This paper deals with spatial models of the auditory properties of speech sounds, in particular vowel space as conceived by Liljencrants and Lindblom (1972) and consonant space as discussed in Laver (1994). Computer models are used to explore the mathematical consequences of these spaces, which pose difficulties for phonetically based approaches to phonology. In each case, the problems with the spatial model point out the need to constrain and define the phonetic space with a phonological structure.

A number of recent papers in phonology have considered the possible role of the auditory and acoustic phonetic properties of segments in determining their phonological behaviour. This research has dealt in particular with contrast and neutralization, both active (expressed in phonological ‘processes’) and passive (expressed in inventory shapes). Steriade (1997) shows that a useful generalization about the neutralization of laryngeal features may be made by observing that the environments in which neutralization takes place are those in which there are fewer opportunities for the expression of phonetic cues that signal the values of those features. For example, a voicing contrast might be preserved only in contexts in which the following segment provides an opportunity to differentiate voice onset times. Flemming (1995) derives inventory shapes from interactions among constraints on the number and strength of auditory contrasts between segments. The need to maintain salient auditory distinctions is weighed against the need to have a certain number of segments in the inventory, and against the desire to minimize effort on the part of the speaker.

One of the chief difficulties faced by this phonetic approach to phonology is the problem of representing and quantifying the notion of auditory contrast. Acoustic differences between sounds can be measured on a spectrogram, but the differences that look significant on a spectrogram do not correspond at all precisely with the differences that sound significant to the human ear. To capture auditory—as opposed to acoustic—differences, a more abstract model is needed. One particularly promising approach is to treat segments as points (or areas) in a multidimensional auditory ‘space.’ Such a space would be abstract, in that its dimensions would correspond to auditory features rather than to physical dimensions, but it would also be quantifiable, because it would be subject to certain mathematical laws. This paper deals with two visions of auditory space. The first of these is a vowel space, proposed by Liljencrants and Lindblom (1972), within which vowels are moved away from one another in order to derive inventories with maximal contrasts. The second is a consonant space suggested by Laver (1994), in which segments are located closer together or farther apart based on their probability of being mistaken.

*Thanks to Keren Rice, Jean Balcaen, and Alexei Kochetov for comments, suggestions, and inspirations.
for one another. As we shall see, these models encode useful ideas, but their internal logic leads the theory in unexpected directions.

1. Vowel space

Liljencrans and Lindblom (1972) describe an attempt to predict the shape of vowel inventories from a mathematical formula for maximizing contrast. In their model, vowels increase their distinctiveness by moving away from one another within the available acoustic space, like equally charged subatomic particles, or strangers in an elevator. Vowel space, for Liljencrans and Lindblom, is defined by the frequencies of the first, second, and third formants; each formant becomes a dimension. The shape and size of this space are of course limited by the range of the human voice and ear; Liljencrans and Lindblom take the shape of their space from earlier work by Lindblom and Sundberg (1969:1971). They then compress the space into two dimensions by incorporating the (rather small) third-formant dimension into the second-formant dimension. The resulting two-dimensional space is described in linear mel units. The horizontal axis of the available vowel space (which corresponds to the first formant) extends from 350 mel to 850 mel (about 250 Hz to 750 Hz). The vertical axis (second and third formants combined) extends from 800 mel to 1700 mel at the left edge of the space, and narrows asymmetrically to a point (1150 mel) at the right edge. The top and bottom of the space are defined by the following two half parabolas:

\[
\text{top: } y = 1150 + 550 \sqrt{\frac{850 - x}{500}}
\]

\[
\text{bottom: } y = 1150 - 350 \sqrt{\frac{850 - x}{500}}
\]

In rough terms, lower \(x\) values correspond to higher vowels and higher \(x\) values to lower vowels; higher \(y\) values correspond to vowels that are farther front or less rounded and lower \(y\) values to vowels that are farther back or more rounded. Figure 1 (based on Liljencrans and Lindblom’s Figure 3) shows approximate positions for fourteen vowels.
Having defined a vowel space, Liljencrants and Lindblom then go on to model the placement of vowels within that space. Three to twelve vowel points “are evenly placed on a circle of radius 100 mel, with its center at [x] = 600 and [y] = 1200 mel” (Liljencrants and Lindblom 1972:842). (Figure 1 indicates this circle with a dashed line.) Using the metaphor of vowels as electrons, the ‘energy’ of the system is calculated—each vowel acts on each other vowel with a force equal to the inverse of the square of the distance between them. The energy (E) of the system is the sum of the forces generated by each pair of vowels. A computer program (written in FOCAL) then calculates for each vowel the change in E that would result from moving the vowel a set distance in each of “a number of directions, usually six” (Liljencrants and Lindblom 1972:842). The program chooses the direction that results in the lowest value for E, and continues to move the vowel in that direction until E no longer decreases or the edge of the vowel space is reached, whereupon it chooses a new direction. The whole procedure is repeated until no further reductions in energy
are obtained. It is not clear from Liljencrants and Lindblom’s description whether the vowels are moved simultaneously or in sequence—that is, whether the movement of the second vowel, for instance, is calculated using the original or the moved position of the first vowel. To move the vowels simultaneously makes more sense, as different sequential movements of the same vowels could produce quite different results.

Liljencrants and Lindblom’s program predicted the following shapes for inventories of three to twelve vowels:1

Liljencrants and Lindblom then go on to compare their model’s predictions with inventories reported in actual languages by other researchers, in particular Trubetzkoy (1929), Hockett (1955), and Sedlak (1969). The comparison is made

1In Liljencrants and Lindblom’s (10), ü and û are shown as separate vowels, and there is no œ. However, the ten-vowel inventory in their Figure 2 and Figure 4 has a schwa and only four close vowels, in positions similar to those of the close vowels in the seven- and eight-vowel inventories. I have restored the schwa and collapsed ü and û here based on these figures. Otherwise, Table 1 reflects the vowel symbols used in Liljencrants and Lindblom’s (3)-(12), and, as closely as possible, the relative positions they show the vowels.

Table 1.

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>u</td>
<td>i</td>
<td>u</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>u</td>
<td>u</td>
<td>i</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>u</td>
<td>i</td>
<td>u</td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Predicted inventories (based on Liljencrants and Lindblom’s (3)-(12) and Figure 2)
somewhat awkward by the fact that the attested inventories are described in phonemic
terms, while the predicted inventories are phonetic, and by the variety of symbols
used by the different writers. (Where, exactly, does Trubetzkoý’s Siang-Tang /i/ sit
in Liljencrants and Lindblom’s vowel space, or Hockett’s Mandarin /i/?!) However,
some of the merits and faults of the computer model do come readily to light. The
program’s inventories show much of the same sort of symmetry as naturally
occurring inventories, and its choices for three- and six-vowel systems in particular
owe very closely indeed with at least some of the attested systems.

The preference of the program for vowels at the edge of the space is
understandable in light of its basic principle of repulsion; however, this preference is
not taken to such extremes in the attested inventories. The program does not
generate more than one ‘non-edge’ vowel in any system, and it generates no non-
edge vowels at all in systems of fewer than ten vowels. In particular, the program
never generates the sound [o], which appears frequently in attested inventories with
seven or more vowels. Meanwhile, the same centrifugal tendency that leads the
program to posit too few mid central vowels also produces too many high vowels.
While the program generates five close vowels in nine-, eleven- and twelve-vowel
systems, none of the attested inventories has more than four vowels at any height.
This fact is not readily predictable from the shape of the vowel space alone; the
space appears to be able to accommodate five high vowels as easily as four mid
vowels, and there are languages with four mid vowels.

The facts can be made to follow, however, from phonological rather than
phonetic considerations—and indeed this has been done, by Rice (1995). Rice argues
for the use of only two monovalent vowel-place features, Peripheral (which
collapses Labial and Dorsal) and Coronal. This allows for at most four phonemic
vowels at any given height: one with no features (i or u), one with Coronal only (i),
one with Peripheral only (u or u), and one with both place features (ü). In languages
such as Turkish, in which there is a contrast between vowels that surface
phonetically as [u] and [u], the [u] may be analyzed phonemically as /i/, since it
behaves phonologically as if unmarked for place. One possible way of curbing the
computer program’s fondness for high vowels, then, might be to restrict the
phonetic quest for contrast through phonological universals: if there are only two
place features in Universal Grammar, then one cannot have more than four vowels at
any height, no matter how wide the available vowel space may be. This approach
fits better with the data than simply stating that five or more high vowels would be
too acoustically close to be permitted, for the latter approach would in turn lead one
to predict fewer mid and low vowels. Highly symmetric large inventories such as the
attested examples in Table 2 would not be expected:

---

2Liljencrants and Lindblom cite the following exemplars of the inventories in Table
2: (a) Trukese, Thai, Temaoyan and Mazahua Otomi, and English (Hockett 1955);
Kannada, Banda-Linda, Karen, and English (Sedlak 1969); (b) Estonian (Hockett); (c)
Tibetan (Sedlak).
Table 2.

<table>
<thead>
<tr>
<th>(a) Liljencrants and Lindblom’s (9a)</th>
<th>(b) Liljencrants and Lindblom’s (9dα)</th>
<th>(c) Liljencrants and Lindblom’s (12a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i i u</td>
<td>i ü u</td>
<td>i ü i u</td>
</tr>
<tr>
<td>e ə o</td>
<td>e ö o</td>
<td>e é ø o</td>
</tr>
<tr>
<td>æ a ø</td>
<td>æ ø a</td>
<td>æ ø a</td>
</tr>
</tbody>
</table>

**Symmetrical nine- and twelve-vowel inventories**

Boersma (1997) remarks that Liljencrants and Lindblom’s model tends to predict systems with disproportionately many place contrasts for their height contrasts. Boersma’s claim is that “Symmetries [in inventories] are the language-specific results of general human limitations on the acquisition of perceptual categorization and motor skills” (Boersma 1997:1). A more straightforward approach might be to say that symmetry results from the human tendency to categorize, to build systems. If Universal Grammar forces us to categorize vowels using two place features, then the feature system explains why we can have no more than four high vowels, and why we can have as many as four low ones. Boersma uses an array of gestural and perceptual constraints to describe the interplay of symmetries and gaps in inventories: symmetry is inherent in the constraint set as a whole, and gaps arise from local hierarchical interactions between constraints. However, similar generalizations may be captured more simply through universal limits on featural representations. Rather than positing and permuting constraints on the perception of individual feature values, one can simply state what features are available. Symmetry then arises from the mathematical possibilities for combining features, and from the fact that human beings think in terms of abstract categories. The availability of phonetic space can still be used to explain why a language is more likely to have four high vowels than four low ones, but phonetic possibilities must be filtered through the systematizing principles of language before they can become phonological realities.

Returning to Liljencrants and Lindblom’s algorithm, we can find even in their very phonetic model an analogue of phonological feature specifications. Recall the initial placement of the vowels: “As an arbitrary initial condition, the points are evenly placed on a circle of radius 100 mel, with its center at $[x] = 600$ and $[y] = 1200$ mel.” Is it any wonder, then, that the program refused to produce an $[ø]$? Each vowel, repelled by the others, would tend to move straight outward from the circle in a path as direct as the direction-choosing algorithm allowed. A different set of starting points could drastically change the results. Suppose one vowel were placed in the centre of the circle, with the rest evenly spaced along the circumference as before. The centre vowel, under equal force from each direction, would not move at all. Only when the shape of the space itself disrupted the symmetry of the surrounding vowels would the balance of forces be lost and the centre vowel disturbed. Thus not all of the centrifugal tendencies of the program are inherent in the notion of repulsion; these tendencies arise from the initial placement of the vowels as well.

The program in Appendix A operates on the same principles as Liljencrants and Lindblom’s program, and was written to test the observations above. The program is written in Turing, a language similar to Pascal developed at the University of Toronto. (See Holt (1993) for a comprehensive explanation of Turing syntax.) Like
Liljencrants and Lindblom’s program, it treats vowels as if they were electrons repelling one another with their electrical charges. However, it does differ from their program in a few aspects, mostly to take advantage of the greater computing power afforded by the quarter century that has passed since their study was published. Rather than moving vowels by set amounts in set directions and evaluating which move is best, this program calculates the exact force exerted on each vowel by each other vowel, moves the vowels simultaneously, and then calculates again. This program does not consider the energy of the whole system, but moves each vowel according to the magnitude and direction of the forces acting upon it.

The program works as follows: The procedure `plot` translates coordinates in the vowel space into coordinates on the screen. The arrays `top` and `bottom` contain the y-coordinates of the two parabolas that bound the vowel space. Each vowel is represented as a record containing the vowel’s current position, the next position to which it will be moved, its name, and the colour in which it is drawn on the screen. The procedure `setvel` evaluates the force of each other vowel in the system on a given vowel, and computes the next position for the vowel accordingly. Vowels are not allowed to move beyond the edge of the vowel space; if the next computed position for a vowel is out of bounds, it is adjusted to fall directly on the edge rather than beyond it. The force of each vowel is multiplied by a constant (here 100), so that the vowels move at a visible rate. The procedure `move` moves a vowel to its determined next position and plots it on the screen; this function is kept separate from `setvel` so that the new position of each vowel can be calculated before any vowel is actually moved. Since the old position of the vowel is never erased from the screen, each vowel leaves a trail indicating where it has been; this makes the vowels easier to see. The main body of the program prompts the user for the number of vowels to be modelled, and for the name and starting position of each vowel. It then draws the vowel space, places the vowels within it, and enters an infinite loop in which the positions of the vowels are continually updated. (Eventually the vowels move far enough apart that their effect on one another is no longer perceptible, or the edges of the space prevent them from moving any further.)

As expected, the initial placement of the vowels has a great effect upon the output of the program. For example, Liljencrants and Lindblom report that their program prefers the three-vowel inventory in Figure 2 (a). So does the program in Appendix A—if the initial positions of the vowels are generally as in Figure 2 (b):

**Figure 2.**

(a) ![Preferred three-vowel inventory and corresponding starting positions](image)
However, the three vowels could be equally placed along (or within) Liljencrantz and Lindblom’s arbitrary circle in any number of different ways. The inventories in Figure 3, which resemble other attested inventories, can be derived by the program given the appropriate initial placements:

Figure 3.
(a) (b)

```
i
\hat{\circ}
a
```

(c) (d)

```
i
\hat{\circ}
a
```

*Other attested inventories, with starting positions*

Given the right input, the program can produce any of the observed three-vowel inventories, and probably most, if not all, of the attested inventories with more vowels. However, given a ‘wrong’ input such as the one in Figure 4 (b), it could produce the unlikely system of Figure 4 (a):

Figure 4.
(a) (b)

```
e \hat{o} o
```

*Generating an unattested inventory*

If the purpose of the program is to derive phonetic inventories, then the input starting positions may be seen as the phonological specifications of those vowels. In Figure 2, we could say, it is the feature Peripheral that tells the program that [u] has a somewhat lower F2 than the other vowels, and Low that tells it that [a] has a
somewhat higher F1; the task of the program is to work out exactly how much is somewhat. The program, under this view, is doing the work of phonetic enhancement rules in assigning specific phonetic values to phonological contrasts. The input to the program may be constrained by universal statements about the markedness of features; for example, the inventory in Figure 4 could be ruled out by saying that no language may have both Coronal and Peripheral unless it has Low. This approach leaves the program with a considerably lesser role than Liljencrants and Lindblom seem to have imagined for it, but their program as it stands (or even with some of the modifications they suggest) cannot be a shaper of inventories. From the results they present, the program undergenerates—they show only one predicted inventory shape for each number of vowels. On the other hand, if we consider the possibilities that result from different inputs (even merely from rotations of their circle of starting points), the program overgenerates. It produces more systems than are attested. Either the set of well-formed inputs must be defined, which solution leads back to the phonological view, or the algorithm itself must be extensively revised.

Some caution should be used, of course, in making inferences about Liljencrants and Lindblom’s program from the behaviour of the program in Appendix A. Since Liljencrants and Lindblom are not entirely clear about some aspects of their program, I have not attempted to replicate its workings precisely, but rather have tried to make as consistent an implementation as possible of the conceptual model behind it. One particular point of departure to note is their direction-choosing algorithm. Since they do not specify any of the “number of directions, usually six” that their program tests, or explain under which circumstances this number is not six, the program in Appendix A computes an exact direction based on the forces operating in the system, rather than choosing among a set of approximate directions. Because of this difference, it is impossible to tell from the program in Appendix A whether Liljencrants and Lindblom’s program could also have generated vertical and horizontal inventories such as those in Figure 3 (c) and Figure 4. In the program in Appendix A, the middle vowel in such an inventory—the schwa—is acted upon in precisely opposite directions by the vowels on either side of it. If the outer vowels are equally spaced, the schwa will not move at all; if they are not, the schwa will move toward the farther one, but the schwa will not depart from the axis on which all three vowels lie. In Liljencrants and Lindblom’s program, however, movement is motivated by the energy of the system as a whole. Moving the schwa in either direction perpendicular to the three-vowel axis would make it farther from each of the other two vowels, and thus decrease the overall energy. However, moving the same distance in either perpendicular direction would yield the same decrease in energy, and so there is no non-arbitrary way to choose between the two. Since Liljencrants and Lindblom do not report which directions their program tries, or what it does when opposite directions fare equally well, it is impossible to say what their program would do with an input of three collinear vowels. However, it seems reasonable to speculate that either their program would fail to generate the attested Figure 3 (c), or else it would also generate the unattested Figure 4 (a).

Another issue to consider is how thoroughly the analogy with electrons is realized in the model. In the program in Appendix A, the movement of each vowel at any step in the program is determined solely by the positions of the other vowels...
at that moment; the velocity imparted upon the vowel in earlier steps has disappeared entirely. In other words, the vowels, unlike electrons, have no momentum. A second version of the program, given in Appendix B, makes the vowels behave more like physical objects by allowing them to keep the velocity imparted to them at each step. The new velocity imparted by the set of forces acting on a vowel at each stage does not replace the old velocity, but is rather added to it. The most important consequence of this refinement of the model is that there is no longer any guarantee that the movement of a vowel at any given moment has the result of reducing the energy of the system as a whole. One vowel, being repelled by a second, may accelerate so greatly that it cannot slow down soon enough to avoid coming too close to a third vowel that lies in its path. The presence of momentum leads the revised program to generate very different inventories from those of the original version. For example, the initial positions shown in Figure 5, which yield the unlikely [u, æ, a] in the original program, produce [i, u, a] in the revised version.

Figure 5.

Vowel 1, as expected, moves leftward, toward the [u] position, while vowels 2 and 3 move rightward and away from each other. When 2 and 3 hit the top and bottom boundaries of the space, they continue to move rightward along the edges. Because the space is slightly asymmetrical, vowel 2 reaches the [a] position before vowel 3. The momentum of vowel 3 causes it to continue moving rightward, even though it is now much closer to vowel 2 than to vowel 1. Only when vowel 3 is almost touching vowel 2 is the force of vowel 2 enough to cancel out the momentum of vowel 3. Vowel 3 then moves very rapidly leftward until it hits the left edge of the vowel space at a point just below vowel 1. Vowels 1 and 3 then repel each other; vowel 1 moves straight upward and vowel 3 moves downward. When the system finally becomes stable, vowel 1 is [i], vowel 3 is [u], and vowel 2 is [a]. The resulting inventory is, of course, a very common one, but the manner in which the program arrives at it is rather capricious. The behaviour of the revised program shows that strengthening the analogy of electrons can only weaken the model’s ability to maximize contrast, as the presence of momentum allows vowels to move in counterproductive directions.

One logical way out of the problems with Liljencrants and Lindblom’s program would be to create an algorithm independent of any initially specified positions for the vowels. Instead of moving vowels by trial and error or in imitation of electromagnetic forces, the program would simply calculate the optimal set (or sets) of positions for a given number of vowels. Given a finite space and a finite number of points to place in that space, the program would find the arrangement(s)
of points that would result in the lowest total energy. Such an approach would allow
the model to make purely contrast-based predictions, independent of phonology and
physics. Not only the nature, but even the number of the solutions such a program
would find would be determined only by the shape of the space and the number of
vowels—it is entirely possible that there would be more than one optimal
arrangement for a given number of vowels. (Consider the hypothetical case of three
vowels in a circular vowel space: as long as the vowels are evenly spaced along the
circumference, they can be rotated through an infinite number of positions without
increasing the total energy.)

The main practical problem with evaluating vowel systems in this fashion is
the computational complexity involved. The number of inventories to be considered
increases greatly with small increases to the number of vowels in the inventory or to
the number of positions to be considered for each vowel. Furthermore, the number
of these positions increases greatly with small decreases in the distance between
positions. A truly thorough evaluation of the possibilities for even a small
inventory would take a great deal of computing time using this method. However, a
rough estimation of the ideal five-vowel inventory has been made. This inventory
was calculated by a script written within the database program FileMaker Pro. Given
the distance between possible vowel positions (which defines the precision of the
search), the program first calculates all the positions it needs to consider. It then
examines all possible five-vowel inventories made up of subsets of those positions,
calculating the ‘energy’ of each inventory as it goes. If the energy of the system the
program is currently evaluating is the lowest it has seen so far, the program stores
the inventory in order to compare it with subsequent possibilities. For this test, the
program was told to consider vowel positions at 125 mel units apart, of which 26
may be fit into Liljencrants and Lindblom’s vowel space. Using these positions, it
is possible to construct 65,780 unique five-vowel inventories. To evaluate these
inventories took the computer (a Power Macintosh 7100/80) approximately four and
a half hours. The inventory with the lowest energy is shown in Table 3:

| i | u | æ | a |

*Table 3. The five-vowel inventory with the lowest ‘energy’*

This is not a frequent inventory among the world’s languages. Liljencrants and
Lindblom mention three languages with inventories of a similar shape: Tabassaran
and Kyuri (both Caucasian languages; the latter is also known as Lezghian) have [i,
ü, u, e, a] (Trubetzkoy 1929), and Huichol (Uto-Aztecan) has [i, i, u, e, a] (Hockett
1955). Of the inventories surveyed by Maddieson (1984), the closest to the predicted
pattern are Papago (Uto-Aztecan) [i, i, u, ç, a], and Acoma (Keresan) [i, i, u, e, ą].

The inventory predicted by Liljencrants and Lindblom’s program, given in
Table 4 as they present it in their (5), has a slightly higher total energy than the one
predicted by the FileMaker program.
Table 4.

\[
\begin{array}{ll}
i & \text{u} \\
\varepsilon & \text{a} \\
\end{array}
\]

Five-vowel inventory predicted by Liljencrants and Lindblom

However, this inventory is closer in shape to the extremely common five-vowel inventory in Table 5:

Table 5.

\[
\begin{array}{ll}
i & \text{u} \\
\text{e} & \text{a} \\
\end{array}
\]

Widely attested five-vowel inventory

The relative infrequency of the inventory chosen by the FileMaker program suggests that the need to maximize distance is seldom the pre-eminent concern reflected in the structure of vowel inventories. Liljencrants and Lindblom’s program, with its arbitrary ‘phonological’ input, fared better; again the logical way to refine our predictions about phonetic inventory shape seems to be to eliminate the arbitrariness by using phonological universals.

The precise formulation of a phonological approach to cross-linguistic preferences in inventory shapes is, of course, a complicated matter which can be fully worked out only in the larger context of phonological theory. However, there is no shortage of possible approaches. Rice and Avery (1993) observe that, among both vowels and consonants, fewer places of articulation are contrasted in the more sonorous segments. Just as languages tend to have fewer low vowels than high vowels, they also tend to have fewer liquids than nasals, and fewer nasals than obstruent stops. Rice and Avery account for this parallelism through the feature geometry: both sonorants and vowels are distinguished from obstruents by the presence of an SV (Sonorant Voice or Spontaneous Voice) node, and distinguished from one another by the various dependents of this node. The more sonorous a segment is, the more SV structure it has. Consequently, low vowels are inherently more complex than high vowels, and so have less ‘room’ to accommodate the additional structure required for place distinctions. With this feature system, the inventory in Table 3 could be characterized as more marked than the inventory in Table 5 (and thus dispreferred) because it contains the extra structure imposed by the presence of two low vowels. Rice and Causley (1998) add another facet to markedness by proposing a constraint against fully underspecified segments such as high central vowels, which are neither marked as low nor marked for place. Thus the inventory in Table 3 could be dispreferred either for its complex low vowels or for its nondescript [u].

A related view of markedness is offered by Béjar (1998), who uses a weight metric to derive feature specifications from inventory shape. According to this metric, whenever one subset of the inventory is distinguished from its complement set by the assignment of a monovalent feature (as per Dresher’s (1998) Successive Binary Algorithm), the subset marked by the presence of the feature must be the smaller of the two. Under this view, it would be possible to relate the desire to
reduce markedness in the system as a whole to a desire to make the smaller subset in each complementary pair as small as possible. Thus in both Table 3 and Table 5, low vowels would be a marked subset, because they are fewer than their non-low counterparts, but the marked subset in Table 5 is smaller than the marked subset in Table 3. Of course, merely minimizing the size of each marked subset in an inventory would not necessarily reduce the complexity of the whole, because it would increase the number of features required. However, if the number of different features used in a system is also considered as a measure of complexity, then Table 5 may be preferred over Table 3 for its symmetry. Once the low vowel [a] has been dealt with, the remaining four vowels in Table 5 form a tidy rectangular pattern that can be fully and logically described with two features. It is not so easy to find symmetry within Table 3. Cutting off [u] does leave a similar rectangle to the one in Table 5, but why should [u] be separate from the other high vowels? Béjar’s algorithm for feature assignment implies preferences both for symmetry and for smaller sets of marked items; although these preferences sometimes conflict with each other, it can be argued that Table 5 satisfies either of them better than Table 3 does. Whatever phonological mechanism is used to choose the inventory in Table 5 over the one in Table 3, it is clear that something more abstract than the range of the human voice is at work here. Underlying the phonetic shape of any inventory is a phonological structure.

In addition to pointing out the usefulness of phonological specifications, the FileMaker program raises other questions about the idea of maximal contrast. The program found only one optimal system; the ‘energy’ of this system was $4.43 \cdot 10^{-5}$, while the energy of Liljencrants and Lindblom’s predicted five-vowel system is approximately $5.06 \cdot 10^{-5}$. How sensitive are human speakers to differences in contrastiveness? By how much do the energy levels of two systems have to differ for one of them to be preferred over the other? When more than one inventory shape is attested, as is the case for systems of all sizes, how much of the variation is attributable to the presence of alternatives which are more or less equally contrastive, and how much to the other requirements with which the need for contrast interacts? How frequent or infrequent need an attested inventory be to justify describing it as theoretically ‘preferred’ or ‘dispreferred’? In order to incorporate a phonetically based notion of contrastiveness into the theory, it is necessary to understand more fully the nature of the model itself as well as the phonological considerations that constrain it.

2. Consonant Space

The difficulties inherent in working with consonant space are even more basic than those met with in vowel space in the previous section. There is as yet no clear notion of what consonant space looks like, or even of how many dimensions it has. While Liljencrants and Lindblom’s organization of vowel space according to the frequencies of the first three formants certainly represents a simplification of phonetic reality, it is at least a clearly reasonable approximation. Consonants, however, cannot be characterized with such ease; in connected speech, their differences are more acoustically salient on the vowels they abut than on themselves. Furthermore, consonants do not fall into continua as readily as do vowels; for
instance, one can produce a continuous vowel sound that gradually moves from [i] to [a], passing through all points in between, but one cannot produce an analogous continuum of, say, voiced stops from [b] to [G]. Even if such an articulatory continuum were possible, it would not necessarily correspond to an acoustic or auditory continuum; it is possible, for example, that [k] and [p] are auditorially closer to each other than either is to [t].

One attempt at quantifying the shape of consonantal space comes from Laver (1994), who offers Table 6 as “an initial attempt to give a global suggestion of auditory distance (and hence of acoustic dissimilarity) between segment-types representing the consonantal phonemes of English (RP)” (Laver 1994:392).

Table 6.

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>t</th>
<th>k</th>
<th>f</th>
<th>s</th>
<th>j</th>
<th>v</th>
<th>b</th>
<th>d</th>
<th>g</th>
<th>m</th>
<th>n</th>
<th>w</th>
<th>z</th>
<th>l</th>
<th>h</th>
</tr>
</thead>
</table>
| p | 25 | 15 | 65 | 70 | 90 | 95 | 85 | 80 | 95 | 95 | 30 | 60 | 70 | 65 | 60 | 70 | 75 | 85 | 85
| t | 20 | 85 | 65 | 80 | 85 | 75 | 90 | 90 | 45 | 20 | 35 | 75 | 65 | 75 | 75 | 70 | 85 | 85 | 85 | 85
| k | 80 | 85 | 95 | 90 | 85 | 85 | 85 | 35 | 45 | 20 | 65 | 70 | 55 | 60 | 80 | 80 | 85 | 90 | 90 | 85 | 85
| f | 25 | 65 | 70 | 20 | 65 | 70 | 65 | 75 | 90 | 85 | 90 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| s | 70 | 65 | 35 | 25 | 70 | 65 | 80 | 75 | 85 | 85 | 85 | 90 | 85 | 90 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| j | 70 | 65 | 35 | 25 | 70 | 65 | 80 | 75 | 85 | 85 | 85 | 90 | 85 | 90 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| v | 60 | 70 | 90 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| b | 30 | 25 | 30 | 30 | 60 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80
| d | 30 | 60 | 30 | 60 | 80 | 80 | 80 | 75 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90
| g | 55 | 65 | 55 | 55 | 65 | 85 | 75 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85
| m | 20 | 20 | 55 | 75 | 85 | 85 | 75 | 75 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90
| n | 25 | 65 | 70 | 80 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90
| l | 35 | 70 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85
| w | 30 | 30 | 30 | 30 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85
| j | 50 | 45 | 85 | 90 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| l | 40 | 85 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95
| h | 1 | 85 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95 | 95

Auditory distances between consonants (based on Laver’s Figure 13.1)

Here distance between two segments represents specifically the improbability of confusing them; thus [p] and [z], which are 95 units apart, are much less likely to be confused than [p] and [k], at 15 units apart. The distances are on a theoretical scale of 0 (identity) to 100 (absolute dissimilarity); as it turns out, no two segments are closer than 15 units apart or farther than 95 units apart. Laver claims that “it would be hypothetically possible to locate all segment-types in multidimensional auditory space” (Laver 1994:391). However, the set of distances in Laver’s table (which was “developed by the author from subjective auditory impressions (supported by some theoretical assumptions)” (Laver 1994:393)) cannot possibly correspond to a set of points in \( n \)-dimensional Euclidean space.
Imagine a set of points that do inhabit Euclidean space. Any three points A, B, and C in this set will be coplanar; unless they are collinear, they will define a triangle \( \Delta ABC \). The distance between each pair of points (AB, BC, CA) will be the length of one side of the triangle. Now consider the line segments AB and BC, which meet at point B. If they meet at an angle of 180°, then A, B, and C are collinear, and the length of CA is equal to the sum of the lengths of AB and BC. If AB and BC meet at a narrower angle, then CA will be correspondingly shorter. Under no circumstances can CA be longer than AB + BC.

If the ‘auditory space’ represented in Laver’s table were subject to the axioms of Euclidean geometry, we could expect any three consonants to be the points of a possible triangle. However, this is not the case. Consider the set of distances Laver posits for the segments [\( \delta \)], [b], and [k]:

Table 7.

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta )</td>
<td>50</td>
<td>95</td>
</tr>
<tr>
<td>b</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Distances between \([\delta, b, k]\)*

These consonants cannot be the points of a triangle \( \Delta \delta bk \), because the distance from \([\delta]\) to \([k]\) (95) is greater than the sum of the distances from \([\delta]\) to \([b]\) and \([b]\) to \([k]\) (50 + 35 = 85). Nor is this group of consonants unique; Laver’s table contains 51 such impossible triads, which were found by the Prolog program listed in Appendix C. A predicate called *corresp* enables the program to match items occupying corresponding positions in two different lists. The predicate *distance* encodes Laver’s table; the predicate *impossible* describes the conditions under which three points cannot coexist in Euclidean space.

When queried for all values for A, B, and C such that *impossible*(A, B, C) is true, the program yielded the results summarized in Table 8. (As written, the program actually finds each triad twice—once as A, B, C and once as C, B, A.)

Table 8.

<table>
<thead>
<tr>
<th>bd</th>
<th>dm</th>
<th>bm</th>
<th>dn</th>
<th>g</th>
<th>n</th>
<th>gn</th>
<th>b</th>
<th>m</th>
<th>d</th>
<th>( \theta )</th>
<th>f</th>
<th>v</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>bd</td>
<td>d</td>
<td>g</td>
<td>d</td>
<td>t</td>
<td>z</td>
<td>g</td>
<td>t</td>
<td>n</td>
<td>b</td>
<td>( \delta )</td>
<td>f</td>
<td>l</td>
<td>v</td>
</tr>
<tr>
<td>bg</td>
<td>m</td>
<td>d</td>
<td>k</td>
<td>t</td>
<td>d</td>
<td>t</td>
<td>( \delta )</td>
<td>g</td>
<td>w</td>
<td>a</td>
<td>b</td>
<td>( \theta )</td>
<td>d</td>
</tr>
<tr>
<td>bg</td>
<td>n</td>
<td>d</td>
<td>k</td>
<td>( \delta )</td>
<td>f</td>
<td>( \theta )</td>
<td>( \delta )</td>
<td>j</td>
<td>w</td>
<td>( \eta )</td>
<td>b</td>
<td>( \delta )</td>
<td>5</td>
</tr>
<tr>
<td>bk</td>
<td>m</td>
<td>d</td>
<td>m</td>
<td>n</td>
<td>g</td>
<td>k</td>
<td>n</td>
<td>k</td>
<td>t</td>
<td>d</td>
<td>f</td>
<td>v</td>
<td>f</td>
</tr>
<tr>
<td>bk</td>
<td>d</td>
<td>n</td>
<td>t</td>
<td>g</td>
<td>m</td>
<td>b</td>
<td>m</td>
<td>t</td>
<td>d</td>
<td>g</td>
<td>n</td>
<td>f</td>
<td>v</td>
</tr>
</tbody>
</table>

*Impossible triads from Table 6*
If Laver’s table of distances is correct, then auditory space is non-Euclidean in some way—for example, the dimensions of this space may be bent through still other dimensions. There may well be more departures from Euclidean geometry in Laver’s table than are revealed in Table 8; the program did not check for impossible quartets (sets of four consonants which can be neither coplanar nor vertices of a tetrahedron), quintets, sextets, and so on. If the notion of auditory distance is to be put to phonological use (as it is, for example, in Flemming (1995)), then we need a clearer picture of the space in which that distance is to be measured.

Departing from Euclidean geometry is not inherently a bad thing; however, as it represents a drastic expansion of the logical possibilities of the notion of space itself, it is not a step to be taken lightly. Laver makes an even more fundamental departure from Euclidean space in his next table of distances, but this departure is acknowledged and explained. Laver’s table 13.2 shows a table of ‘distances’ based on the likelihood that one consonant would be misheard as another by an automatic speech-recognition system (Laver 1994:394). In this table, distance is not commutative—A is not necessarily as far from B as B is from A. For example, [D] is more likely to be mistaken for [v] than [v] for [D], and so the distance from [D] to [v] is only nine units, while the distance from [v] to [D] is 16.

We now have pairs of points that cannot exist in one-dimensional Euclidean space, but we also have an experimentally tested reason for them. The situation might be modelled by imagining that [v] and [D] are indeed points on a line, but that this line is curved or tilted through a second dimension so that [D] is ‘higher’ than [v]. The auditory similarity of the two consonants is then expressed not in terms of distance, but as the time in which one can cover that distance by applying a fixed amount of force. Moving ‘uphill’ from [v] to [D] thus takes longer than moving ‘downhill’ from [D] to [v]. However, there would have to be more than one ‘downhill’ for the model to reflect the larger system of relationships. Consider the distances for [f], [d], and [b], which are summarized in Table 9.

<table>
<thead>
<tr>
<th>Seg. 1</th>
<th>Seg. 2</th>
<th>1 to 2</th>
<th>2 to 1</th>
<th>Higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>d</td>
<td>19</td>
<td>13</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>7</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>f</td>
<td>14</td>
<td>13</td>
<td>f</td>
</tr>
</tbody>
</table>

Distances between [f, d, b]

In each pair of consonants in (14), the distance from the first segment to the second is longer than the distance from the second to the first, and so in the spatial model the second member of each pair would be represented as higher than the first. Thus [f] is higher than [b], which is higher than [d], which is higher than [f]. Not only is distance not commutative; height is not transitive. In order to capture this fact without creating a paradox, the dimension along which [f] is higher than [b] must be perpendicular to the direction specified as ‘downward’ in at least one of the other pairs. Such a model would account for the asymmetries of confusability, but it would also leave us with a segment space that consists of a yet unknown number of dimensions, distorted through at least two other dimensions, and supplied with a law
of gravity. (A remaining perplexity for the spatial model is the fact that the 'distance' from each consonant to itself is greater than zero. Since the 'distance' from A to B is really a measure of how unlikely A is to be heard as B, the distance from any segment to itself simply indicates the overall susceptibility of that segment to misperception. However, this piece of information has no clear analogue in the spatial model.) Clearly the limitations of this very complex space must be better understood before it can be put to phonological use.

3. Conclusions

Both Liljencrants and Lindblom’s algorithm for modelling vowels as electrons and Laver’s system of distances between consonants contain useful notions for a theory of the geometry of auditory space. However, both of these approaches have mathematical consequences which their authors have not fully explored. Liljencrants and Lindblom’s vowel program is too dependent upon its arbitrary starting points, which could, if their arbitrariness were eliminated, form the heart of a phonological account. Laver’s vision of distance removes auditory space from the purview of Euclidean geometry, creating an unspecified number of dimensions whose significance is left unclear. (For example, the shape of Liljencrants and Lindblom’s vowel space correctly suggests that there is less room for different places of articulation among low vowels than among high vowels; however, it is impossible to see the corresponding asymmetry between sonorants and obstruents reflected in Laver’s consonant space, because the space itself is impossible to visualize.) In each case, further clarification of the nature of segment-space and the forces that operate within it could lead to a more robust theory, from which generalizations about contrast might be made to follow mathematically.

References


Appendix A: Vowel space program (Turing)

procedure plot(x: real, y: real, c: int)
    drawdot(round(((x-349)/501)*maxx), round(((y-499)/1201)*maxy), c)
end plot

var top: array 350..850 of real
var bottom: array 350..850 of real
for lcv: 350..850
    top(lcv):=1150+550*sqrt((850-lcv)/500)
    bottom(lcv):=1150-650*sqrt((850-lcv)/500)
end for

type vowel: record
    position: record
        x: real
        y: real
    end record
    newpos: record
        x: real
        y: real
    end record
    c: int
    name: string
end record

procedure setvel(var v: vowel, var s: array 1..* of vowel, n: int, p: int)
    var dist: real
    var distx: real
    var disty: real
    v.newpos.x:=v.position.x
    v.newpos.y:=v.position.y
    for lcv: 1..n
        if lcv not=p then
            distx:=v.position.x - s(lcv).position.x
            disty:=v.position.y - s(lcv).position.y
            dist:=sqrt(distx**2+disty**2)
            v.newpos.x:=v.newpos.x+(100*distx/(dist**3))
            v.newpos.y:=v.newpos.y+(100*disty/(dist**3))
            if v.newpos.x < 350 then
                v.newpos.x:=350
            end if
            if v.newpos.x > 850 then
                v.newpos.x:=850
            end if
            if v.newpos.y < bottom(round(v.newpos.x)) then
                v.newpos.y:=bottom(round(v.newpos.x))
            end if
            if v.newpos.y > top(round(v.newpos.x)) then
                v.newpos.y:=top(round(v.newpos.x))
            end if
        end if
    end for
end procedure
end if
end for
end setvel

procedure move(var v: vowel)
  v.position.x:=v.newpos.x
  v.position.y:=v.newpos.y
  plot(v.position.x, v.position.y, v.c)
end move

var tot: int
put "How many vowels? ".
get tot
var allvwl: array 1..tot of vowel
for lcv: 1..tot
  put "Name for vowel #", lcv, "? ".
  get allvwl(lcv).name
  allvwl(lcv).c:=lcv+1
  put " x for vowel #", lcv, "? ".
  get allvwl(lcv).position.x
  put " y for vowel #", lcv, "? ".
  get allvwl(lcv).position.y
end for
setscreen("graphics")
for lcv: round(bottom(350))..round(top(350))
  plot(350, lcv, 1)
end for
for lcv: 350..850
  plot(lcv, top(lcv), 1)
  plot(lcv, bottom(lcv), 1)
end for
for lcv: 1..tot
  plot(allvwl(lcv).position.x, allvwl(lcv).position.y, allvwl(lcv).c)
end for
loop
  for lcv: 1..tot
    setvel(allvwl(lcv), allvwl, tot, lcv)
  end for
  for lcv: 1..tot
    move(allvwl(lcv))
  end for
end loop
Appendix B: Vowel space program with acceleration (Turing)

procedure plot(x: real, y: real, c: int)
        drawdot(round(((x-349)/501)*maxx), round(((y-499)/1201)*maxy), c)
end plot

var top: array 350..850 of real
var bottom: array 350..850 of real
for lcv: 350..850
        top(lcv):=1150+550*sqrt((850-lcv)/500)
        bottom(lcv):=1150-650*sqrt((850-lcv)/500)
end for

type vowel: record
        position: record
                x: real
                y: real
        end record
newpos: record
        x: real
        y: real
        end record
veloc: record
        x: real
        y: real
        c: int
        name: string
end record

procedure setvel(var v: vowel, var s: array 1..* of vowel, n: int, p: int)
        var dist: real
        var distx: real
        var disty: real
        v.newpos.x:=v.position.x
        v.newpos.y:=v.position.y
        for lcv: 1..n
                if lcv not=p then
                        distx:=v.position.x - s(lcv).position.x
                        disty:=v.position.y - s(lcv).position.y
                        dist:=sqrt(distx**2+disty**2)
                        v.veloc.x:=v.veloc.x+(distx/(dist**3))
                        v.veloc.y:=v.veloc.y+(disty/(dist**3))
                        v.newpos.x:=v.newpos.x+(distx/(dist**3))
                        v.newpos.y:=v.newpos.y+(disty/(dist**3))
                end if
        end for
        if v.newpos.x < 350 then
                v.newpos.x:=350
        end if
if v.newpos.x > 850 then
    v.newpos.x:=850
end if
if v.newpos.y < bottom(round(v.newpos.x)) then
    v.newpos.y:=bottom(round(v.newpos.x))
end if
if v.newpos.y > top(round(v.newpos.x)) then
    v.newpos.y:=top(round(v.newpos.x))
end if
end if
end for
end setvel

procedure move(var v: vowel)
v.position.x:=v.newpos.x
v.position.y:=v.newpos.y
plot(v.position.x, v.position.y, v.c)
end move

var tot: int
put "How many vowels? ".
get tot
var allvwls: array 1..tot of vowel
for lcv: 1..tot
    put "Name for vowel #", lcv, "? "
    get allvwls(lcv).name
    allvwls(lcv).c:=lcv+1
    allvwls(lcv).veloc.x:=0
    allvwls(lcv).veloc.y:=0
    put "  x  for vowel #", lcv, "? "
    get allvwls(lcv).position.x
    put "  y  for vowel #", lcv, "? "
    get allvwls(lcv).position.y
end for
setscreen("graphics")
for lcv: round(bottom(350))..round(top(350))
    plot(350, lcv, 1)
end for
for lcv: 350..850
    plot(lcv, top(lcv), 1)
    plot(lcv, bottom(lcv), 1)
end for
for lcv: 1..tot
    plot(allvwls(lcv).position.x, allvwls(lcv).position.y, allvwls(lcv).c)
end for
loop
for lcv: 1..tot
    setvel(allvwls(lcv), allvwls, tot, lcv)
end for
for lev: 1..tot
    move(allwvs(lev))
end for
end loop
Appendix C: Impossible consonants program (Prolog)

corresp(Item1, List1, Item2, List2):-
  List1 \neq [ ],
  List2 \neq [ ],
  List1 = [Head1|Tail1],
  List2 = [Head2|Tail2],
  ((Item1 = Head1, Item2 = Head2);
   corresp(Item1, Tail1, Item2, Tail2)).

distance(A, B, X):-
  Clist = [p, t, k, f, theta, s, esh, v, edh, z, ezh, b, d, g, m, n, engma, w, j, r, l, h],
  Distlist = [
    [0, 25, 15, 65, 70, 90, 95, 80, 85, 95, 95, 20, 55, 30, 60, 70, 65, 70, 75, 85, 85],
    [25, 0, 20, 85, 65, 80, 85, 70, 75, 90, 90, 45, 20, 35, 75, 65, 75, 75, 75, 70, 65, 85, 85],
    [15, 20, 0, 80, 85, 90, 85, 95, 95, 85, 35, 45, 20, 65, 70, 55, 60, 80, 90, 95, 85, 85],
    [65, 85, 80, 0, 25, 65, 70, 20, 65, 70, 65, 75, 85, 80, 85, 85, 85, 90, 95, 95, 95, 65],
    [70, 65, 85, 25, 0, 70, 65, 35, 25, 70, 65, 80, 75, 85, 85, 95, 85, 95, 90, 75, 75, 95, 65],
    [90, 80, 95, 65, 70, 0, 35, 85, 80, 40, 60, 95, 85, 95, 95, 90, 95, 95, 95, 95, 95, 85],
    [95, 85, 90, 70, 65, 35, 0, 80, 75, 55, 45, 95, 90, 95, 95, 85, 90, 95, 95, 95, 95, 65],
    [80, 80, 85, 20, 35, 85, 80, 0, 25, 65, 30, 55, 60, 65, 55, 65, 65, 65, 65, 60, 60, 60, 80],
    [85, 75, 95, 65, 25, 80, 75, 25, 0, 60, 25, 50, 45, 75, 65, 65, 70, 70, 70, 70, 70, 80],
    [95, 90, 95, 70, 70, 40, 55, 65, 60, 0, 20, 85, 65, 85, 85, 85, 75, 80, 85, 80, 70, 90, 85],
    [95, 90, 85, 65, 65, 60, 45, 30, 25, 20, 0, 80, 70, 85, 85, 65, 75, 80, 75, 75, 90, 85],
    [20, 45, 35, 75, 80, 95, 95, 55, 50, 85, 80, 0, 30, 25, 25, 65, 60, 55, 85, 85, 80, 95],
    [55, 20, 45, 85, 75, 85, 90, 60, 45, 65, 70, 30, 0, 30, 60, 60, 65, 80, 80, 70, 75, 95],
    [30, 35, 20, 80, 85, 95, 95, 65, 75, 85, 85, 25, 30, 0, 55, 65, 30, 50, 70, 85, 70, 95],
    [60, 75, 65, 80, 85, 95, 95, 55, 65, 85, 85, 25, 60, 55, 0, 20, 20, 55, 75, 85, 75, 90],
    [70, 65, 70, 85, 85, 90, 95, 85, 65, 75, 65, 65, 30, 65, 20, 0, 25, 65, 75, 80, 70, 90],
    [65, 75, 55, 85, 90, 95, 95, 65, 70, 80, 75, 60, 65, 30, 20, 25, 0, 35, 70, 85, 75, 90],
    [60, 75, 60, 80, 85, 95, 90, 55, 70, 85, 80, 55, 80, 50, 55, 65, 35, 0, 30, 30, 30, 85],
    [70, 70, 80, 95, 95, 95, 95, 60, 70, 80, 75, 85, 80, 70, 75, 70, 70, 0, 50, 45, 85],
    [75, 65, 90, 95, 95, 95, 60, 70, 70, 75, 85, 80, 85, 80, 85, 30, 40, 40, 85],
    [85, 85, 85, 95, 95, 95, 90, 60, 70, 90, 90, 80, 75, 70, 75, 70, 70, 0, 40, 85],
    [85, 85, 85, 65, 65, 85, 85, 80, 80, 85, 85, 95, 95, 95, 90, 90, 85, 85, 85, 85, 85, 0]],
  corresp(A, Clist, Distlist,
  corresp(B, Clist, X, Distlist),
  impossible(A, B, C):-
  distance(A, B, X),
  distance(B, C, Y),
  distance(C, A, Z),
  Maxac is X + Y, Z > Maxac.