AN EVOLUTIONARY SYSTEM FOR DESIGN EXPLORATION

PATRICK H. T. JANSSEN
National University of Singapore

ABSTRACT: This paper reports on the development of a multi-objective evolutionary developmental design environment, called EDDE. The goal of the system is to make it easier for designers to use evolutionary techniques to explore design possibilities. The system consists of a generic evolutionary core into which a set of design specific scripts need to be plugged in. The system uses a web based client-server architecture that can either be run on a single computer or on multiple computers in parallel. Initial experiments have shown the system to be effective in evolving designs.

KEYWORDS: Generative, evolutionary, genetic, search, optimisation, simulation

RÉSUMÉ : *Cet article présente le développement d'un environnement de conception évolutive, multi objectif, appelée EDDE. Le but du système est de faciliter l'usage de techniques évolutives par les concepteurs pour explorer les possibilités de conception. Le système se compose d'un noyau générique évolutif dans lequel une série de scripts spécifiques doivent être connectés. Le système utilise une architecture WEB client-serveur qui peut être exécutée sur un seul ordinateur ou sur plusieurs ordinateurs en parallèle. Les premières expériences ont démontré que le système est efficace dans la conception évolutive*

MOTS-CLÉS : *Génératif, évolutif, génétique, recherche, optimisation, simulation*

1. INTRODUCTION

Evolutionary design systems are loosely based on the neo-Darwinian model of evolution through natural selection. A population of individuals is maintained and an iterative process applies a number of evolutionary steps that create, transform, and delete individuals in the population.
Each individual has a genotype representation and a phenotype expression: the genotype representation encodes information that can be used to create a model of the design, while the phenotype expression is the actual design model. The phenotypes are evaluated, and based on these evaluations individuals is assigned relative fitness values. By repeatedly breeding the fittest individuals, the population is able to evolve and adapt to satisfy the objectives set by the designer. Breeding involves creating new genotypes from parent genotypes using 'genetic operators' such as crossover and mutation.

The evolutionary process replaces the traditional process of design exploration, in which typically only a small number of options will be considered. The advantage of the evolutionary system is that it is able to cyclically develop and evaluate large populations of design variants. This approach has proved to be well suited to design processes that are typically divergent and exploratory (Frazer and Connor 1979; Graham *et al.* 1993; Frazer 1995; Bentley 1999; Bentley and Corne 2002; Rosenmann 1996; Shea 1997; Coates *et al.* 1999; Funes and Pollack 1999; Sun 2001; Janssen 2004).

This paper presents a multi-objective evolutionary developmental design environment, called EDDE, created with the aim of making it easier for designers to use evolutionary algorithms. The paper consists of the following sections: section 2 introduces evolutionary design research, section 3 consists of an overview of EDDE; section 4 reports on initial experiments carried out using EDDE; and section 5 draws conclusions and discusses future work.

2. EVOLUTIONARY DESIGN RESEARCH

Evolutionary design encompasses a wide range of research areas exploring how design variability and design evolvability is affected by the fundamental evolutionary steps of development, evaluation, survival, and reproduction.

This has led to many evolutionary design systems differing significantly from more canonical genetic algorithms. In some cases, researchers have been experimenting with novel types of evolutionary steps, testing out new variants for the rules used to perform the evolutionary steps. Other researchers have focused on the overall structure of the evolutionary algorithm, and have been inventing variants that are better suited to particular types of problems.

Two key evolutionary steps that are particularly relevant to the design domain are the developmental step and the evaluation step. With canonical genetic algorithms, the process of fitness evaluation consists of a four sub-processes: 1) the genotype is mapped to a set of parameters, 2) the parameters are used to generate a phenotype, 3) the phenotype is used to perform a set of evaluations, and 4) the evaluations are used to calculate the fitness. Many researchers have found it useful to explicitly define the phenotype and the evaluations as distinct from the fitness. This is especially the case in the design field where the phenotype represents the actual design, and evaluations represent design performance.

The development step creates a phenotype for each new genotype. In many cases, this process is a straightforward mapping process. In other cases, the developmental process may be much more complex. In the design domain, this process may involve an iterative rule based growth processes from which the phenotype will emerge. Various researchers have argued that the inclusion of an explicit developmental step is advantageous. (Frazer 1979, Angeline 1995, Bentley 1999, O'Neill and Ryan 2000, Janssen 2004)

The evaluation step produces an evaluation for each new phenotype. The evaluations are created by assessing the performance of the phenotype with respect to a particular objective. In the design domain, these evaluations may consist of very complex types of calculations that simulate the performance of the design of a period of time under different conditions. Since these evaluations will need to be performed thousands of times for different design variants, the evaluation step will often become the main bottleneck in the speed of execution of the evolutionary algorithm.

3. OVERVIEW OF EDDE

A multi-objective evolutionary developmental design environment has been implemented, called EDDE, and is available under an Open Source BSD license at http://www.evo-devo-design.net. EDDE is that latest version of a number of previous evolutionary systems that have been developed and tested. It has been implemented in the Python programming language.

3.1. EDDE Design Goals

EDDE has been design with one main goal in mind: to make it easier for designers to use evolutionary design exploration tools. When considering the various hurdles that designers are facing, then the two most important are the complexity of the software and the speed of execution.

- Evolutionary design systems typically have to be custom made for each new design scenario. This is because most existing evolutionary systems are typically not flexible enough, and therefore severely limit the types of design that can be evolved. Implementing a custom made evolutionary systems using programming libraries is typically too complex as it requires advanced programming and other computational skills.

- Evolutionary design systems typically have computationally demanding design evaluation routines that may invoke advanced analysis and simulation software. Evaluating designs can therefore be relatively slow when compared to other domains, which means that the overall execution time can be prohibitively slow – sometimes measured in days.

EDDE aims to overcome these two hurdles by creating an evolutionary system that is both flexible and fast. It should be noted from the outset that this is not an easy task, as is highlighted by the well known No Free Lunch theorem, which states that any two optimization algorithms are equivalent when their performance is averaged across all possible problems. EDDE attempts to sidestep the No Free Lunch theorem by using a software architecture that can be run in parallel on multiple computers.

3.2. The evolutionary process

Evolutionary algorithms differ significantly from one another in how the evolutionary process is defined. Two key differences are the evolution mode and the control structure.

The evolution mode refers to how the evolutionary steps process individuals in the population. In nature, the evolutionary steps are applied in parallel. At any moment in time, some organisms may be in the process of being born, others may be in the process of living, and yet others may be in the process of dying. This natural evolutionary process may be described as an asynchronous evolution mode, in that the life-cycles of the individuals in the population are not synchronised. Most evolutionary algorithms use a synchronous evolution mode. In this case, individuals in the population are processes in a synchronised manner, with each evolution step being applied to the whole population in turn. The synchronous application of all the evolutionary steps to the individuals in the population is described as a generation.

The control structure refers to how the evolutionary steps are controlled. In nature, the evolutionary process is an emergent phenomenon that arises as a result of the behaviour of individual organisms. The evolutionary steps are applied locally and independently from one another. This is referred to as a decentralised control structure. Most evolutionary algorithms use a centralised control structure, whereby the application of the evolutionary steps to individuals in the population is centrally orchestrated.

3.2.1. The EDDE evolutionary algorithm

In the case of EDDE, the evolutionary process uses an asynchronous evolution mode in combination with a decentralised control structure. The system is implemented using a client-server architecture: the server maintains a population of designs, and clients carry out the evolutionary steps by downloading individuals from the server, processing these individuals, and then uploading the results back to the server. The server does not control any of the evolutionary steps - instead, it passively waits to be contacted by the clients that execute the evolutionary steps.

This approach results in a population consisting of a mix of design variants at four different life stages: 1) at the first stage, an individual only has a genotype; 2) at the second stage, it will have gained a phenotype; 3) at the third stage it will have gained a set of performance scores; 4) and finally at the fourth life stage it will have survived one or more attempts to kill it.

Each of the clients requires design variants at a particular life stage. It is the responsibility of the server to ensure that each client is sent design variants at the appropriate life stage, referred to as 'candidates'. When the server receives a request from one of the clients, it will first identify all possible candidates in the population and will then randomly select the required number of design variants from these candidates and send them to the client.

One consequence of using a decentralised approach is that the population size will naturally tend to vary over time. The change in the population size will depend on the rate at which the reproduction client adds new design variants to the population and the rate at which the survival client culls design variants. If some form of control were not applied, the population size would be unlikely to remain stable. The server therefore controls the size of the population by selectively rejecting requests from either the reproduction and survival clients at appropriate times.

3.2.2. Comparison with other evolutionary algorithms

The great majority of evolutionary algorithms use a synchronized evolution mode in combination with a centralised control structure. The four main types are genetic algorithms, evolution strategies, evolutionary programming, and genetic programming. Of these, genetic algorithms are the best known. The synchronised mode encompasses three more specific approaches commonly referred to as the generational, elitist and steady-state evolution modes.

The asynchronous decentralised approach used by EDDE seems to be very rare – in fact, a review of the literature did not discover any other evolutionary systems using this approach. A number of systems were found to use asynchronous centralized architectures, (Cantu-Paz 1997, 1998; Alba and Troya 1999; Nowostawski and Poli 1999), while others were found to use other specialised types of architectures such as the distributed peer-to-peer models (Chong and Langdon 1999; Arenas *et al.* 2002).

The asynchronous decentralised approach has been taken to ensure that EDDE is both scalable and robust. First, an asynchronous evolutionary process allows the evolutionary steps to be executed in a totally parallelized manner. This can significantly reduce the execution time and is highly effective in situations where the development and evaluation steps are costly. This is especially pertinent for evaluation clients, since the simulation or analysis of the performance of a building design can be time consuming. Second, a decentralised control structure in combination with a client-server model allows client computers to be easily added and removed from the evolutionary process. The architecture is robust in that it can cope with failure of client systems in a graceful manner.

In addition, it should also be noted that the asynchronous decentralised approach is actually a superset of many other more conventional approaches, such as generational, elitist, and steady-state GAs. Within EDDE, these more conventional types of evolutionary algorithm can be run simply by setting up the design schema in the appropriate way.

3.3. Customizing EDDE

A key concept in the proposed evolutionary design approach is an algorithmic model developed by a designer that has both built-in variability as well as built-in constraints. This algorithmic model is referred to as the 'design schema' and the designer creating the schema is referred to as the 'schema author'.

3.3.1. Design schemas

The design schema captures the essential and identifiable character of a varied family of designs. It encompasses those characteristics common to all members of the family, possibly including issues of aesthetics, space, structure, materials and construction. Although members of the family of designs share these characteristics, they may differ considerably from one another in overall organization and configuration (Janssen 2004).

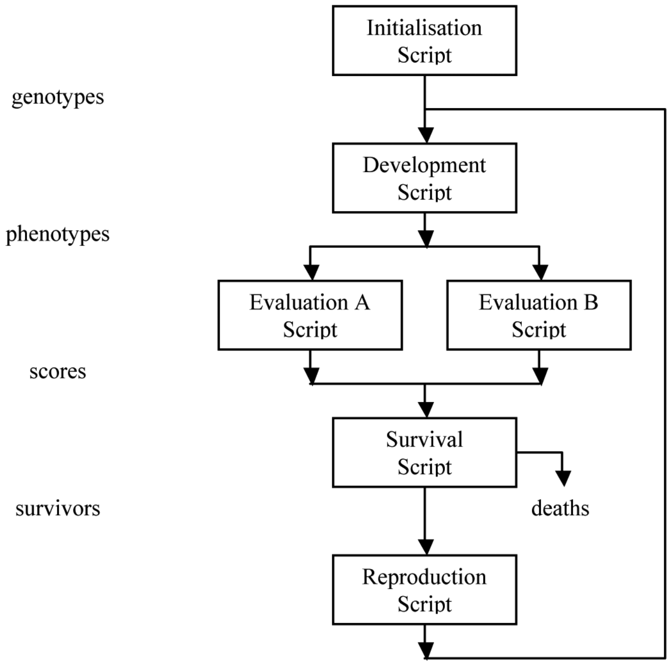
Broadly, the schema based approach consists of two main stages:

- In the schema codification stage, the design schema is encoded in a form that can be used by the evolutionary system. This involves creating a set of design specific schema scripts that can be executed by the evolutionary system.
- In the design evolution stage, the encoded schema is used by the evolutionary system to evolve a population of design variants. This will result in a Pareto-optimal set of design variants, which can be explored for further design development.

  EDDE consists of a generic evolutionary core into which design-specific schema scripts can be plugged in. The generic evolutionary core performs all the tasks related to starting, stopping, and managing individual evolutionary runs. The design schema scripts perform the design-specific evolutionary steps. The five scripts are as follows (see Figure 1):

- The initialisation script creates a new population of individuals with genotypes.
- The development script appends a phenotype model to each individual.
- The evaluation scripts append one or more performance scores to each individual.
- The survival script produces a verdict on who survives, and who has to die.
- Finally, the reproduction script produces new genotypes from parents that have survived.

FIGURE 1: *THE FIVE TYPES OF SCRIPTS PERFORMING THE EVOLUTIONARY STEPS. THE TYPE OF DATA THAT EACH SCRIPT CONSUMES AND PRODUCES IS SHOWN BY THE LABELS ON THE LEFT. TWO EVALUATION CRITERIA ARE BEING CONSIDERED, LABELLED AS A AND B. NOTE THAT THE OUTPUT FROM ONE SCRIPT DOES NOT LINK DIRECTLY TO THE INPUT TO THE NEXT SCRIPT - SCRIPTS ARE LINKED ONLY INDIRECTLY VIA THE POPULATION SERVER – SEE FIGURE 2.*



3.4. Running EDDE

In order to run EDDE, the first step is to start the EDDE web server. The user can then log onto the web server using a standard browser and access EDDE's graphical user interface, which allows various server management tasks to be performed.

One of the first tasks that will need to be performed is the pluging-in of the design schema. The user interface allows users upload schema zip files onto the server, and to start and manage multiple evolutionary runs.

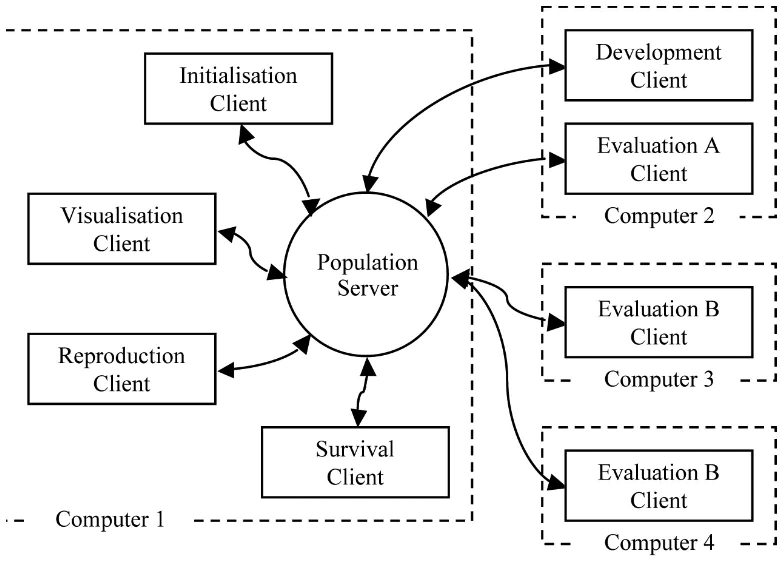
Once the EDDE server has been started and the schema uploaded, each of the client application must then be started. As a minimum setup, the whole system can actually be run on a single computer, in which case the computer will be acting as both a server and a client. Five client applications need to be started – one for each evolutionary step. Once the clients have been started, they will cyclically keep logging onto the server, downloading individuals at particular life stages, executing the scripts to generate new data for those individuals, and uploading them back to the server. Gradually, the population will evolve and adapt.

3.4.1. Running EDDE on multiple computers

In order to run EDDE on multiple computers, the same approach can be taken as above, except that the clients can be running on separate computers.

Once again, five client applications will need to be started – at least one for each evolutionary step. However, multiple clients can be assigned to steps that are particularly slow to execute, such as the development or evaluation steps (see Figure 2). Furthermore, these additional clients can be added and removed from the evolutionary process without requiring and changes to the server, even in the middle of an evolutionary run. This allows computing resources to be flexibly and efficiently allocated in environments where computers are used for multiple purposes such as design offices and research labs.

FIGURE 2: *ONE POPULATION SERVER AND EIGHT CLIENTS RUNNING ON FOUR COMPUTERS. TWO EVALUATION CRITERIA ARE BEING EVALUATED, LABELLED A AND B. IN THE CASE OF B, TWO COMPUTERS ARE BEING USED TO PROCESS INDIVIDUALS IN PARALLEL, BOTH PERFORMING THE SAME EVALUATION.*



4. AN EXAMPLE OF A EDDE SCHEMA

A number of simple example schemas have been developed in order to test and demonstrate the EDDE approach. Initially, a series of non-architectural cases were developed, using common optimisation problems from the research literature. These included the one-armed bandit problem, the water buckets problem, and the travelling salesman problem. These tests allow the functionality of EDDE to be tested, and confirmed that the system was actually successful in evolving solutions.

Following these tests, a simple architectural example was then developed that allowed EDDE to be further tested, and to explore how other existing applications could be integrated into EDDE. This schema consisted of optimizing the size of a series of roof overhangs in order to minimize two conflicting objectives. This schema, referred to as the overhangs schema, is presented in more detail below.

4.1. The Overhangs Schema Scripts

The problem defined by the overhangs schema is to optimise the size of a series of overhangs shading the windows in the walls of a small single-spaced concrete building, shown in Figure 3. The space is a 5 x 5 x 5 cubic meter space with an overhanging roof. The design objectives were to minimise both the annual energy consumption and construction cost. The Florida (USA) weather file was used to perform the evaluations.

4.1.1. Representations

The two most important representations used by the schema are the genotype representation, and the phenotype representation.

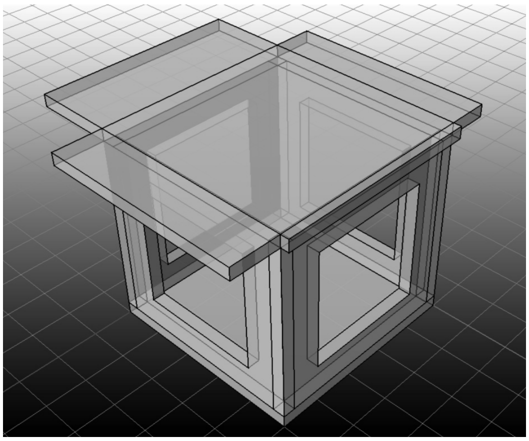
The genotype representation consists of a list of four integers that represent the length that the overhangs project out over the windows. Since the maximum overhand was set to 2m, these numbers could range from 0 to 2000.

The phenotype representation consists of two 3D models of the buildings using different file formats, which are related to the two programs required by the schema. The Rhino3D CAD modeling program is used by the development script in order to generate a 3D model. The simulation program EnergyPlus is use by the evaluation script to calculate the annual energy consumption of the space. The first phenotype model is created as a 3dm file, the format used by Rhino3d. The second phenotype model is created as an idf file, the format used by EnergyPlus.

TABLE 1. *THE FIVE SCRIPTS PERFORMING THE EVOLUTIONARY STEPS.*

Script name	Script description
Initialisation script	For the initialization script, there is no input and the output consists of a population of 100 random genotypes, each consisting of a list of four random integers between 0 and 2000.
Development script	The development script first uses the Rhino3d api to call various CAD functions in order to generate a detailed model of the building, including wall thicknesses and window openings. This model is saved as a 3dm file, to be used by one of the evaluation scripts. The development script then extracts coordinate data out of the Rhino3d model in order to produce a simplified model with additional parameters for the EnergyPlus simulation program. This model is saved as an idf file, which is a text based EnergyPlus input file.
Evaluation scripts	The script for evaluating the energy consumption first executes the EnergyPlus simulation program, providing the idf file that it receives as part of the input phenotype, and when the simulation has completed, it reads the resulting output files. Based on the results in these files, it then calculates the annual energy consumption. The script for evaluating construction cost uses a simplified method that assumes that the construction cost will be proportional to total volume of concrete. The script therefore uses the 3dm model that it receives as part of the input phenotype in order to calculate the volume of concrete.
Survival script	For the survival script, the input consists of 20 design variants that have been fully evaluated. The script uses standard Pareto-ranking methods to rank design variants and to decide which should die and which should survive.
Reproduction script	For the reproduction script, the input consists of 20 design variants that have been through the survival process, and the output consists of 20 new design variants. The script uses crossover and mutation operators to generate new genotypes from the parent genotypes. It does not perform any selection of the parents, but instead simply takes each pair of parents in turn.

FIGURE 3. *A 3D MODEL OF ONE OF THE DESIGN VARIANTS FOR THE OVERHANGS SCHEMA.*



4.2. EDDE Graphs

The EDDE graphical interface includes a series of graphs that dynamically update during the evolutionary process to give feedback to the user about how evolution is progressing. There are three main types of graphs:

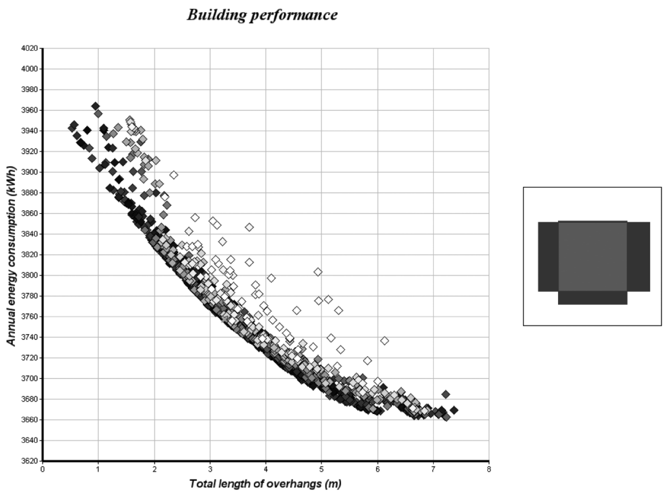
- Progress graphs plot changes in the overall performance scores of the population over time. For multi-objective schemas, multiple graphs will be displayed.
- Pareto graphs plot fitness values for a selected set of design variants. These graphs are therefore not time-based. These types of graphs allow the user to see the Pareto-front.
- Population graphs plot the number of design variants that are waiting to be processed by each script. These graphs allow users to see which evolutionary steps are taking the most time, and as a result where to add more computing power if it becomes available.

4.2.1. Pareto graphs

The Pareto graph for the overhangs schema is displayed in Figure 4. Each point in the graph represents one design variant. In total 10,000 design variants were evolved.

The graphs allow you to view what the design actually looks like. Hovering over a point in the graph with the mouse displays the image of the design on the right hand side as shown below. Clicking on a point in the graph downloads the 3D model from the server and opens the model in the default application.

FIGURE 4. *THE PARETO GRAPH FOR THE OVERHANGS SCHEMA, SHOWING THE ROOF PLAN OF ONE OF THE SOLUTIONS ON THE PARETO FRONT.*



5. CONCLUSIONS AND FUTURE WORK

EDDE is a multi-objective evolutionary developmental design environment that aims to make it easier for designers to use evolutionary techniques in the design process. Initial experiments have show that EDDE is effective in evolving design variants.

However, the overall speed of EDDE was slower than expected. Since the system supports parallelization, this could be overcome by adding more computers, but it was felt that further improvements in performance could be made by changing the architecture of the system.

Further testing revealed that a significant amount of time is taken up with reading and writing files. This is because the coupling between the system and the scripts is based on file input and file output. Each time a script is executed, the following steps occur: 1) the system provides the script with a set of input files; 2) the scripts reads the input files; 3) the scripts processes the data; 4) the script writes a set of output files; and 5) the system fetches the output files. The problem of file reading and writing is made worse by the fact that each file has to be read and written multiple times – by the script, the client, and the server.

Various other problems were also identified, the most important being the lack of flexibility in how the schema is define. EDDE hard-codes the life-stages that an individual in the population will pass through, and the types of scripts that process individuals. This was found to be limiting in many situations. For simple problems, the hard-coded stages and script types were too complex, while for complex problems they were often too simple.

A new system is now under development that aims to overcome the limitations and problems discovered with EDDE. The overall concept of the system is the same, but the underlying architecture has fundamentally changed. In particular, the coupling between the scripts and the system now no longer relies on file reading and writing, and the life stages and script types are now no longer hard coded.

ACKNOWLEDGEMENTS

This research was supported under Australian Research Council's *Discovery Projects* funding scheme (project number DP 0880813). The research assistants working on the project were Jamshid Arshadi, Graham Stratton, Richard Penman, Danny Bishop, and Kong Guan Tan.

REFERENCES

Alba, E. and Troya, J.M., 1999, A Survey of Parallel Distributed Genetic Algorithms, *COMPLEXITY*, 4(4):31–51.

Angeline, P.J., 1995, Morphogenic Evolutionary Computations: Introduction, Issues and Examples, *in* J. McDonnell, B. Reynolds, and D. Fogel, (eds), *Evolutionary Programming IV: The Fifth Annual Conference on Evolutionary Programming*, pp 387–401, MIT Press.

Arenas, M.G., Collet, P., Arenas, M.G., Collet, P., Eiben, A.E., Jelasity, M., Merelo, J.J., Paechter, B., Preuß, M., and Schoenauer, M., 2002, A Framework for Distributed Evolutionary Algorithms, *in* J.J. Merelo Guervos, P. Adamidis *et al.* (eds), *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pp. 665–675, Springer-Verlag.

Bentley, P.J. (ed.), 1999, *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, CA.

Bentley, P.J. and Corne, D.W. (eds), 2002, *Creative Evolutionary Systems*, Academic Press, London, UK.

Caldas, L., 2001, *An Evolution-Based Generative Design System: Using Adaptation to Shape Architectural Form*, Doctoral dissertation, Massachusetts Institute of Technology.

Cantu-Paz, E., 1997, A survey of Parallel Genetic Algorithms, *in Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141-171.

Cantu-Paz, E., 1998, Designing Efficient Master-Slave Parallel Genetic Algorithms, *in* J. Koza, W. Banzhaf, J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. Fogel, D.E. Goldberg, H. Iba and R. Riolo (eds), *Genetic Programming: Proceeding of the Third Annual Conference*, San Fransisco, CA. Morgan Kaufmann.

Coates, P., Broughton, T. and Jackson, H., 1999, Exploring Three-Dimensional Design Worlds Using Lindenmayer Systems and Genetic Programming, *in* P.J. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, CA., pp. 323-341.

Frazer, J.H., 1995, *An Evolutionary Architecture,* AA Publications, London.

Frazer, J.H. and Connor, J., 1979, A Conceptual Seeding Technique for Architectural Design, *in Proceedings of International Conference on the Application of Computers in Architectural Design and Urban Planning (PArC79)*, pp. 425–434, Berlin, AMK.

Funes, P. and Pollack, J., 1999, Computer Evolution of Buildable Objects, in P.J. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco, CA., pp. 387-403.

Graham, P.C., Frazer, J.H. and Hull, M.C., 1993, The Application of Genetic Algorithms to Design Problems with Ill-Defined or Conflicting Criteria, *in* R. Glanville and G. de Zeeuw, (eds), *Proceedings of Conference on Values and, (In) Variants*, pp. 61-75.

Janssen, P.H.T., 2004, *A Design Method and a Computational Architecture for Generating and Evolving Building Designs*, Doctoral dissertation, School of Design Hong Kong Polytechnic University (submitted October 2004).

Janssen, P.H.T., Frazer, J.H. and Tang, M.X., 2005, Generative Evolutionary Design: A Framework for Generating and Evolving Three-Dimensional Building Models, *in Proceedings of the 3rd International Conference on Innovation in Architecture, Engineering and Construction (AEC 2005).*

Nowostawski, M. and Poli, R., 1999, Parallel Genetic Algorithm Taxonomy, *in* L.C. Jain (ed.), *Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering* Systems (KES'99), pp. 88-92, Adelaide, IEEE Press.

Chong, F.S. and Langdon, W.B., 1999, Java Based Distributed Genetic Programming on the Internet, *in* W. Banzhaf, J. Daida *et al.* (eds) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pp. 1229, Orlando, FL, Morgan Kaufmann.

O'Neill, M. and Ryan, C., 2000, Incorporating Gene Expression Models into Evolutionary Algorithms, *in* A. Wu (ed.), *Proceedings of GECCO 2000 Workshop on Gene Expression*, pp. 167-173, San Francisco, CA, Morgan Kaufman Publishers.

Rasheed, K.M., 1998, *GADO: A Genetic Algorithm for Continuous Design Optimization*, Doctoral dissertation, Department of Computer Science, Rutgers University, New Brunswick, NJ, Technical Report DCS-TR-352.

Rasheed, K.M. and Davidson, B.D., 1999, Effect of Global Parallelism on the Behaviour of a Steady State Genetic Algorithm for Design Optimization, *in Proceedings of the Congress on Evolutionary Computation (CEC'99)*, vol. 1, pp. 534-541, IEE Press.

Rosenman, M.A., 1996, An Exploration into Evolutionary Models for Non-Routine Design, *in AID'96 Workshop on Evolutionary Systems in Design*, pp. 33-38.

Shea, K., 1997, *Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search*, Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Sun, J., 2001, *Application of Genetic Algorithms to Generative Product Design Support Systems*, Doctoral Dissertation, School of Design, Hong Kong Polytechnic University.