SoloJam Island : a platform for rhythmic experimentation in spatially segmented multi-agent systems

Pierre Potel[1], **Frank Veenstra**[2], and **Kyrre Glette**[2,3]

[1]**ENSTA Paris**, France
[2]*Department of Informatics*, **University of Oslo**, Norway
[3]*RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion*, **University of Oslo**, Norway

ABSTRACT

The development of musical robots which create musical performances or assist musicians during their performances has been an active subject of research in the last few years. In this paper we consider a multi-agent system for the improvisation of rhythmic patterns, where agents compete to take the lead and play musical solos. This competing mechanism is based on evolutionary algorithms and auctions. As such a natural evolution of this system, is to investigate the ways in which separating these agents into multiple subpopulations, where some agents occasionally migrate from one subpopulation to the other and try to bring with them rhythmic knowledge of their past experience, affects the musical dynamics of these subgroups. Simulation's results show that migrating agent's can disrupt the musical play in those subgroups and ensure diversity in the musical compositions.

1. INTRODUCTION

Generating interesting and creative rhythms has been an active subject of research in computer music [1–3]/.

Vuust and Witek define a rhythm as "a pattern of discrete durations and is largely thought to depend on the underlying perceptual mechanisms of grouping" [4]. Therefore a rhythm makes sense relative to a musical context and depends on the previous musical patterns that have been played to fit into the musical composition. However, a rhythm can also become stale if it becomes too repetitive and bores out the listener. This duality is a complex task undertaken by the musician performing an improvisation who *"proceeds by attempting to continue an antecedent musical situation in such a way that the piece fulfills the latent expectations implied by the beginning while traversing a musical obstacle course that delays gratification and creates tension"* according to Tirro [5].

Musical systems which perform in a band-like manner have been explored in the past [6–10]. They allow for users without former musical training to perform and create interesting sounding musical patterns and become an active participant of the musical composition, through human-robot interaction or robot-robot interaction.

The SoloJam system presented in this paper is built upon is one of these systems [7]. What is interesting in our case is the ability for those systems to create novel musical patterns out of a set of simple predefined rules. This system conducts a series of auctions where agents bid with proposed rhythmic patterns to take the responsibility of performing the next musical solo. Each agent that did not win this auction then applies a mutation to their rhythmic pattern to try to perform better at the next auction.

This idea of mutating musical patterns to better suit a utility evaluation is inspired by evolutionary algorithms which are a class of algorithms particularly suited to provide creative solutions to problems [11]. Subdividing populations of agents into islands [12] or teams [13], where agents only occasionally migrate between those subgroups has been an effective method to preserve diversity of populations and presents high performing solutions to problems. In this paper, the spatial clustering of musical agents into "bands" where agents migrate between when they are "bored" is investigated to try to provide diverse solutions.

Indeed as Tirro stated : "The jazz improviser reuses and reworks material from previous performances; and, [...] musical ideas evolve through the passage of time and during subsequent performances" [5]. This way migrating agents could try to bring the knowledge from the musical patterns of the "band" they were previously in, and try to apply it to their new environment. Therefore the system should be designed to promote remembering knowledge from previous environments, for example, through the design of the utility function. Encoder based approaches have also been proven effective at not forgetting knowledge from previous tasks to allow for lifelong learning [14].

The contributions of this paper are listed as follows. In this study, we start by discussing the ways intelligent agents can generate rhythmic patterns relevant to the musical situation they are in. Furthermore in a second part we aim at understanding the dynamics of inter-subgroups migrations of musical agents and how they affect the ways agents play in a given group after the arrival of a migrating agent, compared to the case of isolated subgroups.

To meet these goals first a simple Unity simulator will be designed and then its abilities will be extended to allow for the migration of musical agents

2. THE SOLOJAM MUSIC SYSTEM

2.1 Presentation

The SoloJam algorithm [7] passes the responsibility to play a musical solo among agents in a group. At the beginning of the musical performance, each agent's musical pattern is initialised randomly and one leader is selected in the group. It will play its musical solo until it is not a leader anymore.

A musical pattern is defined by a binary array deciding if the agent should play a note at a specific beat, 0 being silent and 1 playing a note[1].

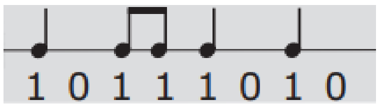

Figure 1. An example of a musical pattern.

After the musical pattern is defined each agent's utility is assessed, which means it's current musical pattern is compared to the leader's following Equation 1 :

$$u_i = \frac{c}{(1 + aD_l)(1 + bT_l)} \tag{1}$$

u_i being the utility of agent i, D_l representing the Hamming distance between the agent's musical pattern and the leader's, and T_l being the amount of time the agent has been playing solo. a and b are weights associated to these measures and c is a normalization constant.

Two notable exceptions are (1) if an agent which isn't the leader is considered too close to the leader's musical pattern —and its Hamming distance is under a certain threshold ϵ —then its utility becomes 0 to promote creativity, and (2) if the agent has just played a solo it gets a utility of 0 for one musical pattern to recover.

Therefore, agents try to adapt to the leader's musical pattern to maximise their utility function, while trying to remain just different enough so they don't get a utility of 0. This adaptation is performed by mutating the agent's musical pattern just before assessing its utility, i.e. flipping the value of each bit of the musical pattern with a probability of 1/λ; λ being the length of the musical pattern.

Furthermore, genetic crossovers between the adapting agent's pattern and the leader's are performed. A genetic crossover is the action of combining information between two parents (here the agent's and the leader's musical patterns) to create a new solution as shown in Figure 2. Our crossover implementation only decreases the Hamming distance between musical patterns, because it directly copies part of the leader's musical pattern. Therefore a two-points crossover with a probability p_c of happening, is added before the mutation to try to improve each agent's utility.

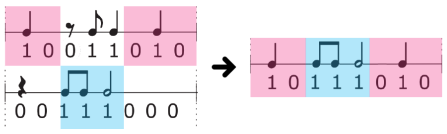

Figure 2. An example of a 2 points crossover between 2 musical patterns.

Because the leader's utility only decreases over time with the T_l parameter, another agent will eventually take over, and the responsibility of playing the solo freely circulates among the group illustrated in Figure 3.

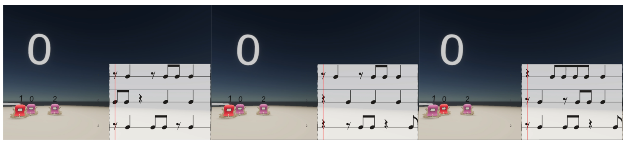

Figure 3. Agents adapting their musical pattern until eventually one takes over the group and becomes the new leader (highlighted in red).

2.2 Example dynamics

An experiment is conducted with the following parameters : $a = 1$; $b = 0.05$; $c = 2$; $\epsilon = 0.1$; $p_c = 0.6$ musical patterns of length 8 and 3 agents playing. The circulation of the responsibility to play musical solos is validated by experimental results.

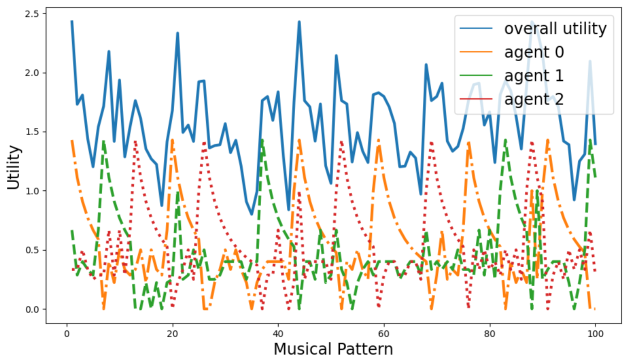

Figure 4. Agents' utilities during a baseline SoloJam experiment.

Indeed, it can be seen that the utilities of the agents regularly spike in Fig. 4, meaning that this agent has become the leader of the musical performance. Just after an agent has passed the leadership it's utility goes back to 0 for one musical pattern and then goes back to a normal value.

3. THE SOLOJAM ISLAND EXTENSION

3.1 Description

Subpopulations in island models have been used to help maintain diversity [12] which helps maintain diversity of the solutions. We therefore introduce the SoloJam Island

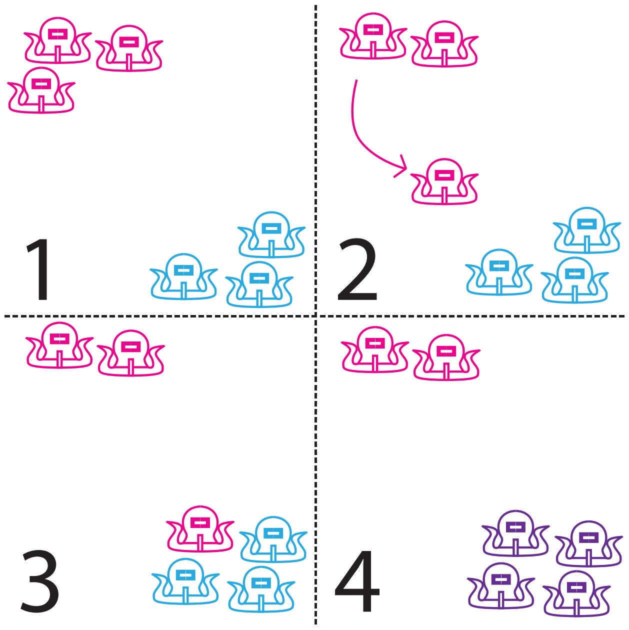

Figure 5. The principle of the SoloJam Island framework.

framework illustrated in Fig. 5, which operates as follows:

At first, the situation can be interpreted as though multiple instances of the SoloJam algorithm detailed in 2 work in parallel (1). Among those groups agents propose performing solutions close to their leader's musical pattern to take the lead and play their musical pattern. However, sometimes when an agent is free, i.e. not going to be designated as the next leader, it can decide to migrate to another island, another subgroup where agents are playing (2). Upon arriving at its new island the agent then follows the auction process of the group and tries to take the leadership in its new environment (3). Because the agent comes from a different island, it brings with it the knowledge from its previous environment in the form of its musical pattern. This pattern then influences the rest of the group if the agent plays a solo because the other members will try to make their patterns resemble the musical pattern of this migrating agent. With this process, the migrating agent has changed the search trajectory of the group which brings diversity to the musical patterns of the island (4).

3.2 Algorithm implementation

A new parameter called p_m is added, the probability for an island that an agent of this island starts migrating if it is free. This probability is tested during each auction taking place on an island. If the test is successful, an agent of this island is then uniformly randomly designated to leave the island and go to another random island, which it does if it is free and there are more than 2 agents on this island. This is done to avoid the scenario where an agent is stranded on an island and there's no more circulation of the musical solo.

To ensure the agent navigates correctly to its destination, it knows at any time the position of each island, its own position and the area the agent can walk on, using a Custom Navigation Mesh[1] component so that the moving agent goes around non moving agents and tries to reach the island even if the path is obstructed by other agents.

Finally a sensing range is added, which is the distance at which an agent can perceive another agent and know information about it such as the agent's state or its utility.

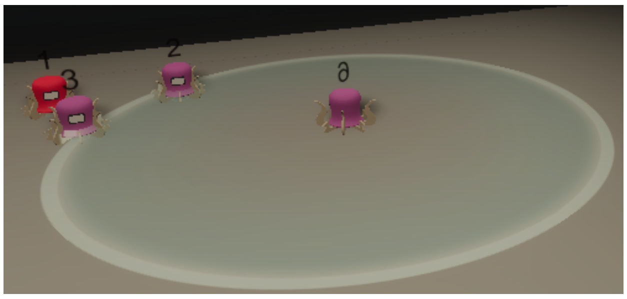

Figure 6. The sensing range of an agent.

For example, in Fig. 6, the migrating agent has access to the information of the two closest agents but can't access the information of the furthest one yet.

This framework is designed to limit the amount of information exchanged between agents. Therefore adapting agents only need to know the musical pattern of their current leader and the state of other agents in their sensing range i.e. if they are migrating or if they are the leader, and the leader only needs to know the utilities of agents in range as well as their state.

4. RHYTHMIC DYNAMICS ON THE ISLANDS

4.1 New utility function and crossover probability

Migrating agents have a special role in these experiments compared to static agents because they bring with them the knowledge from their previous environment.Therefore a new utility function needs to be designed to reward closeness to the leader's pattern but also closeness with its previous leader's musical pattern, with the importance of the latter decreasing over time.This function is defined as follows:

$$\begin{cases} u_{mig} = \frac{1}{2A_p}[(A_p - T_n)\cdot f(MP_f) + (A_p + T_n)\cdot f(MP_c)], \\ T_n < A_p \\ u_{mig} = f(MP_c),\ T_n \geq A_p. \end{cases} \tag{2}$$

- f is the utility function defined in Equation 1.
- MP_f represents a snapshot of the leader's musical pattern when the migrating agent left its former island and MP_c is the musical pattern of the leader of the island the agent currently is on.

[1] A navigation mesh is a 2D surface in the 3D space composed of several polygons indicating to a navigating agent the unobstructed areas in the space it evolves in that it can navigate on. Regular pathfinding algorithms such as A* are then applied on this mesh so that the agent can navigate to its destination.

- T_n counts how many musical patterns ago the agent joined its new island.
- A_p is the adaptation period representing how many musical patterns for the agent should take into account the musical pattern of the island he used to be on.

This new utility function rewards the migrating agent for creating a pattern that is close to its new island leader's, but that is also close to the last leader's pattern it has seen on its former island. This function starts by giving an equal importance to the two patterns, and the longer the agent stays on it's new island the lesser the contribution of its former island pattern weighs in, meaning it is required to adapt to its new environment. Eventually, after the adaptation period has passed or the agent becomes the leader, it is evaluated just like the other static agents following the utility function defined in section 2.

Furthermore, migrating agents can now perform crossovers with the snapshot of their former island's leader's pattern during their adaptation period. To make sure the agent still adapts to what's played in its new island, this probability to perform a crossover is now adjusted by the amount time the agent has spent in its new island following this formula:

$$\begin{cases} P(T_n) = p_c \cdot \frac{A_p+T_n}{2A_p} \ for \ MP_c \\ P(T_n) = p_c \cdot \frac{A_p-T_n}{2A_p} \ for \ MP_f \end{cases} \tag{3}$$

This probability ensures a migrating agent is as likely to perform a crossover with MP_f than with MP_c on its arrival to its new island. After that, the longer the agent stays on this island, the higher the probability to mix its musical pattern with MP_c while the probability to copy musical material from MP_f decreases. At the end of this adaptation period this crossover probability returns to its usual value of a probability of p_c to cross over MP_c and a probability of 0 to cross over MP_f.

4.2 Simulation parameters and metrics of interest

Simulations are run with the same parameters as 2.2, 3 islands with 3 agents in the beginning, a probability of migration p_m=0.05. For each simulation 150 runs of a duration of 100 musical patterns are simulated to average results.

Two cases are going to be investigated, one with A_p=20 which means migrating agents totally adapt to their new island 20 musical patterns after arrival and another with A_p=100 meaning agents will never fully adapt to their new environment and always retain some bias towards using musical material from their previous environment.

Three metrics are taken into consideration. Firstly, the Hamming distances between islands to see if different islands end up converging to the same zones of the search space because of the migration of agents. However, these two simulations show musical patterns of the different islands' leaders stay uncorrelated, so this metric is not investigated further.

Secondly, the distance between MP_f and the musical pattern of the leader of the new island the migrating agent arrived in. This metric shows if the migrating agent effectively manages to use musical material from its former island to influence the musical composition in its new environment.

Finally, the autocorrelation of MP_L, the list containing an island's leader's pattern after the arrival of a migrating agent, is explored. This autocorrelation is defined in these terms :

$$R(\tau) = mean(\sum_{MP_i, MP_{i+\tau} \ in \ MP_L} \frac{L - 2H(MP_i, MP_{i+\tau})}{L}) \tag{4}$$

- MP_i is the musical pattern at index i, $MP_{i+\tau}$ the musical pattern at index i+τ i.e the musical pattern played τ patterns after MP_i, provided it exists.
- L is the length of the musical patterns.
- H is the Hamming distance function.

This autocorrelation function yields a value of 1 if the patterns are the same, a value of 0 if they are uncorrelated and a value of -1 if they are inversely correlated. Therefore its variations give information about the evolution of leaders' musical patterns over time. If a migrating agent influences the evolution of an island's musical patterns, the autocorrelation of the island's musical pattern after the arrival of a migrating agent should evolve differently than in the case without migrating agents.

4.3 Experimental results

The two simulations yield the following results :

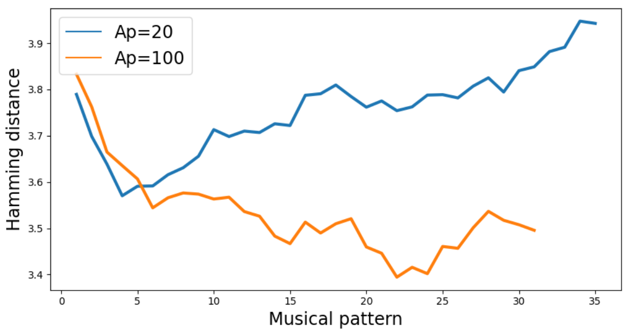

Figure 7. The Hamming distance between migrating agents' patterns and MP_f (A_p=20 in blue, A_p=100 in orange)

Fig. 7 shows that, when A_p=20, while the trend during the first musical patterns points to a decreasing distance between the snapshot of the former island, MP_f and the pattern that is being played by the migrating agent's new island's leader MP_c , the two patterns stay largely uncorrelated. Indeed the patterns here are binary arrays of length 8. Therefore for each bit there is a ½ chance that agents share the same bit which entails that uncorrelated patterns should share 4 bits and have 4 inverted bits leading to a Hamming distance of 4. A Hamming distance of 8 would mean that agents' patterns are inversely correlated.

However, when A_p=100, despite the fact that the Hamming distance between MP_c and MP_f remains important, extending the adaptation period has proven effective at decreasing this value during this period. This means migrating agents had been able to share the musical knowledge from their former islands more successfully.

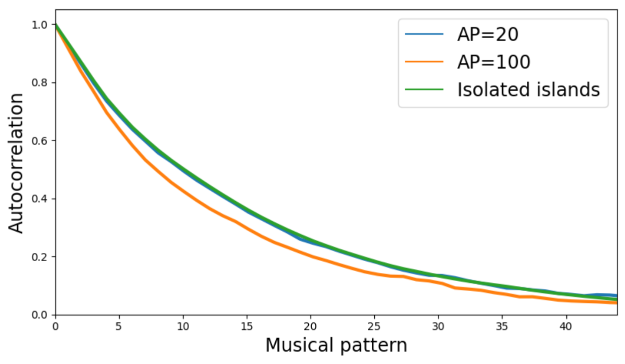

Figure 8. : The mean autocorrelation of an island's leader's patterns after the arrival of a migrating agent (blue A_p=20, orange A_p=100) or in the case of isolated islands (green).

Fig. 8 shows that the migration of agents when A_p=100 has influence over the evolution of an island's leader's musical patterns because the autocorrelation of an island after the arrival of a migrating agent doesn't follow the same trajectory as the autocorrelation of isolated islands. The autocorrelation values being lower in the case of the migrating agent indicates this migrating agent brings some disruptions to the group. Indeed, the musical patterns of the subsequent leaders tend to be less correlated with each other compared to the case where agents don't migrate between islands.

However, one may argue that in the case where A_p=20 the agent doesn't have enough time to adapt to its new environment which prevents the knowledge from its previous island to be shared, underlined by the fact the graphs representing the autocorrelation of isolated islands and when A_p=20 are merged.

These results indicate that the longer the adaptation period the higher the level of disruption the migrating agent entails. This is to be expected because the agent has less incentive to get close to what's played on the island during the first patterns after its migration.

5. CONCLUSIONS

In this work we designed a Unity simulator [2] which allows for rhythmic experimentation in multi-agent systems using spatial clustering of agents. This platform helped underline and explain the ways in which migrating agents can affect the rhythmic scenarios of the subgroups they arrive in.

SoloJam Island is thought as a versatile and easily extensible simulator. Future works could investigate methods to bring more important changes after the arrival of a migrating agent and tune more finely the weight of the agents' prior knowledge relative to the state of the musical composition, including but not limited to :

- Reinforcement learning approaches using autoencoders [14] and other deep neural networks' based solutions.
- Including the whole history of the migrating agent's musical performances and not only from the last seen environment. In that case, it would be especially interesting to investigate how changes in the number of islands affect the rhythmic dynamics of the subgroups.
- Developing more complex rhythm representation schemes which could take into account syncopation or the tone and the amplitude of the musical note, for example through the use of dynamic time warping.
- Implementing this framework on real robotic platforms to investigate the impact of message passing and communication delays in this decentralized system.

5.1 Acknowledgments

This work was conducted as a M.Sc. student project and was partially supported by the Research Council of Norway through its Centres of Excellence scheme, project number 262762.

6. REFERENCES

[1] A. Milne, S. Herff, D. W. Bulger, W. Sethares, and R. Dean, "Xronomorph: Algorithmic generation of perfectly balanced and well-formed rhythms," in *NIME 2016*, 2016.

[2] M. Mcvicar, S. Fukayama, and M. Goto, "Autorhythmguitar: Computer-aided composition for rhythm guitar in the tab space," in *Proceedings ICMC|SMC|2014*, 2014.

[3] G. Weinberg, B. Blosser, T. Mallikarjuna, and A. Raman, "The creation of a multi-human, multi-robot interactive jam session," in *NIME 2009*, 2009.

[4] P. Vuust and M. Witek, "Rhythmic complexity and predictive coding: A novel approach to modeling rhythm and meter perception in music," in *Frontiers in Psychology*, 2014.

[5] F. Tirro, "Constructive elements in jazz improvisation," in *Journal of the American Musicological Society*, 1974, pp. 285–305.

[6] K. Jennings and M. Witek, "Toy Symphony: An international music technology project for children," in *Music Education International*, 2003, pp. 3–21.

[2] Available at `https://github.com/67K-You/SoloJam-Island`

[7] A. Chandra, K. Nymoen, A. Voldsund, A. R. Jensenius, K. Glette, and J. Torresen, “Enabling participants to play rhythmic solos within a group via auctions,” in *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. Queen Mary University of London, 2012, pp. 674–689.

[8] 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*, 2017.

[9] K. Nymoen, A. Chandra, K. Glette, and J. Torresen, “Decentralized harmonic synchronization in mobile music systems,” in *IEEE 6th International Conference on Awareness Science and Technology (ICAST)*, 2014.

[10] K. Tatar and P. Pasquier, “Musical agents: A typology and state of the art towards musical metacreation,” in *Journal of New Music Research*, 2019.

[11] K. Miikkulainen, “Creative AI through evolutionary computation,” in *Evolution in Action: Past, Present and Future*, 2020.

[12] D. Whitley, S. Rana, and R. Heckendorn, “The island model genetic algorithm : On separability, population size and convergence,” in *Journal of Computing and Information Technology*, 1999, pp. 33–47.

[13] P. Lichocki, S. Wischmann, L. Keller, and D. Floreano, “Evolving team compositions by agent swapping,” in *IEEE Transactions on Evolutionary Computation, vol. 17, no. 2*, 2013, pp. 282–298.

[14] R. Triki, R. Aljundi, M. Blaschko, and T. Tuytelaars, “Encoder based lifelong learning,” in *2017 IEEE International Conference on Computer Vision (ICCV), vol. 1*, 2017, pp. 1329–1337.