Proceedings of the International Computer Music Conference (ICMC 2009). Montreal, Canada from 16–21 August 2009

STREAMING FROM MIDI USING CONSTRAINT SATISFACTION OPTIMIZATION AND SEQUENCE ALIGNMENT

Ben Duane
Northwestern University
Music Theory and Cognition
ben-duane@u.northwestern.edu

Bryan Pardo
Northwestern University
Electrical Engineering and Computer Science
pardo@northwestern.edu

ABSTRACT

We present a new system for separating streams in musical pieces encoded as MIDI files. Our approach is to: (1) divide the music under analysis into short segments, (2) analyze each segment using constraint satisfaction optimization, and (3) connect these analyses using a sequence alignment algorithm. Parameters for the system are learned automatically on a small training corpus and generalize reasonably well across a variety of pieces. We report performance results on 108 pieces of Baroque, Classical, and Romantic music: J.S. Bach's two-part inventions (0.95 accuracy by the *F*-measure), three-part sinfonias (0.92), and fugues from the *Well-Tempered Clavier*, book I (0.93) and book II (0.92); and string quartets by Haydn (0.81) and Brahms (0.76).

1. INTRODUCTION

Psychologist Albert Bregman describes an "auditory stream" as a grouping of distinct acoustic events (such as multiple footsteps heard in succession) into a single perceptual entity (the sound of a person walking) [1].

Music seems often to be heard in parallel auditory streams. The typical art song, for example, has a piano part and a vocal part. Rock songs are often recorded with each instrument and voice on a different track. Piano rags tend to have a melody, bass line, and block chords. In each case, the music could be heard as two or more concurrent streams, each containing an independent sequence of notes.

There is abundant research on stream separation. Some researchers have done empirical studies on streaming in human audition [1]. Others have modeled the findings of these studies computationally ([5], [8], [9], [14], [15]). Still others have engineered computer systems to separate streams in digital audio (see [16]). Our goal is different. We propose a computational system for finding streams in symbolically represented music—specifically, in MIDI files [2].

In the context of MIDI files, we define a stream as a sequence of non-simultaneous, non-overlapping notes that a listener would likely perceive as one entity. (This definition is elaborated and formalized in Section 3.1.) The exact nature of streams, even under this limited formulation, varies from one musical style to another. We thus limit ourselves to two cases: fugal music, in which streams are essentially contrapuntal voices, and string quartets, in which individual streams typically, but do not always, correspond with the note sequence produced by a single instrument.

One cannot depend on the creators of MIDI files to make the music's streams explicit. While Type 1 MIDI files may be organized into parallel tracks, the tracks do not reliably correspond to streams (a piano part with multiple streams, for instance, is often encoded as a single track). The same is true of MIDI channels, which are often shared by multiple streams (two violin parts might occupy single channel, for example). Yet the automatic retrieval of streams from MIDI files could be highly beneficial. One might wish, for example, to analyze individual streams in a MIDI corpus, or to mine MIDI files while making a database of melodies.

We first review existing work on computational streaming of symbolically represented music (section 2). We then explain our system (Section 3). In Section 4, we report results of experiments on our system's performance on pieces by Bach, Haydn, Beethoven and Brahms. Sections 5 and 6 contain conclusions and acknowledgements.

2. EXISTING WORK

A number of researchers have developed systems to find streams in symbolically represented music. One such system, the purpose of which was to model stream separation in music perception, treated streaming as motion tracking [9]. When notes were fed into the model, they produced activations that both spread to neighboring pitch receptors and took time to decay. As a result, when notes were close in pitch and time, their activations would overlap. This yielded "motion traces," which were interpreted as streams.

Two other systems were based entirely on hard-and-fast rules. The first used a list of rules, each of which had a Boolean condition and an action to take if the condition

was true [3]. The conditions assessed pairs of notes, and the actions were either to put the notes in the same stream or not. The second model gathered several features from pairs of notes (the pitch interval, for example) [4]. Using these features and a machine-learning algorithm, the authors found a logical predicate for deciding whether notes should be put in the same stream. These two models, however, were each fine-tuned to one musical piece, and, despite performing well on these pieces, were not proven to generalize beyond them. We test our model on numerous pieces.

Other systems employ only one or two rules, supplemented by some form of mathematical optimization. Such systems find analyses that both obey the rules and optimize a mathematical function. Two used a combination of well-formedness rules, expressed as constraints, and preference rules, embodied in a scoring function ([5], [6]). The first model [5], however, uses a dynamic programming algorithm, whereas the problem of streaming has not been proven to have the optimal sub-structure necessary for this algorithm to guarantee the best solution. And the second model [6] employs a short sliding window, testing every possible analysis for each frame. This approach is combinatorially explosive, which not only slows down the algorithm, but also limits the size of the analysis windows. Our model adds a constraint that reduces the complexity of the search.

One model minimized inter-note distance using a clustering algorithm with slight modifications [10]. Another found segments of the music in which only one solution could satisfy the model's constraints; it then used these segments as guideposts when analyzing the others [11]. The advantage of these systems is their speed, each of them building streams one pair of notes at a time. A window of just two notes, however, may lead to errors when a larger context must be considered. Our system is designed to let the user choose the desired balance between speed (by selecting a small window) and accuracy (by selecting a larger window).

Two other systems, unlike ours, found streams containing not just single notes at once, but also multi-note chords ([12], [13]). One of these used sequence alignment in a way analogous to our system's use of it [12]. Finding streams containing chords, however, is a fundamentally different problem than ours for two reasons. Notes must be grouped as chords—a task we do not face. And since streams can contain simultaneous notes, there are fewer of them. The two types of systems thus search different solution spaces—one with both sequential and simultaneous connections to choose from, another with more sequential possibilities but no simultaneous ones.

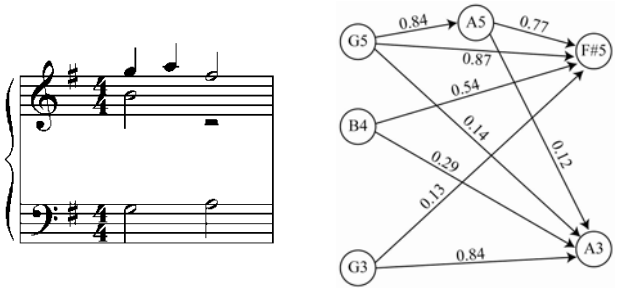
The testing corpora used for existing systems are usually limited to a handful of contrapuntal keyboard works by J. S. Bach. This leaves open the question of how well each approach generalizes to other composers and styles. We test our system on a corpus that includes works from the Baroque, Classical and Romantic periods.

3. A NEW SYSTEM

In this section, we first formalize both our definition of streams and the problem of streaming, modeling the music under analysis as a directed graph. We then explain our system, which operates in two basic steps. In the first step, the music is divided into short segments, and these segments are analyzed individually through constraint satisfaction optimization. In the second step, the resultant analyses are connected using sequence alignment.

3.1. Preliminaries

We model the music under analysis as a directed graph, which comprises a set, $\mathcal{V}$, of *vertices* and another set, E, of *edges* that connect pairs of vertices. In this paper, each vertex represents a note, and each edge represents a possible pairing of notes. Figure 1 shows an example. Given two vertices, u and v, the edge (u, v) is added if and only if u's note can precede v's note in a stream, according to constraints we define. Each edge is assigned a weight in the range [0,1], using a function described later. Higher edge weights indicate a greater likelihood of including both notes in the same stream.



a. The score. **b.** Graph representation.

Figure 1. Example of our graph representation of scores.

Our goal is to create a set of disjoint paths through the graph, with each path representing a stream. Figure 1 shows a graph representation of a score. Figure 2 shows an analysis containing three streams.

We define our problem as follows. Let $G = (\mathcal{V}, E)$ be a graph of the music, and let w: $E \rightarrow [0, 1]$ be the weighting function. We seek the sub-graph $G' = (\mathcal{V}, E' \subseteq E)$ such that:

1. For each vertex $v \in \mathcal{V}$, there exists at most one edge in E' that originates from v.
2. For each vertex $v \in \mathcal{V}$, there exists at most one edge in E' that terminates on v.
3. Any additional constraints we define are satisfied.
4. Equation 1 is maximized.

$$f(E') = \sum_{(u,v)\in E'} w(u,v) \quad (1)$$

Conditions 1 and 2 ensure that E' constitutes a set of disjoint paths, as in Figure 2.

In our model, each vertex v has the following fields associated with it:

- FREQ[v], the fundamental frequency of v's note in Hertz, which is extrapolated from MIDI pitch.
- ON[v], the onset time of v's note in seconds, translated from MIDI timing values.
- OFF[v], the offset time of v's note in seconds .
- PATH[v], the path to which v is assigned (an integer).

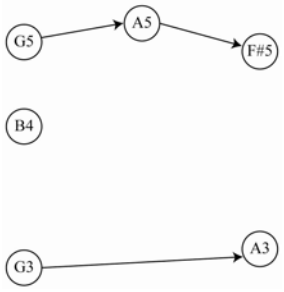


Figure 2. Sub-graph $G' = (\mathcal{V}, E' \subseteq E)$ representing the correct analysis of Figure 1.

Prior to analysis, the notes are sorted by onset, and all notes having sufficiently similar onset times are put in the same *onset group*. (The grey rounded rectangles in Figure 4, for example, enclose onset groups.) Formally, let $\mathcal{V}$ be the set of vertices modeling the music under analysis. Each $v \in \mathcal{V}$ is contained in one onset group, and this onset group contains all other vertices, $u \in \mathcal{V}$, such that $|\text{On}[u] - \text{On}[v]| < \theta_{\text{sim}}$, where θ_{sim} is a free parameter (see 3.2). Together, these onset groups form a partition of $\mathcal{V}$, such that $\cup_{\forall i} V_i = \mathcal{V}$, where V_i is the set of vertices in the i^{th} onset group. All MIDI files we used for testing, moreover, were such that $\cap_{\forall i} V_i = \emptyset$.

3.2. Constraints

We define constraints of two basic types. The first dictates which notes can be connected. The second prohibits streams from crossing.

We allow notes to connect if and only if they are: (1) non-simultaneous and (2) non-overlapping. If the MIDI data is not quantized, both conditions are problematic. MIDI files generated from human performance typically have variation in onset times between notes intended to sound simultaneously. Further, on keyboard instruments, perceptually non-overlapping notes are often performed with slight overlap.

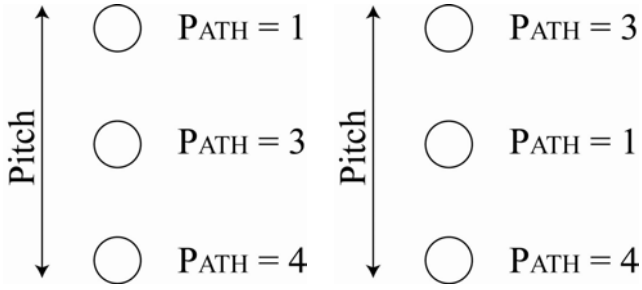
To solve the first problem, we again employ the threshold θ_{sim}, past which notes are considered simultaneous. To address the second problem we define the threshold θ_{over}, past which notes are considered to overlap. For all tests below, θ_{sim} = 10 ms and θ_{over} = 10 ms—values we found through trial and error.

Given these thresholds, our first constraint is defined as follows. Let $G = (\mathcal{V}, E)$ be the graph of the music, and let u and v be vertices from $\mathcal{V}$. The edge (u, v), representing the possibility of connecting these notes, is added to E if and only if:

1. $\text{ON}[v] - \text{ON}[u] > \theta_{\text{sim}}$
2. $\text{OFF}[u] - \text{ON}[v] < \theta_{\text{over}}$

To prevent stream crossings, we first order the notes of each onset group by their pitch. We then number each path in order, starting at 1. Finally, for each onset group, we specify that the order of pitches must correspond to the order of paths. Figure 3 illustrates, showing both a legal and an illegal analysis. Formally, for each onset group V_i:

$$\forall u, v \in V_i : (\text{FREQ}[u] > \text{FREQ}[v]) \Leftrightarrow (\text{PATH}[u] < \text{PATH}[v]) \quad (2)$$



a. Legal analysis. **b.** Illegal analysis.

Figure 3. Illustration of the no-crossing constraint, as defined in Equation 2. Each panel shows the set of vertices whose notes have the same onset time, as well as assignments of these vertices to paths.

3.3. Weighting the Edges

To weight the edges, we desire a function, w: $E \rightarrow [0,1]$, that awards higher weights to more plausible connections. Intuitively, the weight function should be inversely proportional to both the pitch difference and the silence separating two notes—their offset-to-onset interval, that is. We thus define two functions—one for pitch, another for time.

The pitch function is:

$$w_{\text{pitch}}(u,v) = \left(\frac{\min(\text{FREQ}[u], \text{FREQ}[v])}{\max(\text{FREQ}[u], \text{FREQ}[v])} \right)^{\alpha} \quad (3)$$

where min(x, y) and max(x, y) return the minimum and maximum of x and y, and α is a free parameter to be optimized. To set α, we constructed two sets of edges from a corpus of MIDI files: E_{pos}, edges representing

correct connections, and E_{neg}, edges representing incorrect connections. Using a hill-climber, we minimized:

$$\frac{1}{\left|E_{\text{pos}}\right|}\sum_{(u,v)}\left(1-w(u,v)\right)+\frac{1}{\left|E_{\text{neg}}\right|}\sum_{(u,v)}w(u,v) \qquad (4)$$

Our rationale for Equation 4 is that, with an ideal weighting function, correct edges would score as close as possible to 1, and incorrect edges would score as close as possible to 0. The result from optimization was $\alpha = 3.1$, which we used for all tests reported below.

The time function is:

$$w_{\text{time}}(u,v)=\frac{\beta}{\text{ON}[v]-\text{OFF}[u]+\beta} \qquad (5)$$

where β is another free parameter. In the same way we determined α, we set $\beta = 5$, which we used for all tests we report.

We define the edge-weighting function as the geometric mean of the pitch and time functions:

$$w(u,v)=\sqrt{w_{\text{pitch}}(u,v)\,w_{\text{time}}(u,v)} \qquad (6)$$

We opt for the geometric mean because if a connection is implausible in either pitch or time, it is likely implausible overall. If two notes are four octaves apart, for instance, they should probably not be connected, even if they are separated by no silence. Mathematically, this would mean that: $w_{\text{pitch}}(u, v) \approx 0 \vee w_{\text{time}}(u, v) \approx 0 \Rightarrow w(u, v) \approx 0$. The geometric mean ensures this.

3.4. Analyzing Segments of Onset Groups

After the music is sorted into onset groups (see 3.1), we divide it into segments of l contiguous onset groups, where l is a parameter chosen by the user. Figure 4 illustrates for $l = 3$. For all tests reported below, $l = 4$.

We then find the optimal analysis of each segment. That is, if the graph $G = (\mathcal{V}, E)$ represents a given segment, we find the sub-graph $G' = (\mathcal{V}, E' \subseteq E)$ that meets the four conditions in 3.1, where condition 3 comprises the no-crossing constraint defined in 3.2. Computationally, we encode this sub-graph as a set P of paths through G'. This set is indexed such that $p_i[j]$ is the j^{th} vertex along the i^{th} path. We thus rewrite Equation 1 as:

$$f(P)=\sum_{p\in P}\sum_{(u,v)\in E[p]}w(u,v) \qquad (7)$$

where $E[p]$ denotes the set of edges in path p. $f(E')$ and $f(P)$—as defined by Equations 1 and 7—are equivalent, since $E' = \bigcup_{p\in P} E[p]$.

Finding the correct sub-graph is a constraint satisfaction optimization problem (CSOP), as defined in reference [3]. To solve this CSOP correctly every time, an algorithm would need to examine all possibilities. For our problem, the complexity of this search would be $O(m^n)$, where there are m paths and n vertices. We must therefore limit the number of vertices, lest the search become intractable. This is done by keeping the length, l, of the onset group window low, and our experience suggests it should not exceed 6. We also employ two techniques to speed the search.

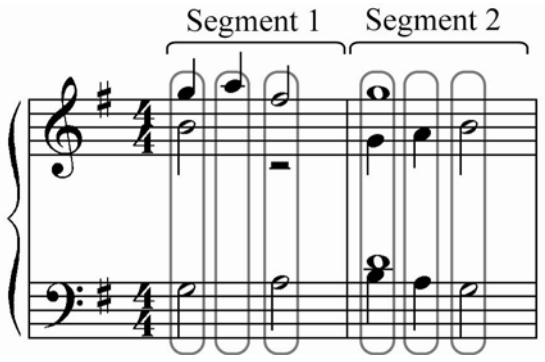


Figure 4. Segmentation of music into segments of onset groups. The grey rounded rectangles enclose onset groups, and the brackets span onset-group segments.

First, we use the "branch-and-bound" heuristic, as suggested in reference [3]. Each analysis is built step-by-step—assigning one vertex at a time to a path—and at each step a heuristic function is applied. This function returns the best possible score of the analysis, given its current state. If this score is lower than that of a previous analysis, then the current analysis, in all possible completions, can be safely abandoned. Formally, let P be the partial set of paths of the analysis being built. The heuristic function is:

$$h(P)=f(P)+r(P) \qquad (8)$$

where:

$$r(P)=\left|\mathcal{V}\right|-m-\sum_{p\in P}\left|E[p]\right| \qquad (9)$$

where V_i is the i^{th} onset group, and m is the number of paths being sought. In Equation 8, $f(P)$ is the score of the partial analysis. In Equation 9, $\sum_{p\in P}|E[p]|$ is the number of edges added so far, and $\mathcal{V} - m$ is the maximum possible number of edges, since no edge can have more than one edge originating from it, and the last vertex in each path will have no such edge. So, with the maximum edge weight being 1, $r(P)$ is the largest possible increase of $f(P)$.

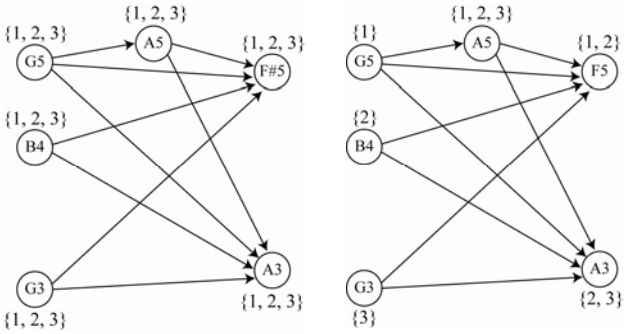
Second, we exploit a combinatorial simplification implicit in our no-crossing constraint. Each vertex can be thought of as having a set of paths it can be assigned to. If we were searching for three paths, Figure 5a would represent the naïve assumption that each vertex could be assigned to each path. This would not be the case, though, since the no-crossing constraint (see 3.2) would limit the possible assignments to those in Figure 5b. Following

Equation 2, the set of paths to which the vertex $v_{(i,j)} \in V_i$ can be assigned is:

$$\{s, \ldots, m - |V_i| + j\} \tag{10}$$

where V_i is the i^{th} onset group; j is the vertex's ordinal number in V_i, as defined by Equation 2; m is the number of paths sought; and s is defined recursively:

$$s = \begin{cases} 1 & \text{if } j = 1 \\ \text{PATH}\left[v_{(i,j-1)}\right] + 1 & \text{if } j > 1 \end{cases} \tag{11}$$



a. All possible assignments included.

b. Illegal assignments omitted.

Figure 5. Graphs showing the set of paths to which each vertex can be assigned without the no-crossing constraint (a) and with the constraint (b).

To find the optimal analysis of a segment of onset groups, we use the algorithm in Figure 6. In this pseudocode:

- P* is the best analysis found so far.
- b is the score of this analysis.
- m is the number of paths being sought.
- t is the onset group being examined.
- k is the vertex being examined.
- W is a matrix containing the weights of all edges in the graph. If two vertices are not connected, the respective entry in W is $-\infty$.
- IS-LEGAL(p_i, W, v) returns *true* if adding vertex v to path p would violate no constraints and *false* otherwise.
- APPEND(p, v) appends vertex v to path p.
- REMOVE(p, v) removes vertex v from path p.
- The vertices, $\mathcal{V}$, are indexed such that $v_{(i,j)}$ is the j^{th} vertex in the i^{th} onset group.

CSO is essentially a standard algorithm for solving CSOPs (see [17]), save two things. First, to accommodate the indexing of $\mathcal{V}$, it uses two indices, t and k, instead of one (see lines 1 and 13–15). Second, according to Equations 9 and 10, it examines only analyses that will not violate the no-crossing constraint from 3.2 (see lines 6–9).

It is, of course, possible that ANALYZE-SEGMENT will not find a legal analysis, given some number of paths, m. We therefore run it multiple times for each segment, starting with m = 1 and incrementing m each time until a solution is found or m = 8. We chose this value because all pieces in our test corpora had less than eight simultaneous paths.

```
ANALYZE-SEGMENT(𝒱, W, m)
1.  global P* ← ∅
2.  global b ← –∞
3.  P ← a set of m empty paths, p_i
4.  CSO(P, 𝒱, W, m, 1, 1)
5.  return P*

CSO(P, 𝒱, W, m, t, k)
1.  if t > |𝒱|
2.     then if f(P) > b
3.             then b ← f(P)
4.                  P* ← P
5.  elseif h(P) > b
6.     then if k = 1
7.             then s ← 1
8.             else  s ← PATH[v_(t, k – 1)] + 1
9.          for i ← s to m – |v_t| + k
10.            do if IS-LEGAL(p_i, W, v_(t, k))
11.                  then APPEND(p_i, v_(t, k))
12.                       PATH[v_(t, k)] ← i
13.                       if k = |v_t|
14.                          then CSO(P, 𝒱, W, m, t + 1, 1)
15.                          else  CSO(P, 𝒱, W, m, t, k + 1)
16.                       REMOVE(p_i, v_(t, k))
17.                       PATH[v_(t, k)] ← NIL
```

Figure 6. ANALYZE-SEGMENT and CSO, the functions used to find the optimal, legal assignments of vertices to paths in each segment of onset groups.

3.5. Connecting the Segment Analyses

Once each segment is analyzed, the analyses must be "stitched together." Figure 7 shows the analyses of two adjacent segments. As illustrated, two sets of vertices are relevant: L, which contains the last vertex of each path in segment 1, and R, which contains the first vertex of each path in segment 2. Our task is to find a legal, optimal, one-to-one mapping from L to R. We thus draw an edge between each pair in $L \times R$ and weight it as explained in 3.3. If the edge is illegal—if it violates the first constraint in 3.2—its weight is $-\infty$.

We seek a mapping, $L \rightarrow R$, that meets two conditions:

1. Pitch order is maintained, as specified in Equation 2.
2. Equation 11 is maximized.

$$g(L \to R) = \sum_{(u,v)\in E[L\to R]} w(u,v) \tag{11}$$

where $E[L \to R]$ is the set of edges contained in the mapping $L \to R$.

Since illegal edges are weighted as $-\infty$, the maximization of Equation 11 entails the satisfaction of the first constraint in 3.2.

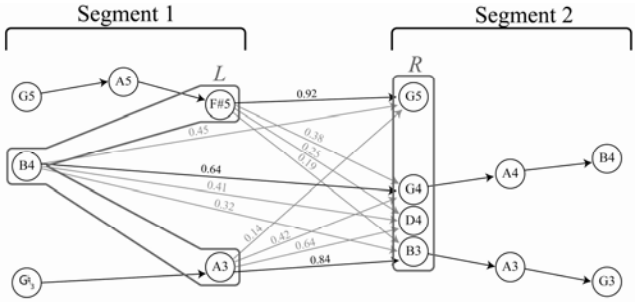


Figure 7. Illustration of how two adjacent analyses are connected. The grey rounded shapes enclose sets L and R, between which a one-to-one mapping is sought. The weighted edges connect each pair in $L \times R$. The correct ones—those in $L \to R$—are darkened.

Such a mapping can be found using a sequence alignment algorithm, like those used to compute edit distance (see [19] or [20]). Let G be a matrix of mapping scores, and let A be a matrix of arrows. Both matrices are $(|L| + 1)$-by-$(|R| + 1)$. The mapping scores are derived from Equation 11, and the arrows encode the mapping itself. To find the target mapping, we first initialize the leftmost columns and topmost rows of G and A with zeros and NIL values. We then fill the rest of the matrices recursively:

$$g_{ij} = \max \begin{cases} g_{i-1\,j-1} + w(l_i, r_j) & \text{(option 1)} \\ g_{i-1\,j} & \text{(option 2)} \\ g_{ij-1} & \text{(option 3)} \end{cases} \tag{12}$$

$$a_{ij} = \begin{cases} \nwarrow & \text{if option 1 in Eq. 16} \\ \uparrow & \text{if option 2 in Eq. 16} \\ \leftarrow & \text{if option 3 in Eq. 16} \end{cases} \tag{13}$$

After this, the mapping can be retrieved by following the arrows from the bottom-right to the top-left of A.

In the case of Figure 7, the filled matrices are:

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.92 & 0.92 & 0.92 & 0.92 \\ 0 & 0.92 & 1.56 & 1.56 & 1.56 \\ 0 & 0.92 & 1.56 & 2.2 & 2.4 \end{bmatrix} \tag{14}$$

$$A = \begin{bmatrix} \text{NIL} & \text{NIL} & \text{NIL} & \text{NIL} & \text{NIL} \\ \text{NIL} & \nwarrow & \leftarrow & \leftarrow & \leftarrow \\ \text{NIL} & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ \text{NIL} & \uparrow & \uparrow & \nwarrow & \nwarrow \end{bmatrix} \tag{15}$$

Notice that the edges in $L \to R$ are those corresponding to the diagonal arrows on the path from bottom-right to top-left in A.

4. EVALUATION OF THE MODEL

4.1. Obtaining the Ground Truth

For testing, we selected MIDI files whose tracks are likely to correspond to individual streams. With the fugal pieces we tested, each voice was encoded as a separate track. With the string quartets, each instrument had its own track. Tracks, therefore, seem to be reasonable estimates of streams in our corpora: usually listeners perceive either each polyphonic voice or each instrument as a single textural entity.

Our definition of a stream does not allow simultaneous notes, so the ground-truth streams cannot have them either. This becomes troublesome with multi-stops in string quartets, wherein one instrument plays several notes at once. This results in occasional points where multiple streams are in a single track. We resolve this problem by processing individual MIDI tracks using the streamer software. Since the MIDI tracks are simple—rarely containing more than three notes at once—we expect our system to perform well, thus giving highly accurate estimates of the correct streams. This lets us generate harder test-cases with known answer keys by re-assembling the streams found in all tracks of a piece

Nonetheless, our answer key will not always agree with some definitions of streams. In fugues, for instance, two voices will often join together for a few beats while a third voice states the subject. And in string quartets, the melody will sometimes be passed from one instrument to another. Unfortunately, occasional discrepancies are unavoidable, given our definitions and our approach to ground-truth generation.

4.2. Quantifying Accuracy

Let $G = (\mathcal{V}, E)$ be the graph of the music under analysis, let $G_M = (\mathcal{V}, E_M \subseteq E)$ be the sub-graph of the model's analysis, and let $G_C = (\mathcal{V}, E_C \subseteq E)$ be the sub-graph of the correct analysis. The model can err in two ways: by failing to add edges it should have, and by adding edges it should not have. To accommodate both types of errors, we evaluate the model using the F-measure [1]:

$$performance = 2\frac{precision \cdot recall}{precision + recall} \tag{16}$$

where *precision* = $|E_M \cap E_C| / |E_M|$ and *recall* = $|E_M \cap E_C| / |E_C|$.

4.3. Tests

We tested six corpora:

- J. S. Bach's two-part inventions (fifteen inventions, each tested separately)
- J. S. Bach's three-part sinfonias (fifteen sinfonias, each tested separately)
- The fugues from J. S. Bach's *Well-Tempered Clavier, Book I (WTC I)* (twenty-four fugues, each tested separately)
- The fugues from J. S. Bach's *Well-Tempered Clavier, Book II (WTC II)* (twenty-four fugues, each tested separately)
- Haydn's *String Quartets*, Op. 1 (twenty-seven movements, each tested separately)[1]
- Brahms' *String Quartets*, Opp. 51 and 67 (three quartets, not separated by movement)

Table 1 summarizes the distribution of performance scores for each corpus. It includes both the overall performance and the mean performance on the individual segments of each piece (i.e. before analyses are connected). These means are weighted by the number of edges in the correct sub-graph of each segment.

We performed a one-way ANOVA on the performance results summarized in Table 1, grouping by corpus. Performance was significantly different across corpora ($p < 0.0001$). Bonferroni comparisons revealed that the system performed significantly better on the inventions, sinfonias, and fugues than on either set of string quartets. There were, however, no significant differences within these two groups.

The mean performance on the inventions is deceptive, for the score on invention ten, at 0.64, was an extreme outlier. Indeed, by removing this datum, the mean performance on this corpus jumps from 0.95 to 0.98, and the standard deviation drops from 0.9 to 0.2. We thus performed a second ANOVA with outliers omitted from each corpus. An outlier was defined as anything outside the range $[Q_1 - (Q_3 - Q_1), Q_3 + (Q_3 - Q_1)]$, where Q_1 and Q_3 are the first and third quartiles for the respective corpus. Under this definition, invention ten, along five outliers in other corpora, was withheld. This ANOVA again revealed a significant effect of corpus ($p < 0.0001$). Now, however, performance was significantly better on the inventions than on the other five corpora, according to Bonferroni comparisons. This was the only change from the previous ANOVA.

We believe the higher performance on the inventions resulted partly from there being fewer streams in the inventions—and thus fewer opportunities for mistakes. And, in all likelihood, this factor also contributed to the lower performance on the string quartets, which have four basic streams rather than two or three. To investigate this possibility, we compared, for the individual segments, the number of streams found to the system's performance. If more streams cause lower performance, one would expect a negative correlation between the two variables. This, in fact, was the case for Bach's sinfonias and fugues and for Haydn's string quartets (sinfonias, $r = –0.16$, $p < 0.0001$; WTC I, $r = –0.19$, $p < 0.0001$; WTC II, $r = –0.17$, $p < 0.0001$; quartets, $r = –0.28$, $p < 0.0001$). But on the other two corpora, correlations were negligible (inventions, $r = 0.01$, $p = 0.72$; Brahms quartets, $r < 0.01$, $p = 0.41$).

Pitch range might also have been a factor. In the inventions, the two voices tend to occupy very different registers, making the voices easier to separate based on pitch, whereas this registral separation is not as pronounced in the sinfonias, fugues, and quartets.

Based on our analysis of the system's errors, the poorer performance on the string quartets was partially caused by their greater textural complexity. In particular, it is common in string quartets for fewer than all voices to sound at once. This does, of course, happen in fugal pieces, but when fugal voices drop out, it tends to be for relatively long times. In string quartets, however, instruments will often exit and enter sporadically. Each segment thus has fewer onset groups in which all of its streams sound, and this makes the no-crossing constraint less effective in guiding the search.

Also problematic were cases in which streams actually cross. Our system, with its no-crossing constraint, cannot analyze such instances correctly. This problem occurred with all six corpora, though most often with the string quartets.

Errors on the full pieces were significantly higher than errors on the individual segments ($p = 0.03$ by a paired, two-tailed *t*-test; $p < 0.0001$ by a sign test). With outliers removed as explained above, this difference remained significant under the sign test ($p < 0.0001$) but not under the *t*-test ($p = 0.06$). Thus, constraint satisfaction optimization produces relatively few errors, and more errors result from the sequence alignment phase.

Corpus	Overall		Segments	
	Mean	St. Dev.	Mean	St. Dev.
Inventions	0.95	0.09	0.98	0.03
Sinfonias	0.92	0.02	0.95	0.02
WTC I	0.93	0.03	0.95	0.02
WTC II	0.92	0.03	0.94	0.03
Haydn	0.81	0.06	0.83	0.06
Brahms	0.76	0.02	0.80	0.03

Table 1. *F*-measure of the system's accuracy on the six corpora, both overall and for the individual segments analyzed by constraint satisfaction optimization.

[1] Quartet 5, movement 2 was omitted, due to technical problems with the MIDI file.

5. CONCLUSION

We have proposed a new system for separating streams in MIDI files, which has several advantages not found collectively in any previous system: flexibility sufficient to analyze music of different genres and periods, empirically derived parameters, and windows of analysis large enough to avoid many errors characteristic of local search.

We tested our system on a corpus of music both larger and stylistically more diverse than those used by previous authors. The system's accuracy, by the *F*-measure, was 0.92–0.95 on the fugal pieces and 0.76–0.81 on the string quartets. Many errors stem from one our model's constraints—that which prohibits streams from crossing. This constraint, though useful, forced errors when streams actually crossed. Thus, a more flexible approach to avoiding stream crossings would likely bring improvement. Performance was higher for onset-group segments than for whole pieces, and exploring better ways to connect segments would likely be another fruitful avenue for future research.

This work is a promising step forward in the creation of a MIDI streamer useful across a wide variety of pieces of tonal music from different genres and by different composers. Such a streamer would be invaluable for several areas of music information retrieval, such as musical database search and melodic analysis of large MIDI corpora.

6. ACKNOWLEDGEMENTS

We thank Richard Ashley, Robert Gjerdingen, and David Temperley for their advice on this project. This work was funded in part by National Science Foundation Grant number IIS-0643752.

7. REFERENCES

[1] Bregman, A. *Auditory scene analysis: The perceptual organization of sound*. MIT Press, Cambridge, MA 1994.

[2] MIDI Manufacturers Association, Complete MIDI 1.0 Detailed Specification, November 2001.

[3] Marsden, A. "Modeling the perception of musical voices: A case study in rule-based systems," *Computer Representations and Models in Music*, Academic Press, London, 1992.

[4] Kirlin, P. and P. Utgoff. "VoiSe: Learning to segregate voices in explicit and implicit polyphony," *Proceedings of ISMIR 2005*, Queen Mary, University of London.

[5] Temperley, D. *The cognition of basic musical structures*. MIT Press, Cambridge, 2001.

[6] Madsen, S. and Widmer G. "Separating voices in MIDI," *Proceedings of ICMPC 2006*, Bologna, Italy.

[7] Lerdahl F. and R. Jackendoff. *A generative theory of tonal music*. MIT Press, Cambridge, 1983.

[8] Cambouropoulos, E. "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception* 26-1, 2008.

[9] Gjerdingen, R. "Apparent motion in music?," *Music Perception* 11-4, 1994.

[10] Szetzo, W.M. and M.H. Wong. "A stream segregation algorithm for polyphonic music databases," *Proceedings of the Seventh International Database Engineering and Applications Symposium*, Hong Kong, 2003.

[11] Chew, E. and X. Wu. "Separating voices in polyphonic music: A contig mapping approach," *Computer Music Modeling and Retrieval: Second International Symposium*, Berlin, 2004.

[12] Karydis, I., A. Nanopoulos, A.N. Panadopoulos, and E. Cambouropoulos. "VISA: The voice integration/segregation algorithm," *Proceedings of ISMIR 2007*, Vienna.

[13] Killian, J. and H. Hoos. "Voice separation—A local optimization approach," *Proceedings of ISMIR 2002*, Paris.

[14] Beauvois, M.W. and R. Meddis. "Computer simulation of auditory stream segregation in alternating-tone sequences," *Journal of the Acoustical Society of America* 99-4, 1996.

[15] McCabe, S.L. and M.J. Denham. "A model of auditory streaming," *Journal of the Acoustical Society of America* 101-3, 1997.

[16] Wang, D. and G. J. Brown. *Computational auditory scene analysis: principles, algorithms, and applications*. IEEE Press, Piscataway, NJ, 2006.

[17] Tsang, E. *Foundations of constraint satisfaction*. Academic Press Limited, San Diego, 1993.

[18] Russell, S. and P. Norvig. *Artificial intelligence: A modern approach*. Prentice Hall, Upper Saddle River, 2002.

[19] Durbin, R., S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge, 1998.

[20] Cormen, T., C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. MIT, Cambridge, 2001.

[21] Manning, C. D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. 2nd ed. Cambridge University Press, New York, 2008.