The power of a Multiagent Orchestra

A trying time for all

Halvor Sogn Haug

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Institute of informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2021

The power of a Multiagent Orchestra

A trying time for all

Halvor Sogn Haug

© 2021 Halvor Sogn Haug

The power of a Multiagent Orchestra

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Contents

Abstract

I’m so cool yo

Chapter 1

Introduction

Chapter 2

Background

2.1 Introduction

Music has long been one of the central ways for humans to express themselves creatively. From all the way back in prehistoric times, until now in the modern world, music has been a form of creative expression. The ubiquity and popularity of music as a creative task makes it a very attractive task for artificial intelligence (AI) research, not just for the creation of music itself, but also for research into creative activities in general. This field, concerned with using technology and algorithms to work with music, is called Musical Metacreation. Musical metacreation is defined as "*a subfield of computational creativity that adresses music related tasks*" (1). This does not only cover the creation of new music itself, but also real-time improvisation and accompaniment.
Multiagent systems are systems that instead of using a single agent to make decisions, uses multiple agents making individual decisions instead, that can create a greater whole together through emergence. Emergence is a term describing properties a system can have, that appears despite none of the individuals in the system having them.

2.2 Musical metacreation

The approaches to the process of musical metacreation are many and varied, depending on the goal. For example, you can hardcode in rules from music theory, or use machine learning to find the rules by itself. There's also a big difference between if the system itself is autonomous, interactive, or something in between. The system can also be offline, where the output isn't concerned with any real world interaction, or it can be in real-time, where the timing in relation to the real world matters.

2.2.1 Interactive vs. Autonomous

The difference between an interactive and an autonomous system, is that the interactive system allows the user to influence the behaviour, meanwhile an autonomous system will need no input after it has started.
The benefit of using an interactive system is that it allows a human to control

and influence how the output will sound. The system is therefore almost more like an instrument, in how the user uses it to "play". This also means that the system might not need a strict evaluation of its own output, as the user is able to influence it, and therefore able to act as an evaluator to the output. The main drawback of the direct systems is that you need a user to actively interact with the system for it to work well, and it could be argued that the system is not producing pleasant sounds on its own, but rather by the user's evaluation. This can be done in various ways, for example, you can have the user directly curating the output (2), is responsible for organizing the system (3), or the human user is directly giving inputs into the system that it adjusts and adapts to (4).
When the user is curating they the output, the system is creating many different melodic ideas, in which the user is curating and removing those that they find displeasing. In Studley et al.'s paper, the user is playing a game. This encourages the user to remove the musical ideas they don't like, while also constantly having new musical ideas show up, creating a dynamic experience. This approach is also different from other interactive approaches, where the human user gives some input that the system react to, here it's the system that creates musical ideas, and it's up to the user to react and respond to it.
For systems where the user is organizing it, as in Bennet's paper on Neurhythmic, where central pattern generators (CPGs), who are oscillators that can influence each other, the user is responsible for creating and placing these generators. This is able to create novel musical ideas.
Lastly, there are the systems where the user gives input directly. They are generally reactive systems, and will not produce musical ideas on their own. Rather, they work on embellishing on the ideas given as input, and are therefore often used for accompaniment.
On the other hand there are autonomous systems, who do not require any input from humans at all, and can run on their own. As these systems are working on their own, they need some kind of evaluation to determine how good the created music is, however they are also able work offline, which means that they can look at the whole of the musical output to create a structure, as opposed to real time systems, which are closer to improvisation.

2.2.2 Harcoded rules vs. Machine Learning

The main benefit of hardcoding the rules, is that you can ensure it follows some kind of musical framework, which in turn will increase the quality of output produced. The drawback on the other hand is that it can often get restricted by the rules, which can limit the variety of the musical output.
When you hardcode musical rules, there are multiple approaches you can take, that also depends on if the system is interactive or autonomous. For an autonomous system, some degree of stochasticity is needed to not have the system have the exact same output over and over. And example of this is Farbood et. al's paper on synthesizing music for Palestrina-style counterpoint (5). In the paper, they set up tables with probability transitions between notes in terms of the intervals, and used that combined with Markov Chains to predict the next note that should be played. It is also worth mentioning that while they hardcoded the musical theory rules, they still took advantage of machine learning techniques to combine it together for the final output.

Using counterpoint was also built on by Plutt et. al, in their paper on LazyVoice (6). LazyVoice is a multi-agent system focused on creating pleasant harmonies. This is done by using voice leading, specifically that notes should move to the closest possible note that fits into the chord selected. Each individual agent is represented as being a voice line, and has parameters for how long each note will be, how deep into the chord it will look to find an appropriate note (so it can play 7th and 9th chords), and is also able to be influenced into moving either away from or towards other notes.
The opposite to hardcoding rules would be musical systems that takes advantage of machine learning in some way. This can either be done by having the system learn the rules from a corpus, or it can try to learn the rules through some evaluation function.
Learning from a corpus, the main goal is to in some way imitate the music the corpus consists of, and use that as a basis for learning rules. For example, in Tatar et. al's paper on using self-organizing maps (7), the training set consists of music of one style that should be imitated. Self-organizing maps are a method for reducing the dimensionality of a set of data, and this is used to classify different "sections", or repeating ideas in the corpus. This allows the entire musical piece to be sliced into smaller parts that are each marked as a category, and then they can use that string of categories to train a Markov Chain. However, it is important to note here that the way the self-organizing maps are able to reduce the dimensions, is by looking at the features extracted from the audio. These features, which are based on a calculation of the valence and arousal of any particular sound, are implemented by hardcoding in a formula for them. This is a general principle for most systems that take advantage of machine learning in some way, that some part of it will have to be hardcoded. This is because the use of heuristics are a natural part of Artificial Intelligence, as the space the algorithm can work in tends to be too big otherwise.
The self-organizing maps from MASOM was also used in StyleRank, a system for classifying music based on style in some way, whether it be genre or composer (8). As it is focused on learning the rules, to be able to classify the music, it is mainly using Machine Learning. However, as with MASOM, the way the music is able to be classified is by extracting features, which requires some hardcoding.
For systems where the user is responsible for organizing the system, such as Neurhythmic mentioned earlier, at least part of the hardcoded rules or machine learning is substituted by the human participant. In Neurhythmic for example, the user is responsible for structuring the CPGs and connecting them together, which is then used to generate rhythm (3). As the rhythm generation is done by playing according to the CPGs, whose main behaviour is largely dependent on their connections and whether they resonate with them,

2.3 Multi-agent systems

There are generally two types of multiagent system:
The first one is swarm intelligence, which uses simple agents and takes a lot of inspiration from nature. These agents follow simple rules often based on the 'senses' of the agent.
The second one is based on economics and sociology, and uses more complex

agents applying game theory to take the action that benefits themselves the most.

2.3.1 Swarm intelligence

Swarm Intelligence is characterized by six properties of the agents (9). Those six are:

1. Autonomy
2. Distributedness
3. Simple agents
4. Emergence
5. Stigmergy
6. Self-organization

Autonomy means that each agent is able to move about and operate on its own, independent from any other agents. This is a trait common to all agents in a multi-agent system, also agents that apply game theory.
Distributedness means that there is no central control sending signals and directing the agents. Rather, the agents are taking all decisions independently, and on their own. This means that the system only requires the simulation of the agents and environment, and not an organizer that keeps track of every agent.
Simple agents means that the agents do not have complex thoughts, but instead make decisions based on a set of rules, usually by using some kind of sensor input to determine what action to take. For example, in the ant colony optimization algorithm the agents will choose which path to take randomly, but will be more likely to choose paths with a heavy pheromone trail.
Emergence means that the simple interactions the agents have with each other, when seen as the whole, will create some kind of greater effect. This emergent effect could be said to be the goal of the swarm system itself.
Stigmergy is how the agents communicate with each other, which is not directly, but rather indirectly by changing the environment. This can be by laying out pheromone trails, making sounds, or by stopping and becoming an obstacle in the environment themselves. This also means there's no need to implement communication protocols between agents, since they all instead simply interact with the environment.
Self-organization means that the swarm of agents organizes itself on its own. This specifically matters in situations where there are different tasks that needs to be done, or where more than one agent is needed to complete a task. The main benefit of this, is that it makes the swarm flexible and robust, so even if some agents either go offline or crash, the rest of the swarm can still operate, or that there are no essential individual agents.
Lastly, while this isn't a formal requirement, it is usual for swarm intelligence systems to consist of a large number of agents, usually 100 or more. This allows the system to be more stable in regards to self-organization, as the loss of one agent is more critical the fewer total agents the are. As the agents will always follow simple rules, it's also easier to scale up the number of agents, as each agent needs less processing power.

Interactive Swarm Orchestra

At the Zurich University of the arts, Bisig et al. has been working on applying swarms to an interactive orchestra, that they've been conducting quite a few experiments with (10). The system consists of different modules, but the ones of most interest here are the synthesis and flock modules. The synthesis module is focused on converting various inputs into an audio output. As it is a general framework, there is no specific method introduced for converting to audio. The flock module is also a general framework for doing swarm simulations, whose parameters in the space will be passed along to the synthesis as either input or control parameters.
The system has later been expanded to include experiments where the whole system has been turned into an instrument, with specific tools for a human participant to influence the swarm and therefore its output (11). Out of this has also come research into what parameters are good to use when using swarms for musical metacreation.There have also been forays into having the system interact with external devices that work as instruments, which can interact with and influence the swarm. Suggested has been both using the position of the swarm(s) to determine the amplitude and frequency of oscillators, or to combine it with granular synthesis. It is also suggested that you could use the distance between the swarm agents as parameters.
The main interest with this system is it being one of the first general swarm music systems, meaning that it is not bound to any particular implementation or algorithm, but rather is able to support any of them if implemented. In this way, this is more of a framework than one particular implementation, but it does also suggest a structure for similar projects, in that the swarm implementation and synthesiser could be kept seperate to avoid interaction. There may be some benefits to having them interact though, such as letting audio events influence the actions of the swarm directly.

Boids

One of the most common algorithms to use for creative swarms is the boids algorithm, which simulates the flocking behaviour of birds. The algorithm was presented by Reynolds in 1987 (12), and it consists of individual agents who follow three simple rules:

1. The agents should adjust their path so they don't collide with nearby agents.

2. The agents should attempt to match the velocity of the agents around them.

3. The agents should try to stay close to the agents around them.

These three rules will ensure that the agents stay together with out colliding, and that they move in the same directions. The rules can also be expanded to include obstacle avoidance, which lets the agents move around in a simulated environment.
The boids model fits all the requirements for a swarm model, as the agents are autonomous and make decisions on their own, they only follow a set of simple rules, the flocking behaviour emerges from the agents following those rules, they

change the environment for communication by themselves moving around, and the swarm is very flexible to the number of agents present, even if some were to be defective.
The reason the boids algorithm is used so much, is that unlike many other swarm algorithms, it is very open-ended. Where most swarm algorithms have an optimization goal they want to achieve, and that the agents will work towards solving, the boids do not have that, and the main behaviour is rather moving around in a coordinated fashion. This is particularly good for musical endeavours, as coordinated movements is something that often show up in music.
One of the first implementations of boids for musical purposes was done by Unemi et. al (13). In their paper, the boids agents moved around in a 3D space, where the coordinates are mapped to represent pan, pitch, and velocity. In addition, the swarm was able to be interacted it by filming human participants, who the swarm then corresponded to in how they moved. The various parameters that swarm algorithms could use to convert into musical ideas was also expanded on by Bisig et. al in 2008 (14). The most basic is similar to what they did with boids, where the coordinates are directly translated into pan, pitch and velocity for the notes. It's also possible to take it a step further and have the coordinates correspond to parameters for oscillators, which will then generate music when added together. Alternatively, it is possible to use granular synthesis, a method where the audio is split up into small samples that are then later modified by adding them together or modifying their parameters such as the speed they're played at, their amplitude, their phase, and so on. You can then have those parameters decided by the agents' position, velocity, and distance to other neighbours, to create audio output.
Using a camera to have the user interact has also been done in the SwarmArt project, where boid swarms could be interacted with by human participants over film. (15). Here, the swarm is given the coordinates of objects that move in front of the camera, and use those to determine where to go. The main benefit of using a camera for human participation is that it allows for an easy and intuitive way for anyone to be able to interact with the swarm.

Multi-swarms

Multi-swarms are swarm systems that take advantage of having multiple separate swarms existing in the space (16). The main benefits to this approach is that they can be better for solving problems where the search landscape is dynamic, and can change over time. Blackwell have already used this approach to create a system that produces musical output, both by its own and in conjunction with human participants (4). It works by using multiple particle swarm optimization swarms, who consist of particles being attracted toward the best spot they've found themselves as well as the best spot the swarm has found, and these swarm then exist simultaneously in the same environment. The 3d environment space represents the loudness, pulse, and pitch of the tone. It then captures both internal and external audio events, and uses them as basis for the target points of the swarm. How the targets are decided is determined by scripts, who process the audio events, mostly to prevent convergence and to make sure the swarms are spread out.

2.3.2 Game Theory

Game theory agents differ from swarm intelligence agents in that instead of just following rules, they try to observe the environment around them, and deliberately choose the option that will benefit themselves the most (17). They are also able to establish communication protocols for directly talking with each other, so they can try and come to decisions that are mutually beneficial. As the agents have to evaluate their environment, and then predict the most beneficial actions for themselves based on what other agents will choose, game theory agents require significantly more processing power than swarm agents. Due to this, the number of game theory agents you can have running in simulation will be fewer, and they will also be more expensive to manufacture for real world operations.
Unlike swarm intelligence systems, game theory systems usually require more processing power from each agent, as they're expected to evaluate and predict other agents' actions as well. This means that the number of agents tends to be smaller than for a swarm system, and as the rules are more explicitly coded into the agents as well, there is less of an "emergence", regarding the behaviour of the system.
The common aspect to all of the following examples using game theory agents is that unlike the swarm systems, the rules are very finely tuned towards the task they're supposed to solve. In addition, the number of agents are generally quite few, but they can allow for human participants to take part as one of the agents, instead of influencing the entire system as is common in the swarm projects.

Agents talking with each other

There are multiple ways to implement game theory agents, and one of them, is implementing agents which directly talk with each other out loud in real time and try to anticipate what to do based on the audio input they get. Examples of this type of agent can be seen in Boersen et. al's paper on Chatterbox, an art installation for the purpose of simulating multicultural communication (18). This is done by having Chatterbox agents, who are able to generate gibberish vocal streams, that has no meaning. This is done by using self-organizing maps to categorize human speech, specifically the MASOM system mentioned earlier. Afterwards, the agents are given instructions on how to interact with each other, with different layers. Layer 1 is for example to start speaking if there's silence, and to stop after it has "finished speaking" so to say. Additional layers of rules, including trying to predict when other agents will finish speaking so they can start speaking themselves, leads to a dynamic behavior where the agents are in constant conversation with each other.
This installation is also similar to Rokeby's n-Cha(n)t installation, in that it features various agents communicating with each other, that can also listen to the human speakers who are there (19). However, while in Chatterbox the agents try to hold a conversation by speaking in turn and not interrupting each other, in n-Cha(n)t the agents will rather synchronize if left unattended, and start chanting. If they recieve outside stimuli from a human, they will try to process what they hear, and add on to that.
An interesting part of these types of set-up is how they work through real space, and also incorporate human participants, which allows for the system to

be both automatic and interactive. This could also be done in some kind of virtual environment where the user is allowed to walk around, to maintain the idea of space and location mattering, which can be quite useful for auditory tasks. Another interesting factor is how the systems all listen to each other and try to act according to what they hear, essentially having to anticipate what the other agents will do, which is very useful for musical tasks, as you need to anticipate what the other agents will play so you can harmonize with them. This is especially useful when it comes to improvisational tasks, or where the system works real-time.

Passing it around

Another technique that can be used for implementing game theory agents, is the mechanism of passing either responsibility or the output around between the agents, and in the period where an agent have responsibility, they have full control. This can be done both in real-time or offline, depending on the system. An example of this comes up in Chandra et. al's paper on SoloJam, which uses auctioning theory to distribute the time a participant can play a solo (20). This is done by having the agent/node that is currently performing a solo auction off the right to continue the solo, using a sealed-bid, second price auction, also known as a Vickrey auction. Each node bids with its own rhythmic pattern, and the amount bid corresponds to the evaluation of the rhythmic pattern. This will ensure that it's always the best rhythmic pattern that will have the solo. This system works in real-time, and the responsibility for the solo is passed around between the participants as they play.
Another example of this can be seen in Kirke et al.'s paper on multiagent poetry.(21) By giving each of the agents a "mood", based on valence and arousal, they had the agents select appropriate words corresponding to their mood for their poems, then it was passed around to another agent. After reading the poem that they've received, their mood is affected by the contents of the poem, and then they added text to the poetry based on their new mood. This process continues until the program is stopped. The result is a set of poems that are filled with repetition, but has a somewhat recursive structure to itself. While the poems in text form appear quite nonsensical, this might work better in a musical context, as musical ideas for different emotions are more vague and less directly affected by the direct reading as words are. As opposed to SoloJam, this instead works offline, and every agent is processing a piece of output at the same time, and passing it around. This system is therefore also more suitable for composition as opposed to improvisation, unlike the other game theory agents that have been presented.

Chapter 3

Design and Implementation

Chapter 4

Evaluation

Chapter 5

Conclusion and Future Work

Bibliography

[1] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov, "An introduction to musical metacreation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 1–14, 2017.

[2] J. D. Thomas Studley and N. Scott, "Exploring competitive musical creativity in digital composition games," in *Proceedings of the 7th International Workshop on Musical Metacreation* (B. L. Sturm and R. M. Keller, eds.), 2019.

[3] D. Bennett, P. Bennett, and A. Roudaut, "Neurythmic: A rhythm creation tool based on central pattern generators.," in *NIME*, pp. 210–215, 2018.

[4] T. Blackwell, "Swarm music: improvised music with multi-swarms," *Artificial Intelligence and the Simulation of Behaviour, University of Wales*, vol. 10, pp. 142–158, 2003.

[5] M. Farbood and B. Schöner, "Analysis and synthesis of palestrina-style counterpoint using markov chains," in *ICMC*, 2001.

[6] C. Plutt and P. Pasquier, "Lazyvoice : Efficient voice-leading calculation," in *ICMC*, 2021.

[7] K. Tatar and P. Pasquier, "Masom: A musical agent architecture based on self organizing maps, affective computing, and variable markov models," in *Proceedings of the 5th International Workshop on Musical Metacreation (MUME 2017). Atlanta, Georgia, USA*, 2017.

[8] J. Ens and P. Pasquier, "Quantifying musical style: Ranking symbolic music based on similarity to a style," *arXiv preprint arXiv:2003.06226*, 2020.

[9] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Théraulaz, G. Theraulaz, *et al.*, *Swarm intelligence: from natural to artificial systems.* No. 1, Oxford university press, 1999.

[10] D. Bisig, M. Neukom, and J. Flury, "Interactive swarm orchestra a generic programming environment for swarm based computer music," in *ICMC*, 2008.

[11] D. Bisig and P. Kocher, "Tools and abstractions for swarm based music and art," in *ICMC*, 2012.

[12] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, 1987.

[13] T. Unemi and D. Bisig, “Music by interaction among two flocking species and human,” in *Proceedings of the third international conference on generative systems in electronic arts, Melbourne, Australia*, pp. 171–179, 2005.

[14] D. Bisig and M. Neukom, “Swarm based computer music-towards a repertory of strategies,” *computer music*, vol. 12, p. 13, 2008.

[15] J. E. Boyd, G. Hushlak, and C. J. Jacob, “Swarmart: interactive art from swarm intelligence,” in *Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 628–635, 2004.

[16] T. Blackwell and J. Branke, “Multi-swarm optimization in dynamic environments,” in *Workshops on Applications of Evolutionary Computation*, pp. 489–500, Springer, 2004.

[17] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[18] R. Boersen, A. Liu-Rosenbaum, K. Tatar, and P. Pasquier, “Chatterbox: an interactive system of gibberish agents,” in *Proceedings of the 26th International Symposium on Electronic Art (ISEA)*, pp. 55–62.

[19] D. Rokeby, “n-cha(n)t,” 2001.

[20] A. Chandra, K. Nymoen, A. Voldsund, A. R. Jensenius, K. Glette, and J. Torresen, “Market-based control in interactive music environments,” in *International Symposium on Computer Music Modeling and Retrieval*, pp. 439–458, Springer, 2012.

[21] A. Kirke and E. Miranda, “Emotional and multi-agent systems in computer-aided writing and poetry,” in *Proceedings of the Artificial Intelligence and Poetry Symposium*, pp. 17–22, 2013.