An Agent-based System for Robotic Musical Performance

Arne Eigenfeldt
School of Contemporary Arts
Simon Fraser University
Burnaby, BC
Canada
arne_e@sfu.ca

Ajay Kapur
School of Music
California Institute of the Arts
Valencia, CA
USA
akapur@calarts.edu

ABSTRACT
This paper presents an agent-based architecture for robotic musical instruments that generate polyphonic rhythmic patterns that continuously evolve and develop in a musically "intelligent" manner. Agent-based software offers a new method for real-time composition that allows for complex interactions between individual voices while requiring very little user interaction or supervision. The system described, *Kinetic Engine,* is an environment in which individual software agents, emulate drummers improvising within a percussion ensemble. Player agents assume roles and personalities within the ensemble, and communicate with one another to create complex rhythmic interactions. In this project, the ensemble is comprised of a 12-armed musical robot, *MahaDeviBot*, in which each limb has its own software agent controlling what it performs.

Keywords
Robotic Musical Instruments, Agents, Machine Musicianship.

1. INTRODUCTION
MahaDeviBot [11, 12] is a robotic drummer comprised of twelve arms, which performs on a number of different instruments from India, including frame drums, shakers, bells, and cymbals. As such, it is, in itself, an ensemble, rather than a single instrument; to effectively create music for it – particularly generatively in real-time performance - an intelligent method of interaction between the various instruments is required.

The promise of agent-based composition in musical real-time interactive systems has already been suggested [23, 18, 16], specifically in their potential for emulating human performer interaction. Agents have been defined as autonomous, social, reactive, and proactive [22], similar attributes required of performers in improvisation ensembles.

The notion of an "agent" varies greatly: Minsky's original agents [15] are extremely simple abstractions that require interaction in order to achieve complex results. Recent work by Beyls [1] offers one example of such simple agents that individually have limited abilities, but can co-operate to create high levels of musical creation.

The authors' view of agency is directly related to existing musical paradigms: the improvising musician. Such an agent must have a much higher level of knowledge, but, similar to other multi-agent systems, each agent has a "limited viewpoint" of the artistic objective, and, as such, collaboration is required between agents to achieve (musical) success.

Kinetic Engine [6, 7], created in Max/MSP, is a real-time generative system in which agents are used to create complex, polyphonic rhythms that evolve over time, similar to how actual drummers might improvise in response to one another. A conductor agent loosely co-ordinates the player agents, and manages high-level performance parameters, specifically *density*: the number of notes played by all agents. Each agent manages one of the percussion instruments of *MahaDeviBot*, aware of its function within the ensemble, and its specific physical limitations.

2. RELATED WORK
2.1 Multi-agent Systems
Multiple-agent architectures have been used to track beats within acoustic signals [10, 5] in which agents operate in parallel to explore alternative solutions. Agents have also been used in real-time composition [21, 3]. Burtner suggests that multi-agent interactive systems offer the possibility for new complex behaviours in interactive musical interfaces that can "yield complexly organic structures similar to ecological systems". Burtner's research has focused upon performance, and extending instrumental technologies, rather than interactive composition; as such, his systems are reactive, rather than proactive, a necessary function of agency.

Dahlstedt and McBurney [4] developed a multi-agent model based upon Dahlstedt's reflections on his own compositional processes. They suggest such introspection will "yield lessons for the computational modeling of creative processes". Their system produces output "that (is) not expected or predictable – in other words, a system that exhibits what a computer scientist would call *emergent properties*."

Wulfhorst et al. [23] created a multi-agent system where software agents employ beat-tracking algorithms to match their pulse to that of human performers. Although of potential benefit for real-time computer music and robotic performance, the research's musical goals are rather modest: "Each agent has a

defined rhythmic pattern. The goal of an agent is to play his instrument in synchronism with the others."

Murray-Rust and Smaill's *AgentBox* [17] uses multi-agents in a graphic environment, in which agents "listen" to those agents physically (graphically) close to one another. A human conductor can manipulate the agents - by moving them around - in a "fast and intuitive manner", allowing people to alter aspects of music "without any need for musical experience". The stimulus behind *AgentBox* is to create a system that will "enable a wider range of people to create music," and facilitate "the interaction of geographically diverse musicians".

2.2 Rhythm Generation

Various strategies and models have been used to generate complex rhythms within interactive systems. Brown [2] describes the use of cellular automata (CA) to create monophonic rhythmic passages and polyphonic textures in "broad-brush, rather than precisely deterministic, ways." He suggests "CA provide a great deal of complexity and interest from quite simple initial set-up". However, complexity generated by CA is no more musical than complexity generated by constrained randomness. Brown recognises this when he states that rhythms generated through the use of CA "often result in a lack of pulse or metre. While this might be intellectually fascinating it is only occasionally successful from the perspective of a common aesthetic."

Pachet [19] proposes an evolutionary approach for modeling musical rhythm, noting that "in the context of music catalogues, [rhythm] has up to now been curiously under studied." In his system, "rhythm is seen as a musical form, emerging from repeated interaction between several rhythmic agents." Pachet's model is that of a human improvisational ensemble: "these agents engage into a dynamic game which simulates a group of human players playing, in real time, percussive instruments together, without any prior knowledge or information about the music to play, but the goal to produce coherent music together."

Agents are given an initial rhythm and a set of transformation rules from a shared rule library; the resulting rhythm is "the result of ongoing play between these co-evolving agents." The agents do not actually communicate, and the rules are extremely simple: i.e. add a random note, remove a random note, move a random note. The system is more of a proof of concept than a performance tool; it developed into the much more powerful *Continuator* [20], a real-time stylistic analyser and variation generator.

Martins/Miranda [13] describe a system the uses a connectionist approach to representing and learning rhythms using neural networks. The approach allows for the computer to learn rhythms through similarity by mapping incoming rhythms in a three dimensional space. The research is part of a longer project [16, 14] in which self-organising agents create emergent music through social interactions; as such, the emphasis is not upon the interaction of rhythms as in the emergence of new and/or related rhythmic patterns.

Gimenes [9] explores a memetic approach that creates stylistic learning methods for rhythm generation. As opposed to viewing rhythmic phrases as consisting of small structural units combined to form larger units (a more traditional method of musical analysis), the memetic approach suggests longer blocks that are dependent upon the listener (suggesting a more recent cognitive method of rhythmical analysis that utilizes "chunking"). *RGeme* "generates rhythm streams and serves as a tool to observe how different rhythm styles can originate and evolve in an artificial society of software agents."

Kinetic Engine, in collaboration with *MahaDeviBot*, builds upon such previous efforts; however, it is fundamentally different in two respects: firstly, it is a real-time system with performance as its primary motivation; secondly, the software controls a physical instrument that requires mechanical movement.

3. AGENT-GENERATED RHYTHM

It is important to recognize that rhythmic intricacy can result not only from the evolution of individual rhythms, but also through the interaction of quite simple parts; such interaction can produce musical complexity within a system. The interrelationship of such simple elements requires musical knowledge in order to separate interesting from pedestrian rhythm. Such interaction suggests a multi-agent system, in which complexity results from the interaction of independent agents.

Existing musical models for such a system can be found in the music of African drum ensembles and Central and South American percussion ensembles (note that Indian classical music, which contains rhythmic constructions of great complexity, is fundamentally solo, and therefore lacks rhythmic *interaction* of multiple layers). Furthermore, models for the relationship of parts within an improvising ensemble can be found in jazz and certain forms of Techno. For more information on such modeling, see [8].

4. TOOLS

4.1 MahaDeviBot

Figure 1. *MahaDeviBot* controlled by *Kinetic Engine*.

The development of the *MahaDeviBot* serves as a paradigm for various types of solenoid-based robotic drumming techniques, striking twelve different percussion instruments gathered from around India, including frame drums, bells, finger cymbals, wood blocks, and gongs. The machine even has a bouncing head that can portray tempo to the human performer. The *MahaDeviBot* serves as a mechanical musical instrument that extends North Indian musical performance scenarios, which arose out of a desire to build a pedagogical tool to keep time and help portray complex rhythmic cycles to novice performers in a way that no audio speakers can ever emulate. It accepts MIDI messages to communicate with any custom software or hardware interface.

4.2 Kinetic Engine

Kinetic Engine is a real-time composition/performance system created in Max/MSP, in which intelligent agents emulate improvising percussionists in a drum ensemble. It arose out of a desire to move away from constrained random choices and utilise more musically intelligent decision-making within real-time interactive software.

The principle human control parameter in performance is limited to density: how many notes played by all agents. All other decisions - when to play, what rhythms to play in response to the global density, how to interact with other agents – are left to the machines' individual agents.

Agents generate specific rhythms in response to a changing environment. Once these rhythms have been generated, agents "listen" to one another, and potentially alter their patterns based upon these relationships. No databases of rhythms are used: instead, pre-determined musical rules determine both generation and alteration of rhythmic patterns.

5. AGENTS

Agent-based systems allow for limited user interaction or supervision, allowing for more high-level decisions to be made within software. This models interactions between intelligent improvising musicians, albeit with a virtual conductor shaping and influencing the music.

There are two agent classes: a conductor and an indefinite number of players (although in this case the agents are limited to the twelve instruments of the robot).

5.1 Conductor Agent

The conductor agent (hereafter simply referred to as "the conductor") has three main functions: firstly, to handle user interaction; secondly, to manage (some) high-level organisation; thirdly, to send a global pulse.

Kinetic Engine is essentially a generative system, with user interaction being limited to controlling only a few global parameters:

- *individual on/off* – individual agents can be forced to "take a rest" and not play.
- *density* – the relative number of notes played by all agents. (Described in section 6.1).
- *global volume* – the approximate central range of an agent's velocity. Agents vary their velocities independently, and will "take solos" (if they feel they are playing something interesting) by increasing their velocity range; however, their central velocity range can be overridden by the conductor.
- *agent parameter scaling* – the user can influence how the individual agents may react. (Described in section 5.2).
- *new pattern calculation* – agents can be forced to "start again" by regenerating their patterns based upon the environment.

Metre, tempo, and subdivision are set prior to performance by the user, and remain static for a composition. The conductor also sends a global pulse, to which all player agents synchronise.

5.2 Player Agents

Upon initialisation, player agents (hereafter referred to simply as "agents") read a file from disk that determines several important aspects about their behaviour; namely their *type* and their *personality*.

Type can be loosely associated with the instrument an agent plays, and the role such an instrument would have within the ensemble. Table 1 describes how *type* influences behaviour.

Table 1. Agent *type* and influence upon agent behaviour.

	Type *Low*	Type *Mid*	Type *High*
Timbre	low frequency: • frame drums	midrange frequency: • gongs • shakers	high frequency: • hand drum • tambourine
Density	lower than average	average	higher than average
Variation	less often	average	more often

The stored personality traits include *Downbeat* (preference given to notes on the first beat), *Offbeat* (propensity for playing off the beat), *Syncopation* (at the subdivision level), *Confidence* (number of notes with which to enter), *Responsiveness* (how responsive an agent is to global parameter changes), *Social* (how willing an agent is to interact with other agents), *Commitment* (how long an agent will engage in a social interaction), and *Mischievous* (how willing an agent is to disrupt a stable system). A further personality trait is *Type-scaling*, which allows for agents to be less restricted to their specific types. For example, low agents will tend to have lower densities than other types, but a low agent with a high type-scaling will have higher than usual densities for its type. See Figure 2 for a display of all personality parameters.

Figure 2. Example personality parameters for a player agent.

6. RHYTHMIC CONSTRUCTION

6.1 Density

Agents respond to the global *density* variable – this correlates to the number of notes playing within a measure. Agents are unaware of the exact global density required, and instead rely upon the conductor to rate the requested density as "very low", "low", "medium", or "high" and broadcast this rating. Agents know the average number of notes in a pattern based upon this rating, which is scaled by the agent's type and type-scaling parameter. Agents apply a Gaussian distribution around this average, and choose an actual density from within this curve, thereby maintaining some unpredictability in actual density distribution.

The conductor collects all agent densities, and determines whether the accumulated densities are "way too low/high", "too low/high", or "close enough" in comparison to the global density, and broadcasts this success rating.

[1] if the accumulated density is "way too low", non-active agents can activate themselves and generate new densities (or conversely, active agents can deactivate if the density is "way to high").

[2] if the accumulated density is "too low", active agents can add notes (or subtract them if the density is "too high").

[3] if the accumulated density is judged to be "close enough", agent densities are considered stable.

6.2 Density Spread

An agent's density (i.e. seven notes) is "spread" across the available beats (i.e. four beats) using fuzzy logic to determine probabilities, influenced by the agent's downbeat and offbeat parameters (see Figure 3 for an example of probability weightings spread across four beats). Thus, an example spread of seven notes for agent A, below, might be (3 1 2 1), in which each beat is indicated with its assigned notes.

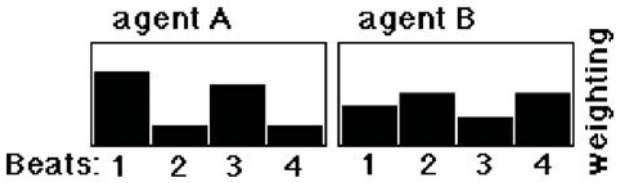


Figure 3. Example density spread weightings for two agents, 4/4 time with different downbeat and offbeat parameter values.

Agents determine the placement of the notes within the beat using a similar technique, but influenced by the agent's syncopation parameter.

6.3 Pattern Checking

After an initial placement of notes within a pattern has been accomplished, *pattern checking* commences. Each beat is evaluated against its predecessor and compared to a set of rules in order to avoid certain patterns and encourage others.

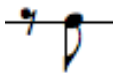
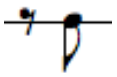
Previous beat	Pattern A	Pattern B
		
	30%	90%

Figure 4. Example pattern check: given a previous beat's rhythm, with one note required for the current beat, two "preferred" patterns for the current beat.

In the above example, if the current beat has one note in it, and the previous beat contains the given rhythm, a test is made (a random number is generated between 0 and 1). If the generated number is less than the coefficient for pattern A (.3, or a 30% chance), the test passes, and pattern A is substituted for the original pattern. If the test fails, another test is made for pattern B, using the coefficient of .9 (or 90%). If this last test fails, the original rhythm is allowed to remain. Using such a system, certain rhythmic patterns can be *suggested* through probabilities. Probability coefficients were hand-coded by the first author after extensive evaluation of the system's output.

7. SOCIAL BEHAVIOUR

Once all agents have achieved a stable density and have generated rhythmic patterns based upon this density, agents can begin social interactions. These interactions involve potentially endless alterations of agent patterns in relation to other agents; these interactions continue as long as the agents have a social bond, which is broken when testing an agent's social commitment parameter fails. This test is done every "once in a while", an example of a "fuzzy" counter.

Social interaction emulates how musicians within an improvising ensemble listen to one another, make eye contact, and interact by adjusting and altering their own rhythmic pattern in various ways. In order to determine which agent with which to interact, agents evaluate other agent's *density spread*. Evaluation methods include comparing density spread averages and weighted means, both of which are fuzzy tests.

Table 2. Example density spreads in 4/4: comparing agent 1 with agents 2 and 3.

Agent #	1	2	3
Density spread	3 1 2 2	1 2 2 1	2 3 3 3
Similarity rating		0.53	0.48
Dissimilarity rating		0.42	0.33

An agent generates a *similarity* and *dissimilarity* rating between its density spread and that of every other active agent. The highest overall rating will determine the type of interaction: a dissimilarity rating results in rhythmic polyphony (interlocking), while a similarity rating results in rhythmic heterophony (expansion). Note that interlocking interactions (dissimilarities) are actually encouraged through weightings.

Once another agent has been selected for social interaction, the agent attempts to "make eye contact" by messaging that agent. If the other agent does not acknowledge the message (its own social parameter may not be very high), the social bond fails, and the agent will look for other agents with which to interact.

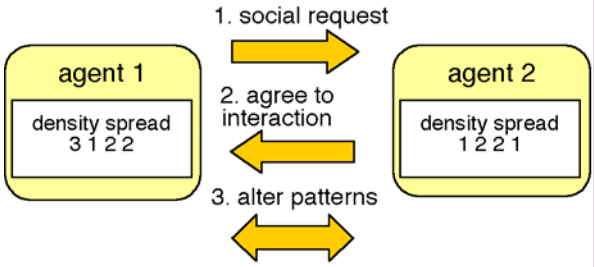


Figure 5. Social messaging between agents.

7.1 Interaction types: Polyphonic

In polyphonic interaction, agents attempt to "avoid" partner notes, both at the beat and pattern level. For example, given a density spread of (3 1 2 2) and a partner spread of (1 2 2 1), both agents would attempt to move their notes to where their partner's rests occur (see Figure 6). Because both agents are continually adjusting their patterns, stability is actually difficult to achieve.

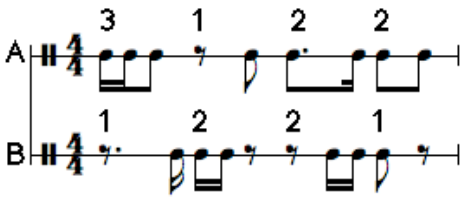


Figure 6. Example polyphonic interaction result between agents A and B, with density spreads of (3 1 2 2) and (1 2 2 1). Note that not all notes need to successfully avoid one another (beats 3 and 4).

7.2 Interaction types: Heterophonic

In heterophonic interaction, agents alter their own density spread to more closely resemble that of their partner, but no attempt is made to match the actual note patterns (see Figure 7).

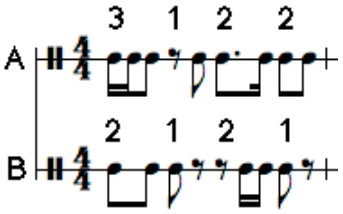


Figure 7. Example heterophonic interaction result between agents A and B, with density spreads of (3 1 2 2) and (2 1 2 1). Agent B had an initial spread of (1 2 2 1).

8. ADDITIONAL AGENT KNOWLEDGE

Because each agent is sending performance information, via MIDI, to a specific percussion instrument, agents require detailed knowledge about that instrument. Each instrument has a discrete velocity range, below which it will not strike, and above which it may double strike. These ranges change each time the robot is reassembled after moving. Therefore, a velocity range test patch was created which determines these limits quickly and efficiently before each rehearsal or performance. These values are stored in a global array, which each agent directly accesses in order to appropriately choose velocities within the range of its specific instrument.

Similarly, each instrument also has a physical limit as to how fast it can re-strike; this limit is also determined through a test patch used to inform the program regarding potential tempo limitations. For example, the frame drums have limits of approximately 108 BPM for three consecutive sixteenths (138 ms. inter-onset times) while the tambourine and hand-drum can easily play the same three sixteenths at over 200 BPM (better than 75 ms inter-onset times). The conductor will limit the overall tempo and subdivisions so as not to exceed these limitations; furthermore, individual agents will attempt to limit consecutive notes for each drum at contentious tempi.

9. CONCLUSION

Kinetic Engine has been used previously as an independent ensemble, both autonomously (as an installation) and under performance control (via a network on nine computers for the composition *Drum Circle*); its use as a generative environment for the control of *MahaDeviBot* has been discussed here. This "collaboration" has been used in performance, in which the first author controlled *Kinetic Engine*'s conductor agent via a Lemur control surface, and the second author performed on ESitar [11]. In this case, the experience was very much like working with an improvising ensemble, in that high-level control was possible (density/volume/instrument choice), but low-level control (specific pattern choice or individual agent control) was not possible. At the same time, the intricacy of musical interaction created by the intelligent agents resulted in the perception of the robot being a complex organism, capable of intelligent musical phrasing and creation, rather than a simple tool to play back pre-programmed rhythms; combined, they provided a genuinely new and powerful interface for musical expression.

10. ACKNOWLEDGMENTS

We would like to thank Trimpin and Eric Singer for their support in building the MahaDevibot.

11. REFERENCES

[1] Beyls, P. Interaction and Self-Organization in a Society of Musical Agents. *Proceedings of ECAL 2007 Workshop on Music and Artificial Life (MusicAL 2007)* (Lisbon, Portugal, 2007).

[2] Brown, A. Exploring Rhythmic Automata. *Applications On Evolutionary Computing, Vol. 3449* (2005), 551-556.

[3] Burtner, M. Perturbation Techniques for Multi-Agent and Multi-Performer Interactive Musical Interfaces. *Proceedings of the New Interfaces for Musical Expression Conference (NIME 2006)* (Paris, France, June 4-8, 2006).

[4] Dahlstedt, P., McBurney, P. Musical agents. *Leonardo, 39, 5* (2006), 469-470.

[5] Dixon, S. A lightweight multi-agent musical beat tracking system. *Pacific Rim International Conference on Artificial Intelligence* (2000), 778-788.

[6] Eigenfeldt, A. Kinetic Engine: Toward an Intelligent Improvising Instrument. *Proceedings of the 2006 Sound and Music Computing Conference (SMC 2006)* (Marseille, France, May 18-20, 2006).

[7] Eigenfeldt, A. *Drum Circle:* Intelligent Agents in Max/MSP. *Proceedings of the 2007 International Computer Music Conference (ICMC 2007)* (Copenhagen, Denmark, August 27-31, 2007)

[8] Eigenfeldt, A. Multi-agent Modeling of Complex Rhythmic Interactions in Real-time Performance, *Sounds of Artificial Life: Breeding Music with Digital Biology*, Eduardo Miranda, ed., A-R Editions (forthcoming in 2008).

[9] Gimenes, M., Miranda, E. R. and Johnson, C. A Memetic Approach to the Evolution of Rhythms in a Society of Software Agents. *Proceedings of the 10th Brazilian Symposium on Computer Music* (Belo Horizonte, Brazil 2005).

[10] Goto, M., Muraoka, Y. Beat Tracking based on Multiple-agent Architecture - A Real-time Beat Tracking System for Audio Signals. *Proceedings of The Second International Conference on Multiagent Systems,* (1996), 103-110.

[11] Kapur, A., Davidson, P., Cook, P.R., Driessen, P.F., and W. A. Schloss. Evolution of Sensor-Based ETabla, EDholak, and ESitar. *Journal of ITC Sangeet Research Academy, Vol. 18* (Kolkata, India, 2004).

[12] Kapur, A, Singer, E., Benning, M., Tzanetakis, G., Trimpin Integrating HyperInstruments, Musical Robots & Machine Musicianship for North Indian Classical Music. *Proceedings of the 2007 Conference on New Interfaces for Musical Expression (NIME 2007)* (New York, New York, June 6-10, 2007).

[13] Martins, J., Miranda, E.R. A Connectionist Architecture for the Evolution of Rhythms. *Lecture Notes In Computer Science, Vol. 3907*, (2006). Springer, Berlin, 696-706.

[14] Martins, J. and Miranda, E. R. Emergent rhythmic phrases in an A-Life environment. *Proceedings of ECAL 2007 Workshop on Music and Artificial Life (MusicAL 2007)* (Lisbon, Portugal, September 10-14, 2007).

[15] Minsky, M. *The Society of Mind.* Simon & Schuster, Inc (1986).

[16] Miranda, E.R. Evolutionary music: breaking new ground. *Composing Music with Computers*. Focal Press (2001).

[17] Murray-Rust, D. and Smaill, A.: The AgentBox. http://www.mo-seph.com/main/academic/agentbox

[18] Murray-Rust, D., Smaill, A. MAMA: An architecture for interactive musical agents. *Frontiers in Artificial Intelligence and Applications, Vol. 141* (2006).

[19] Pachet, F. Rhythms as emerging structures. *Proceedings of the 2000 International Computer Music Conference ICMC 2000)* (Berlin, Germany, August 27-September 1, 2000).

[20] Pachet, F. The Continuator: Musical Interaction With Style. *Journal of New Music Research, 32, 3,* (2003) 333-341.

[21] Spicer, M. AALIVENET: An agent based distributed interactive composition environment. *Proceedings of the International Computer Music Conference (ICMC 2004)* (Miami, Florida, November 1-6, 2004).

[22] Woolridge, M., Jennings, N. R. Intelligent agents: theory and practice. *Knowledge Engineering Review, 10, 2* (1995) 115-152.

[23] Wulfhorst, R.D., Flores, L.V., Flores, L.N., Alvares, L.O., Vicari, R.M. A multiagent approach for musical interactive systems. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems.* ACM Press, New York, NY, 2003, 584-591.