Improvised Music with Swarms

T. M. Blackwell and P. Bentley
Department of Computer Science
University College London
Gower St.
London, UK
tim@theHotandtheCool.co.uk, P.Bentley@cs.ucl.ac.uk

Abstract: This paper describes SWARMUSIC, an interactive music improviser. A particle swarm algorithm is used to generate musical material by a mapping of particle positions onto events in MIDI space. Interaction with an external musical source arises through the attraction of the particle swarm to a target. SWARMUSIC is the first application of swarm intelligence to music.

I INTRODUCTION

The field of computer music is not a new one, but there are still very few systems that can play music in an improvised style, or indeed interactively. Most effort has been spent on systems that only interact weakly with humans (for example, accompaniment programs that adjust pre-prepared material according to external parameters), or music composition programs that embody some encoding of musical knowledge (a grammar, for example) but do not produce music in real time. But the chief characteristic of an improvisation is that it happens in real time and without revision. In fact interaction and improvisation have a symbiotic relationship, and this is to be heard most clearly in 'free improvisation'.

Freely improvised music (which differs from form-based improvisation as exemplified by jazz), is so named because it is free from many conventional musical constraints. Improvised music is not structured through tonality, form and metre, but relies on:

1. a distinctive musical language for each improviser, and
2. an ability to couple this with the other contributions arising from the ensemble.

It is proposed here that this coupling is primarily *expressive* - performers pay attention to the change in a musical landscape. This suggests that artificial improvisers would not need to have large expert systems hardwired with musical knowledge. Instead, through interaction, the computer could attend to the expressive attributes of the music. This would arise through a monitoring of the primitive musical parameters of pitch, loudness and pulse.

In order to find a strong and individual musical language for an artificial improviser, it is worth looking at some other non-musical system that could be *interpreted*.

Certain properties of swarms and flocks are very suggestive. A swarm (or a flock) has a persistent shape, and it may suddenly change its direction of flight, with every member almost spontaneously responding. Birds in a flock avoid collisions yet remain attracted to the flock centre. Swarms have similar behaviour but without the velocity matching of near neighbours that is so characteristic of flocks. For example, insects may also be attracted towards a distant target, which they swarm around on arrival. In general, swarms and flocks exhibit self-organisation.

In music, melody moves around a key centre, and has a 'shape' – the pattern of rising and falling tones. Large intervals between notes are rare, and variations can be visualised as gradual changes in shape. Notes tend to 'avoid' one another, but not by too much. Human rhythms are a pattern of accents and placements about a beat (i.e., fixed mathematical pulse). In other words, rhythm is attracted to the beat. Chords are groups of notes with an avoidance of small intervals and an attraction to consonant intervals. Good harmony follows the rules of voice leading – notes make the minimum movement necessary for the harmonic progression. Chord progressions are 'attracted' to a key centre, but they may actually avoid the tonic for some time, delaying resolution.

This analogy suggests that music, and improvised music in particular, is a self-organising system: local interactions lead to structure, even though there is no central organisation (as there is in a composition). The relevant local interactions that lead to self-organisation are attraction and repulsion [12].

In fact the ideas of attraction and avoidance do extend into music. Musicians, as they interpret a piece of music, will constantly vary expressive parameters such rhythm and dynamics. However these variations (often minute) are within limits set by stylistic constraints. For example, adjacent notes will be played with similar, but not identical, intensities. There is attraction towards a stylistic norm, but an avoidance of the banal.

Structure in improvised music derives from spontaneous (i.e., unplanned) changes in musical direction. This can only occur in a flock-like way. Each musician is faced with a constant dilemma: new expressive initiatives may be followed, or they may be ignored. The tension between expressive attraction and avoidance leads to the sudden, apparently orchestrated, changes that characterise this music.

The idea underlying SWARMUSIC is that a musical interpretation of a particle swarm (implementing both attraction and repulsion) will correspond to a distinctive musical language. Furthermore, a coupling to an external musical source can be made by mapping that source to an attractive target.

II BACKGROUND

SWARMUSIC is a music improviser. But what is improvised music? Improvised music can be typified as a deliberate avoidance of the usual musical constraints, but a more positive heuristic is to think of musicians *collaborating*, but doing so without a prepared structure (i.e., 'free form') and without rehearsal. It is conjectured here that this collaboration, if it succeeds, does so because of the sensitivity of the musicians to the expressive dimension of the music. This is an important concept, and is best explained by considering levels of musical organisation.

Formally, music is a hierarchical organisation of sound. Each sound has pitch, loudness and timbre. At the most primitive, sounds are organised according to pulse or beat. At the next level, small groupings of notes are organised into melody, rhythm (i.e., the temporal placing of notes within a pulse), harmony, and texture. Finally, these are grouped into a form. However, this hierarchy says nothing about how the music is realised in performance. These formal elements need to be *expressed* by the performer. Interpretation arises from individual decisions about, for example, dynamics (changes in loudness within a group of notes), vibrato (changes in loudness and pitch of a single note), swing (particular rhythms found in jazz) and tempo (how many beats per minute). These expressive parameters are usually stabilised within an accepted style – a set of conventions that govern interpretation. In improvised music, the formal elements are almost inconsequential to the interpretation. Instead, interaction and expression have a very high priority.

As well as being an improviser, SWARMUSIC is an interactive musical source. Surprisingly, there are very few examples of computer programs that can be said to *interact* with a musician. One review of contemporary approaches to computer music suggests that the focus of most research is on composition, but there are four systems (Texture, IBVA, Vox Populi, and M) that are designed for real time music production [1]. Although these systems are interactive, the human-computer interface is a computer keyboard, a mouse, or a set of electrodes. In other words, the systems do not interface with a human musician actually playing an instrument. However two other interactive systems do allow for real time input from an instrument: John Biles's *GenJam* [2] and the experiments of the trombonist George Lewis [3]. GenJam is perhaps closest in spirit to SWARMUSIC. A genetic algorithm produces melodic phrases in response to a real time input. The improvisations are in a (mainstream) jazz style and depend on prior storage of the harmonic form.

Computer music, as a whole, is preoccupied with the composition of music, and the means to this end is logic. In other words, algorithms are used to *generate* music. The algorithms are based on a synthesis of the logical elements discussed above. Clearly the algorithms are developed with a certain outcome in mind, and as such they encode a certain amount of musical knowledge. The success of the system in generating music will depend critically on the constraints and procedures used in the algorithms, and this will depend on the composer's own intentions and musical knowledge.

Such rule-based or algorithmic approaches (which we term *A-type*) can lead to music lacking in expression. Miranda, too, is aware of these problems. He concludes his study ([1], p206) with "…computers are very good at complying with systemisations and rules, but they are useless at breaking them", and quotes the Brazilian composer Richter, "In music, rules are made to be broken. Good composers are those who manage to break them well".

It might be possible to move away from algorithmic generation and towards a machine that can musically interact on equal terms with a human collaborator – *I-types*. The idea behind an I-type approach is that it *interprets, interacts and improvises.* An I-type would interpret some system whose properties may have a musical analogue. Schoenberg defined music as "repetition and variation" (Wolf and Thomson in [2]). This is a very high level description, and misses many elements important to the production of music, but is very interesting for our purposes here. What happens if Schoenberg's observation is inverted: "Systems whose properties include repetition and variation have musical interpretations"? SWARMUSIC is an exploration of this idea.

III SYSTEM OVERVIEW

There are three important processes in SWARMUSIC: **capture** of external events and their placement as targets, the particle swarm **update algorithm** and the **interpretation** of particle positions as MIDI events.

The update algorithm in SWARMUSIC is based on the algorithm of Reynolds' flock simulations, but without velocity matching [3]. The algorithm therefore imitates a swarm rather than a flock. Each particle in the swarm feels an attractive force toward the swarm centre of mass and an inter-particle repulsion (avoidance). The avoidance is to discourage particles from occupying the same locality – which would be interpreted as the same set of musical ideas. The attraction to the swarm centre opposes the avoidance force and helps to establish coherency (i.e., form and shape). An examination of these forces is contained in an accompanying paper [11]. The general idea is that form will emerge from the balance of opposing tendencies [12].

Additionally, the algorithm incorporates attraction towards a target in a similar way to the early particle swarms [4]. However, the algorithm differs from the usual particle swarm optimisation (PSO) algorithm since there are no concepts of global and neighbourhood 'bests' [5]. This is because the musical swarm is not trying to optimise any function (which would be an encoding of musical knowledge). Instead, the target force is included to link the swarm to the external source, and to constrain the swarm to the target cube (see below).

The total acceleration a_i experienced by particle i (= 1,…N) at position x_i is shown in Box 1. Here, x_{centre} and x_{target} are the centres of mass of the swarm $\{x_i, v_i\}$, i = 1,…N and the target swarm $\{x^T_i\}$, i = 1,…M. The particles move in a space of dimension n. The three accelerations are parameterised by the constants $\{C_{avoid}, p, p_{min}, a_{core}, C_{centre}, C_{target}\}$.

The two attractive accelerations $a_{i\ centre}$ and $a_{i\ target}$ are linear spring forces. These two terms are similar to the accelerations in the PSO algorithms - replace x_{centre} and x_{target} by local and best positions. The avoidance acceleration $a_{i\ avoid}$ is zero for separations greater than p – this encourages the attractive accelerations. The particles experience an inverse square repulsion between a core radius p_{core} and a limit of perception p, and a constant 'core' acceleration at separations less than p_{core}. This core acceleration can be made equal to the acceleration at p by setting $a_{core} = C_{avoid} / p_{min}^2$, which ensures piecewise continuity at the core boundary. The particles can be made to experience a constant repulsion for all separations less than p by setting $p_{min} = p$. Note that $p_{min} > 0$ due to the singularity in the inverse square law.

$$
\begin{aligned}
a_i &= a_{i\ avoid} + a_{i\ centre} + a_{i\ target}, \\
a_{i\ avoid} &= 0, & r_{ij} &\geq p \\
a_{i\ avoid} &= \sum_{j \neq i} \frac{C_{avoid}}{r_{ij}^2}, & p_{core} &< r_{ij} < p \\
&= a_{core}, & r_{ij} &\leq p_{core} \\
\text{where } r_{ij} &= x_i - x_j, \\
a_{i\ centre} &= C_{centre}(x_{centre} - x_i) \\
a_{i\ target} &= C_{target}(x_{target} - x_i).
\end{aligned}
$$

BOX 1 PARTICLE ACCELERATION

The update parameters, UP, are constituted from the acceleration constants, a clamping velocity v_{max} and a target cube length x_{max}: UP $= \{C_{avoid}, p_{core}, p, a_{core}, C_{centre}, C_{target}, v_{max}, x_{max}\}$. The update algorithm is shown in Box 2. An unusual feature is that velocity clamping occurs after position update [11].

```
Choose dimensions {n, N, M}
Initialise swarm $\{x_i, v_i\}$
Place target swarm $\{x^T_i\}$ in cube $T = [0, x_{max}]^n$
Initialise UP
**Loop**
        **if** ( *interact* ) **capture** events
        update $\{x^T_i\}$
        Find $x_{centre}$ , $x_{target}$
        **for each particle**
                $v_i = v_i + a_i$
                $x_i = x_i + v_i$
                **if** ( $|v_i| > v_{max}$) $v_i = (v_{max} / |v_i|)\, v_i$
        **endfor**
        **if (** *play* ) **interpret** swarm
**until** stopping criterion is met
```

BOX 2 UPDATE ALGORITHM

External interaction is included in this model through the possibility of adjusting the swarm in response to streamed audio events and is invoked by setting *interact* to true. The **capture** algorithm parses an input audio stream and adds targets, up to a maximum number. Then, as new events are parsed, the targets are re-positioned according to a target update algorithm. This means that the dynamics of the target swarm are determined by the interaction of the external musician. The attraction between the target swarm and the particle swarm corresponds to the coupling between musicians discussed above. The internal dynamics of the swarm and the **interpretation** algorithm (which is invoked if *play* is true), encodes SWARMUSIC's own improvisational ideas.

IV INTERPRETATION AND CAPTURE

Swarm interpretation is directly analogous to the interpretation of a score. A simple interpretation must be found that is musical and sensitive to the swarm's time development. The inspiration for the mapping to *music space* came from the flock simulations of Reynolds [4]. The aim was to find a musical interpretation of a flock or swarm. Suppose that an observer is looking towards the coordinate origin and at the flock (Figure 1).

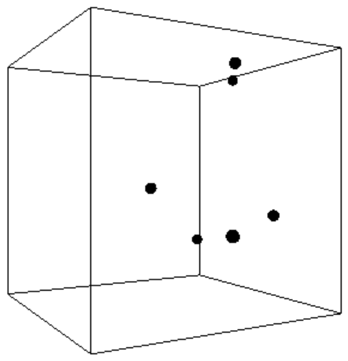

FIGURE 1 A VIEW OF THE SWARM

By replacing each 'boid' with a note, the swarm can be read as a score. In a score, the order of events is read from left to right, and the duration of each event (which may be a note or a rest) is represented by the type of symbol. Furthermore, notes are grouped into bars, where each bar corresponds to a unit of time determined by an underlying pulse in beats per minute (BPM). In other words, a score provides *relative* timing information. Finally, instructions about intensity are placed above or below small groupings of notes.

An obvious idea for reading the swarm is to allow particles closer to the viewer to represent louder notes, and 'higher' particles to represent higher-pitched notes.

The interpretation of time is not so evident. Suppose the time dimension of music space corresponds to the time *separation* Δt between the start of successive events. Then particles further to the observer's right correspond to notes of a longer duration. Each snapshot (or frame) of the swarm after a complete update then represents a succession of events played in some pre-determined order, and occupying a time interval equal to the sum of the individual Δt's of each event. Any underlying beat will arise as an emergent property of the swarm, depending on the spatial relationship between the particles, and not on the absolute position.

To summarise, the swarm moves in n-dimensional phase space, but only the position coordinates are used for a mapping to music space [7]. Music space is populated by musical events, each one of which corresponds to a note played at a certain time and with a definite loudness. The three axes of music space are therefore loudness, pulse and pitch. Figure 2 shows a single event in music space. The event corresponds to middle C (C3), played at loudness MIDI 68, and sounding at a time interval corresponding to a beat of 120BPM after the preceding event. MIDI note and loudness values range in integral steps from 0 to 127, but we may wish our MIDI events to lie within a smaller range. Pulse is also restricted to a range between some arbitrary minimum and maximum BPM. In other words, (music space) event coordinates $\{X_j\} = (X_{loudness}, X_{pulse}, X_{pitch})$ are constrained to lie in the interval $[X_{j\ min}, X_{j\ max}]$.

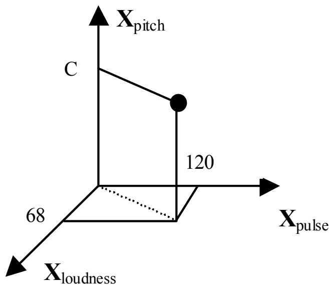

FIGURE 2 AN EVENT IN MUSIC SPACE

Formally, interpretation is a mapping I: $\{x_j\} \rightarrow \{X_j\}$ from the position space of the particles to the music space of events. There is, however a problem, since the particle swarm is unbounded in position space, but music space is bounded.

To solve the problem, if we stipulate that target coordinates $\mathbf{x}_{target}$ must lie within a **target cube** $\{0 \leq x_j \leq x_{max}\}$, then the particles will, given enough time, swarm inside this cube as long as there is attraction to the target. Positions in this cube can then be linearly mapped to music space. Particles that are temporarily outside the cube can be interpreted by projection onto the nearest boundary.

Notice also that the boundaries of music space are adjustable parameters which enable SWARMUSIC to be 'tuned' for a particular environment or context. This might be useful if notes below some MIDI value are inaudible, or if SWARMUSIC wants to contract its pitch values to the range of a particular instrument, for example double bass.

The mapping $x_1 \rightarrow X_{pulse}$ is actually nonlinear in event rate (i.e., linear in $1/\Delta t$) since this corresponds better to our perception of pulse, but is linear in pitch and loudness.

Directly after interpretation to MIDI, there is much control over how these events are actually sent to the synthesizer. For example, events could be queued and played in block as a chord, or they could be queued and played in order of pitch, or notes can be played staccato… there are many possibilities. These are matters of style; it may be a requirement that SWARMUSIC improvises within the broad constraints of a musical style. The style can be quantified into a number of style parameters [7].

The capture algorithm searches for musical events in the audio stream and parameterises them into start time, end time, loudness and pitch. (The fundamental pitch is extracted through a Fast Fourier Transform. There is scope here for future enlargement of music space to include timbral parameters, and to allow swarming in these dimensions too.) A target is then placed (or moved) in music space with coordinates determined by these extracted parameters, which amounts to inverse interpretation. The exact placement of the target and adjustment of style parameters is determined by a *script* – an algorithm that organises and controls the complete process of capture, updating, and interpretation.

The interaction between human and machine is provided by the swarming motion around the target, giving the musical sensation of listening and responding, whilst the swarm's own uncertain dynamics provide novel musical ideas.

V MUSICAL RESULTS

It is very hard to quantify musical output, but in this section we will describe two notable recordings, one of which demonstrates SWARMUSIC's own improvisational language, and one which was a particularly successful duet with a singer [8, 9].

In both cases, a swarm of five particles was released in n = 3 dimensions from random starting positions and velocities, with a single target. The acceleration constants were determined as fractions of certain limit values, except for p_{core} which was set arbitrarily to 1.0. The target cube dimension x_{max} was set to 128.0. These choices for p_{core} and x_{max} were made for interpretative reasons [7]. The clamping velocity v_{max} was also chosen as a fraction x_{max}. The limits to the acceleration constants were determined by the requirement that position updates should be on a scale commensurate with x_{max}. The relationships are set out in tables 1 and 2.

TABLE 1 LIMITS TO ACCELERATION CONSTANTS

$C_{avoid\ lim}$	p_{lim}	$a_{core\ lim}$	$C_{attr\ lim}$	$v_{max\ lim}$
$2x_{max}\ p_{core}^{2}$	$n^{1/2}x_{\ max}$	$2x_{max}$	$2\ n^{-1/2}$	X_{max}

TABLE 2 VALUES FOR UPDATE PARAMETERS USED IN RECORDINGS

$\mathbf{C}_{target,}\ \mathbf{C}_{centre}$	$\mathbf{C}_{avoid}$	$\mathbf{p}_{core}$	**P**
$0.5C_{attrlim}$	$0.5C_{avoidlim}$	1	$0.5p_{lim}$
$\mathbf{a}_{core}$	$\mathbf{v}_{max}$	$\mathbf{x}_{max}$	
$0.25a_{corelim}$	$0.25v_{maxlim}$	128	

The graphs that follow show a plot of a single coordinate (interpreted as MIDI pitch value) for each particle as a function of time in update units (one unit = one iteration). Each dot in the graphs corresponds to a single particle and the continuous wavy line and continuous straight lines are plots of the pitch coordinate of the swarm centre and target centre respectively.

A. Solo Improvisation

This example was produced using a script which randomly resets a target in the target cube when the separation from the swarm centre to the target is less than 10 units. In addition, the style parameters were set by the script so that a three note chord was played with probability 0.25, and, with the same probability, the swarm was interpreted in ascending order of pitch. Recording 1 lasts for just over two minutes and a plot of pitch against time is given in Figure 3.

The most prominent feature of the recording is that the melody swings repeatedly from low-pitched notes to mid-range notes, and at a fast tempo. The melodic shape is clearly visible in Figure 3. The particles in the swarm stay close together, following the path of the swarm centre as it oscillates about the target. This produces the melodic shape. The period of oscillations is 5 – 7 seconds (the horizontal axis in Figure 7.5 is calibrated in *update* units – five events are played in real time for each update), which can be discerned in the recording as the time between very low-pitched events.

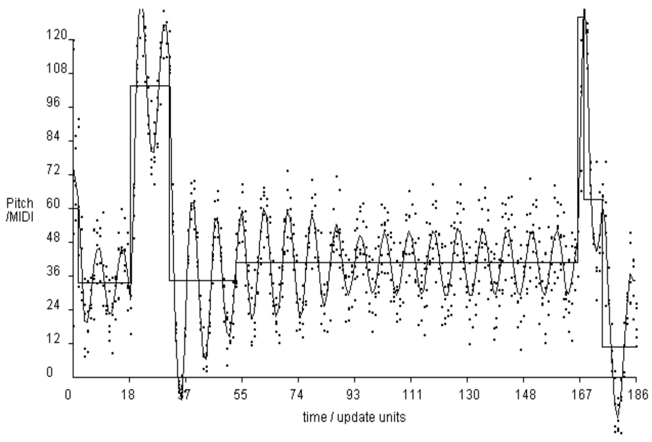

FIGURE 3 SOLO IMPROVISATION

Another feature of the plot which can be heard quite easily is the ascending series of notes occurring just after update 18, between 15 and 17s from the start of the track. This coincides with a target jump to a high pitch value. The very high notes, beyond MIDI 120, do not sound on the synthesizer, and gives rise to longer gaps between notes at the top of the two 'spikes' above the target .

At 25 seconds from the start, the target jumps back to a low pitch. The swarm follows this new musical direction very quickly, responding with a four note falling phrase ending with some rumbling low notes just one second later.

The piece also ends with an upwards jump to a target (right on the edge of the target cube) at iteration 167. There is a longer gap between notes as the swarm flies outside music space and becomes inaudible, before the target falls again and the swarm plays a rapid rising figure. This ends the recording.

Swarming can be heard in the other two dimensions too, but is not so prominent. An oscillation in loudness and rhythm is also occurring, but the amplitude is small. This makes the improvisation very *expressive*. The small variations in pulse impart swing, and the changes in loudness are very reminiscent of human performance. The occasional chord and ordering of notes into scalar groups also add to the sense that the music has been produced by an intelligent improviser.

B. Duet With Singer

This recording is a duet between a singer, Robin Higgins, and SWARMUSIC. The same style script from the solo improvisation is used, but this time the target is placed by the capture algorithm. The improvisation, which lasted for 2 minutes and 36 seconds, was recorded after five minutes of familiarisation between the singer and SWARMUSIC. A plot of pitch against update number is given in Figure 4.

The recording is noticeable for its evenness – there are no abrupt changes from either performer. Figure 4 shows that the target placement is between MIDI 52 and 74. The external events are mainly in the range MIDI 53 to 65. The SWARMUSIC is playing mostly quavers at 120 BPM, with occasional bars of crotchets, and there are small dynamic variations from both performers.

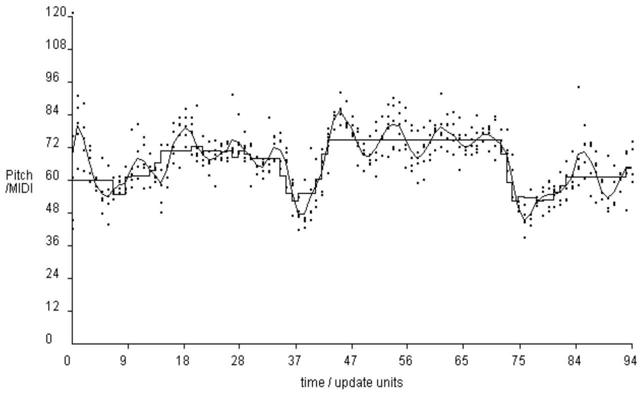

FIGURE 4 DUET WITH SINGER

The piece starts with SWARMUSIC playing a remarkable four bar melody in the key of Bb, which has been transcribed using music notation in Figure 5.

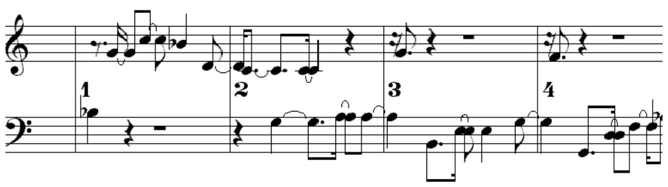

FIGURE 5 OPENING MELODY

Robin's first note is D, occurring just at the start of bar 5, some 7 seconds from the start. The response to this is an increase in pulse, pitch and loudness, but the sense of tonality is lost as the SWARMUSIC plays without a clear tonal centre.

The next phrase from the singer consists of three ascending notes, E, G and A. At this point, SWARMUSIC starts to play higher and noticeably quieter (updates 17–19 of Figure 4 and bars 13-14 of Figure 6). The tonality in bar 13 is C major or A minor, which agrees with the singer's harmony. SWARMUSIC continues with a sequence of chords (bars 15-16).

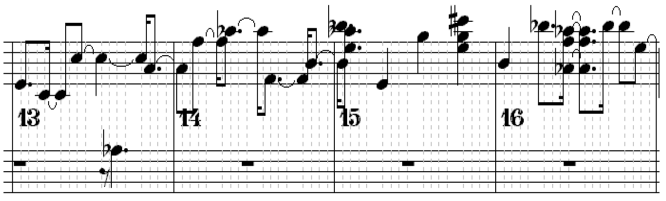

FIGURE 6 SWARMUSIC'S RESPONSE TO AN ASCENDING PHRASE

Overall, the improvisation sounds like an equal collaboration, and bears many features in common with improvisations amongst humans: although there are periods with a common purpose but there is also conflict, and this enhances the music. The singer was deliberately given very little time to acquaint himself with SWARMUSIC's characteristics. The singer commented: "The striking characteristic of the swarm is its sensitive responsiveness – like a good improvising partner, it picks up musical structures I gave it, and were I more attuned to its behaviour, might well give me in return a feedback which I could use more effectively" [10].

VI CONCLUSIONS

This work demonstrates that novel interactive and improvising musical systems can be developed from swarm and flock algorithms. Furthermore, such an approach has the best chance of success if the interaction is expressive.

The solo improvisation is exciting and resourceful – it is hard to believe that this is not of human origin. Organisation is apparent in the improvisation even though none was programmed: musical structure is an emergent property of the musical swarm. This recording demonstrates that SWARMUSIC has a rich musical language of its own.

The duet with a singer displays interaction, and the result is a coherent (and moving) piece of music. This goes some way to validating the conjectures that improvisation succeeds through expressive coupling. Certainly SWARMUSIC seems capable of free improvisation.

SWARMUSIC is the first use of swarm intelligence in a computer music application, and is interactive and improvisational. The key to SWARMUSIC's ability to interact lies in the attention to expression. This already distinguishes SWARMUSIC from other interactive musical systems. SWARMUSIC can be said to improvise since it composes in real time, and without revision. Furthermore, the swarming behaviour leads to musical patterns which give the improvisations a sense of coherency. SWARMUSIC seeks to interpret, rather than generate according to an encoding of musical knowledge. In this sense, it is some way towards an I-type approach. SWARMUSIC is also highly adaptable. The use of scripts means that more refined scripts and combinations of scripts can be developed.

SWARMUSIC has great potential as a solo improviser, as a collaborator, and can even be played as an instrument with real time adjustment of style, animation and script.

References

[1] Miranda E.R. "*Composing Music with Computers,"* Oxford: Focal Press, 2001

[2] Bentley P.J. and Corne D.W. "*Creative Evolutionary Systems*," San Diego: Academic Press, 2001

[3] Winkler, T "Composing Interactive Music," Cambridge, Mass: MIT Press, 2001

[4] Reynolds C.W. "Flocks, herds and schools: a distributed behavioural model," Computer Graphics, Vol 21, pp 25-34, 1987

[5] Kennedy J. and Eberhart, R.C. "Particle Swarm Optimisation," Proceedings of the IEEE International Conference on Neural Networks, IV, pp1942-1948, 1995

[6] Kennedy J. and Eberhart R.C. "Swarm Intelligence," Morgan Freeman, 2001

[7] Blackwell T. M. "*Making Music with Swarms*," MSc thesis, University College London, 2001

[8] Blackwell T.M. track1, "*Swarm Music*" ownlabel cd 001, 2001

[9] Blackwell T.M. and Higgins R. track3, "*Swarm Music*" ownlabel cd 001, 2001

[10] R.Higgins, private communication, 2001

[11] Blackwell T.M. and Bentley P.J. *"Don't Push Me! Collision-Avoiding Swarms*", submitted to CEC-2002

[12] Bonabeau E., Dorigo M. and Theraulax, G. "*Swarm Intelligence: From Natural to Artificial Systems"* Oxford University Press, 1999