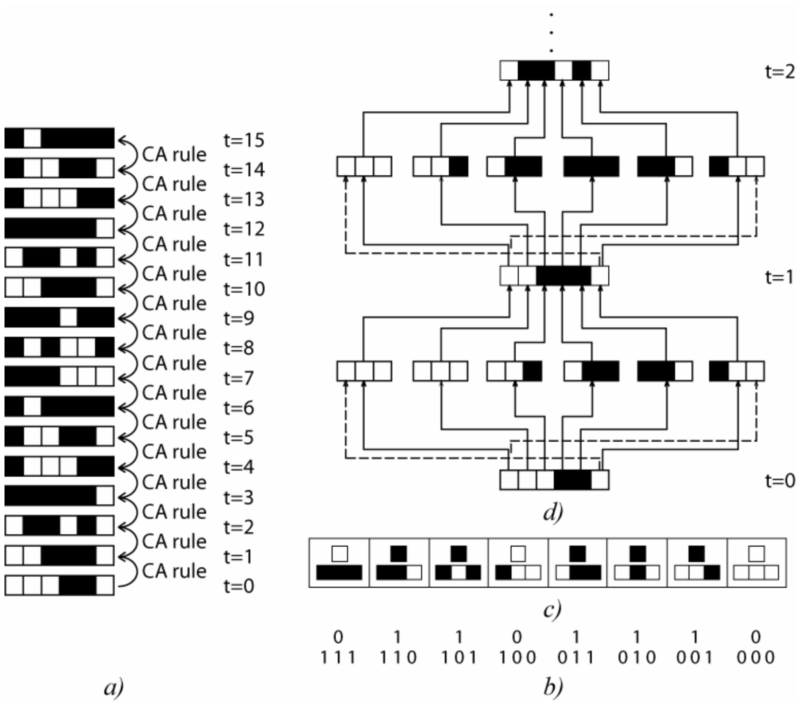


FIG. 12: a) Process of iteration of a simple CA staring with an initial configuration of cells, b) Numerical representation of the CA rule used in a), c) Graphical representation of the same CA rule as in b), d) Details of the mechanism of generation of subsequent cell configurations (t=1,2,etc.,)

The CA rule shown in Fig. 12b specifies all possible (8 in this case) cell values of the local neighborhood of size three (bottom row) and determines the values achieved by the central cells at the next time step (top row). Thus, if we assume the same ordering of the local neighborhoods as shown in the bottom row of Fig. 12b then any elementary CA rule can be defined by a single eight-digit binary number specifying the values achieved by the central cells at the next time step for all possible combinations of cell values in the local neighborhood.

Increasing the number of cell values or size of the local neighborhood causes a rapid growth in the number of possible CA rules and hence a rapid increase of the size of the search space. For example, changing the number of cell values to 3 with the same size of the local neighborhood yields 7,625,597,484,987 possible CA rules compared 256 possible CA rules for elementary CAs. There is, however, a way to significantly reduce it by introducing a concept of a totalistic CA rule. The idea of a *totalistic rule* is to assume that the new value of each cell depends only on the *average* value of cells in the local neighborhood, and not on their individual values (Wolfram, 2002). For example, there are only 2187 possible totalistic CA rules with 3 values and neighborhood of size three compared to 7,625,597,484,987 rules found in the corresponding standard CAs.

4.2 Representations of Steel Structural Systems in Tall Buildings

Emergent Designer implements several types of generative representations of steel structural systems in tall buildings based on cellular automata. Fig. 13 shows an example of a generative representation of a wind bracing system in a tall building utilizing one-dimensional cellular automata (1D CAs). This type of representation consists of two parts: encoding a 'design embryo' and encoding of a 'design rule' (see Fig. 13b). The design embryo forms an initial configuration of the first story of a wind bracing system from which the entire design concept of a wind bracing system (design configuration) is developed. It corresponds to the initial arrangement of cell values in a CA (see Fig. 12). The design rule, which corresponds to the CA rule shown in Fig. 12b-c, is applied to the design embryo and the result of this operation defines a configuration of the second story of a wind bracing system. This process is repeated until the entire design concept of a wind bracing system is fully developed (see Fig. 13d). A detailed description of this type of generative representation can be found in

(Kicinger et al., 2004). Other types of generative representations (e.g., utilizing totalistic CAs and two-dimensional CAs) implemented in *Emergent Designer* are described in detail in (Kicinger, 2004).

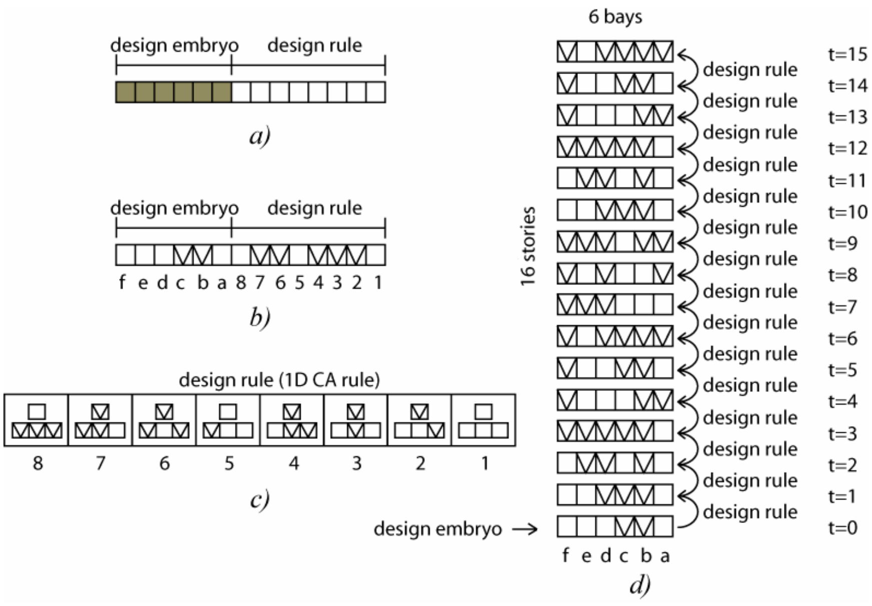


FIG. 13: a) Schematic structure of a generative representation of a steel structural system, b) Genome encoding a specific design concept of a wind bracing system, c) A design rule based on one-dimensional cellular automaton that 'grows' a design concept from the design embryo, d) The process of growing a design concept of a wind bracing system from its generative representation

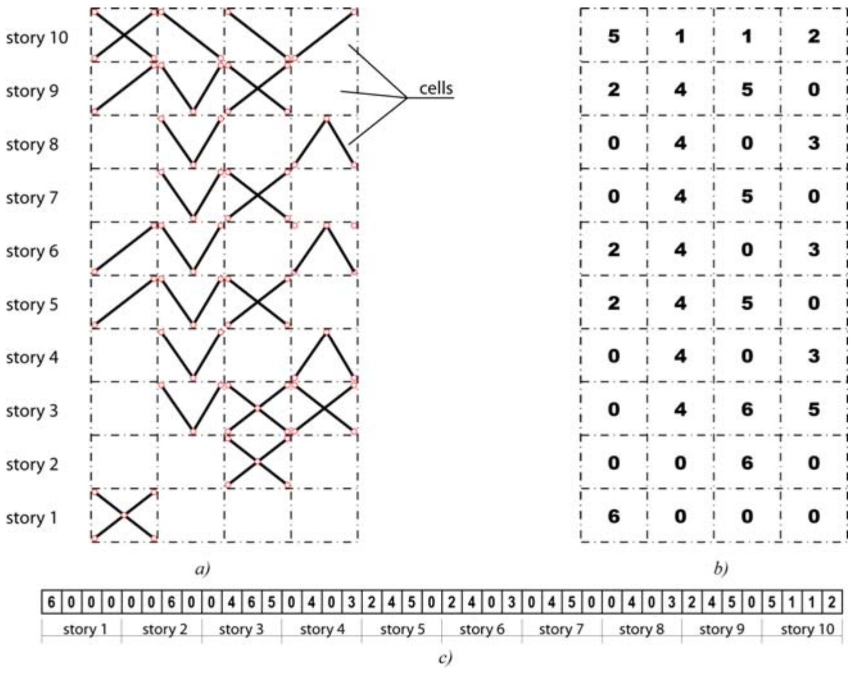


FIG. 14: a) Configuration of structural members of a wind bracing system, b) The same configuration expressed in terms of symbolic attributes, c) Genome of a parameterized representation of a wind bracing system

The generative representation described above is conceptually different than traditionally employed parameterized representation. In the former case, the representation does not encode the entire design concept but rather rules how to grow the design concept from its initial design embryo. In the latter case, there is a one-to-one mapping between the attributes describing an engineering system and its representation. Fig 14 shows that all structural members of a wind bracing system (see Fig. 14a) are directly represented by symbolic attributes (see Fig. 14b) which are subsequently encoded in a linear genome (see Fig. 14c). In this case, no growing/developmental process is necessary to determine the complete configuration of a wind bracing system because it is entirely defined by the genome.

4.3 Experimental results

Emergent Designer has been used to conduct several design experiments in the domain of steel structural systems in tall buildings. Experimental parameters and their values used in these experiments are presented in TABLE 1. It shows that 30-story buildings with 5 bays were investigated. The bay width was equal to 20 ft (6.01 m) and the story height was equal to 14 ft (4.27 m). In the experiments, 7 types of wind bracing elements, 2 types of beams, and 2 types of supports were considered. These parameters could be easily specified using *Emergent Designer*'s GUI as shown in Fig. 5 (left). In the conducted experiments, several types of representations of steel structural systems were investigated, including parameterized representations and generative representations based on one-dimensional CAs (standard and totalistic). These representations and their parameters were determined by *Emergent Designer*'s *Representation and Decomposition Component* presented in Fig. 5 (right).

TABLE 1: Experimental parameters and their values

Parameter	**Value(s)**
Domain parameters:	
Number of bays	5
Number of stories	30
Bay width	20 feet (6.01 m)
Story height	14 feet (4.27 m)
Distance between transverse systems	20 feet (6.01 m)
Structural analysis method	first order
Beams	pinned, fixed
Column	fixed
Supports	pinned, fixed
Wind bracings	no bracing, diagonal bracing (/), diagonal bracing (\), K bracing, V bracing, simple X bracing, and X bracing
CA representation parameters:	
CA type	1D, 1D totalistic
CA neighborhood radius	1
Number of CA cell values	7 (bracings), 2 (beams)
Evolutionary algorithm parameters:	
EA	ES
Pop. sizes (parent, offspring)	(1,25), (5,25), (1,125), (5,125)
Generational model	overlapping
Selection (parent, survival)	(uniform stochastic, truncation)
Mutation rate	0.05, 0.1, 0.3, or 0.5
Crossover (type, rate)	(uniform, 0.0), (uniform, 0.2), (uniform, 0.5)
Fitness	weight of the steel structure (minimization problem)
Initialization method	random
Constraint handling method	death penalty (infeasible designs assigned 0 fitness)
Simulation parameters:	
Termination criterion	1000 fitness evaluations
Number of runs	5 (in each experiment)

Fig. 6 shows the range of available sizing and topology optimization algorithms offered by the system's *Concept Generation and Optimization Component* and the flexibility of specifying their parameters. Thus, it can be used to conduct extensive experimental studies investigating the influence of certain parameters, and their combinations, on the performance of design processes.

The evaluation model employed in the conducted experiments was specified using *Evaluation and Simulation Component* presented in Fig. 7. The top part of Fig. 7 shows the objective with respect to which the designs were optimized as well as types of loads and load combinations applied in the structural analysis. The progress of design processes can be monitored graphically in real-time as shown in the bottom part of Fig 7.

When the design experiment is completed, a detailed report summarizing its results as well as applied parameters and their values can be generated, similarly as presented in Fig. 11. Then, advanced statistical analysis tools and methods may be applied to more accurately analyze the experimental data and compare results of several experiments. For example, Fig. 15 shows a comparison of the performance of parameterized and generative representations. The experimental data presented in Fig. 15 have been initially processed by *Advanced Statistical Analysis Component* to more accurately estimate average performance and confidence intervals. Specifically, the 95% confidence intervals were calculated using Johnson's modified t test (Johnson, 1978) and subsequently visualized using *Emergent Designer*'s *Visualization component*.

Emergent Designer also supports visualization of generated designs using SODA's visualization capabilities. For example, Fig. 16 compares the structural shaping patterns produced by generative representations based on CAs and the ones produced by parameterized representations. This type of visual analysis allowed distinguishing significant qualitative differences produced by these two types of representations, a fact that is important from the structural design perspective.

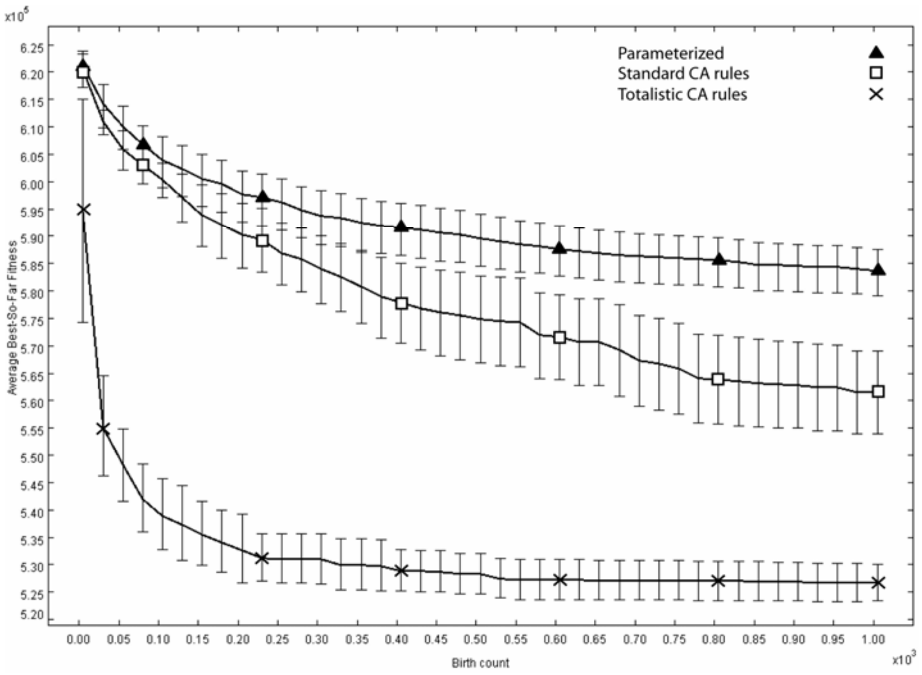


FIG. 15: Typical average best-so-far curves obtained in the design experiments with parameterized and generative representations of the entire steel structural systems in tall buildings

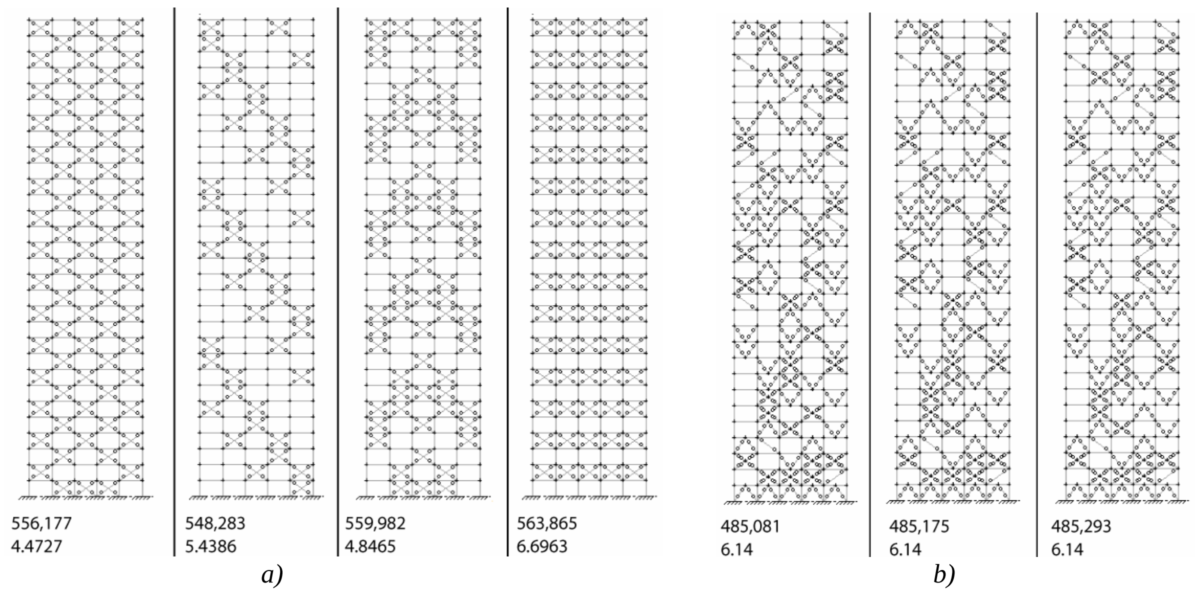


FIG. 16: Comparison of the structural shaping patterns produced using a) generative representations, b) parameterized representations

5. CONCLUSIONS

Engineering design is in the process of transition to meet the globalization and innovation challenges of the 21st century. The traditional focus on quantitative/numerical aspects of engineering is simply insufficient when dealing with these challenges. In particular, our analytical approaches to design and creativity limit the progress and impact on the society. Thus, new perspectives, methods, and tools are needed to successfully address these problems.

One of such emerging tools, called *Emergent Designer*, is discussed in this paper. It is a complex computer system which belongs to a new generation of design tools being developed at George Mason University. They result from several years of intense research on evolutionary designing, on complex adaptive systems, and on cellular automata. It allows conducting design experiments in the area of structural design and for the analysis of their results using various statistical and time series analysis methods. It implements state-of-the-art representations supporting generation of novel design concepts and efficient mechanisms for their subsequent optimization at the topological and sizing levels.

Emergent Designer can be considered as a powerful knowledge acquisition tool. It can be used by an experienced researcher, or an engineer, to acquire knowledge about a given engineering system and its behavior through conducting various kinds of experiments and generating data about its many characteristics. Such information, integrated and analyzed using the tools implemented in *Emergent Designer* and subsequently interpreted by an expert, may become a body of knowledge on a given engineering domain.

As of now, *Emergent Designer* is intended mainly for research purposes, but in the future it will be adapted for the practical engineering design applications. The initial experience with *Emergent Designer* has clearly demonstrated that the concept of the integrated research and design support tool is feasible. It is too early to provide a balanced evaluation of the system’s advantages and disadvantages, but the authors believe that its use will soon lead to various discoveries related to design and engineering creativity, particularly related to emergence.

6. REFERENCES

Arciszewski T. and De Jong K.A. (2001). Evolutionary computation in civil engineering: research frontiers. Eight International Conference on Civil and Structural Engineering Computing (Topping B. H. V., ed.), Saxe-Coburg Publications, Eisenstadt, Vienna, Austria.

Arciszewski T., Michalski R.S. and Wnek J. (1995). Constructive induction: the key to design creativity. Preprints of the Third International Round-Table Conference on Computational Models of Creative Design (Gero J. S. and Maher M. L., eds.), Heron Island, Queensland, Australia, 397-426.

Axtell R. and Epstein J.M. (1996). Growing artificial societies: social science from the bottom, MIT Press.

Baron P., Fisher R., Mill F., Sherlock A. and Tuson A. (1997). A voxel-based representation for the evolutionary shape optimization of a simplified beam: a case-study of a problem-centered approach to genetic operator design. 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2).

Bentley P.J. (1999). An introduction to evolutionary design by computers. Evolutionary Design by Computers (Bentley P. J., ed.), Morgan Kaufmann Publishers, San Francisco, CA.

Bentley P.J. and Kumar S. (1999). Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. Genetic and Evolutionary Computation Conference (GECCO'99) (Banzhaf W., Daida J., Eiben A. E., Garzon M. H., Honavar V., Jakiela M. J. and Smith R. E., eds.), Morgan Kaufmann Publishers, Orlando, Florida, USA, 35-43.

De Jong K.A. (to appear). Evolutionary computation: a unified approach, MIT Press, Cambridge, MA.

d'Humieres D. and Lallemand P. (1986). Lattice gas automata for fluid dynamics. Physica, Vol. 140A, 326-335.

Franceschetti D.R., Campbell B.W. and Hamneken J.W. (1992). Hamming nets, Ising sets, cellular automata, neural nets and random walks. American Journal of Physics, Vol. 61, 50-53.

Frazer J. (1995). An evolutionary architecture, Architectural Association Publications, London.

Funes P. and Pollack J.B. (1999). Computer evolution of buildable objects. Evolutionary design by computers (Bentley P. J., ed.), Morgan Kaufmann Publishers, San Francisco, CA.

Gerhadrt M. and Schuster H. (1989). A cellular automaton describing the formation of spatially ordered structures in chemical systems. Physica, Vol. 36D, 209-221.

Grierson D.E. (1989). Computer-automated optimal design for structural steel frameworks. NATO ASI Conference on Optimization and Decision Support Systems in Civil Engineering (Topping B. H. V., ed.), Kluwer Academic Publishers, Edinburgh, UK, 327-354.

Hajela P. and Kim B. (2001). On the use of energy minimization for CA based analysis in elasticity. Structural and Multidisciplinary Optimization, Vol. 23, No. 1, 24-33.

Inou N., Uesugi T., Iwasaki A. and Ujihashi S. (1998). Self-organization of mechanical structure by cellular automata. Fracture and strength of solids. Part 2: Behavior of materials and structure (Tong P., Zhang T. Y. and Kim J., eds.), 1115-1120.

Johnson N.J. (1978). Modified t tests and confidence intervals for asymmetrical populations. Journal of the American Statistical Association, Vol. 73, No. 363, 536-544.

Kane C. and Schoenauer M. (1995). Genetic operators for two-dimensional shape optimization. Artificial Evolution (Alliot J.-M., Lutton E., Ronald E., Schoenauer M. and Snyers D., eds.), Springer Verlag.

Kessler D.A., Levine H. and Reynolds W.N. (1990). Coupled-map lattice model for crystal growth. Physics Review, Vol. 42A, No. 6125.

Kicinger R. (2004). Emergent Engineering Design: Design creativity and optimality inspired by nature, Ph.D. Dissertation, School of Information Technology and Engineering, George Mason University, Fairfax, VA, USA.

Kicinger R., Arciszewski T. and De Jong K.A. (2004). Morphogenic evolutionary design: cellular automata representations in topological structural design. Adaptive Computing in Design and Manufacture VI (Parmee I. C., ed.), Springer-Verlag, London, UK, 25-38.

Kicinger R., Arciszewski T. and De Jong K.A. (2005). Generative design in structural engineering. Proceedings of the 2005 ASCE International Conference on Computing in Civil Engineering, Cancun, Mexico, July

12-15, 2005 (Soibelman L. and Peña-Mora F., eds.), American Society of Civil Engineers Press, Reston, VA.

Kita E. and Toyoda T. (2000). Structural design using cellular automata. Structural and Multidisciplinary Optimization, Vol. 19, 64-73.

Kita E. and Toyoda T. (2001). Truss structural design based on concept of cellular automata. International Association for Shell and Spatial Structures Symposium (IASS2001), IASS, Nagoya, Japan.

Koch P.N., Evans J.P. and Powell D. (2002). Interdigitation for effective design space exploration using iSIGHT. Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 111-126.

Kundu S., Oda J. and Koishi T. (1997). A self-organizing approach to optimization of structural plates using cellular automata. Second World Congress on Structural and Multidisciplinary Optimization (WCSMO-2) (Gutkowski W. and Mroz Z., eds.), Polish Academy of Science, Zakopane, Poland, 173-180.

O'Reilly U.-M., Kangas M. and Testa P. (2000). MoSS: Morphogenetic Surface Structure: a software tool for design exploration. Greenwich 2000: Digital Creativity Symposium, University of Greenwich, London, UK, 71-80.

Stiny G. (1980). Introduction to shape and shape grammars. Environment and Planning B: Planning and Design, Vol. 7, No. 3, 343-351.

Wolfram S. (1994). Cellular automata and complexity: collected papers, Addison-Wesley, Reading, MA.

Wolfram S. (2002). A new kind of science, Wolfram Media, Champaign, IL.

Wolfram S. (2003). The Mathematica book, 5th edition, Wolfram Media, Champaign, IL.