NetNeg: A Hybrid Interactive Architecture for Composing Polyphonic Music in Real Time

Claudia V. Goldman [*] Dan Gang[†] Jeffrey S. Rosenschein
Daniel Lehmann
Insitute of Computer Science
The Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5353 fax: 011-972-2-658-5439
email:clag@cs.huji.ac.il, dang@cs.huji.ac.il, jeff@cs.huji.ac.il, lehmann@cs.huji.ac.il
url: http://www.cs.huji.ac.il/{$\sim clag, \sim dang$}

Abstract

There are musical activities in which we are faced with symbolic and sub-symbolic processes. This research focuses on the question whether there is any advantage in integrating a neural network together with a distributed artificial intelligence approach in the musical domain.

In this work, we present a new approach for composing and analyzing polyphonic music. As a case study, we began experimenting with first species two-part counterpoint melodies. Our system design is inspired by the cognitive process of a human musician. We have developed a hybrid system composed of a connectionist module and an agent-based module to combine the symbolic and sub-symbolic levels to achieve this task.

The network produces aesthetic melodies based on the training examples it was given. The agents choose which are the next notes in the two-part melodies by negotiating over the possible combinations of notes suggested by the network.

[*]Supported by the Eshkol Fellowship, Israeli Ministry of Science

[†]Partially supported by the Jean and Helene Alfassa fund

1 Introduction

The researchers in computer music that chose to use techniques from Artificial Intelligence (AI) and Neural Networks (NN) explore complex musical tasks such as composing, listening, analyzing, and performance. Research in AI and NN can itself benefit from results in music research; in music, for example, terms like time and hierarchical structure are inherent to the domain. Simulating and modeling a musician activities are tasks adequate for experimenting with in the framework of artificial intelligence. The cognitive processes a musician undergoes are complex and non trivial to model. Whenever we are involved in any musical activity we are faced with symbolic and sub-symbolic processes. While listening, our aesthetic judgment is not necessarily achieved by following explicit rules. Applying a learning mechanism has been shown to be a convenient (and appropriate) way to overcome the explicit formulation of rules. Nevertheless, some of these aesthetic processes might have a symbolic representation and might be specified by a rule-based system. We also think the research in Distributed Artificial Intelligence (DAI) [BG88] can contribute to developing new methods for computer music. For example, an interesting research question to investigate is the analogy between the dynamics of the performance of a group of musical instruments or voices in a vocal ensemble to agents [WFG+95] and their interactions in the DAI sense.

The primary purpose of our work is to build a hybrid system architecture for composing polyphonic music in real time. This system can serve as an interactive work tool for a composer or as a real time performance tool. The system is also useful for composition and analysis. The approach we suggest in this work, enables the musician to encode his knowledge, intuitions and aesthetic taste in different modules. The system captures these aspects by computing three distinct functions: rules, fuzzy concepts and learning.

The implementation we present in this paper considers the composition of polyphonic music in real time. The case study we have chosen to experiment with is the polyphonic vocal style of the sixteenth century; more specifically, we investigate two-part species counterpoint (i.e, bicinia). Our system, *NetNeg*, is composed of two main sub-systems. It creates both counterpoint parts dynamically. One sub-system is implemented by a modified version of Jordan's sequential neural network [Jor86]. The second one is a two-agent model based on Distributed Artificial Intelligence [BG88]. These agents negotiate with one another to maximize the global utility of the whole system, which for our purposes should be interpreted as the global quality of the composition.

In spite of the simplicity of the problem our current system can serve as a basis for further investigation of more complex musical problems.

2 The General Architecture

The overall design of our system reflects the cognitive processes that a musician undergoes when he is involved in a musical activity. We distinguish among three main aspects that are important for a musician to take into consideration. We suggest to enable the musician to encode his knowledge (e.g., rules for a known style, or rules he has invented), intuitions (i.e. fuzzy concepts about the music he is interested in composing), and aesthetic taste, in different modules.

The system will be composed of agents [BG88]. The agents will communicate, cooperate and share tasks in order to improve the global performance of the system. Each agent *knows* rules of a specific style and heuristic rules that take into consideration different aspects of the problem, that the system is aimed at solving. The aesthetic taste referred to above, might be captured by a learning mechanism (e.g. a neural network) that will give advice to the agents.

An example, that we have implemented and will present in the following sections, refers to the problem of composing polyphonic music in real time. This implementation demonstrates a specific solution using the approach presented in this section.

3 NetNeg's Architecture

In many musical styles, the composer needs to create different sequences of notes (i.e., melody lines) that will be played simultaneously. Each sequence should follow some aesthetic criterion, and in addition the sequences should sound appropriate when combined. This overall composition is the result of many interactions among its components. The musician achieves his overall result by compromising between the perfection of a single component and the combination of sequences as a whole. Thus, in this activity there is a constant tradeoff between the quality of a single sequence versus the quality of the combined group of sequences. When a musician is faced with such task, he is involved in a cognitive process, that we suggest it might be seen as a negotiation process. He has to compromise between the melodies' notes by choosing from among the permitted notes those that are preferable.

The case study we chose to experiment deals with first species of two part counterpoint melodies. In NetNeg, we create both parts dynamically, in real time. Therefore, the system is not allowed to perform backtracking. However, the current context is taken into account to produce the next note in the melody. A general view of the architecture of *NetNeg* is shown in Figure 1.

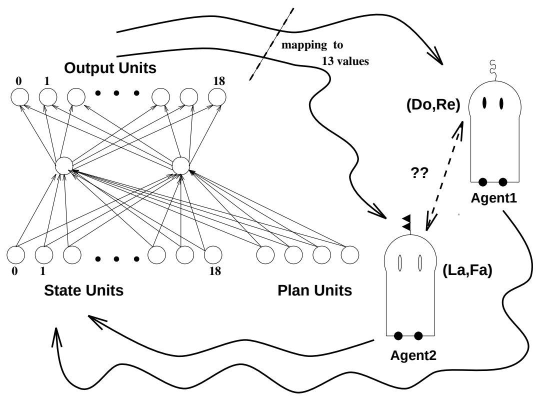

Figure 1: The NetNeg Architecture

NetNeg is composed of two sub-systems: a connectionist sub-system and a DAI-based subsystem. Each part of the melody is produced independently by a neural network. In the training phase, the network learned to reproduce a series of learning examples that were taken from [Jep92]. In the generalization phase, the network predicts in the output layer a vector of expectations for the next note in the melody. Each agent receives a different output vector from the network. On the one hand, each agent has to act according to his voice's aesthetic criteria; and on the other hand, he has to regard the other voice-agent such that both together will result in a two-part counterpoint. Both agents will have to negotiate over all the other possible combinations to obtain a globally superior result. Thus, they influence the context with their agreement. Then, the network will predict another output vector given this new context and the initial values of the plan units. This process continues sequentially until the melodies are completed.

3.1 The Connectionist Subsystem

Each part of the melody is produced independently by a neural network impemented in Planet [Miy91]. Todd [Tod89] has already suggested a sequential neural network that can learn a sequence of melody notes. Currently, our neural network is based on the same idea, although we have extended it to include the representation of the contour of the melody.

We built a three-layer sequential net, that learns series of notes. Each series is a one part melody (i.e., cantus firmus). Each sequence of notes is labeled by a vector of plan units. The net is a version of a feedforward backpropagation net with feedback loops from the output layer to the state units (in the input layer). The state units in the input layer and the units in the output layer represent the pitch and the contour. The state units represent the context of the melody, which is composed of the notes produced so far. The output unit activations vector represents the distribution of the predictions for the next note in the melody for the given current context. The role of the plan units is to label different sequences of notes. In the generalization phase, we can interpolate and extrapolate the values of the plan units so as to yield new melodies (i.e., new cantus firmi). At each step, the net is fed with the output values of the previous step in the state units together with the values of the plan units. These values will cause the next element in the sequence to appear in the output layer and it will be propagated as feedback into the state units (these connections do not appear in Figure 1). The current values of the state units are composed of the previous values multiplied by a decay parameter and the current output values.

The state units and the output layer can represent the notes in different ways. In this implementation, we choose to represent the notes as a vector of 19 units. The first eight units encode the pitch. The next nine units represent intervals between the notes. The last two units describe whether the movement of the melody is ascendent or descendent (we will refer to these units as the movement units). For example, if the current tone is DO(C), the net predicts both RE(D) and FA(F) as the next best tones, and the ascendent movement unit is on, then the interval can help us to decide which tone to choose (i.e., one tone or two and a half tones). If after the net has chosen the tone SOL(G), it predicts LA(A) or FA(F) and the interval is of one tone, then we could choose whether to descend or ascend based on the activations of the movement units. In order to exploit the information encoded out in the output units activations, we have combined the pitch activations with the interval and the movement activations We have mapped the activations of the output units into a vector of thirteen activations corresponding to the notes in more than an octave and a half.

Each agent receives the 13-length vector, and feed the state units with their agreement (see Figure 1 and Subsection 3.2). Then, the network will predict another output vector given this new context and the initial values of the plan units. This process continues sequentially until the melodies are completed.

3.2 The DAI-Based Subsystem

We have implemented the agent module using the Mice testbed [MLM+92]. We consider each part (i.e., voice) of the melody as an agent. Each agent's goal is to compose his melody by choosing the *right notes*. On the one hand, he has to act according to the aesthetic criteria that exist for his voice; at the same time, he has to compose the voice in a manner compatible with the other voice-agent such that both together will result in a two-part counterpoint.

At every time unit in our simulations, each agent receives from the network a vector of activations for all the notes among which he can choose. Were the agent alone in the system, he would have chosen the note that got the highest activation from the neural network, meaning that this note is the one most expected to be next in the melody. But, in order to compose species, both agents will have to negotiate [RZ94] over all the other possible combinations to obtain a globally superior result.

In principle, each agent can suggest any of the n possible notes received from the network. Not all of these pairs of notes combinations are legal according to the rules of the species. In addition, there are specific combinations that are preferred over others for the current context. This idea is expressed in this module by computing a utility function for each pair of notes. In this sense, the goal of the agents is to agree on the pair of notes that is legal and has also achieved the maximal utility value among all the other options.

At each time unit, for each pair of notes, the agents start a negotiation process at the end of which a new note is added to each of the current melodies. Each agent sends to the other all of its notes, one at a time, and saves the pair consisting of his note and the other agent's note that is legal according to the first species style rules and that has yielded the maximal utility so far. At the end of this process the pair that has achieved the maximal utility is chosen. Both agents feed the network with this result as the current context so that the network can predict the next output. Each agent, then, receives a new input based on this output and the negotiation step is repeated until the melody is completed.

We define the system utility function to express the rules given by Jeppesen in [Jep92]. A pair of notes is considered legal according to the following rules:

1. The intervals between pairs of notes in the two part melodies should not be dissonant.
2. There should be perfect consonance in the first and last place of the melody.
3. Unison is only permitted in the first or last place of the melody.
4. Hidden and parallel fifths and octaves are

not permitted.

5. The difference between the previous and the current interval (when it is a fifth or an octave) should be two.
6. The interval between both tones cannot be greater than a tenth.
7. At most four thirds or sixths are allowed.
8. If both parts skip in the same direction, none of them will skip more than a fourth.
9. In each part, the new tone is different from the previous one.
10. In our system, we also don't allow to have more than two perfect consonants in the two part contra-punct (not including the first and last notes).

The utility function we chose is one example of a function that computes all the aspects we described in Section 2. The intuitions in this example are captured by preferring contrary motion. The function values will be determined according to whether the pairs of notes are legal or illegal based on the rules given above, and whether they are more preferred or less preferred, based on the net advice and fuzzy concepts given by the musician (e.g., contrary motion).

We have considered the contrary motion in this function because this type of motion produces the most natural and appropriate effect for this kind of music (as noticed by Jeppesen [Jep92]).

4 Experiments

We have first run every subsystem separately to examine the ability of each one of the two approaches (i.e, Neural Nets and agents) in order to cope with the general problem. We, then run the integrated system, and present results from all these simulations. In this way, we can show the ability of the whole system in producing better results, as opposed to the performance of each of the subsystems. Taking into consideration all the modules together gave us a more natural way to deal with the processing and the representation of our task.

4.1 Running the Net module

The task of the net was to learn to produce two new part melodies. This is a different case as the one faced by the whole system, in which only one part melodies were taught. Therefore, we needed to represent both parts of the melody simultaneously. We used the same sequential net that was described in Subsection 3.1. In this case, we doubled the number of the units in each layer to represent two notes simultaneoulsy, one for each part. In the learning phase the net was given four melodies, containing the two parts. One example from this set of melodies follows:

`V1:`*re*8 *do*8 *mi*8 *la* *do*8 *si* *la* *do*8 *re*8
`V2:`*re*8 *la*8 *sol*8 *fa*8 *mi*8 *re*8 *fa*8 *mi*8 *re*8

Since our notes are taken from one and a half octave, we represent the notes by their names (i.e, re) and those in a higher octave have an 8 concatenated to their names (i.e., re8). In this phase the net learned the examples in the set with high accuracy after a short training [1]. Each melody had a different label coded as a unique value in the plan units. After training, we tested the net by supplying as input the four labels 1000, 0100, 0010, 0001, one for each of the four melodies in the set, and the net was able to reproduce completely the sequence without mistakes. In the generalization phase we have chosen to interpolate the values of the plan units to produce new melodies. Todd [Tod89] demonstrated that the resulting sequences have non-linear similarities to the sequences in the learning set depending on the activations of the plan units. An example of a typical result follows:

The plan vector: (0.3 0.7 0.3 0.7).
`V1:`*re*8 *sol* *si* *do*8** *mi* *re**** *do*8 *re*8
`V2:`*re*8 *fa*8* *sol*8 *la*8 *sol*8 *fa*8 *mi*8 *re*8

This resulting sequence reflects typical problems we encountered when dealing with this sim-

[1] For 20 hidden units it took less than 100 epochs to achieve an average error around 0.0001

ple approach. The examples in the learning set imposed two different constraints on the net. The constraints regard the melodic intervals between the pitches in each part, and the combinations of pitches in both parts. The net is not able to cope with both constraints consistently, and then it satisfies each one at a time. For example in $*$, the combination chosen is not allowed in the specific style, although the melodic interval is fine. In $**$ and $***$ the descending skip is not permitted, but the combination is.

4.2 Running the Agents' module

The agents in our system know the rules of the specific style of the melodies we want to compose. They also know how to compute the system utility for a given pair of notes. We have run experiments with the agents' module solely. We cancel the influence of the recommendations produced by the neural network by giving the agents a vector of zero activations for all the possible notes. In this way, we wanted to check that the voices we will get by applying only the knowledge about the interactions between the two parts will lack the features learned by the net in its training phase (i.e. the aesthetics of one part).

We run the module with the utility function described in Section 3.2, where the net's advice was assigned to zero. We observed that melodies can get to a dead end, when there is no pair of notes that can satisfy the rules for the specific style. Since we choose the pair of notes that got the maximal utility value at each step, the result we got is:

`V1:`*re*8 *do*8 *la sol la mi la re*8

`V2:`*re*8 *mi*8 *fa*8 *sol*8 *fa*8 *sol*8 *fa*8 *re*8

The melody lacks the features requested from each part. In both voices there are redundant notes (i.e, the appearance of note la in V1, and the series of notes in V2 from the third place to the sixth). There is no unique climax in none of the voices. There are two continouos skips in the last three notes in V1. There are too many steps in V2 (i.e., there is no balance between the skips and the steps).

4.3 Running NetNeg

In this subsection we present the main simulations performed on the whole system. In the training phase, the network learned to reproduce four melodies that were taken from [Jep92]. See Figure 2.

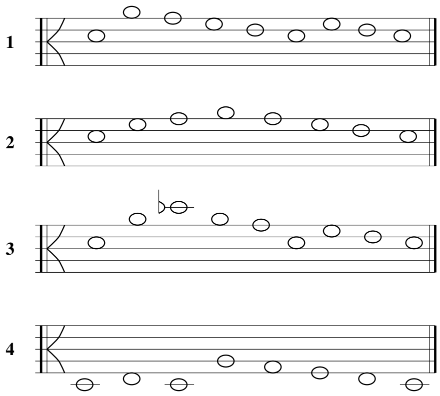

Figure 2: The learning examples

In the generalization phase, given a specific vector of plan units, the network produces a new cantus firmi. We have chosen two different plan vectors for the net that will output the notes for each agent. We run the net, each time with the correspondent plan vectors, and mapped their outputs to two different thirteen activation values. Then, we run the DAI-based module with these inputs. The agents negotiate over the different pairs of possible combinations, computing for each the system utility. Finally, the agents will agree upon a legal pair of notes that has yielded the maximal utility or might decide that no combination is legal, given the previous note in the melody. In our current case, the nets are fed with the agents agreement and the system continues to run. This process is executed until the two part melodies are completed. Currently the length of the melodies is fixed. A melody that resulted from an experiment we performed is shown in Figure 3. The net was presented with two different plan vectors ((0.8 0 0.8 0) and (0

1 0 1)). The agents computed the utility of the system taking into account the rules described in Section 3.2 and the contrary motion term.

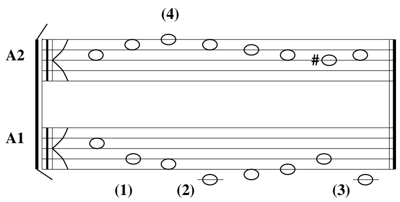

Figure 3: A new two-part melody

From Figure 3, we can observe that the system gives aesthetic results, quite appropriate for the species style we have experimented with. Both parts are consistent with the combinations constraints, as opposed to the simulation we run with the network, where this constraint was not satisfied. Comparing with the simulation run with the agents alone, no redundancy was found in this example. Nevertheless, there is a contour problem as pointed in (1) and (2) in A1's melody in Figure 3. According to Jeppesen [Jep92] it is preferred to descend by a step and then perform a descending skip. After a descending skip, we are expected to have a compensating ascending movement. In (3), we prefer to approach the last note by a step. A2's melody is perfect with regard to the contour. There is a single climax as shown in (4).

5 Summary and Future Work

We have presented a novel computational approach for encoding knowledge, intuitions and aesthetic taste of a musician in different modules. As a case study, we built a system for composing two-part counterpoint melodies. Our system design has been inspired by the cognitive process of a human musician, and the hybrid system is intended to broadly reflect that process. Another work that investigates how the cognitive process can guide the design of the architecture of a system that is involved in some musical activity can be found in [GL95].

The system is composed of a connectionist module and an agent-based module. The agents decide upon the notes in both melodies. The neural network predicts the expectation for every note to be the next one in the melody. The agents then negotiate over the possible combinations of notes until they reach an agreement on notes that, added to the melody, will most greatly "benefit" the system.

We have implemented a specific utility function, but as we have explained in Section 2, our architecture opens the possibility to run different kinds of functions. In a multi agent system, agents are self interested. This can be expressed by giving each agent a different utility function or non deterministic functions. We refer to a function as non deterministic when for the same pair of notes, it will return a different value. In our case, we started to investigate functions in which the contrary motion term was weighted by a coin toss. Issues to be further investigated include more natural ways to integrate both modules, the study of other species (second, third, and fourth species), and polyphonic music in a more flexible and more abstract style. We will in the future examine how other representations and negotiation protocols can influence the performance of the system.

References

[BG88] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[GL95] D. Gang and D. Lehmann. Harmonizing melodies with neural nets. In *Procs. of ICMC*, Banff, Canada, 1995.

[Jep92] K. Jeppesen. *Counterpoint The Polyphonic Vocal Style of the Six-*

teenth Century. Dover, New York, 1992.

[Jor86] M.I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Procs. of The Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, N.J., 1986.

[Miy91] Y. Miyata. *A User's Guide to PlaNet Version 5.6.* Computer Science Dept., University of Colorado, Boulder, 1991.

[MLM+92] T. A. Montgomery, J. Lee, D. J. Musliner, E. H. Durfee, D. Damouth, and Y. So. *MICE Users Guide.* Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan, February 1992.

[RZ94] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter:Designing Conventions for Automated Negotiation Among Computers.* MIT Press, Cambridge, Massachusetts, 1994.

[Tod89] P.M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13:27–43, 1989.

[WFG+95] M. Wooldridge, K. Fischer, P. Gmytrasiewicz, N. Jennings, J. Muller, and M. Tambe, editors. *Working Notes of the IJCAI95 Workshop on Agent Theories, Architectures, and Languages.* Montreal, Canada, 1995.