SOLVING SPECIFIC COMPOSITIONAL PROBLEMS WITH MP1

Sever Tipei

Assistant Professor, School of Music
University of Illinois

MP1 is a computer program for music composition, conceived as an instrument that can help the composer by performing logical operations. Once a certain musical situation is thoroughly defined, MP1 can produce "realizations", it can materialize abstract considerations into musical facts.

In an article on Pierre Boulez, György Ligeti (1958) describes the compositional process, as he sees it at work in "Structures Ia", as comprizing three stages: Decision I - choice of elements, relationships and operations-, Automatism - where elements and operations "are, as it were, fed into a machine, to be woven into structures automatically, on the basis of the relationship chosen", and Decision II - where the resulting "crude" structure is polished by "taking decisions in dimensions that are not employed mechanically". As an example of such dimensions, Ligeti mentions dynamics and register that could be the "left-over paramenters" to receive more attention during this third stage. It is obvious that a computer can succesfully perform the second stage, the Automatism. Moreover, since the third stage, Decision II, is necessary only because the first set of decisions were incomplete, postponing choices that will have to be eventually made anyhow, one can imagine that, on one hand, all decisions could be made at the beginning, operations established not only for one parameter at a time, but also between various sound dimensions and, on the other hand, a more complex Automatism could provide the final results. The latter case requires a more sophisticated and flexibl- program, one that is capable of handling as many sound parameters as the composer desires, one that is capable of considering the interraction and allowing a precise control over the details. This is, actually, what MP1 is meant to do, the contributionit could make to the writing of a musical piece. In short, to use the terminology proposed by Xenakis (1971), once an "outside-time algebra" or "outside-time architecture" has been defined, MP1 can take care of the "in-time algebra" or in-time musical architecture" - the actual realization of the abstract structure.

Another characteristic of MP1 is the fact that it approaches the Computer-Assisted Composition from a musical stand point, not form one outside the field of music. As opposed to the idea - which, undoubtedly, has its merits - that almost any logical or mathematical model can be translated into sounds, MP1 is built around the supposition that, for concrete musical problems, a logical or mathematical model can be found, an algorithm constructed, that will imitate as closely as possible the mental process required during the act of composing. In addition, MP1 is conceived as a coherent entity able to encompass a whole piece, not as a cluster of isolated routines that might solve local problems unrelated to the overall ensemble or operations. There is a backbone of basic operations in MP1 that are sustaining the program as a whole, that are perenial, while allowing new additions at any point. This way, MP1 is not a one-piece program or a one-composition class program. It accumulates abilities that can be put to work or left dormant at the user's discretion.

It goes without saying that MP1 is still a program "in progress" and, by its very nature, it will always be, as a whole, in an experimental stage. It should also be pointed out that, past a certain level, making a program flexible and cooperative might also mean making it mimick a particular style, introducing one composer's idiosyncracies that might turn-off another user, while dealing with more basic situations might accomodate a wider variety of styles and personalities. Bearing in mind the possible risk of transforming it into a too personalized a tool, the refining of existing algorithms was emphasized in more recent projects, where efforts were concentrated toward erasing the difference between computer-composed and composer-written fragments of pieces that combine both modalities.

A succint overview of MP1's main features is probably helpful at this time. They are founded on four premises:

1. the musical space is homogeneous and it can be described as a vector space with sound parameters as its dimensions.
2. equivalence relations modulo m can be defined on every element of the vector space's base.
3. a succession of values at the same parameter can be described either as a Markov chain, a Stochastic distribution or as a random occurence.
4. to compose means to make decisions by choosing from a range of possibilities.

The first premise illustrates the fact that MP1 takes into account a number of parameters between two (duration, pitch) and five (duration, pitch, amplitude, timbre, envelope), or even more (spatial distribution, for example). It also means that the same basic procedures will be used at all parameters. The main loop of the program runs a number of times equal to that of the sound parameters considered. A full run through the whole program coincides with the exhaustive definition of one sound.

The second premise is self-explanatory: It supposes the existence of a system of "octaves" (registers) and classes of elements at all parameters - not only pitch-classes, but also duration-, amplitude-, timbre-, envelope-classes. This makes the programming and the computations more convenient allowing, at the same time, a more precise control over different levels. It also seems to correspond to the way our perception works.

The third premise relates to the fact that sequential strings of events can be considered when the program neither implements a given distribution nor does it deal with random occurences. Initially, MP1 stared by handling almost exclusively probabilities and that stage still constitutes a fundamental layer. There, a preferred value (duration, or pitch, or amplitude, etc.) is heeded, as well as a register and the ambitus of the sound source. If the random option is not in effect, first level restrictions will intervene. The number of possibilities will be narrowed down by a diversity factor - or deviation from the preferred value - and by the degree of coherence desired at that moment. Actually, there are three variables that regulate the latter: coherence between the candidate value and the last n values already assigned at the same part (voice), coherence between the candiate value and the values already assigned at all other simultaneous parts, and coherence between the candidate value and the preferred value requested by the user. This type of restriction represents a feed-back mechanism that

compares a potential (candidate) value with the state of the system at a given moment. Its implementation requires two more features: memory and a way to carry out the comparison.

MP1's memory stores the last n choices (their precise number to be decided by the user) of values at every parameter. After a new choice is made, it is stored as the first element in the memory, the old first value takes the second place, the next to the last becomes last and the last one is "forgotten". As for the comparison itself, a formula of the type $e^{-\tau(\varphi_i-\varphi_p)^2}$ is used, where τ is a coeficient, φ_i the candidate value and φ_p is the preferred value or a value already assigned the candidate one is to be compared to. When the exponent of the expression equals 0, e=1, otherwise, the greater the difference between φ_i and φ_p , the smaller the value of the expression and the chance of the candidate value to be assigned. It is a rather effective way to produce qualitative judgements through quantitative measurements. The final decision is made by matching the probability of each candidate value with a random number: in this way an ellement of chance is introduced. Other members of the expression (not shown here) regulate the degree of variety allowed so that either a higher entropy or a more precise focusing on a given value is obtained. The case of Markov chains will be discussed later.

The forth premise needs no explanation; it only points out that clusters of local decisions add up to major ones similar to the choices a composer supposedly makes. As mentioned before, a choice is completed only when the entire program had run for all parameters of the same sound. It is then only that the printing subroutine is called in and a full description of a new sound is recorded in the print-out.

Coming back to the process of futher reducing the number of candidate values, a second level set of restrictions could be in effect. One of them deals with a minor problem, easy to solve in programing terms but, nontheless, taking care of a bothering detail: voice crossing. Through a variable and time limits set up by the composer, MP1 can allow it or avoid it. Since there is always the possibility for all parameters to share a subroutine, the notion of voice crossing can be extended to, for example, amplitudes: a part can not be assigned a sound that is louder or softer (or both) than some or all other simultaneous parts. Or, if extended to durations, it will forbid values larger or smaller than the ones already assigned at other parts, creating a steady, chordal type of texture.

A more complex and rewarding restriction is represented by the use of sieves. The concept was first introduced in music -to our knowledge- by Xenakis (1971) and represents a method of choosing, through logical operations $(\cup, \cap, -)$ applied to residual classes modulo m, a number of elements out of a continuum. A most obvious example is the selection of various scales out of the total chromatic: diatonic, pentatonic or any other collection of less than twelve pitches.

0	1	2	3	4	5	6	7	8	9	10	11	12	
A	Bb	B	C	C#	D	Eb	E	F	F#	G	G#	A	(1)
1	0	1	1	0	1	0	1	1	0	0	1	1	

Scale: A Minor Harmonic Sieve: $4n+3 \cup 3n+2 \cup 3n \cap 4n$

The procedure can be extended to a micro-tonal context:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	A+	Bb	B𝄫	B	C𝄭	C	C+	C#	C𝄲	D	D+	Eb	E𝄭	E
5	0	5	1	5	0	5	0	5	1	5	1	5	0	5

(2)

15	16	17	18	19	20	21	22	23	24
E+	F	F+	F#	F𝄲	G	G+	G#	G𝄲	A
0	5	1	5	1	5	0	5	1	5

+ = ¼ tone higher
𝄲 = 3/4 tone higher
𝄭 = ½ tone lower
𝄫 = 3/4 tone lower

Values can be entered directly in the DATA set if the sieves refer to traditional scales; for more complex situations, the computer can be very helpful - as Xenakis (1971) already suggested. If instead of simply entering 1 (for "pass",) as opposed to 0 (for "no-pass"), a larger variety of integers is used, one can create a "coloring" (Xenakis,(1971) giving some elements a better chance than others, or even simulate the relative frequency various degrees of the scale have in traditional music:

0	1	2	3	4	5	6	7	8	9	10	11	12
A	Bb	B	C	C#	D	Eb	E	F	F#	G	G#	A
4	0	1	2	0	3	0	3	1	0	0	1	4

(3)

Another example, a fragment of a microtonal scale (using ¼ tones) whose period encompasses 2½ octaves:

..84	85	86	87	88	89	90	91	92	93	94	95	96	97
..Eb	E𝄭	E	F𝄭	F	F+	F#	F𝄲	G	G+	G#	G𝄲	A	A+
..10	6	14	0	10	6	4	4	16	6	12	6	12	0

98	99	100	101	102	103	104	105	106	107	108	109	110 ..
Bb	D𝄭	B	G𝄭	C	C+	C#	C𝄲	D	D+	Eb	E𝄭	F ..
21	0	24	0	15	9	0	9	15	9	12	0	30 ..

(4)

Besides sieves for absolute values (pitches), MP1 can as easily handle sieves for intervals. They turn out to be useful not only when approximating a more or less traditional style (e.g. forbidding melodic tritones, avoiding as much as possible, but not completely, m7 and M7, etc.), but also when dealing with micro-tones. In a situation similar to the one illustrated in (2), it is easy to see how it may become almost impossible to perform (especially sing) large skips involving microtones:

(5)

Modifying only slightly the initial probabilities, a sieve that excludes certain intervals and establishes preferrences among the ones allowed, can correct such an awkward gesture.

For all practical reasons, sieves at the first parameter (duration) deal exclusively with intervals - although setting-up an absolute-time-values sieve, encompassing an entire piece, is a challenging idea. Through duration-sieves irregular subdivisions (1/3, 1/5, 1/7, etc.) can be organically included in the repertoire of values; the lowest common denominator of all subdivisions becomes the smallest time unit to be used in the piece. Since not all possibilities are always acceptable from the musical point of view, a sieve is necessary in order to select the ones to be used. For example, if ♩ ♩3 and ♩5 are used, the smallest time unit will be 1/60 of a ♩ ; however, durations represented by 31, 23 and 49, to point out only to a few, do not make sense when transcribed in traditional notation:

$\frac{31}{60}$ ♩ = $\frac{15+10+6}{60}$ ♩ = ♬3 ♬5 (6)

$\frac{23}{60}$ ♩ = $\frac{18+5}{60}$ ♩ = ♩.5 ♬3

$\frac{49}{60}$ ♩ = $\frac{15+24+10}{60}$ ♩ = ♬5 ♩.3

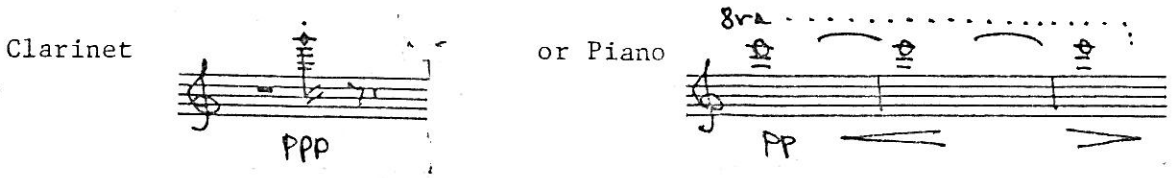
and they will have to be excluded from the list of possibilities if the composer wishes to keep the meter constant. A subroutine, of MP1, METER, makes sure that all note-values are compatible with the system of notation and with the hierarchies prescribed by a certain meter. Actually, what is passed through the sieve is not the whole duration, but just the remainder of a value following the last bar line or the last complete beat. Again, the elements of the sieve can be weighted so that not only certain values will be rejected, but preferences will be established among the accepted ones. The following fragment of a sieve:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	3

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	...	(7)
0	0	0	0	3	0	0	0	2	0	0	0	0	0	4	...	

allows only: ♩ (31): ♬ (16), ♩3 (21), ♩5 (25), ♩3 (11), 𝅘𝅥𝅯 (0, grace note); ♩5 (13) and precisely in that order of preference. When the sieve is as long as the bar-as it probably should be-a hierarchy may be established between beats (1 and 3 accepting more changes than 2 and 4, in a 4/4 meter), not only between subdivisions of the same beat.

Values at all parameters can be passed through both absolute-value or interval sieves. Also, the more sophisticated Multiple-Entry Sieves become useful at higher order parameters (timbre, envelope). They are sieves that operate when specific conditions are met, like the assignment of a certain value at a previous parameter pertaining to the same sound. In fact, a Multiple-Entry Sieve is a matrix correlating the values of two different parameters, a way of avoiding inconguous situations often produced by two or more sets of values assigned independently:



(8)

A third level of restrictions, narrowing even further the list of candidate values, is related to the third premise on which MP1 is based, the possibility of introducing Markov Chains. The program reads one or more patterns, strings of values - usually, but not necessarily - coordinated at all parameters, similar to motives or themes in traditional music. The frequency of single elements as well as that of groups of consecutive elements is recorded in a transformational matrix, the length of the group being associated with the degree of accuracy in reproducing the pattern. Candidate values are checked against the patterns stored in memory. If MP1 finds that one of them could be part of a pattern, the probability of that particular value is increased. Usually, the same candidate value could be part of more than one pattern at a time, especially if the degree of accuracy (i.e. the length of the Markov chain) is low. In that case, MP1 will favorize the pattern with the better chance of being reproduced at more parameters. Once an assigned value is recognized as part of a given pattern, the next candidate value which would continue that particular pattern is assigned a greater probability. If total accuracy in the reproduction of the pattern is desired, all other options of the program are short-circuited and the elements of the pattern are just copied one after the other.

A more complex situation is created when the degree of accuracy in reproducing a pattern varies during the reproduction of the pattern itself. In other words, when the degree of accuracy is stable for a time interval shorter than the pattern itself. Then, fragments of the pattern are copied exactly while other fragments are distorted. Similarly, a lower degree of accuracy will either succeed in reproducing disjunct fragments, or abandon the pattern after a few sounds and then, immediately start again with the same sounds or with elements that should not have been reproduced until a much later time, or with an unwanted transposition. In order to avoid such clumsy handlings of patterns, MP1 keeps tract of an ideal situation in which all the elements of all parameters in a pattern would be reproduced at the right time, compares it with the actual values assigned and increases the probability of those values that are the closest to the ideal situation. The result is that distortions can occur without jeopardizing the pattern as a whole.

The task of correctly reproducing a pattern is further complicated by the use of micro-tones. When both absolute value sieves and interval sieves are in effect, a distortion in the given pattern may easily lead to the wrong transposition of some intervals, introducing more micro-tones than originally planned, or to a dead end where no choice is possible. A solution is to introduce some margin of error in the handling of intervals. For instance, if only ¼ tone, m2, m3, P4 and m6 are allowed as intervals, instead of the sieve:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	0	0	0	1	0	0	0	1	0	0	0	0

15	16	17	18...
0	1	0	0 ...

(9)

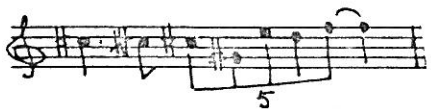
which will banish all other intervals, one could use the sieve:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	3	4	1	0	1	4	1	0	1	4	1	0	0	0

15	16	17	18...
1	4	1	0 ...

(10)

which will allow an error of ¼ tone around the desired intervals, establishing at the same time an unambiguous hierarchy among probabilities. This way, a skip of 9 quarter-tones is made possible in case it is the only solution, but its probability will be much lower than that of a P4 in case both are on the list of candidate values. Indeed, if the beginning of the pattern:

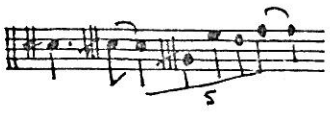


(11)

is distorted:

(12)

the composer might not be pleased to have five times more ¼ tones than initially planned; or even worse, certain sounds may not be permitted at all by the absolute-value sieve. If lets say A, E+ and D+ are not allowed, then the computer will have to stop after G and print an error message, while if the sieve in (11) is in effect, it will print the following sequence:



(13)

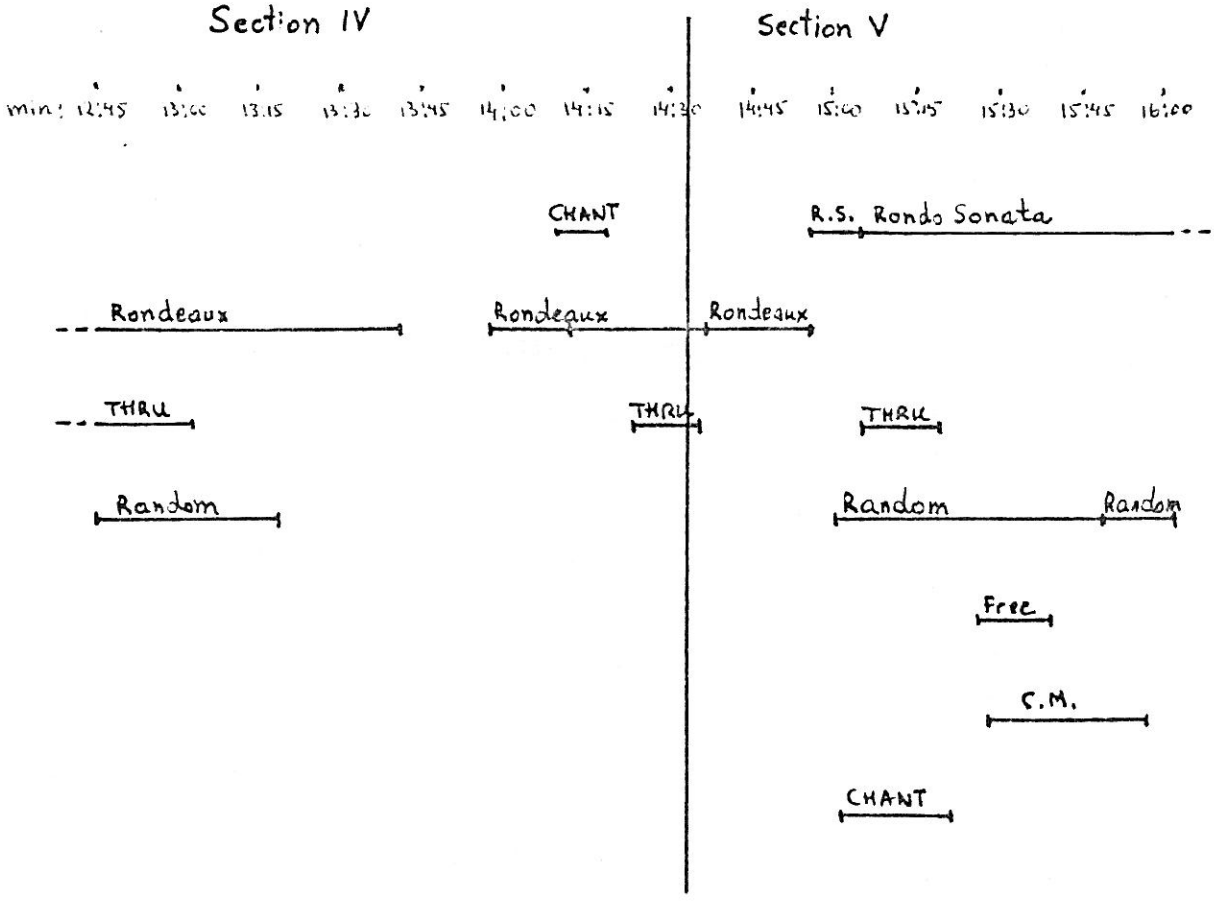
which is, undoubtedly, acceptable.

The vector space, the equivalence relations modulo m, the Markov chains, and, generally speaking, most of MP1's features could conceivably serve as a model for structures outside the domain of sound, or could be applied to music composition in a less conventional sense. Besides mentioning Multimedia events, an appropriate example is that of MP1 dealing not with sounds in a piece, but with a "macro-composition" where sections of the work are treated as sounds in a traditional piece. "Undulating Michigamme", a work for soloists and orchestra, contains seven types of musical form and texture, seven movements that are not heard sequentially, one at a time, but interrupt each other or are superimposed on each other in a way that tries to imitate the way thoughts are wandering through somebody's mind. According to how tightly organized they are, the seven types form a group structure:

-3	Conceptual Music	C.M.
-2	Free rhythmic improvisation	FREE
-1	Directed improvisation	CHANT
0	Chance (random) events	RANDOM
1	Through composed	THRU
2	Rondeaux (Renaissance Forme fixe)	RONDEAUX
3	Rondo-Sonata (tonal)	R.S.

(14)

The following figure shows a moment of high density when all seven types overlap:



(15)

Finally, one more comment on the esthetic implications of using MP1. At the end of this discussion, a valid question still remains: why add such features to a computer program, why try to erase the difference between a composer's work and the computer's product instead of writing music without the machine? A possible answer will have to include the observation that, compared to the human mind, a program of MP1's type is still faster and better in realizing stochastic distributions or dealing with complex sieves and matrices. Moreover, such a program represents a way of introducing some degree of chance as well as a way to accomplish that un-romantic idea of the composer's detachment from the material he is working with. If one remembers again Ligeti's (1958) description of the compositional process, according to it, the creative activity takes place mainly during the first phase, Decision I, while final touches are added during Decision II, a third stage giving the composer another opportunity to unleash his imagination. Although Ligeti never says it himself, it seems reasonable to go one step further and presume that Decision I deals mostly with concepts, as opposed to Decision II which is concerned mainly with craftsmanship. If a computer is made able to participate in the Decision II stage as well, an interesting distribution of roles may take place, with the composer involved in all three stages but being concerned especially with the abstract thinking, the machine providing the strictly local, routine work and sometimes, some of the final touches and the performer occupied with the in-time realization but also being asked to improvise within strict limits. An "Unholy Trinity" keeping a constant watch on the composer's ego.

References:

Ligeti, György 1958. "Pierre Boulez" in Die Reihe vol. 4, Universal Edition, p. 36
Xenakis, Iannis 1971. "Formalized Music", Indiana University Press, pp 164-169 and pp. 194-200.