EVOLVING ADAPTIVE SENSORS IN A SYNTHETIC LISTENER

Peter Beyls
Hogeschool Gent, Belgium &
Computer Music Research, School of
Computing, Communications and
Electronics, University of Plymouth, UK

ABSTRACT

This paper suggests a method to evolve responsive networked sensor configurations using a genetic algorithm. The system integrates symbolic processing in specialized, adaptive sensor functions with sensor fusion using a distributed, connectionist network. The process of interaction itself is viewed as a complex dynamical system featuring wide ranges of remarkable non-linear behavior. An off-line algorithm is proposed to gradually evolve fit networks and discover interesting dynamic behavior that could not have been anticipated by an explicit programmer.

1. INTRODUCTION

This paper addresses the problem of qualitative listening in the realm of man-machine improvisation. This typically suggests a musical climate in constant flux where dynamic relationships unfold in real-time. Important, the scope of the work reported here is limited to the input layer of a synthetic improviser; our simulations feed from an event source of variable complexity but do not modify a given context in any way. We explore responsiveness and adaptation rather than conversational interaction driven by fluctuating internal motivations.

Machine models of improvisation aim the discovery and understanding of interesting yet unknown associations and must accommodate unpredictable behavior by both man and machine. It is understood intuitively that sequences of pre-planned relationships are very limiting as all behavior conforms to static rules. True improvisation excludes the presence of finite goals; the improvisation commonly propels itself generating critical evaluation along the way. We avoid relying on representations or templates of any kind, so rule-based approaches seem inappropriate. Synthetic improvising systems must encourage the spontaneous emergence of man-machine relationships that can be said to exhibit features of human creativity and invention. The emphasis stresses the discovery of unknown mappings within a highly responsive performance system.

Specifically, interactive composing [Chadabe, 83] explores musically rewarding modes of interaction by creating a platform which supports and favors rich behavioral man-machine interplay, paradoxically, the spontaneous creation of coherent structure by active listening and response rather than contemplation.

Approaches include, Improvisation Builder [Walker and Belet, 96] which views musical improvisation is a collaborative activity analogous to conversation. Using Conversation analysis rules for turn-taking (within the jazz idiom), the Continuator project [Pachet, 01] that includes a learning module and the pattern oriented style-modeling work of [Assayag, 03]. These systems explore a certain musical niche in a (largely symbolic) computational framework -- therefore they have trouble coping with unanticipated user input and thus constrain our wish for maximizing behavioral complexity.

Implementation may consider problem decomposition, a method which speculates on successful isolation of sub problems in engineering situations beyond a given complexity barrier. However, this approach is often bound to fail because knowledge to decompose is not explicitly available, because interactions between sub problems are ignored, because the environment is assumed to be static and predictable, and finally, because real-world problems imply a combinatorial explosion of sub problems.

An alternative to the reductionistic method is to look at how nature handles complexity. It is perhaps most candidly acknowledged in the Subsumption architecture advocated in [Brooks, 91]. This work takes inspiration from natural evolution, interpreted as a process of gradual, layered building up of increasing levels of complexity – it proved very successful for the construction of robotic hardware. However, this work uses finite state machines with hardwired logic, and behavior issues from interactions in a hierarchy designed by hand. [Bryson, 93] documents a reactive music system avoiding representation and reasoning in the spirit of the subsumption approach.

Much work in evolutionary robotics [Beer, 90, Nolfi, 93] and in particular [Steels, 94] turned to biology inspired techniques aiming the implicit evolution of robust behavior facing severe realworld constraints. Some such nature-inspired technologies were adopted in the musical domain, including genetic algorithms [Biles, 01], distributed agent architectures [Beyls, 97] and artificial life oriented sound synthesis [Miranda, 03].

Throughout this project, complexity is thought of as an emergent property of the interaction. The interestingness of interaction is directly linked to the interpretation of patterns – incidentally, [Hofstaedter, 95] suggests that intelligence as such is deeply rooted in a capacity to identify, to interpret and to be creative with patterns. The work of [Cope, 91] relates musical style intimately

to the observation of recurrent patterns called 'musical signatures'. Higher level abstractions are generated from the identification of lower level patterns which points to the necessity for sophisticated pattern detectors in interactive music systems.

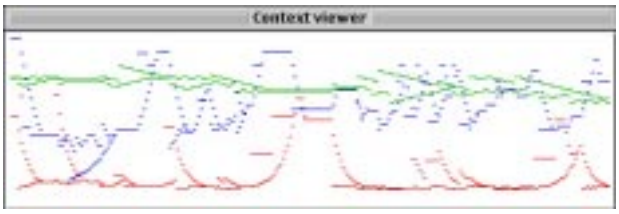

Fig. 1 Context viewer showing adaptive behavior for, from top to bottom, respectively pitch, velocity and duration.

The key requirement throughout is the definition of individual sensors and the appropriate assembly of a particular group of sensors in a process known as sensor-fusion.

2. APPROACH

One fundamental assumption is that listening is a cultural process, colored by social and intellectual forces. In addition, listening is considered a constructive activity [Berger, 01] encompassing filtering, selection and categorization. The listener actively builds expectations thus speculating on the further development of what is heard. Listening can thus be understood as a complex self-organizing process with multiple channels of confirmation and contradiction. In reality, of course, the distinction is blurred and subject to chaotic bifurcation. In addition, perception takes place in a continuous context, far more elusive than an idealized, discrete version of it. Musical imagination thus orbits spaces of relative stability, quasi-periodicity and uncertainty. How could this non-linear behavior be modeled in a machine listener?

A multi-layered, hybrid architecture is proposed incorporating principles of symbolic reasoning as well as reactive, sub-symbolic computational processes. We suggest a distributed connectionist design in a Hebbian spirit. Since we lack the explicit knowledge to design networks by hand, we use an implicit genetic method to evolve potentially interesting configurations.

2.1. Sensor layer

The input layer consists of an array of 64 binary sensors looking for specific features in the incoming MIDI stream with an adaptive sampling-delay of 100 to 1000 msec; the results are collected in a feature-vector for computational efficiency. In addition we apply automatic segmentation according to simple heuristics [Rowe, 93]. Partial sequences are kept in short-term-memory; the last two sequences are always available and are instrumental to infer short-term tendencies while listening. Incoming events are also stored in working memory where they are accessed by the sensor functions. Finally, input is stored in long-term memory; the idea here is 'exploration' i.e. to scan LTM looking for interesting musical gestures as seeds for the synthesis of responses.
Some sensors bear on the most recent input event while others refer to the recent history of the input stream; they document direction and magnitude of changes. For instance, is the performer accelerating, has there been no input for a long time, is there decrease in input density or is there a tendency to move from smooth to angular melodic input?

Thus, the computational complexity of the sensors varies widely. For instance, a neural network, operating in parallel, does real-time tonal inference and feeds the tonal-swing sensor. A simpler scheme views a MIDI input triple {pitch, loudness, duration} as residing in a three-dimensional parametric space; a cube divided in 8 zones, resulting in 8 mutually exclusive individual sensors. Not all sensors are used simultaneously; typically, a variable subset is selected for wiring to the neurons in the next layer. Some specific sensors are discussed next.

• Sensors pertaining to the last perceived event:

Most recent event sensors are equipped with 2 different complementary algorithms, introducing a certain degree of redundancy. The first uses statistical methods, the second, an adaptive procedure.

High-pitch-p and low-pitch-p are sensors providing information on whether the pitch of the last event perceived is high (or low) relative to the most recent history as stored in working memory. Loud-p and soft-p are similar for amplitude and short-p / long-p speak for the duration of the last event. Note that we handle a *zone of uncertainty*; the sensor fires only if the scalar value is sufficiently expressed. For instance, considering pitch, if the range of pitch values in working memory is 48 to 66, a range of 18 semitones, the High-pitch sensor will fire if the last input pitch is greater than the bottom value plus two thirds of the range, i.e. greater than 50. Correspondingly, given the same conditions, Low-pitch sensor will fire if the last input pitch is lower than 54. So instead of using a statistical average as a Boolean barrier, decision-making is biased to the extremes of a scalar continuum yielding a more pertinent analysis.

The alternative versions of the previous 6 sensors are adaptive, relative to the most recent context. This context resides in the last few events and is typically much sorter than the duration of working memory -- thus more accurately reporting the current state of affairs. A simple context sensitive algorithm traces the dynamics of pitch, velocity and duration of the last incoming event. The edges of the current context window are adjusted; the min and max values change, using a multiplicative operator, according whether the event parameter is above, below or inside the context window. This entails a functionality that zooms in into the most recent event. The value of the multiplier ($0.5 < m < 1.5$) controls the hysteresis i.e. the context sensitivity.

• Sensors tracing memory status:

Various second order characteristics are explored here, that is, we look for features of which we hypothesize they deliver valuable insight to the understanding of the behavioral trail left by the human interactor. The data in working memory portrays this trail and serves as source for the following algorithms:

The angularity of the input buffer (working memory) is computed as follows:

```
(defmethod angular-p ((self ear ))
  (> (/ (apply '+ (mapcar 'abs (intervals (buffer self))))
       (nr-events (buffer self)))
     3))
```

The signal is angular if the sum of the absolute values of the intervals of the pitches in the buffer divided by the nr of events in the buffer is greater than a constant, in our case 3. The constant was developed experimentally, by trial and error. The related, smooth-p function yields true when the computed value is lower than 3. Thus, information on intervallic data is not available when the computed value equals exactly 3 -- which is fine as we aim for a degree of expressive polarity.

Expressionist vs. pointillist articulation is considered next. A given melodic sequence is characterized by its expressive, global temporal continuity. Expressionist and pointillist are visual metaphors -- alluding to stylistic features -- to denote how events are chained in time. Expressionist (Cézanne) means tightly connected events while pointillist (Seurat) implies short events in relative isolation. Formally:

```
(defmethod pointillist-p ((self ear))
  (when (> (nr-events (buffer self)) 2)
    (> (nr-rests (buffer self))
       (/ (nr-events (buffer self)) 2))))
```

This method is true when a critical mass of 2 events is exceeded and the number of rests is greater than half of the total number of events in working memory, otherwise, the expression is considered expressionist.

Relative comparative complexity is also to be inferred from working memory. Intuitively, relative complexity is reflected in diversity, similarity, coherence and regularity as interlocking features. Diversity, in percent, is a measure of number of different values in relation to the total number of values, implemented as:

```
(defun diversity (list)
  (round (* 100 (/ (length (remove-duplicates list))
                   (length list)))))
```

Coherence measures relative stability, reflected in the amount of values that remain unchanged while scanning a given sequence.

```
(defun coherence (list) ;; in percent
  (round (* 100 (/ (loop for i from 0 to (- (length list) 2)
                         count (= (nth i list)
                                  (nth (+ 1 i) list)))
                   (length list)))))
```

For example, pitch diversity is considered high if diversity exceeds 60 pct, considered low when below 30 pct, in between, no opinion is available.

```
(defmethod  pitch-diverse-p ((self ear))
  (> (diversity (pitches (buffer self)))
     60))
```

• Motion sensitive sensors:

Some algorithms here loosely follow terminology developed in [Smalley, 86] which suggests a unified theory to handle spectro-morphological complexity in electro-acoustic music. It addresses complexity and its articulation in various dimensions, from the sample level to the appreciation of global structures. It provides hints for macroscopic analysis of MIDI sequences as detailed next.

A sensor tracing interval profiles in a perceived melody reports on the tendency of the melodic pitches to increment, decrement or remain stationary; thus creating 3 mutually exclusive sensors: incremental-p, decremental-p and stationary-p.

```
(defun incremental (list))
  (>= (loop for e in (derivate list)
            count (plusp e))
      (/ (length list) 3)))

(defun stationary (list)
  (>= (loop for e in (derivate list)
            count (zerop e))
      (/ (length list) 3)))

(defmethod incremental-p ((self ear))
  (incremental (pitches (buffer self))))
```

Note we handle a critical bias; all sensor functions evaluate relative to the implied bias value equal to 3.

Directionality of motion is detected by the next sensor. Uni-directionality is more specific and is computed by counting the number of equal signed contiguous values. For instance, after summing both the number of contiguous positive and negative values of the first derivate of the pitch dimension and comparing both sums we gain understanding of directionality. If either sum is much larger than the other, uni-directionality is said to be true. A melody is bi-directional when the sums are roughly equal and there are not many zero intervals. Zero intervals decrease the critical mass and thus increase uncertainty in the inference process.

A melodic gesture is considered reciprocal if the behavioral profile of most of its dimensions is shaped like an arc, more general, when energy builds up and consequently dissipates, or the other way round. Thus, a degree of relative symmetry is implied. Symmetry is

easily inferred; when only 2 zones exist of directed motion (see above), albeit not necessarily of equal length, the signal is considered reciprocal. When a great number of zones exist, the signal is considered ambiguous.

A melody is judged oscillatory when the first derivate of either dimension (pitch, loudness, duration) produces intermittingly positive and negative values and the number of different values roughly equals 2. Stylistic variations i.e. more than 2 unique values are not considered here but are addressed in pattern analysis algorithms (not documented here).

Target detection in motion can also provide valuable information in the context of interactive composing. A target implies a point of reference in some dimension. When the focus value marks the start of repeated sequences, its motion is considered centrifugal. Otherwise, consider short-term memories; they hold the last two identified gestures issued by the human performer and their similarity is thus easily inspected. When the last events in both memories are very similar, it may function as a temporary target. This focused value is related to but not identical to peaks in statistical salience; statistical methods do not typically consider motion in the data they address. In addition, when a single parameter [pitch, loudness or duration] remains roughly stationary while the other two remaining parameters feature wild changes, that parameter is considered a focus.

We may reflect on the nature of the changes inside a given data block, for example, can we detect a growth profile in the motion? How do fresh events numerically relate to the events that are already accommodated in a given memory? The notion of data range in all dimensions is of primary use; a given memory structure may grow outside of its current data range, thus extending that range and producing exogenous growth. In contrast, when data falls inside a given existing data range, that range is not modified by the last event, but endogenous growth is taking place. Amount and type of growth is computed by summing the contributions of all event parameters; pitch, loudness and duration.

Next, we consider the articulation in time; temporal analysis includes conventional tempo tracking, density induction but also the eminence of temporal change, for instance, by using a similarity function to measure the difference between the contents of the two short-term memories in an attempt to detect the absence of motion. Changes in speed such as acceleration are expressive qualities that can be detected from the inter-onset-times of the MIDI events in working memory. Considering the first derivate of the IOT over many events, when consecutive values slightly increase at every step; we know the signal is gradually slowing down. Otherwise, gradually decreasing values (i.e. shorter distances between events) signal acceleration. Note that the incremental function includes a threshold implying the necessity of a significant amount of incrementality before yielding a Boolean true.

```
(defmethod accelerating-p ((self ear))
  (Incremental (derivate (delta-times (buffer self)))))
```

• Distance metrics

The inference of similarity is far more complex, in particular, relating to musical sequences. Our pragmatic, approximate implementation is as follows:

```
(defun similarity (list-1 list-2) ;; in percent
  (if (and list-1 list-2)
    (let ((nd (abs (- (length list-1) (length list-2))))
              ;; difference in length
          (maxl (max (length list-1) (length list-2)))
          (minl (min (length list-1) (length list-2)))
          (c (loop for e1 in list-1
                   for e2 in list-2
                   count (= e1 e2))))
      (round (* 100 (* (/ c minl))      ;; percent
             (/ (- maxl nd) maxl)))) ;; account for length
    0))
```

This algorithm combines the pressures of equality in length with the difference in unique values and provides a robust and computationally compact measure of similarity.

More elaborate distance metric algorithms are typically used to compare 2 melodies. It is useful to compare the various algorithms developed for measuring the metric distance of melodies residing in 3-dimensional dataspace, in particular, if they are fit for real-time applications.

Finally, a sensor is proposed to quantify both the overall harmonic tension in a given pitch sequence and to generate tension profiles using a sliding window. The idea of harmonic tension is related to psychological dissonance in a musical structure, the tension created from relationships establishing order and disorder between its components. [Jaxitron, 85] suggests a subjective yet robust way to quantify tension using a logarithmic scale. The algorithm makes various simplifications -- including abstraction of octave, ignoring repeated pitches and limited resolution -- though it performs remarkably well. It is straightforward to implement and computationally inexpensive. The algorithm assumes there to be harmonic tension in 4 intervals only: the minor-seventh, major-second, major-seventh and minor-second. Implemented as:

```
(defun tension-value (interval)  ;; tension in interval
 (case (mod interval 12)
   (10   1)      ; minor-seventh
   (2   10)      ; major-second
   (11 100)      ; major-seventh
   (1 1000)      ; minor-second
   (t    0)))     ; else 0
```

Now, the total tension in a melodic structure is the sum of the tension contributed by all its constituent intervals while considering only unique pitches. Evaluation of

tension is instrumental for measuring the stability of static structures (harmony) as well as their expansion into melody (tonality). We could use the tension concept as one force in a constraint based real-time harmonization algorithm.

2.2. Neuronal layer

The top layer is a sensor-activator network (SAN) realized as a small non-linear network – taking

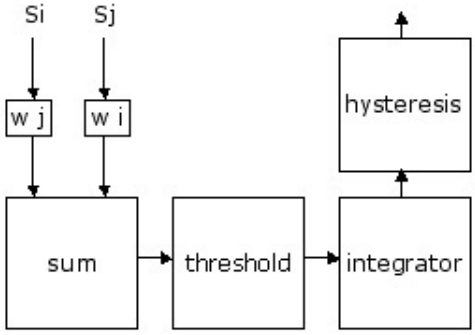

Fig. 2 A single neuronal node

significant inspiration from the work of [Van de Panne & Fiumi, 93] -- with weighted connections between the sensors and the neural nodes of the output layer. The network sports internal delays thereby giving it dynamic properties and a very wide range of oscillatory behavior while the sensors also influence the oscillation pattern. SANs are different from conventional neural networks; they are not used for the purpose of learning. Our version avoids hidden nodes because their computational expense outweighs a beneficial increase in non-linearity. Every node is designed as depicted in fig. 2.

A node will output 1 if the weighted sum of its sensor inputs is positive as with conventional neural networks. However, we expect the node to function as a complex dynamical system by itself since we avoid Pavlovian response in favor of interesting dynamics. Non-linearity is introduced by a time delay implemented by integrator and hysteresis functions. Our current implementation maintains fixed time delays ($-1.0 < d < 1.0$) though further work will exercise parametric control on on-delay and off-delay thus offering critical influence on the periodicity and duty-cycle of the oscillatory pattern. Finally, a hysteresis function, rather than a normal threshold function, further constrains excessive fluctuation.

A neuronal node normally feeds an actuator node that modifies part of the environment, in our case, by contributing machine-originated sound within an interactive setting. Thus, the perceived global sonic system behavior is more complex as it originates from the interlocking circular dynamics of the human performer, the machine listener and the machine player. However, this is beyond the scope of the present paper.

• Tracing neural history

Now, consider a SAN with 16 neurons, 5 inhibitory and 5 exhibitory connections with a random weight ($-2.0 < w < 2.0$) and 5 random sensors for every neuronal node. The behavioral complexity of this SAN is evaluated by having a human performer improvise. Important, we aim the evaluation of degrees of freedom in the network by charting its state space, assuming the performer to be a source of relatively coherent qualitative information, however, this information itself is exploited at a low quantitative level of abstraction.

During interaction, the status of all neurons is captured in a vector, a neural firing configuration that was not encountered before is added to the neural-history list. When an existing pattern reoccurs, its frequency parameter is incremented by one. Figure 3 shows 100 sequentially detected firing patterns, the top pane represents the rate of recurrence, the bottom pane the number of firing neurons. The latter shows a Brownian-like distribution though a clear correlation is observed between both graphs; a small peak in occurrence rate corresponds with a decline in firing density. This area of the state space reveals very low density corresponding with low but non-zero occurrence rate. The large peaks in the upper pane provide evidence of idiosyncrasies of the performer. In addition, the clustering of peaks follows from the inertia in the non-linear network.

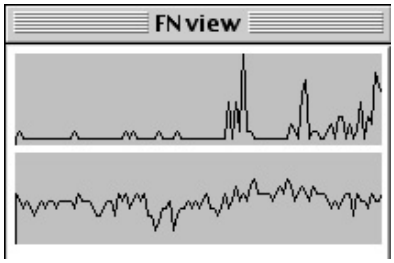

Fig. 3 History trace of 100 unique neural patterns

• Evolved networks

We clearly lack the explicit knowledge to approach network design in a top-down fashion. Top-down design also entails the danger of introducing stylistic biases, which in fact, we aim to avoid in our wish to maximize diversity in support of coherent yet unpredictable behavior. As a further complication, intended non-linear behavior eschews explicit back-engineering. We take on a bottom-up design methodology using a neo-Darwinist algorithm incorporating a generate-and-test strategy. A genetic algorithm is used to explore the huge search space of all possible networks. A matrix, representing the weighted neural connections, is considered a genotype subject to cross-over and mutation. Thus, two components are subject to evolutionary adaptation: first, the number of sensors and their associated weights and, second, the density of the connections between neurons and their respective weights.

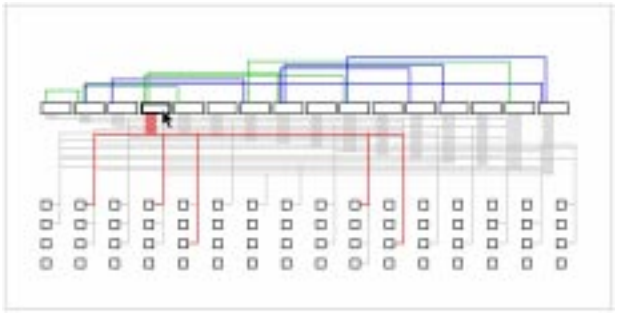

Fig.4 Snapshot of sample sensor-activator network featuring sixteen partially interconnected neurons. In addition, every neuron receives excitation from an array of 64 potential Boolean sensors.

A typical snapshot of an evolved sensor-activator network is in figure 4.

A simple experiment examines the prospect to evolve fit sensor-activator networks. We exclude external selective fitness attribution by a human evaluator and apply an implicit fitness metric.

The fitness of a given SAN is directly proportional to the number of neurons that change state for every perception cycle; the rationale is to evolve sensors that maximize sensitivity facing a given source. We would seemingly need extended periods of interaction with a human interactor, however, instead an off-line method is used which makes it feasible to run the experiment for several hours. We employ a simple, tunable L-systems based sequence generator. We assume the source to exhibit coherent and stable behavior.

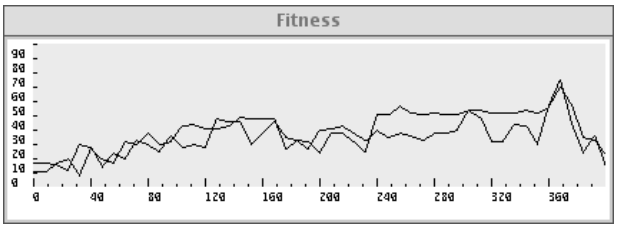

Fig. 5 Evolutionary paths of 2 different sensor-activator networks

The experiment runs as follows. First, generate 10 SAN with random neuronal connections and weights, with every neuron up to 5 sensors (excluding pace makers). Second, we play the test sequence into every SAN and trace fitness. Third, select two fittest networks, apply cross-over and mutation operators and create 10 new offsprings. Cross-over is effectuated by linear interpolation of weights and mixing the sensor functions of both parents. Slight mutation contributes to the evolutive process not getting stuck in local minima. The generate-evaluate-select cycle is repeated and the fitness history accumulated.

For clarity, figure 5 shows the evolutionary path for only 2 individual networks. We observe two particular features. First, both networks retain enough common phenotype to remain in sync for number of time steps; a spike is detected at generation 370. Second, a gradual increase of average fitness is observed, albeit the fitness landscape itself being quite irregular.

3. CONCLUSION

This paper focuses exclusively on the input layer of an interactive system. The combination of adaptive, handcrafted sensor functions and evolved non-linear networks for qualitative listening has proved promising on two axes. First, evolution allows complexity engineering without necessarily understanding the details of the process and second, the evolutionary exploration of state spaces discloses surprisingly reactive configurations that were unknown to exist by the experimenter. We probe areas of relative potential rather than having faith in the power to conceive the ultimate solution right away. This process contrasts sharply with conventional optimization, which optimizes towards a given pre-conceived goal.

Much further systematic work is needed to explore the synthesis and evolution of responsive networks, in particular the impact of wiring densities, the attribution of weights and profitable arrangement of sensors. The preliminary work reported here provides evidence of successful emergent functionality in a real-time setting.

4. REFERENCES

- Assayag, G. Dubov, S. Delerue, O. (1999) *Guessing the composers mind; applying universal prediction to musical style*. Proceedings of the ICMC99, Bejing, China

- Beer, R. (1990) Intelligence as adaptive behaviour, Academic Press, Cambridge, Ma.

- Beyls, P. (1997) *A survey of agents based real-time interactive systems.* Proceedings of the ICMC97, Thessaloniki, ICMA, San Francisco, CA

- Biles, JA. (2001) *Autonomous Genjam; eliminating the fitness bottleneck by eliminating fitness,* GECCO 2001 Workshop, San Francisco

- Brooks, R. (1991) *Intelligence without representation*, Artificial Intelligence, 47 (1-3), pp. 139-160

- Bryson, J (1995) *The reactive accompanist: adaptation and behaviour decomposition in a music system*.

- Chadabe, J. (1983) *Interactive composing: An overview*, Computer Music Journal, Vol. 7, No. 1

- Cope, D. (1991) Computers and musical style, A-R Editions, Inc.

- Hofstadter, D (1995) Fluid concepts and creative analogies; computer models of the funcdamental mechanisms of thought, Basic Books, NY

- Miranda, ER. (2003) On Making Music with Artificial Life Models, Consciousness Reframed 2003

- Nolfi, S. et. al. (1994) How to evolve autonomous robots, Proceedings of Alife IV, MIT

- Pachet, F. (2001) *Playing with virtual musicians: the continuator in practice*. IEEE Multimedia 9:3. pp. 77-82

- Rowe, R. (1993*) Interactive Music Systems, Machine listening and composing*, MIT Press, Cambridge, MA

- Smalley, D. (1986) Spectro-morphology and structuring processes, in: The language of electro-acoustic music, Emmerson, Ed. Macmillan Press, London

- Steels, L. (1994) Emergent functionality in robotic agents through on-line evolution. Proceedings of Alife IV, MIT

- Van de Panne, M. and Fiume, E. (1993) *Sensor-actuator networks*. Proceedings of SIGGRAPH '93, pp. 335-342

- Walker, W. and B. Belet. (1996) *Applying ImprovisationBuilder to Interactive Composition with MIDI Piano*, Proceedings of the International Computer Music Conference (Hong Kong, 1996), ICMA, pp. 386-389.