Fuzzy Graphs and Error-proof Keyboards *

Flaminia L. Luccio
Department of
Mathematical Sciences,
University, 34100 Trieste (Italy)
luccio@dsm.univ.trieste.it

Andrea Sgarro
Department of
Mathematical Sciences,
University, 34100 Trieste (Italy)
sgarro@units.it

Abstract

Fuzzy graphs are used to assess the error-detecting and error-correcting capabilities of usual and unusual keyboards.

Keywords: Fuzzy graphs, error detection, error correction.

1 Fuzzy graphs and their extensions to sequences

A **fuzzy graph** G with vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ is a complete simple graph[1] where each edge $e = (v, v')$ between vertices $v \neq v'$ is given a weight $\mu(e) \in [0, 1]$. The number $\mu(e)$ is interpreted as the *degree of membership* of edge e to the (fuzzy) edge set $\mathcal{E}$. *Crisp* (ordinary) graphs are re-found when $\mu(e)$ is either 0 (edge e is lacking) or 1 (edge e is present). Notice that the vertex set of a fuzzy graph is always crisp, fuzziness is associated with the edge set only (cf e.g. [1] for basics on fuzzy sets). Once a number (a "level") α has been chosen, from G one can construct its α-cut G_α by setting $\mu_\alpha(e) = 1$ iff $\mu(e) \geq \alpha$, else $\mu(e) = 0$; in other words G_α is the crisp graph obtained by retaining only the edges whose degree of membership to the edge set is at least α.

Research partially supported by MIUR.

[1] In a simple graph loop-edges and multiple edges are ruled out; when the graph is complete, all other edges are present (possibly with weight equal to zero). The reader will have observed that fuzzy graphs are a nice way to represent fuzzy relations.

Let $\mathcal{V}^n$ be the n-th Cartesian power of the vertex set $\mathcal{V}$; its elements $\underline{x} = x_1 x_2 \ldots x_n$ are the K^n sequences obtained by juxtaposition of n vertices (of course x_i and x_j might be the same vertex v even if $i \neq j$). We shall now extend the fuzzy graph G with vertex set $\mathcal{V}$ to a fuzzy graph G^n with vertex set $\mathcal{V}^n$; the idea is that two vertex sequences $\underline{x}$ and $\underline{y} = y_1 y_2 \ldots y_n$ ($\underline{x} \neq \underline{y}$) are adjacent when adjacency is "tight enough" in *each* position i. Formally, for the weight $\mu(e)$ of the edge $e = (\underline{x}, \underline{y})$ we set:

$$\mu(e) \;=\; \mu(\underline{x}, \underline{y}) \;=\; \min_j \, \mu(x_j, y_j)$$

where the minimization is performed over all positions j such that $x_j \neq y_j$. Other definitions of power-graphs may be put forward; we have chosen this one because it fits in with the application we have in mind[2]. The following equality for α-cuts is readily proved; it shows that it is the same if one takes the power first and then makes an α-cut, or the other way round (the order of these two operations is irrelevant):

$$(G^n)_\alpha \;=\; (G_\alpha)^n$$

We add one more technicality. Fix a threshold α which, without real restriction, we may constrain to be one of the available edge-weights, inclusive of 1 even if 1 is not an edge-weight (in the examples below, fuzzy edge-sets are never *normal*, i.e. the membership degree $\mu = 1$ is never used). An **α-stable set** of vertices is a subset of vertices such that any edge

[2] This product is inspired by what in the crisp case is called the strong power of a graph; for (crisp) combinatorics an excellent reference is [8].

connecting two of them has weight strictly less than α (any two vertices in an α-stable set are bound to be only "loosely connected"). Actually, an α-stable set of a fuzzy graph G is a stable set of its α-cut G_α in the usual sense of combinatorics, i.e. it is a subset of vertices no two of which are adjacent (no two of which are linked by a crisp edge). We recall that the problem of finding *maximal* stable sets is NP-hard (computationally "intractable "). If $\mathcal{S}$ is an α-stable set of the fuzzy graph G, then its Cartesian power $\mathcal{S}^n$ is an α-stable set of the fuzzy power-graph G^n, as is proved by using the equality above. However, even if $\mathcal{S}$ is a maximal α-stable set of G, $\mathcal{S}^n$ is *not* necessarily maximal in G^n; cf e.g. [8].

In this paper we shall put forward a method meant to assess the error-detecting and error-correcting capabilities of keyboards, e.g. of telephone keyboards. We shall put the method to work upon two examples; the first refers to what can be seen as the "kernel" of a normal telephone keyboard as in use to-day, in the second the design of the telephone keyboard is quite unusual. The performances of the two keyboards will be compared. Much more complicated keyboards might be taken into account, but here we shall concentrate precisely on simple phone keyboards.

Example 1.1. Think of the keys (push-buttons) of a digital telephone keyboard in which digits from 1 to 9 are arranged on a 3×3 grid, left to right, top row to bottom row, and forget about digit 0 which is usually positioned below digit 8 (later we shall come back to this lamentable absence). We now define a fuzzy graph over the nine push-buttons; the edge set consists of those couples made up of two push-buttons $v \neq v'$ which may be inadvertently swapped by the user: $\mathcal{E} = \{(v, v')$ which are easily swapped$\}$. Two keys (two vertices) are joined the more tightly the easier it is to "slip" from one to the other (the more the corresponding edge belongs to $\mathcal{E}$). In this description fuzzy membership degrees are seen as numeric counterparts for "linguistic labels", and so what really counts is the ordering of the labels rather than their precise numeric values:

i) as a rule one pushes the correct button (this would give loop-edges with weight $\mu = 1$; actually, in our formalism loop-edges are not allowed, and so this weight will not be made explicit)

ii) sometimes one inadvertently pushes a neighbour of the correct button on the same row or on the same column ($\mu = 2/3$)

iii) it may also happen that one pushes a neighbour of the correct button situated on the same diagonal ($\mu = 1/3$)

iv) everything else is more or less out of the question ($\mu = 0$)

E.g., $\mu(1,4) = 2/3$, $\mu(1,5) = 1/3$, $\mu(1,6) = 0$.

Example 1.2. Think now of a fancy telephone keyboard in which the 9 push-buttons are arranged from 1 to 9 so as to form the vertices of an enneagon (of a nine-sided regular polygon); cf Figure 1 and also Figure 2.

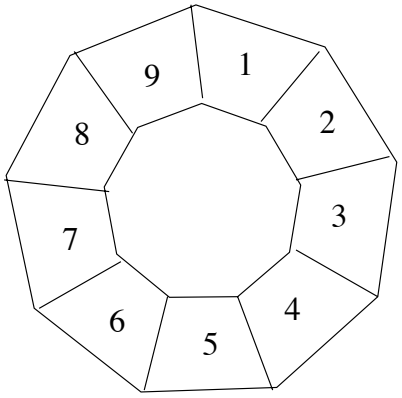

Figure 1: First example of an enneagon keyboard.

We shall describe possible "slips" in the following way:

i) as a rule one pushes the correct button

ii) it does happen that one inadvertently pushes a neighbour of the correct button, the one before, or the one afterwards ($\mu = \alpha^*$)

iii) it does even happen that one presses a button which is situated two positions before or afterwards, though this is quite exceptional ($\mu = 1/6$)

iv) everything else is more or less out of the

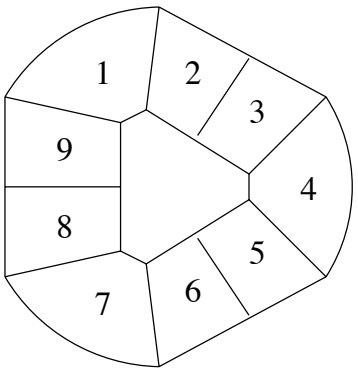

Figure 2: Second example of a nine-button keyboard.

question ($\mu = 0$)

We did not bother to specify the numeric value of α^*, because we want to compare the two keyboards and it is not clear how "physical contiguity" as in ii) and iii) of example 1.1 compares with "physical contiguity" of this example; in the absence of empirical evidence, it is fair to assume only $1/3 \leq \alpha^* \leq 2/3$. E.g., for the enneagon keyboard, $\mu(1,2) = \alpha^*$, $\mu(1,3) = 1/6$, $\mu(1,4) = 0$.

In both our examples we have to think also of slips between "super-keys", i.e. between whole phone numbers (between vertex sequences, in our formalism). The construction of fuzzy power-graphs for vertex sequences will be dictated by our choice of a reliability criterion, as described below at the beginning of Section 2. Preliminarily, think of a fuzzy graph G_i with vertex set $\mathcal{V}^n$ in which two "vertices" $\underline{x} \neq \underline{y}$ are linked when it is easy to slip from one sequence to the other in *exactly* the i-th position; so, we set for the edge-weights $\mu_i(\underline{x},\underline{y}) = \mu(x_i, y_i)$, $\mu_i(\underline{x},\underline{y}) = 1$ if $x_i = y_i$. In a way, G and G_i may be seen as "the same fuzzy graph", since in G_i all components other than the i-th do not really matter. Now, our reliability criterion is such that a slip from one sequence to the other (from one phone number to the other) easily happens only if slips easily happen in *each* position i. Therefore, what we need is a fuzzy graph over vertex sequences which is the fuzzy *intersection* of the graphs G_i. Now, if one uses the standard fuzzy intersection defined through a minimum, what one obtains is precisely the strong power G^n as defined in the preceding section. Let us choose $n = 6$ and go back to example 1.1; one has e.g. $\mu(565711, 555511) = 1/3$ (the minimum is obtained for $i = 4$); going back to the enneagon keyboard of example 1.2, one has instead $\mu(565711, 555511) = 1/6$.

2 Error-detecting keyboards

Our goal is the design of keyboards which are error-correcting (cf next section), or at least error-detecting. Let us agree on a certain phone number length n, e.g., as in our examples, $n = 6$. We shall fix a *reliability level* $\alpha \in [0, 1]$; in practice, and without real restriction, α will be equal to one of the edge-weights of G, inclusive of 1 even if 1 is not an edge-weight. Let us take in $\mathcal{V}^n$ a subset $\mathcal{C}$ of phone numbers, or, as we shall also say, let us take a *codebook* $\mathcal{C}$. The idea is that phone numbers inside $\mathcal{C}$ will be assigned, while those outside cannot be used. The codebook $\mathcal{C}$ is *α-reliable* when it is an α-stable set of G^n; the idea is that whenever a number outside $\mathcal{C}$ is selected, the error is detected, connection is refused and/or an alarm is triggered. Observe that error detection is *always* successful, *unless* one or more slips occur for which $\mu(v, v') < \alpha$. Our reliability criterion is precisely one that leaves out of consideration the occurrence of such slips, as if they were "impossible"; all other errors must be detected. We stress that if α is larger the criterion is *looser*.

An α-reliable code is *optimal* when it is *maximal* among α-stable sets of $\mathcal{V}^n$. As observed above, finding optimal codebooks is an NP-hard problem; in practice this mean that one should look for *good* codes, and usually forget about the optimal ones (this point of view is typical of coding theory, say of algebraic coding theory, as opposed to the more asymptotic and "philosophic" Shannon's information theory).

In our examples G has 9 vertices; standard software can readily identify maximal stable subsets for $n = 1, 2$ but gets bogged down

already for $n = 3$ (G^3 has 729 vertices). One can resort to approximations, and take a way-out like the following:

1st order way out: find a maximal α-stable set $\mathcal{S}$ in G, and then use the codebook $\mathcal{S}^6$ (which is α-stable, but not necessarily maximal); in practice this means that some push-buttons "far enough" from each other are permitted, while the remaining push-buttons are not.

2nd order way out: find a maximal α-stable set $\mathcal{T}$ in G^2, and then use the codebook $\mathcal{T}^3 \subseteq \mathcal{V}^6$; in practice this means that some *couples* of push-buttons are permitted, while the others are not.

In some situations, as is the case with the standard keyboard of example 1.1 both for $\alpha = 1/4$ and for $\alpha = 3/4$, the two ways out turn out to be one and the same. This is because, as it is easy to prove[3], in these two cases optimal codebooks are obtained precisely by taking the Cartesian product of an optimal α-stable set of push-buttons (this is true for each value of n, in particular for $n = 2$ and $n = 6$). Nothing better can be achieved. This state of affairs means that optimal codebook constructions are not especially cute, and it is no relief to learn that these situations are associated to a type of graphs which combinatorialists dub *perfect*. Here we shall not indulge in combinatorial digressions; it will be enough to stress that a long-standing conjecture due to Cl. Berge, called the *perfect-graph conjecture*, points at using keyboards based on odd-sided polygons, like the enneagon keyboard of example 1.2, if one is looking for ways out of the 2nd order which actually improve on the 1st order way out[4]. We shall now present some computational results.

[3]One may use the following well-known and straightforward result of (crisp) combinatorics: if in a graph the maximum size of a stable set coincides with the minimum number of cliques necessary to cover the graph, then an optimal (maximum-size) stable set in the strong-power graph is obtained by taking the Cartesian product of an optimal stable set in the initial graph; cf [8].

[4]Optimal codebooks are associated with a sophisticated asymptotic notion, called *Shannon's graph capacity*; cf [3],[6],[7] or [8]. The Shannon's graph capacity of odd-sided polygons with 7 or more edges is as yet unknown.

Let us take the standard keyboard of example 1.1 and choose $\alpha = 2/3$; one soon hand-checks (cf footnote 3) that an optimal codebook in $\mathcal{V}^n$ is $\{1,3,5,7,9\}^n$; for $\alpha = 1/3$ an optimal codebook is instead $\{1,3,7,9\}^n$. So, for $n = 2$ the optimal sizes are 25 and 16, respectively. Let us go to the enneagon keyboard. For $\alpha = \alpha^*$ and $n = 1$ the optimal size is just 4 and an optimal codebook is $\{1,3,5,7\}$; for $n = 2$ a computer search shows that the optimal size is 18 and an optimal codebook is made up of the couples $\{11,13,25,27,32,39,44,46,51,58,63,65,77,$ $79,82,\ 84,96,98\}$. On the whole, as far as error *detection* is concerned, the enneagon keyboard does not beat the standard one. Before going to the more ambitious goal of error *correction*, we still have to take $\alpha = 1/6$ for the enneagon keyboard; for $n = 1$ the optimal size is 3 and an optimal codebook is $\{1,4,7\}$, as made evident in Figure 2; the 2nd order way out does not bring any advantage, and an optimal codebook is simply $\{1,4,7\}^6$ (use again footnote 3). One might think of other fancy shapes; for example one might juxtapose two pentagon keyboards, so as to have one push-button in common. Unfortunately, the double pentagon is sort of a "squeezed enneagon": this leads to a fuzzy graph with higher connectivity, which is bad both from the point of view of error detection and error correction.

3 Error-correcting keyboards

We begin with a technicality. Once a fuzzy graph G is given, we can resort to a **fuzzy proximity closure**, and introduce a new graph $\widehat{G}$ for which

$$\widehat{\mu}(v,v') = \max_{w \in \mathcal{V}} \{ \min[\mu(v,w), \mu(w,v')] \}$$

The maximum is taken over the vertex set $\mathcal{V}$, which is common to both graphs, agreeing that $\mu(v,v) = 1$ (cf e.g. [4] for fuzzy relations, and in particular for fuzzy proximities). One has $\widehat{\mu}(v,v') \geq \mu(v,v')$ and so connectivity is higher in $\widehat{G}$ than it is in G; consequently, maximal α-stable sets are in general smaller. Take for example the standard keyboard; one

has $\mu(1,7) = 0$ but $\widehat{\mu}(1,7) = 2/3$ (think of the intermediate push-button 4); in the enneagon keyboard one has $\mu(1,3) = 1/6$ but $\widehat{\mu}(1,3) = \alpha^* > 1/6$.

Also in the case of error correction, a reliability criterion is chosen which abstracts from slips for which $\mu(v,v') < \alpha$ (we stress that the weights of the slips, either those allowed or those ruled out, are as in G and not as in its closure $\widehat{G}$). An *optimal* codebook $\mathcal{C}$ is then a maximal α-stable set of the power-graph $\widehat{G}^n$, and not of the power graph G^n, as is the case for error detection; in other words, error-correcting codebooks are obtained in the same way as error-detecting codebooks, only replacing the fuzzy graph G^n by its proximity closure $\widehat{G}^n$ (a proof of this claim is soon obtained by adapting the corresponding proof given in [7] to the "language" of fuzzy graphs, as used here). If a sequence $\underline{x}$ is pressed which lies outside $\mathcal{C}$, it is automatically corrected to the *unique* sequence $\underline{c}$ in the codebook for which $\mu(\underline{c},\underline{x}) < \alpha$ (the uniqueness of $\underline{c}$ is also proved in [7]). Let us present some computational results.

Let us take the standard square keyboard; α-stable sets are $\{1,9\}$ and $\{1\}$ for $\alpha = 2/3$ and $\alpha = 1/3$, respectively. One can show that for $n > 2$ one cannot do anything better than taking Cartesian products, and so *no* error correction is feasible for $\alpha = 1/3$. In particular, for $n = 2$ the optimal codebook sizes are 4 and 1, respectively. The square keyboard performs rather poorly from the point of view of error correction. Let us go to the enneagon keyboard. As one can readily check, the α^*-cut of the closure graph $\widehat{G}$ coincides with the 1/6-cut of the initial graph G; this equality carries over to the corresponding stable sets. This means that the *same* codebook is optimal both for error detection at level $\alpha = 1/6$ and for error correction at level $\alpha = \alpha^*$; the user can select the option he needs. We recall that for $n = 1$ the optimal size is 3, while for $n = 2$ the optimal size is 9. The enneagon keyboard performs better than the standard one as an error-correcting device, and it has the additional advantage of being a double-use device. The fact that 9 is not a prime number may be used to give the keyboard a pleasant appearance as we have tried to do in Figure 2; a zero push-button may be located in the center of the enneagon. If one agrees that it is quite hard to push such a button by mistake, the analysis performed so far need not be modified, since the new push-button can be used freely.

Final remark. In [2], [6], and more systematically in [7], a "soft" approach to coding, and in particular to error correction, has been introduced which is based on *possibility theory* rather than *probability theory*. As argued in [6] and [7], this approach is quite adequate, simpler and at the same time more powerful than the traditional approach to this sort of problems taken in the so called *zero-error information theory* (historically, the zero-error theory, which has a probabilistic basis, was started in the fifties by the late Cl. Shannon [5]; for an excellent overview cf [3]). Actually, the fuzzy graph approach taken in this paper, though necessarily limited to very special problems of error detection and error correction, appears to be even simpler and more direct than the possibilistic approach of [6] and [7]. Unlike what one does usually in the zero-error probabilistic approach, or in the possibilistic approach of [6] and [7], in this paper our interest has not been Shannon-theoretic, but has been more matter-of-fact, as is the case in much of coding theory proper. Zero-error information theory has been always held in high esteem because of the beautiful problems and results that it has provided to "pure" combinatorics; we hope that our effort may help to convince people that zero-error information theory, if suitably re-visited in the spirit of soft mathematics, be it possibility theory or fuzzy set theory, can also exhibit a large degree of practical applicability.

References

[1] D. Dubois, W. Ostasiewicz, H. Prade. Fuzzy Sets: History and Basic Notions. In *Fundamentals of Fuzzy Sets*, ed. by D. Dubois and H. Prade, Kluwer Academic Publishers, 2000.

[2] F. Fabris, A. Sgarro. Possibilistic Data Transmission and Fuzzy Integral Decoding. In *Proceedings of the conference IPMU'2000*, pages 1153-1158, Madrid, Spain, July 3-7 2000.

[3] J. Körner, A. Orlitsky. Zero-error Information Theory. *IEEE Transactions on Information Theory*, 44(6): 2207-2229, 1998.

[4] S. Ovchinnikov. An Introduction to Fuzzy Relations. In *Fundamentals of Fuzzy Sets*, ed. by D. Dubois and H. Prade, Kluwer Academic Publishers, 2000.

[5] C.E. Shannon. The Zero-Error Capacity of a Noisy Channel. *IRE Trans. Inform. Theory* (IT-2): 8-19, 1956.

[6] A. Sgarro. The Capacity of a Possibilistic Channel. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ed. by S. Benferhat and Ph. Besnard, Lecture Notes in Artificial Intelligence, 398-409, Springer, 2001.

[7] A. Sgarro. Possibilistic Information Theory: a Coding Theoretic Approach. *Fuzzy Sets and Systems*, galley proofs, 2002.

[8] J.H. van Lint, R.M. Wilson. *A Course in Combinatorics*, Cambridge University Press, 1992.