Promels

Iago Bonnici, Abdelkader Gouaich

January 22, 2016 LIRMM M. Iago BONNICI
Montpellier research report Dr. Abdelkader GOUAICH

In this research report, we are designing a mathematical object we call a matching *key* or "shape", which can basically do two things:

- A key can be *matched* against another key to produce a deterministic *match* value.
- A key can *mutate* into another close, random key.

Each vertex of a simple, complete, weighted graph will be associated with one of these. We will then expect two things from the keys:

- Static: any graph (set of edges weights values) should be describable by the keys when you match them together. That is, if you know the keys, you know the graph.
- Dynamic: when you successively mutate the keys, the resulting random graph process will take all the possible graph configurations.

If we consider that the graph contains $s \in \mathbb{N}$ vertices, we can easily write down the first point's challenge: being essentially able to describe $\frac{s(s-1)}{2}$ edges' weights with only s symbols. Each key will thus have to contain much information.

This document first formalizes all the requirements we would like our keys objects to meet. There are three things we need to design, our *unknowns*:

- What are the keys? Which kind of mathematical objects? ($k \in K$)
- How do we match two keys together? Which operation is that? ($\star$)
- How do we mutate one key into another? Which operation is that? ($\mu \in M$, $\mathscr{U}(M)$)

The requirements will depend on what we call the *weigth* or *descriptor* of an edge. ($d \in D$). This descriptor d translates the affinity relation defined in M_1.

In the second part, we describe a candidate object for the case $D = [-1, 1]$.

Contents

Scope Statement

1 Statics

1.1 Aim objects: edges weights in a graph

The ultimate role of the keys is to describe relations between objects, which can be represented as edges in a graph.

Let $s \in \mathbb{N}$ be the number of vertices in this this graph.

Let D be the set of all possible descriptors for one edge in the graph.

A "graph" can be equally seen as a matrix containing each edge decriptor:

$$graph : g \in G = \mathscr{M}_{s,s}(D) \tag{1}$$

Note that for a simple graph, this matrix is symmetric and its diagonal elements need not be defined:

$$\begin{aligned} \forall\, g \in G, \qquad \forall\, i \in [\![1,\ s]\!], \quad & g_{i,i} \text{ undefined} \\ \forall\, i \neq j \in [\![1,\ s]\!], \quad & g_{i,j} = g_{j,i} \end{aligned} \tag{2}$$

Many other kinds of graphs will also be describable by those keys. For example, one could relax the above hypotheses and consider a complete, bipartite graph (any non-squared $\mathscr{M}(\mathscr{D})$) for which we would produce all edges' descriptors by matching each key in the top compartment to each key in the bottom one. Any complex network of relations can be viewed as a bipartite graph [2], and bipartite graphs can also describe any hypergraph when interpreted as Levi graphs [3]. In this document, for the sake of simplicity, we assume that we can only work on complete, simple graphs without loss of generality.

The requirements listed hereafter will depend on the nature of D. We assume that D can either be:

- a non-metric, discrete set
- a metric, discrete set
- a metric, continous set
- a non-metric, continous set (even though we haven't investigated much this way)

(s and D are given as input data to the design problem.)

1.2 Keys and match

Keys ($k \in K$) are objects meant to be matched together in pairs. There must be an operation called *match* allowing two keys to match together, producing a result which will be interpreted as an edge's descriptor:

$$match : \begin{cases} K \times K \to D \\ (k_1,\ k_2) \mapsto k_1 \star k_2 \end{cases} \tag{3}$$

1.3 Graph description

If we associate each vertex $i \in [\![1,\ s]\!]$ with a key $k_i \in K$, we can get a result in G by matching each of them with every other key. This transformation of one set of keys into one graph is simply defined as:

$$T : \begin{cases} E = K^s \to G \\ \qquad e \mapsto g \; / \; \forall\, i \neq j \in [\![1,\ s]\!], \quad g_{i,j} = g_{j,i} = e_i \star e_j = k_i \star k_j \end{cases} \tag{4}$$

We call $e \in E$ a "set of keys" a "state of the system". In a nutshell, a set of keys defines a layout of relations between the objects.

Note that, since g is symmetric just like the relation between the objects, the match operation is also symmetric and $k_i \star k_j = k_j \star k_i$.

Case with D metric

If D is metric and $\text{dist} : D^2 \to \mathbb{R}^+$ is the metric over D, then we can also define a metric over G based on dist, let:

$$\text{Dist} : G^2 \to \mathbb{R}^+ \tag{5}$$

be any metric over G based on dist.

We then define a metric over E using Dist:

$$\forall\ (e, e') \in E^2,\ \text{Dist}(e,\ e') = \text{Dist}(T(e),\ T(e')) \tag{6}$$

In a nutshell, two states of the system are close together if they produce close graphs.

(Dist is then given as input data to the problem)

1.4 Richness of the keys

We need the keys to be able to accurately describe any layout of relations between the objects. This requirement writes down differently depending on the nature of D:

Case with D discrete, metric or non-metric

$$T \text{ surjective} \tag{7}$$

(any graph can be obtained by matching a particular set of keys)

Case with D continuous and metric

$$\forall\ g \in G,\ \forall\ \varepsilon \in \mathbb{R}^{+*},\ \exists\ e \in E\ /\ \text{Dist}(T(e),\ g) < \varepsilon \tag{8}$$

(any graph can be approached to an arbitrary precision by matching particular set of keys)

Case with D continuous and non-metric

To be filled if ever needed..

2 Dynamics

2.1 Mutating keys

We need K to exhibit a set of internal transformations called *mutations* $\mu \in M$:

$$mutation : \mu : \begin{cases} K \to K \\ k \mapsto \mu(k) \end{cases} \tag{9}$$

As a requirement, there should exist a null mutation:

$$\exists\ \mu_0 \in M \ / \ \forall\ k \in K,\ \mu_0(k) = k \tag{10}$$

2.2 Mutating states

We can mutate one state of the system into another by simply mutating each key in the system. This operation is defined with the same symbol μ as:

$$\mu : \begin{cases} E \to E \\ e \mapsto \mu(e) \ / \ \forall\ i \in [\![1,\ s]\!],\ (\mu(e))_i = \mu(e_i) \end{cases} \tag{11}$$

2.3 Small mutations

This requirement only holds if D metric: there should exist small mutations.

Case with D continuous and metric

The mutation should potentially be arbitrarily small:

$$\forall\ k_1,\ k_2 \in K, \quad \forall\ \varepsilon \in \mathbb{R}^{+*}, \quad \exists\ \mu \in M \ / \ \mathrm{dist}(k_1 \star k_2,\ \mu(k_1) \star k_2) < \varepsilon \tag{12}$$

Case with D discrete and metric

Then there exists an atomic distance between two elements of D:

$$\begin{aligned} \exists!\ a \in \mathbb{R}^{+*} \ / \ \exists\ (d,\ d') \in D^2 \ / \ \mathrm{dist}(d,\ d') = a \\ \&\ \nexists (d,\ d') \in D^2 \ / \ \mathrm{dist}(d,\ d') < a \end{aligned} \tag{13}$$

So we wish there exists an atomic mutation:

$$\forall\, k_1,\, k_2 \in K,\, \exists\, \mu \in M \,/\, \mathrm{dist}(k_1 \star k_2,\, \mu(k_1) \star k_2) = a \tag{14}$$

2.4 Connexity *via* mutations

Random graph processes are often expected to display some specific properties, at any time (evolving under constraints) [4] or at infinity (converging to a specific form) [5]. In contrast, we wish our random graph process resulting from successively mutating $e \in E$ to be able to reach every possible state without particular constraint, so that mutations can take you anywhere.

Put it another way, we should be able to reach any layout of relations by successively mutating the state of any system. This requirement writes as:

Case with D discrete, metric or non-metric

$$\begin{aligned} \forall\, (g_1,\, g_2) \in G^2,\, (e_1,\, e_2) \in E^2 \,/\, T(e_1) = g_1,\, T(e_2) = g_2, \\ \exists\, n \in \mathbb{N},\, \mu \in M^n \,/\, (\mu_n \circ \mu_{n-1} \circ \cdots \circ \mu_1)(e_1) = e_2 \end{aligned} \tag{15}$$

Case with D continuous and metric

$$\begin{aligned} \forall\, (g_1,\, g_2) \in G^2,\, (e_1,\, e_2) \in E^2 \,/\, T(e_1) = g_1,\, T(e_2) = g_2, \\ \forall\, \varepsilon \in \mathbb{R}^{+*},\, \exists\, n \in \mathbb{N},\, \mu \in M^n \,/\, \mathrm{Dist}\big((\mu_n \circ \mu_{n-1} \circ \cdots \circ \mu_1)(e_1),\, e_2\big) < \varepsilon \end{aligned} \tag{16}$$

Case with D continuous and non-metric

To be filled if ever needed..

2.5 Random motion via mutations

There should exist a random distribution over M, $\mathscr{U}(M)$ such that those two, yet informal, requirements are met:

- The probability of randomly getting a *small* mutation (see 2.3) can be arbitrarily higher than the probability of getting a bigger mutation.

- The probability of never finding a path between two graphs (see 2.4) by successively drawing a serie of mutations is zero.

3 To conclude

We need to design those four objects: $(K,\ \star,\ M,\ \mathscr{U}(M))$, so that all the above points gathered as requirements are fulfilled.

4 Lost on the way:

4.1 Key divergence

This last requirement has to see with the biotic interpretation of D, and so it is difficult to place it in the above. It involves a continuous, metric $D = [-1, 1]$ exhibiting an internal *order* and a particular value of $d = 0 \in D$ which is interpreted as a "null match" (no interaction): the closer a match is from 0, the "weaker" the match is.

The idea is that two keys heavily, independently mutated for a while by $\mu \hookrightarrow \mathscr{U}(M)$ should have a weak match in expectancy:

$$\begin{aligned}
&\text{Let}\quad k_0 \in K,\ n_a,\ n_b \in \mathbb{N},\ \mu_a \hookrightarrow \mathscr{U}(M)^{n_a},\ \mu_b \hookrightarrow \mathscr{U}(M)^{n_b},\\
&\qquad k_a = (\mu_{a n_a} \circ \cdots \circ \mu_{a1})(k_0),\\
&\qquad k_b = (\mu_{b n_b} \circ \cdots \circ \mu_{b1})(k_0),\\
&\qquad X = k_a \star k_b,\\
&\text{Then, as}\ \ n_a, n_b \to \infty,\\
&\qquad \mathbb{E}(X) = 0\\
&\qquad \mathbb{P}(X > 0) = \mathbb{P}(X < 0),\\
&\qquad \forall\, x_1,\ x_2 \in [0,1],\quad x_1 > x_2 \Rightarrow [\,|X|\,](x_1) < [\,|X|\,](x_2)
\end{aligned} \tag{17}$$

where $[X]$ is the density function of the real, random variable X.

Promels

Let us describe here the candidate we've built for $D = [-1, 1]$ which is metric, continuous, and gifted with an internal order and a "null match" value $0 \in D$ (see Scope Statement, section 4.1). We trust it does fulfill all the above requirements even though we haven't proved it yet.

Since a continuous, metric D contains enough information to describe any other discrete set, we think it'll be easy to use the same candidate for other types of edges descriptors just by discretizing D. Put it another way, we think we can work with this particular D without loss of generality.

The idea is to define the key as a continuous function in $\mathbb{R}$, and the match operation as an operation comparing the shapes of such functions together. This idea had already been explored by Edelstein and Rosen in 1978, then modelling enzymes-substrate interaction and their molecular forms [1]. The main difference with our candidate is that the duality enzyme-substrate led them to consider two different types of functions, one of which defined the pattern for reading the other one. Here we define a single object that can be matched against other objects of the same type, and the reading pattern just emerges from the two objects shapes. The common idea is to make use of the rich information contained in any real function. Here, it is to deal with the problem of essentially representing $\frac{s(s-1)}{2}$ edges with only s symbols.

5 Defining K, the key

We state that this specific key is a $\mathscr{C}^\infty$, periodic, angular function over $\mathbb{R}$. Its period is any $T \in \mathbb{R}^{+*}$ and its results are interpreted as angles:

$$\mathscr{C}^\infty \supset K \ni k : \begin{cases} \mathbb{R} \sim [0, T[\to \mathbb{R} \sim [0, 2\pi[\\ t \mapsto k(t) \end{cases} \tag{18}$$

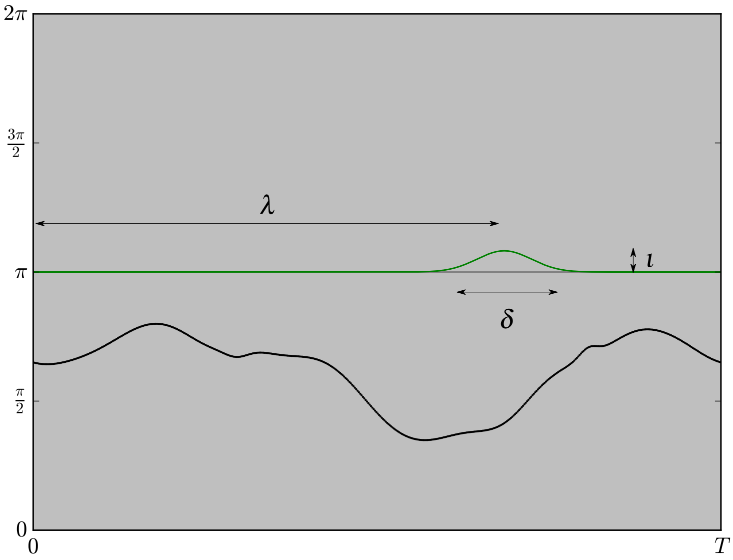

Figure 1: Flat projetion of a key (black) and its mutation kernel (green). The grey line is the kernel zero. The grey area is a flattened torus

Since numbers are interpreted as angles, the neighbourhood of 0 is the the neighbourhood of 2π. We also state that the neighbourhood of 0 is the neighbourhood of T on the base axis. The key can thus be seen as a continous curve on a torus (Figure 1).

6 Defining $\star$, the match operation

6.1 Local match

For any $t \in [0,T[,\ k_1,\ k_2 \in K$, we define the "local match" between k_1 and k_2 by:

$$\mathrm{lm}: \begin{cases} [0,T[\to [-1,1] \\ \quad t \mapsto \mathrm{lm}(t) = \cos\Big(k_2(t) - k_1(t)\Big) \end{cases} \tag{19}$$

The local match will therefore be positive if the direction of the two keys are locally aligned, negative if they locally point to opposite directions.

Note that $k_2(t) - k_1(t)$ is meant to represent the *shortest distance* between the two angles, taking into account the fact that 0 is neighbourg of 2π. In this way, we have $\frac{3\pi}{2} - \frac{\pi}{2} = \frac{pi}{2}$ but $\frac{7\pi}{4} - \frac{\pi}{4} = \frac{2\pi}{4}$. To be neat: "$b-a$" $= \min(|b-a|,\ |2\pi - (b-a)|)$ since a and b are read on the $[0,2\pi[$ circle.

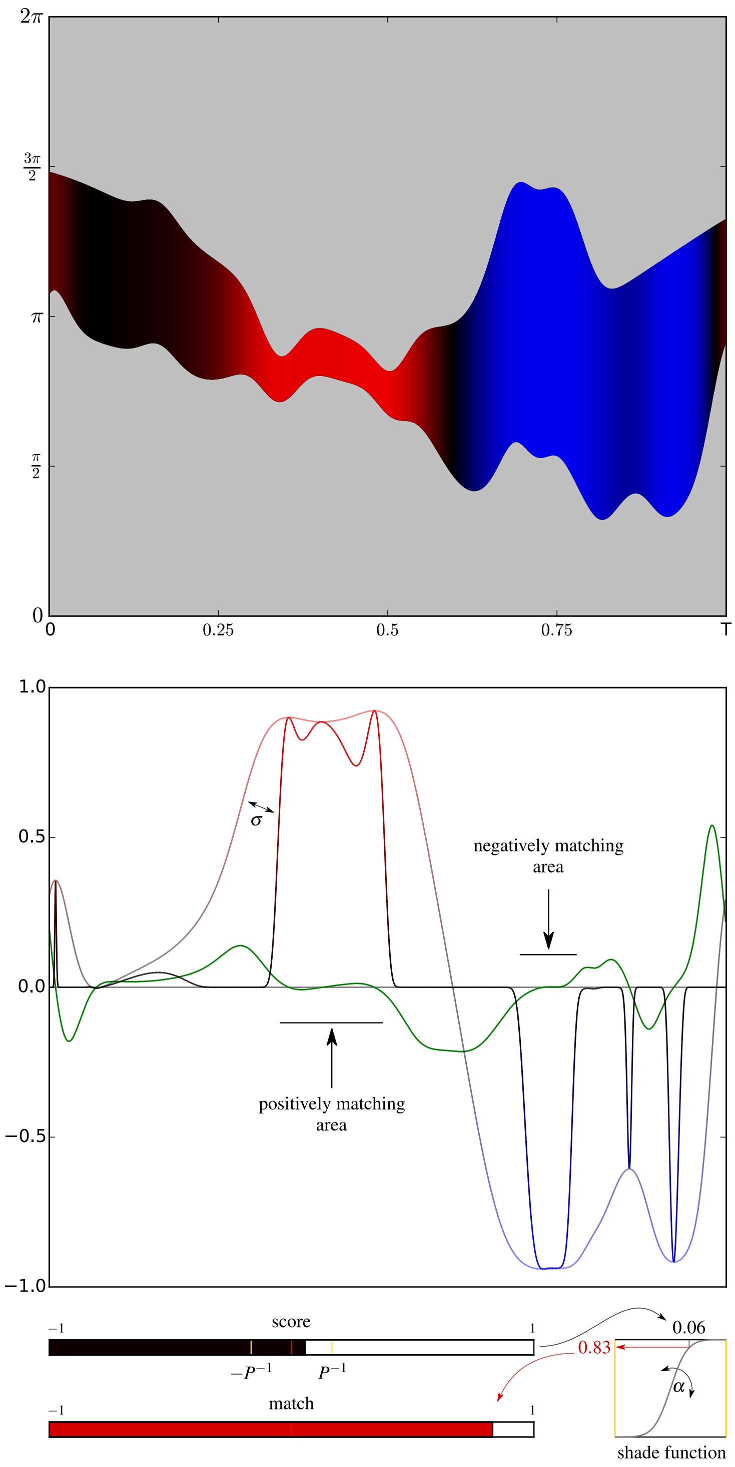

Figure 2: Representation of the matching operation. The colored area above represents the local match along two keys (blue as it approaches -1, red as it approaches 1). On the second plot, one can read the same local match lm (light line), its derivative lm' (green line) and the resulting local score $\mathrm{w}_\sigma \circ \mathrm{lm}'$ (strong line) where the regions with different shapes have been σ-filtered out. The global score is the integral of this local score. The global match is the $\mathrm{shade}_{\alpha,P}$ of the global score, here about 83% for $P = 6, \sigma \approx 5, \alpha \approx 3.4$.

6.2 Score

The idea, for the match operation not to be transitive, is that this local match will only be considered in regions where the keys have the same shape, even if they point to different directions (see blue area figure 2). By introducing this notion of *locality*, we trust that we can make use of all the information contained in the keys. For example, two keys that match well in a specific region of $[0,T[$ might have two different relations to another third key if those relations are defined by another region of $[0,T[$. Regions of $[0,T[$ where the keys don't share the same shape are just made silent by the match operation.

To achieve this, we set up a weight filter w that will only select the regions where the keys share the same shape, that is, where the derivative of lm (green line figure 2) is close to zero, $\forall\ \sigma \in \mathbb{R}^+$:

$$\mathrm{w}_\sigma : \begin{cases} \mathbb{R} \to [0,1] \\ \mathrm{lm}' \mapsto e^{-\frac{1}{2}(\sigma\ \mathrm{lm}')^2} \end{cases} \tag{20}$$

The parameter σ represents the "severity" of the filter: the more high, the more regions with weakly matching shapes will be filtered out.

Weighted local matches will be then summed to produce a total matching *score* $\in [-1,1]$:

$$score : \begin{cases} K \times K \to [-1,1] \\ (k_1,\ k_2) \mapsto k_1 \cdot k_2 \end{cases} \tag{21}$$

$$k_1 \cdot k_2 = \frac{1}{T} \int_0^T \mathrm{w}_\sigma\big(\mathrm{lm}'(t)\big)\,\mathrm{lm}(t)\,\mathrm{d}t \tag{22}$$

6.3 Global match

Finally, the score will be interpreted as a match value by this shade function, $\forall\ \alpha \in \mathbb{R}^+$, $P = \frac{S(S-1)}{2}$:

$$\mathrm{shade}_{\alpha,P} : \begin{cases} [-1,\ 1] \to [-1,\ 1] \\ \qquad s \mapsto \begin{cases} \mathrm{sign}(s) & \text{if } |s| > \frac{1}{P} \\ \mathrm{arctanh}\left(\alpha \tanh(Ps)\right) \end{cases} \end{cases} \tag{23}$$

$$k_1 \star k_2 = \mathrm{shade}_{\alpha,P}(k_1 \cdot k_2) \tag{24}$$

Note that $\mathrm{shade}_{\alpha,P} \nearrow$ & $\mathrm{shade}_{\alpha,P}(0) = 0$. Therefore the higher the score, the higher the match, and a null score is equivalent to a null match.

$\frac{1}{P}$ is the sufficient score to get a match value of 1 (complete match). $\frac{-1}{P}$ is the sufficient score to get a match value of -1 (complete antimatch). The parameter α represents the importance one gives to a certain score: the more high, the more a score close from zero will give a match far from zero. This relation is linear for $\alpha = 1$.

7 Defining M, the mutation operation

We will mutate one key into another by adding a *mutation kernel* to the original key (see Figure 1). We choose as a kernel an element of K with interesting properties: a Gaussian function "circularized" along $[0, T[$ in the following way:

Let N be the normal distribution function, parametrized by μ and σ:

$$N_{\mu,\sigma} : \begin{cases} \mathbb{R} \to \mathbb{R}^+ \\ x \mapsto \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \end{cases} \tag{25}$$

Here is how we define its "circularized" version C:

$$C_{\mu,\sigma} : \begin{cases} [0, T[\to \mathbb{R}^+ \\ t \mapsto \displaystyle\sum_{-\infty}^{+\infty} N_{\mu,\sigma}(t + iT) \end{cases} \tag{26}$$

It turns out that C then writes using the third Jacobi theta function as, $\forall\, t \in [0, T[$:

$$C_{\mu,\sigma}(t) = \frac{\vartheta_3(q,u)}{T} = \frac{1}{T}\left(1 + 2\sum_{i=1}^{\infty} q^{i^2} \cos(2iu)\right), \; q = e^{-2\left(\frac{\pi\sigma}{T}\right)^2}, \; u = \pi\frac{t-\mu}{\sigma} \tag{27}$$

Then here is how we define our mutation kernel, which depends on three parameters

$\lambda \in [0,1],\ \delta \in \mathbb{R}^{+*},\ \iota \in \mathbb{R}$ (see figure 1):

$$K \ni \ker_{\lambda,\delta,\iota} : \begin{cases} [0,\ T[\rightarrow \mathbb{R} \\ \qquad t \mapsto \iota \dfrac{C_{T\lambda,T\delta}(t)}{C_{0,T\delta}(0)} \end{cases} \tag{28}$$

λ simply is the *location* of the mutation over the key (the kernel position)

δ tunes its *delocation* (the kernel width)

ι tunes its algebraic *intensity* (the kernel height)

In the end, the mutation operation can be defined as, $\forall\ k \in K$:

$$\mu(k) = k + \ker_{\lambda,\delta,\iota} \tag{29}$$

As the keys will be successively "bumped" by these mutation kernels, their shape will evolve over time and so will their match against other keys (see figures 1 and 2).

8 Defining $\mathscr{U}(M)$, the mutations distribution

Randomly choosing a mutation is now just as easy as randomly choosing the kernel parameters λ, δ, ι from their respective domains.

However, the distributions from which they will be drawn will determine whether all the above object requirements are met or not. If they are too restrictive, then the resulting random graph process might not be able to reach every possible state, nor to get as close as an arbitrarily small ε, ect.

Nonetheless we trust that, given $s \in \mathbb{N}$ (the number of vertices in the graph), and given one targetted precision $\varepsilon \in \mathbb{R}^{+*}$ over the resulting process, one will always be able to find a filter severity $\sigma \in \mathbb{R}^{+*}$, a shade parameter $\alpha \in \mathbb{R}^{+}$ and a distribution among λ, δ, ι domains allowing one to meet all the requirements specified in the first chapter.

9 Designing keys for a specific graph

Even if this is not a formal requirement for the object, we would also like being able to *design* a set of keys $e \in E$, such that a particular, arbitrary graph $g \in G$ results from it. In a nutshell, find an e solution to $T(e) = g$. In the following, we describe how a particular solution can be found.

Note that, during the time we build this solution, the $K \subset \mathscr{C}^\infty$ hypothesis is relaxed. We will come back to this at the end of the section.

9.1 Slicing down keys

Let K_T be the set of all keys with period $T \in \mathbb{R}^{+*}$. For any $k \in K_T,\ T_1 < T$, we define the following start $\vdash$ and end $\dashv$ operators by:

$$k^{\vdash T_1} = k\big|_{[0,T_1[} \in K_{T_1} \tag{30}$$

$$k^{\dashv T_1} = \left\{ \begin{array}{rcl} [0,T_1[& \to & \mathbb{R} \\ t & \mapsto & k(t+T-T_1) \end{array} \right. \in K_{T_1} \tag{31}$$

This easily leads to the following, intuitive properties, $\forall\ T_1 + T_2 + T_3 = T$:

$$\left(k^{\vdash T_1+T_2}\right)^{\vdash T_1} = k^{\vdash T_1} \tag{32}$$

$$\left(k^{\dashv T_2+T_3}\right)^{\dashv T_3} = k^{\dashv T_3} \tag{33}$$

$$\left(k^{\vdash T_1+T_2}\right)^{\dashv T_2} = \left(k^{\dashv T_2+T_3}\right)^{\vdash T2} \tag{34}$$

Keys can thus be sliced down in several subkeys.

9.2 Joining keys together

Here is the reverse operation, defined as, $\forall\ T_1 + T_2 = T$:

$$\backsim : \left\{ \begin{array}{l} K_{T_1} \times K_{T_2} \to K_T \\ (k_1,\, k_2) \mapsto k_1 \backsim k_2 \end{array} \right. \tag{35}$$

$$k_1 \backsim k_2 = \begin{cases} [0, T_1 + T_2[\to \mathbb{R} \\ \qquad t \mapsto \begin{cases} k_1(t) & \text{if } t < T_1 \\ k_2(t - T_1) \end{cases} \end{cases} \tag{36}$$

Of course, we have:

$$(k_1 \backsim k_2)^{\vdash T_1} = k_1 \tag{37}$$

$$(k_1 \backsim k_2)^{\dashv T_2} = k_2 \tag{38}$$

9.3 Scoring joined keys

By linearity of the integration, we can assert that, $\forall\ T_1 + T_2 = T$:

$$\begin{aligned} &\forall\ k_1 \in K_{T_1},\ k_2 \in K_{T_2},\ k \in K_T, \\ &\qquad (k_1 \backsim k_2) \cdot k = \frac{T_1}{T} k_1 \cdot k^{\vdash T_1} + \frac{T_2}{T} k_2 \cdot k^{\dashv T_2} \end{aligned} \tag{39}$$

(break down the global score between two keys into the sum of their subkeys global scores)

9.4 Finding null keys

We now focus on the set of all constant keys $\overline{K}_T \subset K_T$. And search for a key that has a null score with every key in $\overline{K}_T$. In the following, we will make use of the abusive notation $k(t) = k$ when $k \in \overline{K}_T$.

Let $v_0 \in \overline{K}_T$. We define, for every $i \in \mathbb{N}^*$:

$$v_i : \begin{cases} [0, T[\to \mathbb{R} \\ \qquad t \mapsto \begin{cases} v_0 & \text{if } t \pmod{\frac{1}{T^{i-1}}} < \frac{1}{T^i} \\ v_0 + \pi \end{cases} \end{cases} \tag{40}$$

They have these interesting properties $\forall\ i \in \mathbb{N}^*,\ k \in \overline{K}_T$:

$$\begin{aligned} v_i \cdot k &= 0 \\ \forall\ j \in [\![1,\ i-1]\!],\ v_i \cdot v_j &= 0 \end{aligned} \tag{41}$$

9.5 Building the set

Let $g \in G$, we now have everything we need to build a set of keys $e \in E$ that will solve $T(e) = g$.

The idea is to join $P = \frac{S(S-1)}{2}$ elementary keys together to produce each of the $k_i,\ i \in [\![1,\ s]\!]$ keys in e, and to make a heavy use of the null keys ν to avoid conflicts between keys relations to one another.

Each key k_i in e is obtained by joining together P elementary keys from $K_{\frac{T}{P}}$:

$$\forall\, i \in [\![1,\ s]\!], \quad k_i = k_{i_1} \backsim \ldots \backsim k_{i_P} \tag{42}$$

Therefore, each relation $g_{i,j}$ in g is defined by this formula, derived from 39:

$$\begin{aligned} &\forall\, i \neq j \in [\![1,\ s]\!], \\ &g_{i,j} = \operatorname{shade}_{\alpha,P}\left(\frac{1}{P}\sum_{p=1}^{P} k_{i_p} \cdot k_{j_p}\right) \end{aligned} \tag{43}$$

Let us choose any basal, elementary constant key $\beta \in \overline{K}_{\frac{T}{P}}$. In order to cancel the unwanted terms in the sum (43), we suggest using the following design, here with $S = 5$:

$$\begin{array}{ccccccccccccccccccccc}
k_1 & = & \beta & \backsim & \beta & \backsim & \beta & \backsim & \beta & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \nu_1 \\
k_2 & = & x_{12} & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \nu_1 & \backsim & \beta & \backsim & \beta & \backsim & \beta & \backsim & \nu_2 & \backsim & \nu_2 & \backsim & \nu_2 \\
k_3 & = & \nu_1 & \backsim & x_{13} & \backsim & \nu_2 & \backsim & \nu_2 & \backsim & x_{23} & \backsim & \nu_2 & \backsim & \nu_2 & \backsim & \beta & \backsim & \beta & \backsim & \nu_3 \\
k_4 & = & \nu_2 & \backsim & \nu_2 & \backsim & x_{14} & \backsim & \nu_3 & \backsim & \nu_2 & \backsim & x_{24} & \backsim & \nu_3 & \backsim & x_{34} & \backsim & \nu_3 & \backsim & \beta \\
k_5 & = & \nu_3 & \backsim & \nu_3 & \backsim & \nu_3 & \backsim & x_{15} & \backsim & \nu_3 & \backsim & \nu_3 & \backsim & x_{25} & \backsim & \nu_3 & \backsim & x_{35} & \backsim & x_{45}
\end{array}$$

With this design, and thanks to the null keys, only one term remains in the sum 43, which is now:

$$\begin{aligned} g_{i,j} &= \operatorname{shade}_{\alpha,P}\left(\frac{1}{P}\beta \cdot x_{ij}\right) \\ &= \operatorname{shade}_{\alpha,P}\left(\frac{1}{P}\cos(x_{ij} - \beta)\right) \end{aligned} \tag{44}$$

All we have to do is to compute the remaining unknowns $x_{ij} \in \overline{K}_{\frac{T}{P}}$:

$$x_{ij} = \beta \pm \arccos\left(P \operatorname{shade}_{\alpha,P}^{-1}(g_{i,j})\right) \tag{45}$$

And the set e of all k_i keys is such that $T(e) = g$.

We noticed already that the keys built this way are not $\mathscr{C}^\infty$ anymore. However, since they are periodic, one can approximate them with arbitrary precision by $\mathscr{C}^\infty$ functions using Fourier decomposition, and get a "true" $e \in E$ result such that $\mathrm{Dist}(T(e),\ g) < \varepsilon\ \forall\ \varepsilon \in \mathbb{R}^{+*}$.

Bibliography

[1] Leah Edelstein and R Rosen. “Enzyme-substrate recognition”. In: *Journal of theoretical biology* 73.1 (1978), pp. 181–204.

[2] Jean-Loup Guillaume and Matthieu Latapy. “Bipartite structure of all complex networks”. In: *Information processing letters* 90.5 (2004), pp. 215–221.

[3] Friedrich Wilhelm Levi. University of Calcutta, 1942.

[4] Lionel Tabourier, Camille Roth, and Jean-Philippe Cointet. “Generating constrained random graphs using multiple edge switches”. In: *Journal of Experimental Algorithmics (JEA)* 16 (2011), pp. 1–7.

[5] Monica Van Horn, Angela Richter, Dian Lopez, et al. “A random graph generator”. In: *36th Annual Midwest Instruction and Computing Symposium, Duluth, MN*. 2003.