Automatic Narrative Evolution
A White Paper

May 26th, 2004

Lutz Hamel[1], Judd Morrissey[2], Lori Talley[2]

[1] Dept. of Computer Science and Statistics, University of Rhode Island,
Kingston, Rhode Island 02881, USA
hamel@cs.uri.edu
[2] Art Institute of Chicago, 37 South Wabash
Chicago, Illinois 60603, USA
{jmorrissey,ctalle}@artic.edu

Abstract. We present a tool that enables complex, unfixed temporal structures in digital narrative and also facilitates the creation of works that are a hybrid of human authorship, structural design, and machine writing. The system has basic similarities with other tools for the development of non-linear and interactive narrative, but it is unique in its form and visual/textual behavior because it takes the printed page as its model rather than traditional hypertext. At its core the system can be considered a creative evolutionary system. In particular, we can consider it to be a co-evolutionary, collaborative (1+1)-ES system. In order to cope with the difficulty of assigning a viable semantics to textual constructs we designed a markup language allowing a composer to assign semantics to bodies of text.

1 Introduction

Here we present a tool that enables complex, unfixed temporal structures in digital narrative [8, 10, 25], and that also facilitates the creation of works that are a hybrid of human authorship, structural design, and machine writing. The resulting system has basic similarities with other tools for the development of non-linear and interactive narrative, but is unique in its form and visual/textual behavior because it takes the printed page as its model rather than traditional hypertext. However, the page is rather distinct from the printed page in that it continuously re-writes itself in response to the interactions with the reader whose actions trigger textual transformations. When new text enters into a portion of the page, it is assimilated both visually/structurally and syntactically, so as to be constantly weaving itself together as it is endlessly reconfigured. The effect on the reader is an immersive experience, one in which previous material on the page lingers like a memory and is registered as the page shifts and the reading progresses. This is very different from the hypertext model where all the text on a page is replaced by new text or a new window is launched that contains a new page of text.

The current approach can be seen as the further development of the "The Jew's Daughter" [17]. The approach in this previous work was to divide the page into three regions with hard coded narrative paths as well as visual/structural configurations. The goal of our new system is to break away from this hard coded approach and make the process as dynamic and free flowing as possible.

The tool is developed as a creative evolutionary system in the sense that the engine which drives the constant reconfiguration and text weaving makes use of concepts in evolutionary computation. Here, the digital narrative is structured as narrative nodes which are directly manipulated by the evolutionary engine. In order to work with these representations effectively and side-step some of the issues which surface when working with textual constructions (e.g., semantic representations of text) we developed a markup language which gives the composer of the digital narrative the ability to identify narrative nodes, assign semantics, and insert user interaction points.

We acknowledge that the structure and functionality of our system is heavily influenced by our own view of the creative process of writing: it is an accumulation of accidental movements in language leading to culminating moments. What is most compelling about the act of writing in our view is this sense of accident, the way in which a writer is led or misled by the work, which, in turn, leads or misleads the reader. But perhaps Picasso said it best: art is the lie that tells the truth[1].

The remainder of the paper is structured as follows. In Section 2 we provide a brief overview of the field of creative evolutionary systems. Section 3 highlights some of the issues involved when working with textual representations. Section 4 provides a high-level description of our system. Our markup language is described in Section 5. Section 6 describes our overall system architecture. We look at further research in Section 7. In Section 8 we take a look at some related systems and we conclude with some remarks in Section 9.

2 Creative Evolutionary Systems

We developed our tool in the context of creative evolutionary systems. Typically, creative evolutionary systems are defined as computer systems that make use of some aspect of evolutionary computation and are designed to aid our own creative processes and/or generate results to problems that traditionally required creative people to find solutions {Rooke, 2002 #2}. Creative evolutionary systems often distinguish themselves from other systems, such as knowledge-based systems, by finding highly innovative and novel solutions.

Evolutionary computation itself can be seen as a search strategy loosely based on biological evolution [12]. As a search strategy, evolutionary computation has been shown to be very effective in searching vast solution spaces given appropriate problem constraints. Figure 1 highlights the relationship between computer science, biology, evolutionary computation, and creative evolutionary systems [5].

[1] This quote is generally attributed to Picasso by folklore.

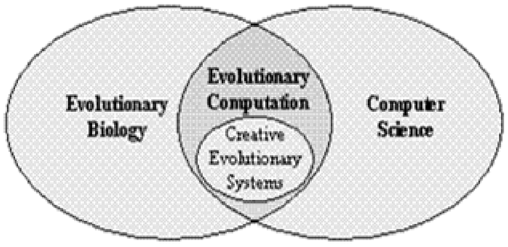

Fig. 1. The relationship between computer science, biology, evolutionary computation, and creative evolutionary systems.

Evolutionary algorithms are usually characterized as iterative, population based algorithms where each individual in the population represents a potential solution to the problem at hand. The iterative nature of these algorithms allows the individuals to evolve under the repeated application of appropriate evolutionary operators such as mutation and crossover (sexual reproduction). At the high level evolutionary algorithms can be described as follows:

1. Determine initial (random) population
2. Select a set of individuals to be parents
3. Perform crossover and mutation
4. Compute the new generation of individuals, new population
5. Does one of the individuals satisfy the problem constraints?
    - Yes – DONE!
    - No – go back to step 2.

In the area of evolutionary computation we can identify three major categories of evolutionary algorithms: a) genetic algorithms which are characterized by their large populations of individuals that evolve under crossover and mutation [12]; b) evolution strategies sometimes called (1+1)-ES, because they often have populations consisting only of a single parent producing a single child at each generation with adaptive genetic operators [22]; c) collaborative evolutionary computation that actively involves the user [6].

Creative evolutionary systems have been successfully applied in a number of areas usually reserved for human creativity using problem constraints that naturally arise in these areas:

- Art & music – aesthetic constraints [7, 23, 24]
- Architectural design - functionality constraints [19]
- Engineering - physical constraints [14, 15]
- Drug discovery – structure constraints [29]

There are two types of evolution we can consider when looking at evolutionary algorithms to solve problems. In g*oal oriented evolution* we attempt to find solutions to problems under given (static) constraints. Here, the sought after solution is characterized by a set of constraints and it is essentially the task of the evolutionary algorithm to search the solution space for a point that satisfies all the given constraints. We can consider this the classical notion of genetic algorithms and evolution strategies as developed by Holland and Rechenberg [12, 22]. In o*pen-ended evolution* or *co-*

evolution a system is "situated", in the sense that it continuously adapts itself to constantly changing constraints. There is no solution, but only the best possible adaptation to the current set of constraints. Typically, both the evolutionary system and the constraints are evolving in response to each other [21].

3 Why is Evolutionary Computation on Text so Difficult?

In building an evolutionary system around text we faced many of the same issues, such as semantic representation and text matching, which researchers in information retrieval also face (e.g. [16, 27]). However, evolutionary computation has some additional challenges. In particular, not only do we need to recognize documents but we also need to be able to *synthesize* coherent documents for the construction of individual solution points which are part of the population in the evolutionary algorithm.

In analyzing the domains where evolutionary algorithms have successfully been applied we found that these domains tend to have highly stylized constraint rules. Consider:

□ Harmony theory in music.
□ Visual composition rules in 2&3 D art.
□ Physics/mathematics in industrial/architectural design.
□ Electrical and mechanical foundations for engineering.
□ Structural chemistry in drug discovery.

Text and narrative, on the other hand, do not possess such stylized constraint rules. The possible different symbol combinations that form words, sentences, paragraphs, pages, *etc.* are almost infinite and typically the legal symbol combinations are governed by semantics rooted in context and common sense which as such is extremely difficult to codify. Even the accessible composition rules based on syntactic contexts can be bent for effect and tension. Given this, a key aspect of our system is to overcome these difficulties and make text manipulation, document synthesis, and narrative evolution accessible.

4 ANE – System Description

Our objective was to create a tool that will enable interactive complex, unfixed temporal structures in digital narrative. In this context, a composer of such work will assemble a set of narrative nodes. Here, each node is a fragment of text written by the composer, garnered off the web, scanned from hard-copy documents *etc.* and constitutes the basic building material for the digital narrative. The idea is that the system will maintain a database of such narrative nodes and will use them appropriately in response to interactions with the user in order to evolve new narrative structures.

When the automatic narrative evolution (ANE) system is first launched, the user is confronted with a page of text. This initial text layout is designed by the composer of the evolving text. Within the page are especially marked words which act as interaction points for the user. Evolution of the narrative is driven by either highlighting or double-clicking one of the marked words. The system interprets this as a constraint and adapts the narrative to this constraint. Adaptation takes the form of swapping out a segment of text and replacing it with new text deemed more appropriate given the current constraint. Once the text has been replaced and absorbed into the visual page the user is confronted by a newly evolved page with a new set of interaction points. The evolution of the text occurs at the granularity of narrative nodes, that is, both the text selected to be deleted from the page as well as the new text which is to replace the old text are based on narrative nodes.

The system is co-evolutionary in the sense that the user's understanding of the text evolves at the same time as the text evolves in order to satisfy the constraints posed by the user. The system does not evolve towards a definite goal, but rather undergoes an "evolutionary arms race" of user understanding vs. constraint resolution.

An interesting aspect of our system is that constraints are not interpreted as syntactic constraints but instead as semantic constraints. For example, double-clicking the word 'danger' which might appear as an interaction point on a page does not mean to evolve the text to contain more of the word danger in order to satisfy the constraint but instead means to evolve the text so that the text is *about* danger. An subtle but important distinction.

This semantics-based evolution is facilitated by our markup language which allows the composer to identify narrative nodes, assign semantics to nodes, and create user interaction points.

5 The Markup Language

The structure of our markup language was heavily influenced by Knuth's TeX language [13]. Consider the following two narrative nodes:

1. Will she disappear? I said to you, "be careful. Today is a strange day, and that was the end of it." I had written impassioned letters that expressed the urgency of my situation. I wrote to you that it would be a violation of our exchange, in fact, a criminal negligence if I were to fail to come through. To hand to you the consecrated sum of your gifts, the secret you imparted persistently and without knowledge, these expressions of your will that lured, and, in a cumulative fashion, became a message.
2. A street, a house, a room.

As these simple examples illustrate, narrative nodes can be paragraphs, simple collections of sentences, a sentence, or just utterances of several words. Figure 2 shows a possible markup of the above narrative nodes.

In our markup language narrative nodes are denoted by the `{\node …}` construct. Individual sentences in each narrative node are denoted with by `{\sen …}`. The `{\lnk …}` construct allows the composer to define sections of text to act as user interaction points. We called these interaction points links due to their conceptual similarity to hypertext links. However, as indicated above, their actual workings are quite different from hypertext links.

```
{\node \meta="incorrigible, wake, refrain, halt, June, begin"
  {\sen Will she {\lnk \meta="vanish" disappear}?}
  {\sen I said to you, "be careful. Today is a strange day, and that
        was the end of it."}
  {\sen I had written impassioned letters that expressed the urgency
        of my situation.}
  {\sen I wrote to you that it would be a violation of our exchange,
        in fact, a {\lnk criminal} negligence if I were to fail to
        come through.}
  {\sen To hand to you the consecrated sum of your gifts, the secret
        you imparted persistently and without knowledge, these
        expressions of your will that lured, and, in a cumulative
        fashion, became a message.}
}

{\node \meta= "disappear, criminal, incorrigible, wake"
  {\sen {\lnk \meta="refrain" A street}, a house, a room.}
}
```

Fig. 2. A simple example of marked up narrative nodes.

The construct that deserves a few words of explanation is the `\meta` construct. The primary purpose of this construct is to assign semantics (or *meta* information) to narrative nodes in the form of a keyword list. The `\meta` construct can also modify a user interaction point by essentially providing alternative targets for the interaction point. Consider the way our system works; when a user interacts with an interaction point this is interpreted as a semantic constraint. For example, if the user interacts with the interaction point `{\lnk danger}`, then the system interprets this as a constraint with the semantic value ‘danger’ and will search for narrative nodes that have a semantic description that contains the word ‘danger.’ The assumption is that inserting a narrative node with the semantic description that contains the word ‘danger’ will satisfy the constraint posed by the user. Now consider a modified interaction point, e.g., `{\lnk \meta="fear" danger}`. This link will be interpreted as the semantic constraint ‘fear’ and will work in the same way as the unmodified interaction point except that a narrative node with the semantics ‘fear’ will be sought in order to satisfy the constraint. In this way we can consider unmodified interaction points as having identity maps into the semantic domain.

6 System Architecture

At its core our system can be viewed as a co-evolutionary, collaborative (1+1)-ES system, drawing from a database of narrative nodes for the narrative evolution using the semantics of the narrative nodes as a way to discriminate between different possible narratives depending on which best satisfies the user constraints. We have already shown that the system can be considered a co-evolutionary system due to the fact that both the user constraints and the narrative evolve in response to each other over time.

That the system is collaborative follows straightforwardly from the fact that the system involves the user as part of its evolutionary computation. That the system can be considered a (1+1)-ES system deserves some remarks. The following is the pseudo code for our ES algorithm:

```
procedure ES
   t := 0
   initialize Page(t)
   display Page(t)
   get Constraint(t)
   while Constraint(t) available do
      Offspring := satisfy_constraint(Page(t),Constraint(t))
      Page(t+1) := Offspring
      display Page(t+1)
      get Constraint(t+1)
      t := t + 1
   end while
end procedure
```

The evolutionary algorithm starts by initializing the first page to be displayed and then waits for the user to supply a constraint. Once a constraint is available the evolutionary algorithm proceeds by computing an offspring (a new page) that satisfies the user constraint. This offspring becomes the new parent, the new parent is then displayed and the system waits for a new constraint from the user. Here is where we diverge from the standard ES algorithm. By construction our offspring is the best offspring of all possible offspring. Also, this offspring is always fitter than the parent (current page being displayed), since it satisfies the user constraint. Therefore, we do not have to perform any additional fitness evaluations after we construct the offspring, but can simply replace the parent with the new offspring as shown in the pseudo code above. The 'satisfy_constraint' procedure computes the offspring by searching for the most appropriate narrative node that satisfies the user constraint and uses it to construct a new page by replacing an existing narrative node with the new one.

As one can see, the system only maintains one parent at any one time and only produces one offspring per iteration. In our current system the search for the most appropriate narrative node is not adaptive, therefore a more appropriate nomenclature for our system today would be (1+1)-GA. However, we do have plans to make this search adaptive, for example, by taking user constraint history into account when selecting a particular narrative node. Thus, we prefer the name (1+1)-ES.

As shown in Figure 2, the markup language allows us to express semantics for the narrative nodes in a fairly straightforward manner. If we interpret the meta construct as a map of a narrative node into its semantic domain and analogously as a map of an interaction point into its semantic domain, then we can say that narrative evolution takes place in a semantic domain. A replacement narrative node that fulfills the user constraints is found in the semantic domain and is then projected into the syntactic domain for assimilation into the evolving text page. Figure 3 illustrates this process. Here, given a set of narrative nodes with their semantics and an interaction point with its semantics interpreted as a constraint, the narrative evolution engine attempts to find the most appropriate narrative node based on the match of constraint against semantics. The most appropriate node is selected and its syntax is then used to compute the next iteration of the evolving page.

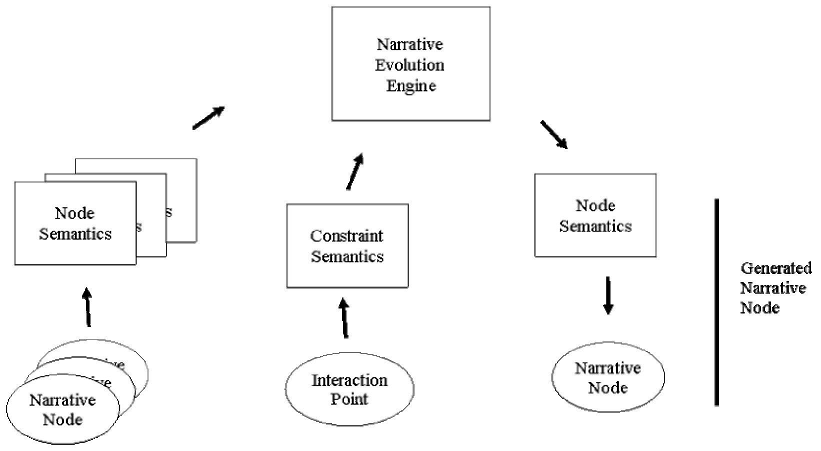

Fig. 3. Process of selecting a node for one evolutionary step. Given a set of narrative nodes with their semantics and an interaction point with its semantics or constraints, the narrative evolution engine attempts to find the most appropriate narrative node based on the match of semantics against constraint. The best node is selected and its syntax is incorporated into the evolving text page.

This selection process is core to our evolutionary computation and is implemented as part of the '`satisfy_contraint`' procedure in the ES algorithm.

The actual construction of the offspring page using the newly generated narrative node is not as sophisticated as we would like it to be: a narrative node close to the node that contained the activated constraint is randomly selected for replacement. We purposefully do not replace a narrative node that contains an active constraint, since this disturbs the continuity of the narrative substantially. Rather than randomly selecting a narrative node to be replaced in the parent page we could make use of the semantic information of these nodes, for example, we could replace the node on the active page that is most dissimilar from the generated node.

Figure 4 illustrates our overall system architecture. Looking at the architecture from right to left we see that the marked-up text of the narrative nodes is preprocessed into a narrative node database which the evolution engine uses to resolve constraints. Today the user interaction is via standard terminal functionality usually by highlighting or clicking on a word that is especially marked as a user interaction point on the text page. However, we envision user interactions beyond the typical terminal functionality, for example, voice activated interaction. Currently we are experimenting with a set up where the user interaction is via the SMS (Short Messaging System) protocol using portable devices such as mobile phones and personal communicators to send messages that contain constraints to the evolution engine. The evolution engine receives these constraints and adapts the evolving text accordingly. Here the evolving text is projected on a large surface for easy readability. An interesting side effect of this is that multiple users can interact with the text. We are still implementing this system, the impact of this multiple user interaction remains to be studied.

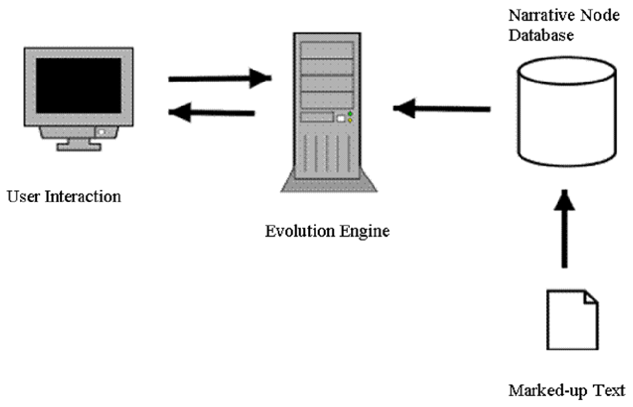

Fig. 4. ANE system architecture

7 Research Directions

We can identify a number of research directions we would like to pursue. Probably the most important one is in the area of semantics. Right now we only consider keywords as our semantics for the narrative nodes. We would like to investigate more expressive semantics following the lead of the information retrieval community. Also, an important part of this new semantics is that is can be computed automatically for the narrative nodes.

The next area of investigation is in the syntactic composition of the evolving text page. We currently do composition by purely syntactic means: "randomly select a narrative node close to the user interaction point and replace it with a new narrative node during evolution." Obviously this is not very satisfying and we would like to incorporate some additional semantic and syntactic compositional rules into the actual construction of the text page. Some ideas based on dissimilarity were sketched out earlier. We also need to refine our text layout engine to match the pristine text rendering in "The Jew's Daughter".

Perhaps one of the more intriguing research directions for this project is the notion of "foreign text" acquisition. By this we mean that not only will the system use narrative nodes that were hand-constructed by a composer but it will also acquire text for narrative nodes in an autonomous fashion via techniques such as web spidering or verbal exchanges with users. However, one of the more difficult issues to consider in this research is the automatic mark-up of the acquired text. In particular, given an acquired text we need to be able to automatically compute a viable semantics for the text so that the evolution engine can use this text in the actual construction of the evolving text page. This of course ties back into our earlier research goals.

8 Related Work

Our work has been influenced as much by modernist and post-modern practices within print literature as by currents within computer science and the digital arts. *Finnegans Wake*, for example, has been compared to both a supercomputer and an evolutionary system [3]. The textual surface of the work, an invented conglomeration of multiple languages whose embedded meanings can only be decoded phonetically, appears to visually transform for the reader as the eye becomes aware of new possible textual combinations and readings. In the more experimental prose of Gertrude Stein, the language seems to have an internal logic that disregards conventional grammar and is characterized by an odd, mechanical repetition [26]. The particular approaches of any number of writers as well as specific techniques, such as the cut-up method [8] of juxtaposition that William S. Burroughs borrowed from painting and developed for literature, have been useful filters for us in the consideration of our project. Constraint-based and generative approaches to the non-digital arts such as those practiced by the literary organization Oulipo [20] and the performance group Goat Island [2] are also of interest to us as we continue to refine our system.

There is some compelling work being done by writers within networked and programmable media as well. Poet and programmer Loss Glazier creates re-arrangeable poems [11] that appear on screen in an odd hybrid language that includes elements of English, Spanish, and Html. John Cayley creates poetic works in which the text undergoes animated transformations over time, and Chinese characters morph into English words and phrases [9]. *The Impermanence Agent* [28] is an online narrative that customizes itself to each of its readers based on an analysis of their internet activities. While most automatically-generated texts consist of fragments that are randomly extracted from a database, projects that implement more sophisticated machine-writing techniques include *Trajectiores* [4], a detective novel that generates portions of its own narrative, created by John Pierre Balpe and the groupe@graphe, and the anonymous hypertext novel *ebbflux* [1].

Trajectoires makes use of a Java engine in conjunction with specially constructed dictionaries for the generation of prose fragments. The generated text is concise and intelligible. While the content is often vivid and bizarre, the style of writing resembles conventional fiction in that the simulation is based upon the expectations of a reader of fiction for a coherent physical setting, plot-related suspense, and narrative action. *Ebbflux*, on the other hand, generates prose that is stylistically abstract and disjointed, without concern for the qualities associated with traditional narrative. Each time the reader clicks on a link in *ebbflux*, a new page is produced from a finite database of fragments in which semantic relationships are designated. These systems are closely related to our system which, like *ebbflux*, is not primarily concerned with the expectations of traditional narrative but does require a compositional logic and self-consistent rigor like that of *Trajectiores*.

9 Conclusions

We presented a tool that enables complex, unfixed temporal structures in digital narrative and also facilitates the creation of works that are a hybrid of human authorship, structural design, and machine writing. The system has basic similarities with other tools for the development of non-linear and interactive narrative, especially the *ebbflux* and the *Trajectiores* systems. We developed our system in the framework of evolutionary creative systems. This framework naturally encompasses the functionality required by our system and allows for interesting extensions of the basic ideas such as adaptive search strategies for constraint satisfaction and user interaction modes other than the canonical keyboard/monitor interactions. Initial experiments with the existing prototype have been very encouraging [18] and we are currently extending this prototype with a Flash based text layout engine and implementing user interactions by means of mobile phones and personal communicators promising a different kind of immersive experience with automatic narrative evolution.

References

1. Ebbflux, http://www.ebbflux.com
2. Goat Island Performance Group, http://www.goatislandperformance.org
3. Armand, L.: Enzymes, Reverse Transcriptions & the Technogeneses of Finnegans Wake, http://www.geocities.com/louis_armand/jjht_technogenesis.html
4. Balpe, J.-P.: Trajectiores: La Fiction De La Fiction. labart, http://www.labart.univ-paris8.fr/recit/balpe-riE.html (2000)
5. Bentley, P.J.: Aspects of Evolutionary Design by Computer. In: *Proceedings of the 3rd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC3)*. (1998)
6. Bentley, P.J., Corne, D.W.: An Introduction to Creative Evolutionary Systems. In: Bentley, P.J., Corne, D.W. (eds.): Creative Evolutionary Systems. Morgan Kaufman (2002) 1-78
7. Biles, J.A.: Genjam: A Genetic Algorithm for Generating Jazz Solos. In: *Proceedings of the International Computer Music Conference*. ICMA, San Francisco (1994)
8. Burroughs, W.S.: The Cut-up Method, AUTHOR HOMPAGE LIBRARY, http://www.spress.de/author/burroughs/
9. Cayley, J., Liang, Y.: Where the Sea Stands Still. ICA, London, http://homepage.mac.com/shadoof/wsss/ (1997)
10. Chomsky, N.: An Interview on Minimalism. In: Belletti, A., Rizzi, L. (eds.): On Nature and Language. Cambridge University Press (2002)
11. Glazier, L.P.: White-Faced Bromeliads on 20 Hectares, http://wings.buffalo.edu/epc/authors/glazier/java/costa1/00.html (1999)
12. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
13. Knuth, D.E.: The Texbook. Addison-Wesley (1984)

14. Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A.: Automated Synthesis of Analog Electrical Circuits. In: Genetic Programming 3. Morgan Kaufman (1999)
15. Linden, D.S., Altshuler, E.E.: Automating Wire Antenna Design Using Genetic Algorithms. Microwave Journal, 39:3. (1996)
16. Littman, M.L., Dumais, S.T., Landauer, T.K.: Automatic Cross-Language Information Retrieval Using Latent Semantic Indexing. In: *Cross Language Information Retrieval*. Kluwer, (1998)
17. Morrissey, J., Talley, L.: The Jew's Daughter, http://www.thejewsdaughter.com
18. Morrissey, J., Talley, L., Hamel, L.: Narrative Time and Machine Writing, Panel Discussion. In: *17th Annual Conference of the Society of Literature and Science (SLS2003)*. Austin, Texas (2003)
19. O'Reilly, U.M., Ross, I., Testa, P.: Emergent Design: Artificial Life for Architecture Design. In: *Proceedings of Artificial Life 7*. (1999)
20. Oulipo, http://www.nous.org.uk/oulipo.html
21. Paredis, J.: Coevolutionary Algorithms. In: Baeck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press (1997)
22. Rechenberg, I.: Evolutionsstrategie: Optimierung Technisher Systeme Nach Prinzipien Der Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
23. Rooke, S.: Eons of Genetically Evolved Algorithmic Images. In: Bentley, P.J., Corne, D.W. (eds.): Creative Evolutionary Systems. Morgan Kaufman (2002) 329-365
24. Sims, K.: Artificial Evolution for Computer Graphics. In: *1991 ACM SIGGRAPH Conference Proceedings*. Las Vegas, Nevada (1991) 319-328
25. Stein, G.: How to Write. 1st ed. Plain edition, Paris (1931)
26. Stein, G.: Tender Buttons: Objects, Food, Rooms. Claire Marie, New York (1914)
27. Vinokourov, A., Shawe-Taylor, J., Cristianini, N.: Finding Language-Independent Semantic Representation of Text Using Kernel Canonical Correlation Analysis, NeuroCOLT Technical Report, NC-TR-02-119 (2002)
28. Wardrip-Fruin, N.: Hypermedia, Eternal Life, and the Impermanence Agent. In: *Proceedings of the ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*. Los Angeles, California (1999)
29. Weber, L.: Evolutionary Combinatorial Chemistry – Application of Genetic Algorithms. Drug Discovery Today, 33. (1998) 379-385