HIER-HEIR: An Evolutionary System with Hierarchical Representation & Contextual Operators applied to Fashion Design

Abhinav Malhotra
Dept. of Electronics and Communication Engg.
Netaji Subhas Institute of Technology
New Delhi, India
abhinav.malhotra89@gmail.com

Varun Aggarwal
Statistics and Machine Learning Team
Aspiring Minds, Gurgaon
India
varun.aggarwal@gmail.com

ABSTRACT

There has been considerable interest in using evolutionary algorithms based techniques to design creative systems. However, these techniques are either too 'creative' and violate design constraints of the domain, or, those catering to a limited search space, but operating within design constraints. Our new evolutionary system 'HIER-HEIR', is not only creative (searches a large space effectively), but creates only such designs which are *valid* with respect to the design domain. Inspired by human design methodology, the representation is a hierarchy of components and the variation is *contextual* acting at all levels of the hierarchy intelligently, facilitating effective search in the design space with explicit control over exploitation and exploration. We have explained our technique with the metaphor of automatic design of a fashion dress in this paper. The experimental results validate our hypotheses with regard to the system. With regard to previous work, our technique is new both with regard to previously published hierarchical systems and those designed for evolving fashion designs. (*Track: Real World Applications*)

ACM Classification Keywords: I.2.4 Knowledge Representation Formalisms and Methods.

General Terms: Algorithms, Design, Experimentation, Human Factors.

Keywords: Genetic Programming, Evolutionary Design, Fashion Design, Hierarchical Representation, Interactive Evolution, Strongly Typed Genetic Programming(STGP).

1. INTRODUCTION

Automatic design of creative and engineering systems using evolutionary algorithms has been of considerable interest [10, 5, 6, 4, 3, 7]. For evolving designs which are useful in a real scenario, these techniques require not only to be creative, but the resulting design must follow certain design constraints to allow actual fabrication and use. The current systems for evolutionary design suffer from a few challenges. While some algorithms design very flexible and creative systems [3, 4], others look at a coarse library of few designs or degenerate to simple parameter optimization for set designs [8, 10, 9]. The second challenge with flexible systems is that they not only lead to many *invalid* designs in the course of evolution [7], but lack of direct mapping between genotype and phenotype leads to a coarse fitness landscape [2] and possible random search.

Our aim is to design an algorithm (i) which creates designs from a large design search space, (ii) has a rich set of variation operators capable of creating every valid design in the search space, (iii) without resulting in any invalid design, (iv) having a smooth fitness landscape (v) with control over exploitation and exploration. Such a system can be used to evolve designs which are not only creative, but also respect the constraints of the specific domain of design.

We have identified that encoding a design as a hierarchy of components solves the challenges discussed above. The knowledge of the hierarchy, the do's (variations) and don'ts (constraints), at each level of the hierarchy is what enables a designer to make *valid* yet creative designs. Whereas changing components at a higher level of hierarchy helps the designer take larger jumps in the search space, tweaking components at the lower level of the hierarchy helps in fine-grained search. The designer respects the component integrity at each level of hierarchy and thus never creates an invalid design.

We represent the dress as a hierarchical set of components (as illustrated in Figure 1) where crossovers and mutations happen at different levels of hierarchy, capable of producing all, but only, valid designs in the search space. The level of hierarchy at which the variation happens controls exploration vs. exploitation.

Whereas in [1], only leaf nodes of the tree can be exchanged, in [9], there is no discussion on how to facilitate variation at different hierarchical levels without creating invalid individuals. Our variation operators work at all levels of hierarchy and yet, only create valid designs. With regard to evolution of dresses, the system in [6] uses a flat representation and in [10], the dress is considered as a sketch, for which coordinates can be moved to create a new dress. It is thus evident, that representing a dress as a hierarchy of components is not obvious.

2. APPROACH

Based on the ideas discussed above, we created a hierarchy of components(*a library of parts*) that combine to make a design(i.e. a fashion dress). This hierarchy is shown in figure 1. The design-representation and variation operators preserve this hierarchy at each level. The hierarchical library of parts is represented as a polymorphic class. Every possible dress or genome is an instance or object of this poly-

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

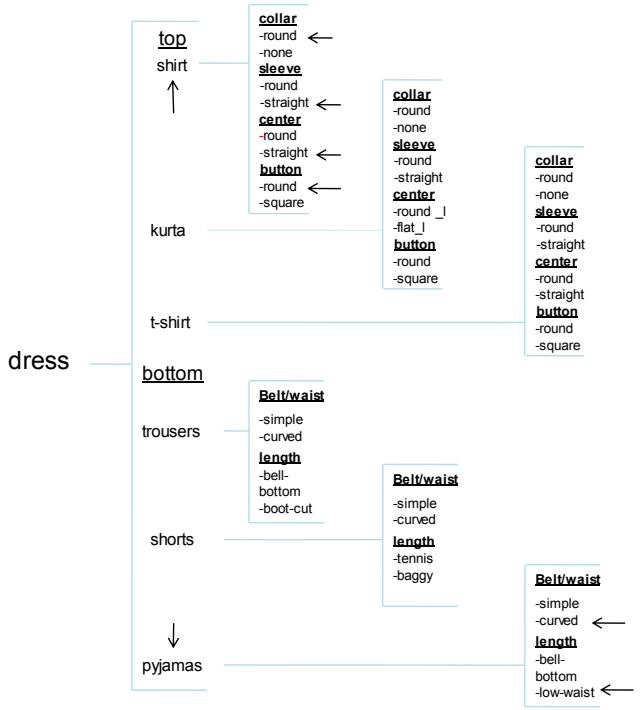

Figure 1: Illustration of Hierarchical Library of Parts, with arrows at each level indicating the particular choice at that level.

morphic class. The genome is a hierarchy of chosen parts. A 'part' is a component/block at each level of the hierarchy. Each part contains the *necessary* and *sufficient* information to construct (or equivalently draw, in this case) the part in 2-D space and to perform effective crossovers and mutations.

The hierarchical mutation operator simply chooses a part of the individual and re-initializes it. The function works iteratively to grow the whole hierarchy of parts under the chosen part. The genome so created is same as the parent except the chosen part and all parts that make the chosen part, ensuring *validity* of designs. The operator provides a knob to control exploitation and exploration in the choice of the level from which the part is mutated. Parts at higher level of the hierarchy (say, a shirt) are mutated to favor exploration(large part of the dress changes), whereas parts at lower level(say, collars) mutate to favor exploitation.

The crossover operator allows flexible exchanges among individuals, yet resulting in *valid* dresses. Valid crossovers are performed by exchanging parts of the same *type* between two individuals. The crossover simply exchanges the parts of same type with all its child parts, which ensures the validity of the new dresses created. For instance, a shirt may be exchanged with a kurta/t-shirt, since they belong to the same type, but not with trousers. Similar to mutation, exploration and exploitation are controlled by the level at which this exchange of parts occurs.

In both mutation and crossover, a probability density function (PDF) determines the probability of choosing a part(to undergo mutation or crossover) at any given level of the hierarchy. The PDF is defined by the algorithm designer and can be varied as a function of generations.

3. RESULTS

The system was displayed as a GUI, showing eight different designs per generation, rated by 28 subjects(17-Male,11-Female) for a total of ten generations. The subjects were asked to fill up a questionnaire after ten generations. Around 86% of the subjects felt that the system was learning their taste of design, and an encouraging 70% of the subjects thought the system found a design they liked. Interestingly, around 60% subjects felt that the system had good diversity amongst its population, and 86% subjects thought the system was learning their taste. This suggests that there is a large search space and the exploration and exploitation dynamics work well through the designed methodology of macro/micro variation operators.

The full-length paper(in PDF) can be accessed here: `http://www.try2doit.com/abhinav/HIER-HEIR.pdf`.

4. REFERENCES

[1] P. J. Bentley and J. P. Wakefield. Hierarchical crossover in genetic algorithms. In *In Proceedings of the 1st On-line Workshop on Soft Computing (WSC1)*, pages 37–42, 1996.

[2] James B. Grimbleby. Hybrid genetic algorithms for analogue network synthesis. In *Proceedings of Congress on Evolutionary Computation (CEC-1999), Washington DC*. Press, 1999.

[3] Martin Hemberg, Una-May O'Reilly, Achim Menges, Katrin Jonas, Michel Gonçalves, and Steven Fuchs. Genr8: Architects' experience with an emergent design tool. In *The Art of Artificial Evolution*, pages 167–188, 2008.

[4] G. S. Hornby, H. Lipson, and J. B. Pollack. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19:703–719, 2003.

[5] R. Kicinger, T. Arciszewski, and K.A. De Jong. Generative design in structural engineering. In *ASCE International Conference on Computing in Civil Engineering*, 2005.

[6] Hee-Su Kim and Sung-Bae Cho. Knowledge-based encoding in interactive genetic algorithm for a fashion design aid system. In *GECCO*, page 757, 2000.

[7] John R. Koza, Forrest H Bennett Iii, David Andre, Martin A, and Frank Dunlap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1:109–128, 1997.

[8] Domine Leenaerts and Wim Kruiskamp. Darwin: Cmos opamp synthesis by means of a genetic algorithm. *Design Automation Conference*, 0:433–438, 1995.

[9] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert. Genetic programming with reuse of known designs for industrially scalable, novel circuit design. In *GPTP V*, pages 159–184, 2008.

[10] Y. Ogata and T. Onisawa. Interactive clothes design support system. In *ICONIP (2)*, pages 657–665, 2007.