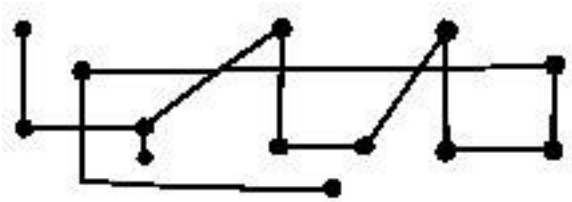
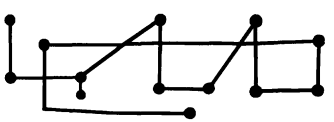


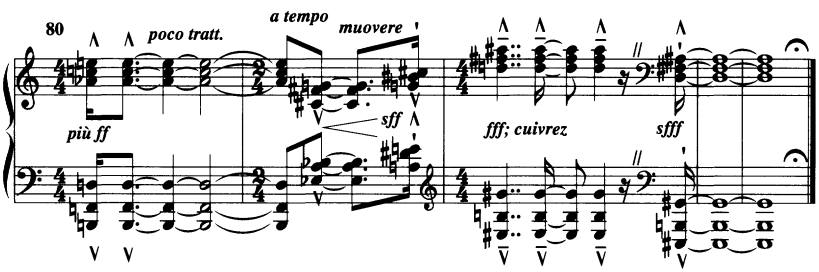
BROKEN AND UNBROKEN INTERVAL CYCLES AND THEIR USE IN DETERMINING PITCH-CLASS SET RESEMBLANCE



MICHAEL BUCHLER

THE THIRD SONG of Dallapiccola's *Quattro Liriche di Antonio Machado* ends with a series of six-note chords in the piano part (Example 1).[1] The first chord, repeated and held through measure 80, is a member of set class (SC) 6-*Z*28 [013569]; the second chord is its literal complement, a member of SC 6-*Z*49 [013479] (they form the two halves of a twelve-tone series). The third and fourth distinct chords (from the end of measure 81 to the end of the excerpt) reverse this pattern at a different transpositional level. While these chords types all have the same interval-class (ic) content (as is the nature of *Z*-related SCs), the subsets and inter-

vals that they project differ greatly. The two realizations of 6-*Z*49 prominently feature two closely-spaced realizations of 3-5 [016]—one in each hand. In these chords, ics 1, 5, and 6 are most salient. By contrast, both realizations of 6-*Z*28 prominently feature a close-position augmented triad (3-12 [048]) in the right-hand part and an open-position diminished triad (3-10 [036]) in the left-hand part. Or, taking the lowest four notes of these 6-*Z*28 chords (including the lowest note in the right hand) yields a complete diminished seventh chord (4-28 [0369]), spaced as two tritones, nine semitones apart.



EXAMPLE 1: DALLAPICCOLA, *QUATTRO LIRICHE DI ANTONIO MACHADO*, SONG 3, PIANO PART, LAST FIVE MEASURES

These differences between the 6-*Z*28 and 6-*Z*49 chords are not merely products of spacing. Even though both set types share the same intervallic profile, 6-*Z*49 embeds neither 3-12 nor 4-28. Because 6-*Z*28 *does* embed these two set types, both of which are complete interval cycles, I will argue that it has the potential of projecting ics 4 and 3 (their cyclic progenitors) more strongly than does its *Z*-equivalent, 6-*Z*49. This article will propose a series of pitch-class-based analytical tools (including a similarity index) that differentiate such set pairs, while still acknowledging their intervallic affinities.

* * *

Marcus Castrén's recent work on measures of pitch-class set resemblance establishes a dichotomy between methods that compare only the sets' interval-class content and those that consider all subset classes.[2] Examples of interval-class-based resemblance measures include Morris's ASIM, Isaacson's IcVSIM and more recent ISIM, my own interval-class saturation similarity measure—or SATSIM(2),[3] and the new ANGLE measure

by Damon Scott and Eric Isaacson. Examples of subset-based measures, which Castrén calls "total" measures, include Rahn's ATMEMB, Castrén's RECREL, and potentially Lewin's REL (depending on which subset classes are included in the TEST group).[4]

Castrén, among others, objects to interval-class-based measures because they tend to produce a smaller number of distinct values than do total measures, and because they do not distinguish between *Z*-related set classes.[5] Total subset-based measures such as the ones mentioned above *do* distinguish *Z*-related set classes, and each of them produces a greater number of values than do any of the aforementioned interval-class based measures. However, I'm not convinced that there is any correlation between the number of distinct values produced and the quality—or effectiveness—of a particular measure. The measure that will be presented later in this article produces hundreds more values than any of these total measures, but I don't believe that this is necessarily an advantage.

Total measures, almost by definition, use different criteria in comparing sets that are not the same size. For example, if one wanted to compare two hexachords using a total measure, one would examine their mutual pentachord-, tetrachord-, trichord-, and dyad-class embeddings. If, however, one wanted to compare a hexachord to a trichord, one could only compare the mutual dyad-class (and perhaps trichord-class) content of the two sets. While each of these so-called total measures includes an algorithm to bring such unequal comparisons into a common range of values, they still create scenarios where different means are used to compare sets of unequal size.

Rather than judging resemblance by comparing interval classes or all available subset classes, I will propose a method that is based upon how each set is partitioned with respect to the six distinct interval-cycles. (Because interval 7- through 11-cycles may be understood as either retrogrades or inversions of interval 5- through 1-cycles, they will not be considered distinct.) This information serves as the basis for a new weighted six-argument vector that resembles the interval-class vector (ICV) in function (or at least in its function as data for similarity indices) but not in design. Each argument of the vector represents the degree to which instances of corresponding interval-class *n* are found in unbroken *n*-cycle segments. The assumption behind the weighting is that, for any set class *X*, the more that instances of interval-class *n* form a particular *n*-cycle, the more likely that *X* will project interval-class *n*. For example, one might reasonably claim that a four-note quartal (or quintal) chord projects ic5 more strongly than does a chord with three cyclically nonadjacent ic5s. Although I am addressing only pitch-class sets and not

their particular orientations in pitch space, I believe that it is still legitimate to assert that many—or even most—realizations of set class [0257] will project ic5 to a great extent.

* * *

Before introducing the new vector types and similarity index, it will be useful to make a few comments on the cycles themselves and the ways in which they can be segmented and concatenated to form "cyclic sets." The group of cyclic sets has been discussed elsewhere in the theoretical literature,[6] but I will be undertaking an approach that is rather different in nature from these studies. My approach will lead toward a method for comparing two set classes based upon their shared and different cyclical construction.

Let us define an *n*-cycle (where *n* is a variable that represents any interval class in standard twelve-pc space) as a closed and finite ordered collection of pitch classes where one element maps onto the next (and the last onto the first) under transposition at a constant interval *n*.[7] The members of an *n*-cycle are defined as $(x+n, x+n^2, x+n^3, \ldots x+n^p = x)$ where *p* is the period of the *n*-cycle. For most values of *n*, there are several distinct *n*-cycles in the 12-pc aggregate. For example, there are four 4-cycles: (048), (159), (26a), and (37b). Because each *n*-cycle has *p* elements, there must be $12/p$ distinct cycles formed by interval *n* (we call this value *m*). The complete *n*-cycles are shown in Example 2.

1 cycle:	(0123456789ab)
2 cycles:	(02468a) (13579b)
3 cycles:	(0369) (147a) (258b)
4 cycles:	(048) (159) (26a) (37b)
5 cycles:	(05a3816b4927)
6 cycles:	(06) (17) (28) (39) (4a) (5b)

EXAMPLE 2: CYCLIC SETS IN TWELVE-PC SPACE

Because all instances of ic *n* occur segmentally within the *n*-cycle(s), any pitch-class set that simply is a complete *n*-cycle naturally features the maximal amount of a given ic *n* for a set of its cardinality. {0, 2, 4, 6, 8, a}, for example, is maximally ic2 saturated; {0, 3, 6, 9} is maximally ic3

saturated, and so forth. The same is true of pcsets that are wholly the union of two *n*-cycles (for a single given *n*). Both {0, 1, 4, 5, 8, 9} and {0, 2, 4, 6, 8, a} can be formed by the union of two 4-cycles,[8] and consequently both hexachords maximally include ic4. For an interval *n* whose cycles have periodicity *p*, then, we know how to identify the "maximally *n*-saturated set types" whose cardinalities are *p* or integer multiples of *p*.

A pcset that is smaller than *p* will maximally saturate ic *n* if it is (again, wholly) a continuous *n*-cycle segment. For *n* = 2 (with *p* = 6), the two- through five-element set classes that maximally saturate ic2 are [02], [024], [0246], and [02468]. A pcset that is larger than *p* will maximally saturate ic *n* if it is the combination of however many complete *n*-cycles cardinality permits (possibly just one) and an incomplete *n*-cycle of whatever length cardinality requires. For *n* = 4 (with *p* = 3), any combination of, for example, {0, 4, 8} and some segment from one of the other three 4-cycles will produce sets that are maximally saturated with ic4 (e.g., {0, 1, 4, 8}, {0, 1, 4, 5, 8}, {0, 2, 4, 8}, and {0, 2, 4, 6, 8}).

We can condense the above conditions for maximal *n*-saturation into a single definition of what we shall call an *n*-set (for interval *n*). An *n*-set is comprised of some number of complete *n*-cycles (possibly none, one, or more than one) and, at most, one incomplete *n*-cycle segment. The complete list of all *n*-sets is the same as Tore Ericksson's maxpoint series.[9] All *n*-sets are maximally saturated with interval *n* and all pcsets that are maximally saturated with interval *n* are *n*-sets.

We will now return to the creation of several new vector types that reflect how the elements of a pcset are distributed with regard to the interval cycles. We will first examine such cyclic distribution, focusing on the number and position of any cyclic adjacencies. Next, we will create a version of the interval-class vector that distinguishes the size and quantity of all *n*-cycle segments. This amounts to a subdivided interval-class vector, the arguments of which will be weighted using a procedure that gives cyclic strings of intervals more prominence than equal numbers of the same intervals that are not all cyclically adjacent. My final construct will be derived by comparing these cyclically weighted interval-class vector arguments to what is possible given any set of the same size. This is what I have elsewhere called a measure of saturation.[10] These adjusted values will provide us with a relatively cardinality-neutral means of relating sets based upon their *n*-cycle subsets.

We will begin our transformation from an objective inventory of the intervals within a set to a weighted cyclic saturation vector by examining the manner in which elements of a pitch-class set are distributed among the *n*-cycles. Example 3 shows the cyclic distribution of set class 6-Z28 [013569] (interval-class vector: <224322>). Each line of the example

shows adjacencies within a particular *n*-cycle simply by ordering the elements of the "most normal" form of 6-*Z*28 along the cycle. Parentheses delineate the cycles, and adjacent pitch classes within the parentheses (including the wraparound) are *n*-cycle adjacencies, each producing a single embedded interval-class *n*. Dashes indicate vacant places in each *n*-cycle.

1-cycle distribution of 6-*Z*28:	(01-3-56--9--)
2-cycle distribution of 6-*Z*28:	(0--6--) (135-9-)
3-cycle distribution of 6-*Z*28:	(0369) (1---) (-5--)
4-cycle distribution of 6-*Z*28:	(0--) (159) (-6-) (3--)
5-cycle distribution of 6-*Z*28:	(05-3-16--9--)
6-cycle distribution of 6-*Z*28:	(06) (1-) (--) (39) (--) (5-)

EXAMPLE 3: DISTRIBUTION OF (SET CLASS) 6-*Z*28 [013569] AMONG THE SIX DISTINCT *n*-CYCLES

Example 3 illustrates how the pcset's elements are distributed among the cycles of any given interval. As a means of summarizing this data, we will create an array called $\text{CycleSeg}_n(X)$. This construct lists the cardinalities of the *n*-cycle segments of *X* from longest to shortest. The sum of $\text{CycleSeg}_n(X)$ numbers equals the cardinality of set *X*. Example 4 shows the cyclic segment lengths of our set, 6-*Z*28; compare these numbers with the patterns in Example 3. In Example 3, we can see that 6-*Z*28's elements fall into four disjunct segments of the 1-cycle, two of two elements and two of one; these are now represented by the array <2, 2, 1, 1>. Any realization of the set class (for example, {C, C♯, E♭, F, F♯, A}) will have two two-note 1-cycle segments ({C, C♯} and {F, F♯}) and two one-note segments ({E♭} and {A}).

The lengths of the various segments indirectly tell us the interval-class content of a pcset. We can see, for example, that the two-note 1-cycle segments are the source of the two ic1s in the set, and also that a single unbroken three-note 2-cycle segment is the source of 6-*Z*28's two ic2s. Similarly, the three ic4s in 6-*Z*28 arise from a single complete 4-cycle subset. We will now create a new vector—the "ic-cycle vector" (abbreviated as ICCV)—that conveys not only that there are two ic1s and two ic2s in set class 6-*Z*28, but also that those two ic1s arise from disjunct 1-cycle segments and that the two ic2s arise from a single 2-cycle segment.

CycleSeg1(6-*Z*28): <2,2,1,1>
CycleSeg2(6-*Z*28): <3,1,1,1>
CycleSeg3(6-*Z*28): <4,1,1>
CycleSeg4(6-*Z*28): <3,1,1,1>
CycleSeg5(6-*Z*28): <2,2,1,1>
CycleSeg6(6-*Z*28): <2,2,1,1>

EXAMPLE 4: THE LENGTHS OF IC *n*-CYCLE SEGMENTS (CycleSeg_n) OF 6-*z*28 [013569]

Each of the ic-cycle vector's six arguments is derived from the corresponding CycleSeg_n values. Instead of showing the number of pcs in each *n*-cycle segment, the ICCV lists the number of ic *n* found in each *n*-cycle segment. For the most part, deriving the number of ic *n* in an *n*-cycle segment simply amounts to subtracting 1 from the size of each segment (for example, a one-note *n*-cycle segment yields no interval *n*s, a two-note segment yields a single interval *n*, and so on). This holds true for all cyclic fragments (i.e., incomplete *n*-cycles); in cases where a complete *n*-cycle is embedded in a set, the number of instances of interval *n* is equal to the period of the cycle, not the length minus one. Consider, for example, a complete 4-cycle such as {0, 4, 8}. Here the period of the cycle—the number of steps until the last element maps onto the first—is 3; therefore, this cycle yields three, not two, interval class 4s. This holds true for all *n*-cycles except 6-cycles; two-element cycles yield only one interval each. Example 5 shows the interval-class-cycle vector of set class 6-*Z*28.

ICCV(6-*Z*28): <{1,1}, {2}, {4}, {3}, {1,1}, {1,1}>

EXAMPLE 5: INTERVAL-CLASS-CYCLE VECTOR (ICCV) OF 6-*z*28 [013569]

For any given set (or SC), the sum of the numbers in each ICCV argument equals the parallel ICV argument; more formally,

$$\text{ICV}_n(X) = \sum_{t \in T} (\text{ICCV}_{n_t}(X))$$

where T is the set of interval strings of ic n and t are the elements of T. This elaboration on the interval-class vector will be essential in developing the notion that two ic n from different n-cycle segments produce a different degree of ic n salience than two ic n from the same n-cycle segment.

Our new vector lists the ic1 content of 6-*Z*28 as {1,1}, which tells us that the two ic1s appear in disjunct locations along the single 1-cycle; the ic3 content of 6-*Z*28 is listed as {4}, which tells us that the four ic3s are all from a complete embedded 3-cycle. Of course, we only know that an ICCV3 entry of {4} indicates a complete 3-cycle because we know that the period of a 3-cycle is 4. That same entry would indicate a cyclic fragment for ic1, ic2, and ic5 (and for ic4 and ic6, it wouldn't even be possible). Cyclic periodicity must also be taken into account in order to understand the degree to which a cycle *can* be fragmented. An ICCV entry of {1,1}, for example, represents cyclic fragments for any ic but 6, for which it represents two complete cycles.

Accordingly, we need to understand how many fragments are possible for the cycles of various ics, with their various periods. Obviously, more fragments will be possible when their lengths are shortest; and the shortest possible fragments—single pcs—can be extracted from a cycle in the greatest number by simply taking every other cyclic element. A maximally fragmented n-cycle, then, would contain $0n$, $2n$, $4n$, and so on, as far as the cycle permits. The maximum number of n-cycle fragments equals the greatest integer that does not exceed half the period of ic n. Of course these one-element fragments produce exactly *no* occurrences of ic n; and we are at least as interested in finding out how *uncyclic* some actual ic-n content can be. The process of determining this would be to skip every third element along an n-cycle. By taking two n-cycle adjacencies, we form a single ic n, but by skipping the third, we avoid cyclic segments longer than two notes (and ic n strings longer than one). Thus, the maximum number of unconnected ic n instances in an n-cycle is the largest integer that does not exceed one-third of the period. The product of this figure and the number of distinct n-cycles (again we call this variable m) equals the maximum number of arguments in each ICCV_n. This can be represented more formally as $m \bullet round(p/3)$. These values (i.e., the maximum number of two-element or larger n-cyclic fragments for each distinct n) are provided in Example 6.

A 1-cycle, for example, can be broken into as many as four disjunct two-note or larger fragments. There is only one set class that has four ic1s, none of which are conjunct: 8-28 [0,1,3,4,6,7,9,a], the octatonic collection. Since it is impossible to add another pc to this pcset without adjoining *two* of the cyclic fragments, there will never be any more than

four elements in $ICCV_1$. In contrast to the single 1-cycle, there are three possible ic3 cycles. Fragmentation of these cycles is not an issue, however, since it is only possible to have two nonadjacent fragments within the same 3-cycle, and both fragments could only be one-pc long, producing no ic3. Therefore, there can only ever be three elements in the $ICCV_3$ vector (representing the three distinct 3-cycles).

n-cycle	*p*	*m*	*m**(*p*/3)(rounded)
1	12	1	$1 \cdot (12/3) = 4$
2	6	2	$2 \cdot (6/3) = 4$
3	4	3	$3 \cdot (4/3) = 3$
4	3	4	$4 \cdot (3/3) = 4$
5	6	1	$1 \cdot (12/3) = 4$
6	12	6	$6 \cdot (2/3) = 6$

EXAMPLE 6: CALCULATION OF THE NUMBER OF POSSIBLE ARGUMENTS IN EACH ICCV INTERNAL VECTOR

* * *

Let us now return to our analysis of 6-*Z*28's cyclic distribution. For the sake of comparison, we will also examine the cyclic distribution and interval-class-cycle vectors for set class 6-*Z*49 (prime form [013479]), the set class *Z*-related to 6-*Z*28. These are shown in Example 7 below. While each interval-class occurs the same number of times in 6-*Z*49 and 6-*Z*28, as is the nature of *Z*-related SCs, their arrangement differs for three of the six ics. Later, we will define an index to compare SC similarity based upon respective ICCVs; to do so, we will clearly need to differentiate values such as {4} and {2,2} (these are the $ICCV_3$ values for 6-*Z*28 and 6-*Z*49, respectively).

The premise of this article—that a single *n*-cycle segment projects ic *n* more strongly (or at least differently) than do multiple shorter ones—necessitates adjusting the ICCV arguments accordingly. Larger numbers in the ic-cycle vector, which indicate a significant cyclic presence, should be weighted more heavily than groups of smaller numbers, which denote fragmentation. For example, the ic-cycle vector argument {3} should be

1-cycle distribution of 6-*Z*49: (01-34--7-9--)

2-cycle distribution of 6-*Z*49: (0-4---) (13-79-)

3-cycle distribution of 6-*Z*49: (03-9) (147-) (----)

4-cycle distribution of 6-*Z*49: (04-) (1-9) (---) (37-)

5-cycle distribution of 6-*Z*49: (0--3-1--49-7)

6-cycle distribution of 6-*Z*49: (0-) (17) (--) (39) (4-) (--)

ICCV(6-*Z*49): <{1,1}, {1,1}, {2,2}, {1,1,1}, {1,1}, {1,1}>

EXAMPLE 7: CYCLIC DISTRIBUTION AND IC-CYCLE VECTOR OF 6-*Z*49 [013479]

weighted more heavily than {1,1,1}, because the latter indicates a greater degree of cyclic fragmentation.

This brings us to our next step: the weighting procedure. Perhaps the easiest method would be to square all the values, then add them together. This would create elements with values of

$$\sum_{n \in N} \{3^2\} = 9 \text{ and } \sum_{n \in N} \{1^2, 2^2, 1^2\} = 3.$$

Any similarity index that examined the difference between these two values (as do all commonly-used indices) would find that ic4 is three times as salient in the former set as in the latter. While I want to differentiate between longer and shorter cyclic segments and establish a bias favoring the former, I do not want create an exaggerated comparison by weighting the former *too* heavily.[11] Therefore, I believe that simply squaring the ICCV_{i_n} values produces a distorted weighting system. Taking the square root of the sum of the squared ICCV_i arguments

$$\sqrt{\sum_{n \in N} \mathrm{ICCV}_{i_n}(X)^2}$$

is one way to temper this roughness.[12] The difference between $\sqrt{9}$ and $\sqrt{3}$ (or $3 - 1.73$) is 1.27—a much smaller number, and one that would still allow for a fairly close relation of these two *Z*-related hexachords. Yet even with this adjustment, the former SC still appears to have 73% more ic4 salience. While this seems more tenable than claiming that it has 300% more ic4 salience (as would be the case if we only used the

squares), the relationship still seems quite exaggerated; and this discrepancy is considerably magnified in cyclic vectors of larger SCs.

We will therefore adopt a variable weighting system that is capable of more linear scaling. This system, designated WEIGHT, is a simple additive formula. We begin with the number 1, which we multiply by a constant value (either a real number or an integer). For the sake of this demonstration, the constant that I will be using is 1.2. The weighted value of the number 1 is 1 times 1.2, or, simply 1.2. If the number we are weighting is 2, we begin with the weighted value of 1 (again, 1.2), add 1 to it, then multiply the sum by 1.2. So, 1.2 plus 1 equals 2.2; the product of 2.2 and 1.2 is 2.64, and this is our weighted value for 2. To weight the number 3, we start with the weighted value of 2, add 1 to it, then multiply that sum by 1.2, and so forth.[13] The values produced by WEIGHT are provided in Example 8.

WEIGHT(0) =	0.00	WEIGHT(7) =	15.50
WEIGHT(1) =	1.20	WEIGHT(8) =	19.80
WEIGHT(2) =	2.64	WEIGHT(9) =	24.96
WEIGHT(3) =	4.37	WEIGHT(10) =	31.15
WEIGHT(4) =	6.44	WEIGHT(11) =	38.58
WEIGHT(5) =	8.93	WEIGHT(12) =	47.50
WEIGHT(6) =	11.92		

EXAMPLE 8: VALUES RETURNED BY FUNCTION WEIGHT WHERE THE CONSTANT IS 1.2

With Example 9, we return to the problem of weighting the two ic-cycle arguments {3} and {1,1,1}. As Example 8 shows, WEIGHT(3) = 4.37 and WEIGHT(1) = 1.2. There are three 1s in the latter vector, so we multiply 1.2 by 3, totaling 3.6. These weighted values, 4.37 and 3.6, will replace the respective arguments of the ic-cycle vector in our new weighted ic-cycle vector (abbreviated WICCV). The derivation of the weighted ic-cycle vectors for both 6-*Z*28 and 6-*Z*49 is shown in Example 10.

The weighted ic-cycle vector is an interpretation of both the interval-class vector and the manner in which the elements of a pcset fall among the six distinct ic cycles. In its current state, it could be used in place of the interval-class vector in any ic-based similarity measure, including Teitelbaum's similarity index, Morris's ASIM, Isaacson's IcVSIM and

ICCV(6-*Z*28): <{1,1}, {2}, {4}, {3}, {1,1}, {1,1}>
ICCV(6-*Z*49): <{1,1}, {1,1}, {2,2}, {1,1,1}, {1,1}, {1,1}>

$WICCV_4$(6-*Z*28) = WEIGHT(3) = 4.37
$WICCV_4$(6-*Z*49) = WEIGHT(1) + WEIGHT(1) + WEIGHT(1) = 3.60

EXAMPLE 9: WEIGHTED INTERVAL-CLASS 4 ($WICCV_4$) CONTENT OF 6-*Z*28 [013569] AND 6-*Z*49 [013479] WHERE THE WEIGHTING CONSTANT IS 1.2

ISIM, and Scott's and Isaacson's ANGLE measure. But before using this new vector as fodder for a similarity index, we will add one more degree of interpretation to it. The ICCV and WICCV numbers should carry different meaning depending upon the cyclic period and set cardinality. For example, the 3-cycle ICCV value {4} ($WICCV_3$ = 4.37), indicates a complete cycle; if the set is reasonably small (e.g., a tetrachord or pentachord), then a high degree of ic3 salience is indicated. That same value ({4}) referring to the ic1 content of an octachord suggests considerably less-salient cyclic presence.

We will therefore compare each argument of the weighted vector to the minimal and maximal possible values for that particular interval class in any set class of the same cardinality. This will help us understand the weighted ic-cycle values in the context of what is possible, and also what is trivial, for any given cardinality. This new comparison forms the "cyclic saturation vector," or CSATV for short. To derive it, simply compare each weighted ic-cycle vector argument to the minimum and maximum values for any set of the same cardinality (these are easily derived by examining all the *n*-sets). These minimum and maximum weighted values are given in Example 11.

Comparing each of the six weighted ic-cycle arguments to the minimum and maximum possible values produces a total of twelve comparisons. These are arranged into two six-place vectors in Example 12. The top vector, abbreviated $CSATV_A$, shows the comparisons of the weighted arguments to either their respective minima or maxima, whichever is closer; the bottom vector, marked $CSATV_B$, shows the more distant comparisons—that is, the comparisons that were not represented in row A.[14] A walk through Example 12 will demonstrate how CSATV is created. The top of this figure shows the minimal and maximal possible values for each ic-cycle vector argument in any hexachord (these are

ICCV(6-Z28): <{1,1}, {2}, {4}, {3}, {1,1}, {1,1}>

$WICCV_1$(6-Z28) = WEIGHT(1) + WEIGHT(1) = 2.40
$WICCV_2$(6-Z28) = WEIGHT(2) = 2.64
$WICCV_3$(6-Z28) = WEIGHT(4) = 6.44
$WICCV_4$(6-Z28) = WEIGHT(3) = 4.37
$WICCV_5$(6-Z28) = WEIGHT(1) + WEIGHT(1) = 2.40
$WICCV_6$(6-Z28) = WEIGHT(1) + WEIGHT(1) = 2.40
WICCV(6-Z28) = <2.40, 2.64, 6.44, 4.37, 2.40, 2.40>

ICCV(6-Z49): <{1,1}, {1,1}, {2,2}, {1,1,1}, {1,1}, {1,1}>

$WICCV_1$(6-Z49) = WEIGHT(1) + WEIGHT(1) = 2.40
$WICCV_2$(6-Z49) = WEIGHT(1) + WEIGHT(1) = 2.40
$WICCV_3$(6-Z49) = WEIGHT(2) + WEIGHT(2) = 5.28
$WICCV_4$(6-Z49) = WEIGHT(1) + WEIGHT(1) + WEIGHT(1)= 3.60
$WICCV_5$(6-Z49) = WEIGHT(1) + WEIGHT(1) = 2.40
$WICCV_6$(6-Z49) = WEIGHT(1) + WEIGHT(1) = 2.40
WICCV(6-Z49) = <2.40, 2.40, 5.28, 3.60, 2.40, 2.40>

EXAMPLE 10: DERIVATION OF THE COMPLETE WICCVS OF 6-Z28 [013569] AND 6-Z49 [013479] WHERE THE WEIGHTING CONSTANT IS 1.2

taken directly from Example 11). For example, the ic1 column shows that it is possible to have as small a value as zero and as large a value as 8.93, which would represent 6 pcs adjacent within a single ic1-cycle (as in set class 6-1 [012345]). The third line in Example 12 contains the weighted ic-cycle vector of our now-familiar set class, 6-Z28. The value 2.40 in the ic1 column is 6.53 less than the maximum and 2.40 more than the minimum for a hexachord. The comparative value "min+2.40" is therefore entered in the ic1 column of CSATV's top row and "max-6.53" is entered in CSATV's bottom row. In its ic3 content, we see that 6-Z28 is closer to maximal than it is to minimal saturation. In that case,

Minimum possible weighted ic-cycle vector values

Cardinality	1	2	3	4	5	6
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	1.2	0	0
6	0	0	0	2.4	0	0
7	2.4	2.4	2.64	3.6	2.4	1.2
8	4.8	4.8	5.28	4.8	4.8	2.4
9	7.92	8.84	7.92	7.92	7.92	3.6
10	12.88	12.88	11.72	11.14	12.88	4.8
11	31.15	18.36	15.52	14.30	31.15	6.0
12	47.50	23.84	19.32	17.47	47.50	7.2

Maximum possible weighted ic-cycle vector values

Cardinality	1	2	3	4	5	6
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	1.2	1.2	1.2	1.2	1.2	1.2
3	2.64	2.64	2.64	4.37	2.64	1.2
4	4.37	4.37	6.44	4.37	4.37	2.4
5	6.44	6.44	6.44	5.57	6.44	2.4
6	8.93	11.92	7.64	8.74	8.93	3.6
7	11.92	11.92	9.08	8.74	11.92	3.6
8	15.50	13.12	12.88	9.94	15.50	4.8
9	19.80	14.56	12.88	13.10	19.80	4.8
10	24.96	16.29	14.08	13.10	24.96	6.0
11	31.15	18.36	15.52	14.30	31.15	6.0
12	47.50	23.84	19.32	17.47	47.50	7.2

EXAMPLE 11: MINIMUM AND MAXIMUM WICCV VALUES FOR ALL CARDINALITIES (ROWS) AND INTERVAL CLASSES (COLUMNS) WHERE THE WEIGHTING CONSTANT IS 1.2

min(1.2, 6)	=	0	0	0	2.40	0	0
max(1.2, 6)	=	8.93	11.92	7.64	8.74	8.93	3.60
WICCV(6-*Z*28)	= <	2.40,	2.64,	6.44,	4.37,	2.40,	2.40 >
$CSATV_A$(6-*Z*28)	**<**	**min+2.40,**	**min+2.64,**	**max-1.20,**	**min+1.97,**	**min+2.40,**	**max-1.20 >**
$CSATV_B$(6-*Z*28)	**<**	**max-6.53,**	**max-9.28,**	**min+6.44,**	**max-4.37,**	**max-6.53,**	**min+2.40 >**
WICCV(6-*Z*49)	= <	2.40,	2.40,	5.28,	3.60,	2.40,	2.40 >
$CSATV_A$(6-*Z*49)	**<**	**min+2.40,**	**min+2.40,**	**max-2.36,**	**min+1.20,**	**min+2.40,**	**max-1.20 >**
$CSATV_B$(6-*Z*49)	**<**	**max-6.53,**	**max-9.52,**	**min+5.28,**	**max-5.14,**	**max-6.53,**	**min+2.40 >**

EXAMPLE 12: GENERATION OF CSATV(6-*z*28) [013569] AND CSATV(6-*z*49) [013479] WHERE THE WEIGHTING CONSTANT IS 1.2

the max-related value is shown in the top row, and the min-related value in the bottom row, and so forth.

One of the reasons for segregating min- and max-related values like this, and not putting all the min-related values in one row of the vector and all the max-related values in the other row, is to make it easier, by a quick glance at the top row, to determine whether each WICCV value is closer to minimal or maximal saturation. A more formal reason for doing do will arise shortly when we define our similarity index. Briefly, though, this alignment will allow us to discriminate on behalf of the nearer comparisons. Unlike other such indices, WICCV will be constructed to interpret arguments that reflect the nearer degree of saturation (those in the top row) as more valuable when comparing two CSATVs.

I have, to this point, been using the terms "min" and "max" rather casually. More properly, I should have notated a value such as "min+2.40" (as in Example 12 above) in a way that indicates *which* "min" is being used. Doing so involves specifying the weighting value, cardinality of set, and the weighted ic-cycle vector argument being compared. If we use the variables w, c, and i to represent these three items (respectively), then "min+2.40" should properly be written as "min(w, c, i) + 2.40." In this specific case where $w = 1.2$, $c = 6$, and $i = 1$, min(w, c, i) = 0.

Having just explained the longer, more precise, way of expressing this measure of cyclic saturation, I will now move in the opposite direction and introduce a couple of abbreviations that are warranted for the sake of space and clarity—the first is trivial, the second more substantive. The trivial change will simply be to drop the "min" and "max" designations altogether. The signed numbers alone will tell us whether each value is min- or max-related, and their placement within particular saturation vectors will implicitly contextualize them (providing the interval cycle information and set cardinality). This means that we will have to distinguish between –0.00 and +0.00. While they are numerically equal, in our abbreviated system, these two arguments represent opposite ends of the spectrum: the first indicating maximal and the latter indicating minimal saturation of a particular WICCV value. Our strictly numerical representation will also simplify the formalization of our similarity index. That matter will be addressed later, however.

The more substantial shortcut involves folding our two-part cyclic saturation vector into a single six-argument vector in which each value represents the relative distance from both min and max. This can be accomplished, but not without some loss of specificity. We begin with our max(w, c, i) value and subtract min(w, c, i) from it. This gives us a "min-adjusted" maximum saturation value. We then take the appropriate

WICCV argument and subtract min(w, c, i) from it, producing a min-adjusted WICCV value. Dividing the min-adjusted WICCV value by the min-adjusted max value produces a percentage that indicates how close a particular WICCV argument is to min(w, c, i) or max(w, c, i). We could not simply divide the WICCV value by max(w, c, i), because the result would not account for the possibility that min(w, c, i) is greater than zero. For example, if max (w, c, i) = 4, min(w, c, i) = 2, and WICCV (w, i) = 2, then simply dividing WICCV by max(2/4) indicates 50% saturation. If we take the min (w, c, i) value into account as described above, however, we get

$$\frac{2-2}{4-2} = \frac{0}{2} = 0,$$

indicating minimal saturation.

We will call the complete vector derived using this shorter system the *cyclic proportional saturation vector*, or CPSATV. The derivation of a CPSATV for our familiar hexachords, 6-*Z*28 [013569] and 6-*Z*49 [013479] are shown in Example 13 below.[15] Formally, CPSATV is defined as follows:

$$\text{CPSATV}_i(X) = \frac{\text{WICCV}_i(X) - \min(w, c, i)}{\max(w, c, i) - \min(w, c, i)}$$

While this new form of the cyclic saturation vector seems much more convenient and certainly less clumsy than our two-part CSATV (and, two CPSATVs can be related to each other rather easily using any vector-based similarity index), I generally prefer the extra degree of specificity provided by CSATV, and I worry that perhaps too much information has been packed into each of our CPSATV arguments. As I mentioned earlier, I will shortly introduce a similarity index that is biased in favor of the upper values (A parts) of each CSATV (the "closer" relations)—something that would be impossible with this shorter alternative. The single-part vector is, however, very convenient for obtaining a quick cyclic profile of a particular set class, and pairs of these vectors can be more easily compared in one's head while analyzing a particular piece of music (even if the vectors themselves cannot be derived on the fly). For this reason, I prefer to use the CPSATV while forming my initial analytical opinions and opt for the longer CSATV when drawing more specific analytical conclusions.

We have nearly arrived at the end of our trail of definitions and are almost ready to put this new saturation vector to analytic use. In doing so, we will feed the cyclic saturation vectors into a similarity index and

min(1.2, 6)	=		0	0	0	2.40	0	0	
max(1.2, 6)	=		8.93	11.92	7.64	8.74	8.93	3.60	
WICCV(6-*Z*28)	=	<	2.40,	2.64,	6.44,	4.37,	2.40,	2.40	>
WICCV - min(1.2, 6, *i*):		<	2.40,	2.64,	6.44,	1.97,	2.40,	2.40	>
max(1.2, 6, *i*) - min(1.2, 6, *i*):			8.93	11.92	7.64	6.34	8.93	3.60	
CPSATV(6-*Z*28):		<	**0.27,**	**0.22,**	**0.84,**	**0.31,**	**0.27,**	**0.67**	>
WICCV(6-*Z*49)	=	<	2.40,	2.40,	5.28,	3.60,	2.40,	2.40	>
WICCV - min(1.2, 6, *i*):		<	2.40,	2.40,	5.28,	1.20,	2.40,	2.40	>
max(1.2, 6, *i*) - min(1.2, 6, *i*):			8.93	11.92	7.64	6.34	8.93	3.60	
CPSATV(6-*Z*49):		<	**0.27,**	**0.20,**	**0.69,**	**0.19,**	**0.27,**	**0.67**	>

EXAMPLE 13: GENERATION OF CPSATV(6-*Z*28) [013569] AND CPSATV(6-*Z*49) [013479] WHERE THE WEIGHTING CONSTANT IS 1.2

examine the results. Almost any ic-based similarity index—including those named earlier—may be adapted to work with a saturation vector. The index which I will use is an extension of my own SATSIM.[16] Explained briefly, SATSIM compares the arguments in each of the top (A) rows of one saturation vector to the corresponding minimum- or maximum-related value of the other vector. This similarity index, which we will now call the cyclic saturation similarity index (or CSATSIM), is a function that compares saturation vectors of two sets, returning a real number between 0 and 1 that serves as an indicator of the two sets' degree of resemblance, following the model set by Morris's ASIM(X, Y).[17] The principal difference between the construction of ASIM(X, Y) and CSATSIM(X, Y) is that the former deals with one-part interval-class vectors while the latter uses values in a two-part saturation vector.

When relating two set classes X and Y using their cyclic saturation vectors, it is necessary to allow for the possibility (in fact, the likelihood) that the respective rows (A and B) of the two vectors might feature different patterns of max- and min-related values. The values in row A of the CSATV always reflect the "closest" comparison between the WICCV(X)$_i$ or WICCV(Y)$_i$ value and either min(w, c, i) or max(w, c, i), and those comparative values will play most heavily in our relation. To relate two CSATVs, we first compare each value in row A of pcset X's saturation vector to the corresponding min- or max-related value in either row A or B of pcset Y's saturation vector. We must then compare each value in row A of pcset Y's saturation vector to the corresponding min- or max-related value in either row A or B of pcset X's saturation vector. Because the comparison of pcset X to pcset Y frequently yields different values from the comparison of pcset Y to pcset X, it is necessary to perform both to insure symmetry.[18]

To compare two vectors using the CSATSIM index, we add the absolute values of the numerical differences found in the above comparison, and divide this sum by the combined vector totals. Vector totals are obtained by adding together the distances between the numerical values in the respective arguments of both vector lines. If, for example, a particular argument in $CSATV_A$ is +4 and the parallel argument in $CSATV_B$ is −1, the distance between +4 and −1 = 5.[19] Cyclic saturation vector total ($\sum$CSATV) is formally defined in Example 14. Saturation vectors will always total the same number for sets of the same cardinality, just as they do in ic vectors.[20] These values are provided in Example 15.

$$\sum \text{CSATV}(X) = \sum_{n}^{6} |\text{CSATV}_{\text{A}}(X)_n - \text{CSATV}(X)_n|)$$

EXAMPLE 14: FORMAL DEFINITION OF CSATV TOTALS (∑CSATV)

c	∑CSATV(X)
0	0.00
1	0.00
2	7.20
3	16.13
4	26.32
5	32.53
6	47.36
7	42.54
8	44.86
9	40.77
10	33.09
11	0.00
12	0.00

EXAMPLE 15: (∑CSATV) FOR ALL SET-CLASS CARDINALITIES (C) WHERE WEIGHT (w)=1.2

For a demonstration of how CSATSIM values are derived, consider two cyclic pcsets, X and Y where X = [012678] and Y = [0369]. These are shown with their CSATV values in Example 16. X has the value (max)-3.65 in the ic1 column of CSATV_{A}, while Y has +0.00 in the parallel place. Because pcsets X and Y are of different cardinalities, min(w, c, i) and max(w, c, i) will represent different extremes for each i. It is therefore impossible to compare a min-related value directly with a max-related value; in this case, we must look to line B of pcset Y's cyclic saturation vector, which shows that [0369] is (max)-4.37 saturated with ic1.[21] The absolute value of the difference between -3.65 and -4.37 (i.e., 0.72) is the value returned for the ic1 column. In the ic2 column,

$CSATV_A(X)$ has the value (min)+2.40, while $CSATV_A(Y)$ row has the value (min)+0.00, yielding a difference of 2.40. In this case, one need not check the value in $CSATV_B(Y)$ since row A had the necessary min-related value. This procedure (step 1 in Example 16) is repeated for each argument of $CSATV_A(X)$. One then compares each argument of $CSATV_A(Y)$ to either row A or B of pcset X's saturation vector, creating a two-part difference vector.[22]

Because only the "A" row entries of one saturation vector are compared to whichever entries match them in the other vector, not all the max- and min-related values are necessarily employed in the comparison. In fact, when both sets have, for example, a max-related value in some ic column of row A, the corresponding min-related values in the B rows are never compared. While an index that does not always consider all available arguments might be viewed as incomplete, by comparing only the *closest* arguments in the CSATVs we greatly reduce the effect of cardinality. If, for example, we compared all the $CSATV_A$ and $CSATV_B$ values of ic4 in sets X and Y, we would see that they are

$$\frac{|+0.00 - +0.00| + |-6.34 - -4.37|}{6.34 + 4.37} = \frac{1.97}{10.71} = 18.4\%$$

different with respect to their $WICCV_4$ values (I added the differences between the min-related values and max-related values and divided that sum by the sum of the distance between $CSATV_{B_4}$ and $CSATV_{A_4}$ for each pcset). Considering that these two sets are maximally similar with regard to their 4-cycle segmentation (for a hexachord and a tetrachord), this difference, occurring solely as a product of their difference in cardinality, seems rather extreme. I therefore chose to omit the $CSATV_B$ comparison when both parallel arguments of the two $CSATV_A$ rows are related to the same extreme (min or max).

As mentioned, the sum of the differences between $CSATV_A(X)$ and the corresponding min- or max-related values in either row of $CSATV(Y)$ are not *necessarily* the same as the differences between $CSATV_A(Y)$ and $CSATV(X)$.[24] This was illustrated in step 1 of Example 16. In order to obtain the same value from a comparison of X to Y and Y to X, it is therefore necessary to add all the difference values together, creating a composite that reflects both comparisons (step 2 in Example 16).

This dual comparison produces a context-free similarity index that has, in large part, dealt with the problem of comparing sets with different cardinalities *a priori*. However, an even greater degree of cardinal-neutrality

Set class	ICV								
X[012678]	<420243>	$CSATV_A$:	<	-3.65	+2.40	+0.00	+0.00	-3.65	-0.00>
		$CSATV_B$:	<	+5.28	-9.52	-7.64	-6.34	+5.28	+3.60>
Υ[0369]	<004002>	$CSATV_A$:	<	+0.00	+0.00	-0.00	+0.00	+0.00	-0.00>
		$CSATV_B$:	<	-4.37	-4.37	+6.44	-4.37	-4.37	+2.40>

Step 1: Compare the vectors, creating a two-part difference vector:

$CSATV_A(X) : CSATV_{row}(\Upsilon)$ =	0.72 +	2.40 +	6.44 +	0.00 +	0.72 +	0.00	= 10.28
$CSATV_A(\Upsilon) : CSATV_{row}(X)$ =	5.28 +	2.40 +	7.64 +	0.00 +	5.28 +	0.00	= 20.60

Step 2: Add together the values in the difference vectors: = **30.88**

Step 3: Add together all the numerical distances between $CSATV_A$ and $CSATV_B$ for each set:

(These values are also given in Example 15)

$\sum CSATV(X)$ =	8.93 +	11.92 +	7.64 +	6.34 +	8.93 +	3.60	= 47.36
$\sum CSATV(\Upsilon)$ =	4.37 +	4.37 +	6.44 +	4.37 +	4.37 +	2.40	= 26.32
							= **73.68**

Step 4: Divide the sum from step 2 by the sum from step 3 to complete the SATSIM function:

$CSATSIM(X, \Upsilon)$ = 30.88/73.68 = **0.42**

EXAMPLE 16: $CSATSIM(X, \Upsilon)$ COMPARISON OF [012678] AND [0369]

is attained by dividing the sum of the differences by the combined totals of the two vectors (step 4 in Example 16).[24] This cardinality adjustment better allows us to compare CSATSIM(X, Y) and CSATSIM(S, T) where $\#S$ or $\#T$ are not necessarily equal to $\#X$ or $\#Y$.[25] CSATSIM is formally defined in Example 17.

$$\text{CSATSIM}(X, Y) = \frac{\sum_{n=1}^{6} \left(\left|\text{CSATV}_A(X)_n - \text{CSATV}_{row}(Y)_n\right| + \left|\text{CSATV}_A(Y)_n - \text{CSATV}_{row}(X)_n\right|\right)}{\sum_{n=1}^{6} \left(\left|\text{CSATV}_A(X)_n - \text{CSATV}_B(X)_n\right| + \left|\text{CSATV}_A(Y)_n - \text{CSATV}_B(Y)_n\right|\right)}$$

Where $\text{CSATV}_A(X)_n$ represents the numerical value found in CSATV_A's nth entry for pcset X. *Row* is a function that determines which row of the CSATV to use.

Function *row*:
If $\text{CSATV}_A(X)_n$ is a max-related value and $\text{CSATV}_A(Y)_n$ is also a max-related value, then the function *row* returns row A ($\text{CSATV}_A(X)_n$ is compared to $\text{CSATV}_A(Y)_n$); otherwise, *row* returns row B ($\text{CSATV}_A(X)_n$ is compared to $\text{CSATV}_B(Y)_n$).

EXAMPLE 17: FORMAL DEFINITION OF THE CYCLIC SATURATION SIMILARITY INDEX—CSATSIM(X, Y)

The value 0.42 that CSATSIM yields comparing [012678] and [0369] (step 4 in Example 16) represents the very great differences in their ic1, ic3, and ic5 content and cyclic fragmentation. It also represents the congruence of values in the ic4 and ic6 columns and the more moderate difference in their ic2 columns, returning a value which indicates that 42% of the WICCV values are equivalent. As mentioned, the number zero indicates an equivalence relation, while the number one indicates maximal dissimilarity. As in many similarity measures, however, maximal dissimilarity is impossible to achieve because no two set classes are completely different with regard to their interval class occurrences and cyclic distribution.

There are a number of equivalences yielded by CSATV and, more narrowly, CSATV_A.[26] When two different set classes can be represented by the same CSATV, we will call them CSATV Z-related, following Allen Forte's well-known ICV protocol. There is only one (traditional) ICV Z-

pair that is also a CSATV *Z*-pair: the two all-interval tetrachords, 4-*Z*15 [0146] and 4-*Z*29 [0137]. Since they both feature one and only one of each interval class, each *n*-cycle is segmented to the same degree. There are a few cases of $CSATV_A$ equivalences: they are all relations between set classes of different cardinalities (this is obviously impossible in ICV *Z*-relations), and they are all *n*-sets. The list of $CSATV_A$ *Z*-relations is provided in Example 18 (the ICVs and complete CSATVs are also shown). Despite the very small number of *Z*-relations found using $CSATV_A$, there is one case of a *Z*-triple and one of a *Z*-quadruple. These *Z*-relations are all independent of my particular weighting constant. While changing WEIGHT clearly affects the CSATSIM values, these cases of CSATSIM equivalence will always remain invariant because the values in $CSATV_A$ represent minimal saturation, maximal saturation, or one cyclic fragment away from minimal or maximal saturation. Just as these CSATV *Z*-relations persist with all values for WEIGHT, so will the distribution of relatively large and small values across all pcset comparisons. For these reasons, it seems unnecessary to demonstrate CSATSIM using different WEIGHT constants.

* * *

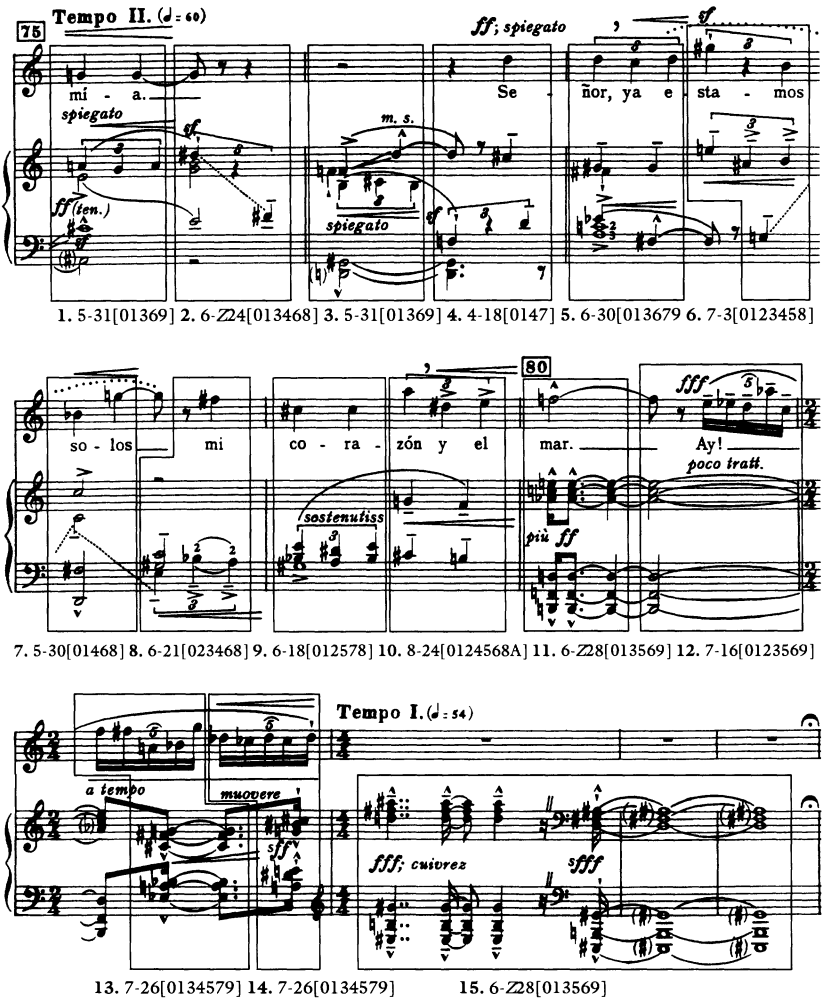
We will now return to the third song of Luigi Dallapiccola's *Quattro Liriche di Antonio Machado*, the final section of which is provided in Example 19. My segmentation (shown in Example 19) is rather simple, drawing upon the piece's conservative metrical structure. I treat each half-measure as a separate group, with only a few exceptions—mostly because of notes held over briefly from one group to the next. The pc content of each of the fifteen groups is labeled using both prime form and Forte designation. Three pairs within the groups present members of the same set class: numbers 1 and 3 (set class 5-31), 11 and 15 (6-*Z*28), and 13 and 14 (7-26). The remaining groups are all members of different set classes.

Example 20 provides a comparison matrix listing the CSATSIM values between all possible pairs of these fifteen groups. The smallest non-zero number in the matrix is 0.072, found between groups 5 and 11 (or, trivially, 5 and 15); the largest number in the matrix is 0.539, between groups 9 and 10. To provide a frame of reference, I have underlined all the CSATSIM values in the matrix that are higher than the average CSATSIM value for all #4 to #8 SCs (underlined values thus represent less than average similarity).[27] Of course, using this average as the dividing point for similar versus dissimilar sets is arbitrary. Rather than the average value for all comparably-sized set classes, I might just as

Set Classes Forte #CSATV(both rows)

CSATSIM group #1	(1/2/3/4/5/6-cyclic sets) (CSATV *Z*-quadruple)							
A [ ]	0-1	<	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00 >
		<	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00 >
B [0]	0-1	<	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00 >
		<	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00 >
C [0123456789a]	11-1	<	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00 >
		<	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00 >
D [0123456789ab]	12-1	<	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00 >
		<	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00 >
CSATSIM group #2	(all-interval tetrachords; non-cyclic sets) (CSATV *Z*-pair)							
A [0146]	4-ZZ15	<	+1.20	+1.20	+1.20	+1.20	+1.20	-1.20 >
		<	-3.17	-3.17	-5.24	-3.17	-3.17	+1.20 >
B [0137]	4-Z29	<	+1.20	+1.20	+1.20	+1.20	+1.20	-1.20 >
		<	-3.17	-3.17	-5.24	-3.17	-3.17	+1.20 >
CSATSIM group #3	(4-cyclic sets) (CSATV$_A$ *Z*-pair)							
A [04]	2-4	<	+0.00	+0.00	+0.00	-0.00	+0.00	+0.00 >
		<	-1.20	-1.20	-1.20	+1.20	-1.20	-1.20 >
B [048]	3-12	<	+0.00	+0.00	+0.00	-0.00	+0.00	+0.00 >
		<	-2.64	-2.64	-2.64	+4.37	-2.64	-1.20 >
CSATSIM group #4	(3/6-cyclic sets) (CSATV$_A$ *Z*-triple)							
A [036]	3-10	<	+0.00	+0.00	-0.00	+0.00	+0.00	-0.00 >
		<	-2.64	-2.64	+2.64	-4.37	-2.64	+1.20 >
B [0369]	4-28	<	+0.00	+0.00	-0.00	+0.00	+0.00	-0.00 >
		<	-4.37	-4.37	+6.44	-4.37	-4.37	+2.40 >
C [0134679a]	8-28	<	+0.00	+0.00	-0.00	+0.00	+0.00	-0.00 >
		<	-10.70	-8.32	+7.60	-5.14	-10.70	+2.40 >
CSATSIM group #5	(3/6-cyclic sets) (CSATV$_A$ *Z*-double)							
A [01369]	5-31	<	+1.20	+1.20	-0.00	+0.00	+1.20	-0.00 >
		<	-5.24	-5.24	+6.44	-4.37	-5.24	+2.40 >
B [0134679]	7-31	<	+1.20	+1.20	-0.00	+0.00	+1.20	-0.00 >
		<	-8.32	-8.32	+6.44	-5.14	-8.32	+2.40 >
CSATSIM group #6	(2/4/6-cyclic sets) (CSATV$_A$ *Z*-pair)							
A [02468]	5-33	<	+0.00	-0.00	+0.00	-0.00	+0.00	-0.00 >
		<	-6.44	+6.44	-6.44	+4.37	-6.44	+2.40 >
B [02468a]	6-35	<	+0.00	-0.00	+0.00	-0.00	+0.00	-0.00 >
		<	-8.93	+11.92	-7.64	+6.34	-8.93	+3.60 >

EXAMPLE 18: SPECIAL CSATSIM EQUIVALENCE GROUPS (CSATV OR CSATV$_A$ *Z*-RELATIONS)



EXAMPLE 19: LUIGI DALLAPICCOLA, *QUATTRO LIRICHE DI ANTONIO MACHADO*, SONG NUMBER 3, LAST TEN MEASURES

reasonably have chosen the average—or the mean—for the group at hand. Alternatively, it would also seem reasonable to examine the distribution of values and look for naturally occurring dividing points. Each of these methods is arbitrary, of course, and the manner in which one interprets data from similarity indices is as subjective as one's choice of a

particular index.[28] Like all analytical decisions, these should be made with a musical context in mind.

In this comparison matrix, there is a slight dearth of values right around 0.2. Since that largely conforms to what I hear as a close relation, it will serve as the cutoff for what I will call a close relation. Another such ebb occurs at 0.1, but this is a very high standard for relatedness, met by only four of our pairs (discounting the set class duplications). We might reasonably say that set pairs yielding a number smaller than 0.1 deserve to be called very closely related.

CSATSIM is particularly useful in the last subsection of this song, which begins in m. 80 and encompasses groups 11 through 15 (and the piano part of which was shown in Example 1). This portion of the CSATSIM comparison matrix has been excerpted in Example 20. There are two set-class duplications in this short span of music: between groups 11 and 15, which present the only sets that trivially match because of Dallapiccola's use of a particular row (we'll discuss that in a moment), and between groups 13 and 14, which present the same set class *despite* being the products of two rows from different row classes.[29] The values in Example 21 indicate that all the sets are similar, using our definition, and that the first or last set and number 12 are very similar indeed. The CSATV similarity between sets 11 and 12 results from their rather close affinity to 3-cycle sets and their relative lack of ic 1-, 2-, 4- and 5-cycle segments.[30] Sets 13 and 14 also contain heavily segmented 1-, 2-, and 5-cycles, and are close to neither the minimal nor maximal saturation of 3-cycle adjacencies. This explains the very close resemblance of sets 11 and 12, and the somewhat less close, but still similar, relations among the other sets in this short excerpt.

Example 22 compares the sets presented in groups 1 through 6 (from the first three measures of Example 1) in another excerpt from the overall matrix with CSATSIM values *greater* than 0.2 underlined. As one can see from the amount of underlining, quite a few set pairs are dissimilar under our criteria. Notice, however, that sets 1, 3 (trivially), and 5 are all similar to each other and are dissimilar only to sets 2, 4, or 6. In more musical terms, the first halves of these measures are all similar to each other. The second halves of these measures are a bit harder to generalize. Sets 2, 4, and 6 are both dissimilar to the odd numbered sets and to each other. They (and particularly set 2) are, however, relatively much more closely related to all the sets at the *end* of the song.

An examination of the first six groups' cyclic subsets reveals the distinguishing feature of group 2. Its most salient cyclic subset is a complete 4-cycle, expressed compositionally as a close-position augmented triad in the pianist's right hand. Group 6 is the only other one that also embeds a

	1. 5-31 [01369]	2. 6-Z24 [01368]	3. (=1.) 5-31 [01369]	4. 4-18 [0147]	5. 6-30 [013679]	6. 7-3 [0123458]	7. 5-30 [01468]	8. 6-21 [023468]	9. 6-18 [012578]	10. 8-24 [0124568A]	11. 6-Z28 [013569]	12. 7-16 [0123569]	13. 7-26 [0134579]	14. (=13.) 7-26 [0134579]	15. (=11.) 6-Z28 [013569]
1.	0.000														
2.	0.365	0.000													
3.	0.000	0.365	0.000												
4.	0.251	0.276	0.251	0.000											
5.	0.120	0.219	0.120	0.282	0.000										
6.	0.376	0.217	0.376	0.235	0.304	0.000									
7.	0.384	0.224	0.384	0.255	0.292	0.246	0.000								
8.	0.300	0.203	0.300	0.168	0.288	0.197	0.226	0.000							
9.	0.315	0.178	0.315	0.231	0.203	0.252	0.254	0.270	0.000						
10.	0.501	0.401	0.501	0.359	0.527	0.440	0.228	0.294	0.539	0.000					
11.	0.206	0.147	0.206	0.245	0.072	0.229	0.207	0.216	0.224	0.453	0.000				
12.	0.182	0.194	0.182	0.190	0.154	0.208	0.266	0.189	0.263	0.422	0.089	0.000			
13.	0.296	0.144	0.296	0.177	0.274	0.209	0.135	0.099	0.286	0.234	0.198	0.191	0.000		
14.	0.296	0.144	0.296	0.177	0.274	0.209	0.135	0.099	0.286	0.234	0.198	0.191	0.000	0.000	
15.	0.206	0.147	0.206	0.245	0.072	0.229	0.207	0.216	0.224	0.453	0.000	0.089	0.198	0.198	0.000

Ordinal numbers at the top and left represent the temporal placement of the SCs in the music; Forte numbers and prime forms are provided just beneath the ordinal numbers at the top of the page. Underlined CSATSIM values correspond to SC pairs that are less similar than average (numbers greater than 0.267).

EXAMPLE 20: CSATSIM COMPARISON MATRIX FOR SET CLASSES IN THE DALLAPICCOLA EXCERPT

	11. 6-Z28 [013569]	12. 7-16 [0123569]	13. 7-26 [0134579]	14. 7-26 [0134579]	15. 6-Z28 [013569]
11.	0.000				
12.	0.089	0.000			
13.	0.198	0.191	0.000		
14.	0.198	0.191	0.000	0.000	
15.	0.000	0.089	0.198	0.198	0.000

EXAMPLE 21: CSATSIM COMPARISON MATRIX FOR THE LAST FIVE GROUPS IN THE DALLAPICCOLA EXCERPT (MM. 80–84)

	1. 5-31 [01369]	2. 6-Z24 [013468]	3. 5-31 [01369]	4. 4-18 [0147]	5. 6-30 [013679]	6. 7-3 [0123458]
1.	0.000					
2.	0.365	0.000				
3.	0.000	0.365	0.000			
4.	0.251	0.276	0.251	0.000		
5.	0.120	0.219	0.120	0.282	0.000	
6.	0.376	0.217	0.376	0.235	0.304	0.000

EXAMPLE 22: CSATSIM COMPARISON MATRIX FOR THE FIRST SIX GROUPS IN THE DALLAPICCOLA EXCERPT (MM. 75–77). CSATSIM VALUES LARGER THAN 0.20 ARE UNDERLINED

4-cycle, and *that* 4-cycle is much less salient, both in abstract terms because it occurs within a larger set (and therefore represents a smaller percentage of the set's overall content), and in compositional terms because it is not in as close a position and also because the pitches that form the augmented triad (C, E, G♯) are not struck simultaneously (as was the augmented triad in group 2). The first and third sets, by contrast, are comprised of a complete 3-cycle ([0369]) plus one other note, yield-

ing as many ic3s as possible in a pentachord. The fifth set is formed by a complete 3-cycle (found in the pianist's left-hand part) plus a tritone from another 3-cycle (we can also think of this as three complete 6-cycles), and the fourth set can be described as an almost-complete 3-cycle with an additional note from another 3-cycle. Sets 1, 3, and 5 are the only ones with complete 3-cycles, and CSATSIM finds them the most closely related of the lot. Sets 2 and 6 are the next-most closely related pair among the first six sets; they are also the only other pair that mutually embed a complete *n*-cycle of the same type (4-cycles). The connections between the beginning and end of Example 19 are also quite strong. In Example 20, we can see that the last five groups are similar to most of the other groups in the excerpt. Additionally, there are very clear set-class connections between the beginning and ending of the song.[31]

The cyclic saturation similarity measure—or any similarity measure that uses the weighted interval-class cycle vectors or cyclic saturation vectors as data—is particularly helpful for a piece such as this one. Dallapiccola used row-classes that divide into *Z*-related hexachords, and, as discussed at the beginning of this article, he compositionally realized the hexachords in ways that take particular advantage of the available cyclic adjacencies. The set class of the last chord is 6-*Z*28; the set class of the penultimate chord is, of course, its complement, 6-*Z*49. Looking back to their 3- and 4-cycle distribution in particular (see Examples 3, 7, and 10), you will recall that the final chord (6-*Z*28) contains a complete 3- and 4-cycle, while the penultimate chord contains two incomplete 3-cycles and three incomplete 4-cycles. It is exactly because of the last set's affinities to both 3- and 4-cycles that CSATSIM finds it relatively closely related to each of the sets at the beginning of the excerpt.[32]

My interval-class cycle vector and weighted cycle vector were designed to illustrate the subtle, yet important, differences between just these sorts of sets by focusing on their cyclic subsets. Anyone so inclined can fine tune these vectors and their employment to reflect other notions of set-class resemblance. One could, for example, simply replace my weighting constant with a different number, replace my particular weighting system with a different algorithm, or compare the cyclically-derived data using a different similarity index. Alternatively, one could use a different (wider or narrower) assortment of cycles[33] or even apply such cycle-based notions of labeling and similarity to other spaces.[34] And, if performing an analysis where you want to equate or more sharply differentiate *Z*-related or complementary sets, there are other systems of resemblance in the waters. My hope is that these cyclic additions to the mix will help diversify the ways in which we think about and compare pitch-class sets in atonal music analysis.

Appendix: CSATSIM Value Group Matrix

Castrén (1994) provides what he calls "value group matrices" to help compare a variety of similarity measures with his own RECREL. I also find such a statistical summary helpful in understanding the range of values that is both possible and average for a given measure of resemblance. Following his format, I provide a similarly drawn CSATSIM (with weighting constant 1.2) value group matrix representing comparisons among all SCs larger or equal to dyad classes and smaller or equal to decachord classes. Each cell of Example 23 represents a statistical summary of the values possible using CSATSIM(X, Υ) where X is a pcset of the X-axis cardinality and Υ is a pcset of the Υ-axis cardinality (or vice versa). The upper left corner of each cell is the lowest CSATSIM value possible in the value group;[35] the upper right corner is the highest CSATSIM value possible in the value group; the middle left value is the lowest non-zero CSATSIM value (this value is not included in Castrén's value group matrices); the lower left corner contains the average of all the values in the group; and the lower right corner contains the number of distinct CSATSIM values in the value group.

The matrix exhibits some patterns. The smallest *average* CSATSIM value between $\#X$ and $\#\Upsilon$ SCs tends to occur where $\#X = \#\Upsilon$. The next smallest average value tends to occur between SCs of $\#X$ and $\#\overline{X}$ (i.e., the compliment of cardinality X). The average CSATSIM value tends to increase the greater the difference between either $|\#\Upsilon - \#X|$ or $|\#\Upsilon - \#\overline{X}|$. Examine, for example, the #3 : $\#\Upsilon$ comparisons (i.e., all CSATSIM comparisons that include a trichord) on the matrix. The smallest average comparison (and also the smallest maximal and minimal CSATSIM values) is between SC pairs X and Υ where $\#X = \#\Upsilon = 3$. The next smallest average value in this case happens to be between #3 and #4 SCs. The third closest average, however is between #3 and #9 SCs. The largest average CSATSIM comparison in the #3 value group is between #3 and #6, which constitutes the largest possible difference between either #3 or #9 and any other size SC.

With the exception of the single case cited above in Example 18 (SC 5-31:SC 7-31), CSATSIM does not suggest equivalence relations between complementary SC pairs, but it does reflect the similar degree of potential cyclic distribution among SCs of complementary cardinalities. It bears reiteration that these values are specific to the weighting value 1.2. Changing the weighting alters the actual CSATSIM values, but does not alter the distribution of relatively large and small values through all value groups.

	#2		#3		#4		#5		#6		#7		#8		#9		#10	
#2	*0.000* 0.333 0.278	0.333 2																
#3	0.000 0.103 0.359	0.597 18	*0.000* 0.149 0.295	0.509 19														
#4	0.107 0.107 0.413	0.574 51	0.000 0.082 0.297	0.561 157	*0.000* 0.091 0.258	0.557 81												
#5	0.231 0.231 0.520	0.730 102	0.148 0.148 0.368	0.638 204	0.062 0.062 0.287	0.582 406	*0.000* 0.015 0.232	0.567 176										
#6	0.226 0.226 0.579	0.795 195	0.206 0.206 0.431	0.733 324	0.080 0.080 0.351	0.634 757	0.000 0.066 0.254	0.650 1067	*0.000* 0.010 0.201	0.604 477								
#7	0.217 0.217 0.540	0.779 156	0.123 0.123 0.399	0.660 272	0.099 0.099 0.327	0.629 606	0.000 0.013 0.250	0.581 768	0.011 0.011 0.232	0.617 1148	*0.000* 0.011 0.227	0.524 418						
#8	0.069 0.069 0.502	0.727 128	0.000 0.154 0.382	0.673 237	0.000 0.042 0.328	0.601 557	0.058 0.058 0.278	0.579 702	0.040 0.040 0.271	0.580 979	0.056 0.056 0.256	0.552 727	*0.000* 0.028 0.258	0.543 299				
#9	0.116 0.116 0.450	0.688 56	0.056 0.056 0.348	0.670 102	0.055 0.055 0.345	0.668 245	0.048 0.048 0.313	0.584 320	0.125 0.125 0.312	0.593 408	0.042 0.042 0.294	0.615 335	0.043 0.043 0.296	0.583 269	*0.000* 0.147 0.292	0.543 48		
#10	0.021 0.021 0.441	0.743 30	0.066 0.066 0.379	0.657 49	0.119 0.119 0.386	0.647 130	0.157 0.157 0.364	0.696 170	0.133 0.133 0.375	0.703 209	0.155 0.155 0.368	0.745 175	0.111 0.111 0.366	0.686 138	0.064 0.064 0.346	0.669 53	*0.000* 0.210 0.359	0.705 13

EXAMPLE 23: CSATSIM VALUE GROUP MATRIX (WHERE WEIGHT = 1.2)

Notes

The author would like to thank Joseph Dubiel for his very careful reading of this manuscript and for his helpful comments.

1. At the end of this article, we will be dealing with the full texture of this excerpt.

2. Marcus Castrén, *RECREL: A Similarity Measure for Set-Classes*, Studia Musica 4 (Helsinki: Sibelius Academy, 1994), 8.

3. SATSIM(2) is a particular case of the SATSIM measure that examines only cardinality 2 subset classes (this is tantamount to saying interval classes).

4. Robert D. Morris, "A Similarity Index for Pitch-Class Sets," *Perspectives of New Music* 18 (1979–80): 445–60; Eric Isaacson, "Similarity of Interval-class Content Between Pitch-class Sets: The IcVSIM Relation," *Journal of Music Theory* 34 (1990): 1–28; and "Issues in the Study of Similarity in Atonal Music," *Music Theory Online* 2,7 (1996); Michael Buchler, "Relative Saturation of Subsets and Interval Cycles as a Means for Determining Set-Class Similarity" (Ph.D. diss., University of Rochester, 1997): 75–79; Damon Scott and Eric Isaacson, "The Interval Angle: A Similarity Measure for Pitch-Class Sets," *Perspectives of New Music* 36, no. 2 (Summer 1998): 107–42; John Rahn, "Relating Sets," *Perspectives of New Music* 18 (1979–80): 483–97; Castrén, 101–43; and David Lewin, "A Response to a Response: On Pcset Relatedness," *Perspectives of New Music* 18 (1979–80): 498–502.

5. My SATSIM measure is even less discriminating than the other ic-based measures since, in addition to *Z*-related set classes, it also cannot distinguish complementary set classes.

6. Some notable examples include: Tore Ericsson, "The IC Max Point Structure, MM Vectors and Regions," *Journal of Music Theory* 30,1 (1986): 95–111; Robert D. Morris, *Composition with Pitch Classes: A Theory of Compositional Design* (New Haven: Yale University Press, 1987), 128–35; George Perle, *Twelve-Tone Tonality*, 2d ed. (Berkeley: University of California Press, 1996), 7–11; and Dave Headlam, *The Music of Alban Berg* (New Haven: Yale University Press, 1996), 13–31.

7. In abstract algebra, this is called a "ring."

8. The former is the union of {0, 4, 8} and {1, 5, 9} and the latter is in the union of {0, 4, 8} and {2, 6, a}

9. Ericksson, 96–100.

10. Buchler, 37–48.

11. "Too heavily" is, of course, entirely subjective; when constructing these vectors and the associated similarity index, I never wanted the weighted value for any single *n*-cycle segment to equal more than twice the sum of the weighted values of any possible smaller *n*-cycle segments that total the same number of ic *n*. For example, {4} should be weighed more heavily than {1,1,1,1}, {2,1,1}, {2,2}, or {3,1}, but not by a factor of 2 or more. This meant moving toward a weighting scaling that increased more gradually than an exponential scale.

12. This resembles the system to compare differences used by Richard Teitelbaum, "Intervalic Relations in Atonal Music," *Journal of Music Theory* 9 (1965): 72–127.

13. While I believe that WEIGHT is easiest to understand as a recursive function, it can also be modeled as a simple formula. Let *n* represent the number that is being weighted and *k* represent the weighting constant.

$$\text{WEIGHT}(n) = \frac{k}{k-1}[k^n - 1]$$

My sincere thanks to Panayotis Mavromatis for constructing this equation.

14. In the case of a tie between minima- and maxima-related values, CSATV_A will show the comparison to the maxima-related value.

15. CPSATV(6-Z28) might also have been expressed as percentages: <27%, 22%, 84%, 31%, 27%, 67%>.

16. Buchler, 51–62.

17. Since a high CSATSIM(X, Y) value indicates a lack of similarity among pcsets X and Y, one might more properly call this a "dissimilarity index," though it could easily be transformed into a true similarity index by subtracting CSATSIM(X, Y) values from 1.

18. While symmetry might not be a necessary pre-condition for such a measure, all context-free similarity measures of which I am aware exhibit this property.

19. I am using integers in this demonstration for the sake of simplicity; actual CSATVs will not primarily contain integer values.

20. In fact, one could arrive at the same vector totals by adding the distances between $\min(w, c, i)$ and $\max(w, c, i)$ for $i = 1$ to 6.

21. Function row, described formally in Example 17, provides a mechanism for determining which row in CSATV(Υ) should be compared to $\text{CSATV}_A(X)$ for each of the six arguments.

22. The term "difference vector," which refers to an ordered list of differences between two vectors being compared is introduced in Isaacson 1990, 16.

23. I.e., $\text{SATV}_A(X) : \text{SATV}_{row}(\Upsilon) \neq \text{SATV}_A(\Upsilon) : \text{SATV}_{row}(X)$ for all values of X and Υ.

24. Again, saturation vector totals are obtained by adding together the distances between the rows for every argument of the vector (step 3 in Example 16 or Example 14).

25. The basic construction of CSATSIM is similar to Morris's ASIM index. (Morris, 1979–80.)

26. If two sets are equivalent in only the CSATV_A vector and not the CSATV_B vector, they will still yield the value 0.00 from CSATSIM.

27. In our analytic example, we are only presented with four- through eight-note sets. The CSATSIM values for sets with four to eight elements ranges from 0.00 for the most similar pairs to 0.65 for the most dissimilar pairs. The average CSATSIM value for cardinalities four through eight is 0.267. Average and extreme CSATSIM values for all cardinality comparisons are provided in an appendix to this paper.

28. And, ultimately every similarity index and method of labeling is also ontologically subjective.

29. Unfortunately, the interesting play of invariance that facilitates such similarity is not within the scope of this paper.

30. One might imagine that sets 11 and 12 are also very closely related because the former (6-*Z*28) is abstractly embedded in the latter (7-16). While it is true that none of 6-*Z*28's three (abstract) seven-note

supersets (7-16, 7-28, and 7-32) significantly disturb the cyclic composition, an embedding/covering relation does not guarantee (or even necessarily imply) a close CSATSIM relation. For example, 6-*Z*3 [012356] is (abstractly) covered by/embedded in six different heptachord classes. 6-*Z*3 is related to one of its supersets (7-4 [0123467]) by a CSATSIM value of 0.070, whereas it is related to another of its supersets (7-16 [0123569]) by a CSATSIM value of 0.236.

31. There are also close SC connections from one song in the cycle to the next, despite the fact that different row classes are used in all but the first and last (fourth) songs of this cycle.

32. Of course, no relation that operates in pitch-class space can be influenced by a composer's particular spacing or instrumentation. Nonetheless, Dallapiccola's setting of the final chord (and also the piano chord in m. 80), which registrally segregates the embedded augmented triad from the embedded diminished seventh chord, makes these analytical results more vivid. By comparison, his setting of the penultimate—and also the antepenultimate—piano chords brings out their interval-class 1 and 6 content.

33. C.f., Morris 1987: 128–35.

34. For example, pitches (in P-space) or beat classes.

35. The value in the upper left corner is italicized if it is 0.000 and that number only represents the trivial case of one SC compared with itself (in cases where $\#X = \#Y$). If the upper left number is 0.000 and there is some CSATSIM *Z*-relation in the value group then the value is not italicized.