Geohash Generalizado: uma metodologia para a implementação de funções de hash geradoras de geocódigos hierárquicos base32

RASCUNHO versão 0.6 – *Peter Krauss*, *Everton Bortolini*, *Thierry Jean*, *Rui Pedro Julião*.

(publicado como urn:doi:10.5281/zenodo.2529558, procurar pela versão mais atual!)

RESUMO

Células de grades espaciais bidimensionais baseadas em indexadores espaciais do tipo Hilbert ou Z-order possuem identificadores numéricos estruturados com base4, de modo a reproduzir a hierarquia espacial na estrutura de dígitos do identificador. Para a representação compacta exata faz-se uso da representação em base16 ou base64, pois são potências de 4, identificando uma célula interior a cada dígito acrescentado. Contudo, na utilização como geocódigo a demanda mais comum é pela base32, que só exibe a hierarquia da grade espacial (estruturada em base4) a cada dois dígitos. A solução adotado pelo Geohash foi representar numa grade espacial degenerada os seus geocódigos com um número ímpar de dígitos. O Geohash Generalizado propõe adotar a mesma estratégia para viabilizar o uso da base32 em outros indexadores, inclusive aqueles baseados em curva de Hilbert. A generalização consiste em oferecer um algoritmo geral para a construção da grade degenerada, a partir da união de duas células vizinhas do nível hierárquico seguinte. Em aplicações orientadas à representação apenas de pontos isolados, ou indexação do tipo Point-Region Quadtree, perda de uniformidade na grade degenerada não causa impacto nem chega a ser perceptível pelo usuário final: com essa hipótese de trabalho foi possível estabelecer uma metodologia simples e abrangente, viabilizando o uso de tecnologias tais como S2geometry como alternativas ao Geohash convencional. O algoritmo de reindexação e a sua geometria são também analisados. Em termos práticos, conclui-se que o Geohash Generalizado pode contribuir no desenvolvimento de sistemas de endereçamento baseados em localização.

SUMÁRIO (19+10 páginas)

Introdução

As curvas fractais de preenchimento do espaço bidimensional têm sido utilizadas em diversos indexadores espaciais, em particular naqueles empregados como identificadores de células de uma grade global discreta (do inglês **DGG** - *Discrete Global Grid*) do tipo **hierárquica**.

Identificadores de célula desse tipo de DGG são passíveis de serem utilizados como **geocódigos**, ou seja, como representações compactas das coordenadas de latitude e longitude de um ponto. Exemplos típicos são as DGGs S2geometry, GeohashReference e Geohash-Hilbert.

O identificador de cada célula da grade, por ser também um indexador, corresponde a um número sequencial que vai de um à quantidade de células na grade. Sendo grades hierárquicas, tipicamente de partição recorrente das células em 4 regiões, o valor do identificador terá sua estrutura hierárquica acessível quando expresso em **base4** (2 bits por dígito). Com comprimento máximo da ordem de 60 bits, os identificadores são capazes de oferecer até ~30 níveis hierárquicos.

A representação compacta, para o uso do identificador como geocódigo, requer base maior que 4, tipicamente base16, base32, ou base64, por serem compatíveis com base4, todas múltiplas de 4 e potências de 2 ($16=4^2$, $32=2\square4^2$ e $64=4^3$). Em termos de representação alfanumérica multimeios (voz ou texto em presença de ruído) o limite é a base36, de modo que a **base32** (5 bits por dígito) é mais se aproxima desse limite, destacando-se também como forma de representação com maior demanda em geocódigos.

Público-alvo e objetivo geral

Depois de uma longa e interessante jornada, os autores encontraram uma solução consistente para um problema relevante, motivados principalmente pelo Projeto CLP. Estamos escrevendo este ensaio (esta versão ainda não é a definitiva) no sentido de documentar as técnicas que desenvolvemos e reflexões teóricas sobre o assunto. Além de apresentar detalhadamente a metodologia, demonstrar seus fundamentos e apresentar resultados que ilustram seu uso, o principal objetivo deste documento é o registro em domínio público, criando-se **evidência contra qualquer tentativa de patente** sobre os métodos ou algoritmos apresentados. Esta obra é dedicada ao domínio público nos termos da Creative Commons CC0.

A seguir, na introdução e nas ilustrações da metodologia, serão detalhados de forma mais didática os conceitos e técnicas de representação numérica, para que os usuários comuns, **sem** formação em Matemática, possam compreender a solução apresentada pelos autores. Na seção Metodologia é apresentado o algoritmo geral para a representação espacial dos geocódigos base32.

Representações numéricas textuais

Os valores de identificadores, na sua representação interna em um software ou banco de dados, assume sua forma mais “pura” como por exemplo um inteiro binário positivo de 64 bits. Entretanto, no momento que surge a demanda por apresentar o valor na tela como texto, para ser legível ao ser humano, é necessário que se tome a decisão pela representação em uma determinada base.

O número **1000** por exemplo, neste texto está na sua representação decimal (base10) ocupando 4 dígitos decimais. O 1000, internamente em um computador, é um valor binário de 10 bits: mesmo representado em outra base, continua sendo portador de 10 bits de informação. Quando representado por exemplo na base4, “33220”, os mesmos 10 bits ocupam mais espaço de texto — cinco dígitos (portando 2 bits cada), portanto um a mais que na base10. Quando representado na base16, “3E8”, ocupa menos espaço de texto — três dígitos (4 bits cada), um a menos que na base10. Bases cujos dígitos são portadores de mais bits garantem representações mais compactas (com menos dígitos).

O número natural expresso em diferentes bases

Vejamos dois casos típicos de "representação padrão" de células de grades distintas e localizando um mesmo ponto, o portão de entrada do MASP em São Paulo, com precisão da ordem de 10 metros:

- Célula na grade Geohash, expressa nativamente na base32, é o valor "**6g.yc.fq.f6**" (inteiro de 40 bits). Expressa na base4 é "**03033.33023.13112.13012**". Assim, percebemos que a quantidade de dígitos vai de oito na base32 para 20 na base4.
- Célula na grade S2geometry, expressa nativamente na base16, é o valor "**94ce59c94c4**". Expressa na base4 é "**22121302303210221 20**".[1] Novamente quantidade de dígitos cresce (de 11 para 19).

As representações em base16, base32 e base64 se encontram bem justificadas e fundamentadas na norma RFC 3548, mas o seu **alfabeto** pode sofrer pequenas adaptações conforme tradição de uso num dado contexto:

> **base4** (dígitos de **2 bits**): dígitos do alfabeto "*0123*".
> Exemplo de conversão da representação decimal para a de base4: 3=*3*, 4=*10*, 10=*22*, 15=*33*.
>
> **base16** (dígitos de **4 bits**): dígitos do alfabeto "*0123456789abcdef*", ou seja, 0=*0*, 1=*1*, 2=*2*, ..., 10=*a*, 11=*b*, 12=*c*, 13=*d*, 14=*e*, 15=*f*. Conforme a tradição dos usuários que estiverem adotando a base, permite-se o uso indistinto de letras maiúsculas, 10=*A*, 11=*B*, etc. Exemplo: 4=*4*, 10=*A*, 16=*10*, 60=*3C*, 90=*5A*.
>
> **base32** (dígitos de **5 bits**): existem diferententes alfabetos para as diferentes "tradições de uso", sendo de interessante ilustrarmos o alfabeto da "tradição Geohash" e a "tradição Internet" (RFC 3548). Conforme o tipo de tradição pode-se ou não fazer distinção entre maiúsculas e minúsculas.
>
> - **Base32-rfc**: dígitos do alfabeto "*ABCDEFGHIJKLMNOPQRSTUVWXYZ234567*", corresponde às 26 letras do alfabeto latino e os dígitos decimais 2 a 7, determinados pela **RFC** 3548.
>   Exemplo: 4=*E*, 10=*K*, 16=*Q*, 60=*B4*, 90=*C2*.
> - **Base32-ghs**: dígitos do alfabeto "*0123456789bcdefghjkmnpqrstuvwxyz*", corresponde aos dígitos decimais e as letras do alfabeto latino sem "aiol", conforme determinadas pela tradição **G**eo**h**a**s**h.
>   Exemplo: 4=*4*, 10=*b*, 16=*h*, 60=*1w*, 90=*2u*.
>
> **base64** (dígitos de **6 bits**): dígitos do alfabeto "*ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_*".
> Exemplo: 4=*E*, 10=*K*, 16=*Q*, 60=*8*, 63=*_*, 65=*BA*, 90=*Ba*.

Apesar da regra simples "quanto maior a base mais compacta a representação", existe um balanço entre grau de compactação e demais necessidades: códigos alfanuméricos (uso somente de dígitos decimais e letras do alfabeto latino como caracteres), conveniência de uso (fala ou escrita em diferentes meios), demanda por resiliência, acesso à hierarquia, e outras. Este balanço em geral resulta na escolha da **base32**.

Alinhamento de representações

Ao usar como referência a representação na **base4** (2 bits por dígito), a tabela abaixo ilustra através do alinhamento vertical os casos onde há ou não "compatibilidade" de outras bases com a base4. O comprimento em bits do dígito de uma base indica exatamente a quantidade de informação passível de ser transmitida com um dígito daquela forma representação. Pode-se supor que o valor numérico típico de um identificador de célula apresenta entre 20 a 50 bits, e esse valor pode ser representado em qualquer base.

As alternativas de representação do mesmo número em diferentes bases serão ou não compatíveis entre si conforme o número de bits do seu dígito. Por serem potências de 4 (16=4² e 64=4³), e por fazerem uso de uma

[1] Foi excluído o índice de face de cubo. A função dentro da biblioteca padrão é S2Cell_id::toString, oferece uma representação mais legível da hierarquia, base4. É por vezes apelidada de key ou Quadkey. Como o S2 inicia pela projeção do globo terrestre num cubo, o primeiro item de Quadkey é o índice de face de cubo (no caso de São Paulo está dentro da face 4).

quantidade de bits por dígito que é múltipla de 2, as bases 16 e 64 sempre possuem alinhamento com a base4.

Tabela 1

Dígitos de **2 bits** (base4) original da hierarquia	
Dígitos de **5 bits** (base32)	
Dígitos de **4 bits** (base16)	
Dígitos de **6 bits** (base64)	

As representações base64 e base16 ficam alinhadas entre si a cada dois dígitos da base64. O alinhamento da base32 com a base4 se dá a cada 10 bits, portanto **a cada 2 dos dígitos base32**. Similarmente a cada dois dígitos da base32 há alinhamento com a base16.

Será convencionado no presente ensaio que o termo “valor do identificador da célula” pode sempre ser associado à representação binária (base2), e que o nível *L* da grade onde a célula é espacialmente representada pode sempre ser associado ao número de dígitos do valor do identificador na base4, $L(id) = D_4(id)$.

Demonstrando algebricamente: sejam *id* o valor do identificador de uma célula, expresso por N_{id} bits, e $D_b(id)$ o número de dígitos de *id* na base *b*. Se o valor de *id* for representado na base32 então teremos $N_{id}=D_{32}(id)\square 5$. Equiparando ao número de bits do valor de *id* na base4, teremos $D_4(id)\square 2 = D_{32}(id)\square 5$. Como $D_{32}(id)$ é a nossa incógnita, batizando-a de *y* e lembrando que $D_4(id)$ é o nível hierárquico, *L*, podemos reescrever $2L = 5y$, ou seja, $\boldsymbol{y = 2L/5}$. Não existe portanto valor inteiro de *y* para um nível *L* qualquer, apenas para múltiplos de 5, ou seja para *L*=5, *L*=10, etc. e, pelo fator 2, teremos sempre *y* par.

Como o problema não é o geocódigo mas apenas a representação espacial da célula identificada por ele, veremos ao longo do ensaio que eventualmente, pela falta de solução exata, podemos flexibilizar admitindo *y* ímpar com níveis hierárquicos *L*=5/2=2½, *L*=7½, etc. de forma a obtermos valores quaisquer 1, 2, 3, etc. para *y*, ajustando valores de *L* conforme a demanda:

$\mathbf{D_{32}(id) = L} \square 0{,}4$
$1 = 2{,}5 \square 0{,}4$
$2 = 5{,}0 \square 0{,}4$
$3 = 7{,}5 \square 0{,}4$
$4 = 10 \square 0{,}4$
...

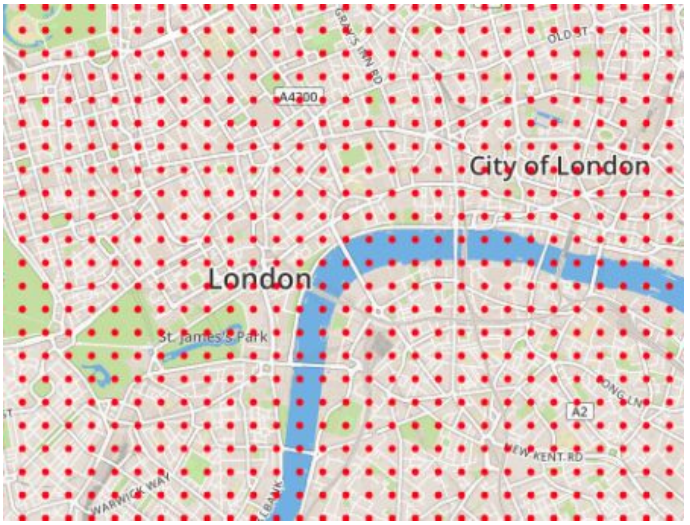
A noção de “nível hierárquico meio” pode ser extendida para qualquer nível, não se restringindo aos múltiplos de 2,5 das soluções da base32. Os algoritmos farão uso também dos níveis intermediários da base4.
A generalização do nível *L*+½ e sua interpretação espacial serão apresentandos neste ensaio como elementos centrais da proposta.

Representação e indexação espaciais

A seguir uma breve introdução aos conceitos relativos à geometria da grade e à localização das células.

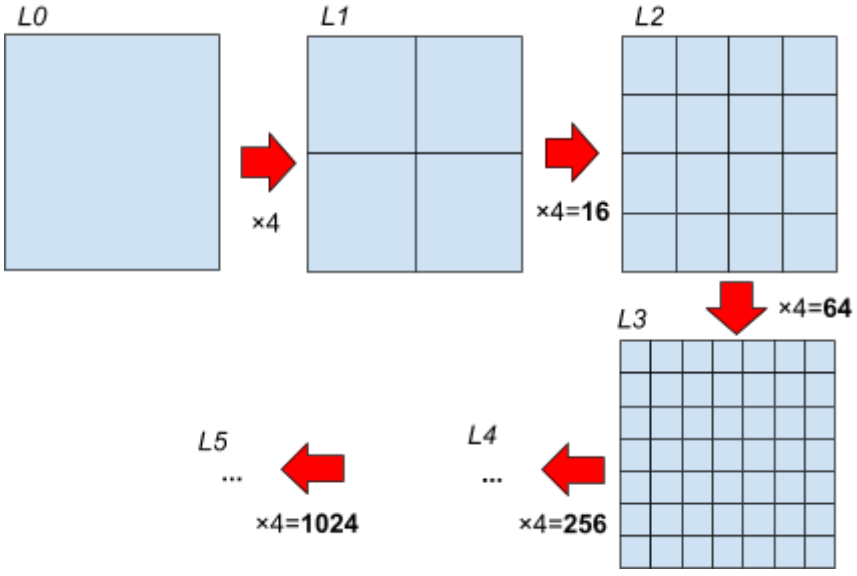
Representações de grades como “mosaico de células” ou como “grade de pontos” (centros de célula) são distintas porém intercambiáveis.

Também será convencionado neste ensaio que que as células de grade sempre possuem um cone de latitude e longitude único (coordenadas usuais associadas por exemplo ao centróide da célula), sem projeção implícita.



Grade hierárquica como partição recursiva do espaço em 4 regiões

Os indexadores baseados em Curva de Hilbert e Curva de Ordem Z são fundamentados na partição uniforme do espaço em 4 regiões. Outros exemplos, de uso prático mais reduzido, tais como a Curva de Peano, também fazem uso da mesma estratégia de partições sucessivas em quatro regiões.

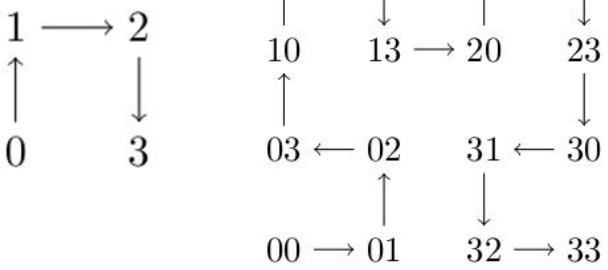


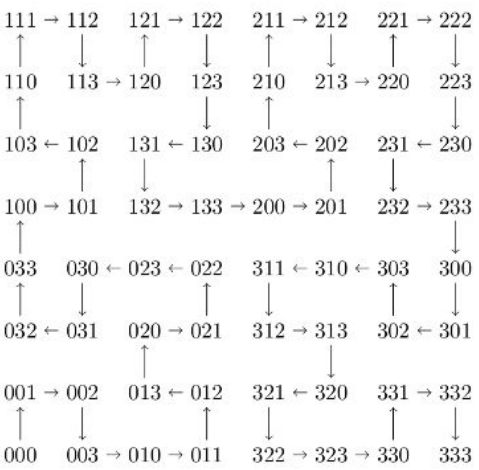
Assim, por serem indexadores espaciais, há sempre uma correspondência exata entre o valor do identificador e a posição no espaço de uma única célula (de nível *L*) na grade.

Representação hierárquica do indexador

Tomemos como exemplo a Curva de Hilbert. O seu indexador primário (sem recursão) é um valor **inteiro de zero a três**, ou seja, um **dígito** expresso na **base4** (2 bits). Na partição hierárquica usual para a representação das células de uma grade discreta indexada por curva de preenchimento, o primeiro dígito representa o nível L_0, o segundo dígito o nível L_1 e assim por diante. Obtém-se desse modo a indexação usual:

Figura-1



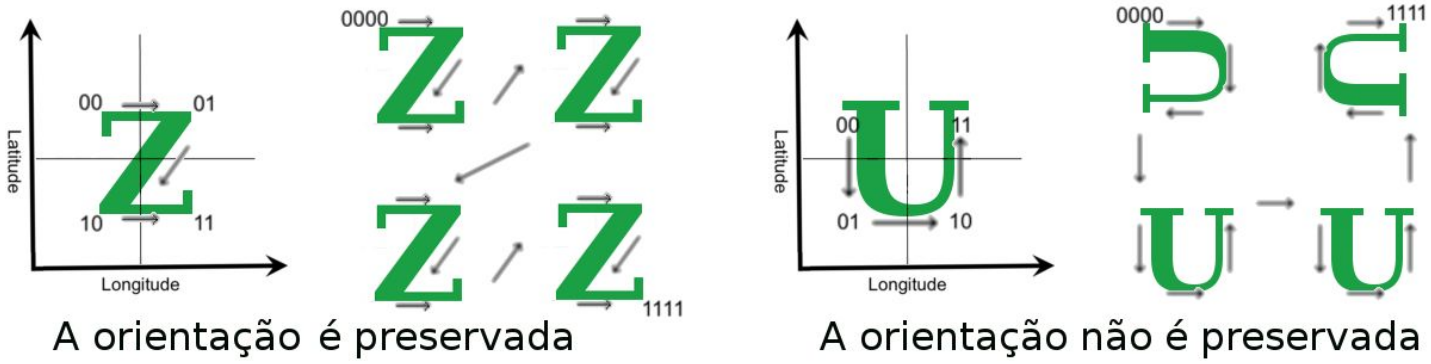


A célula da grade de nível L_1 localizada no canto superior direito é identificada por “2”, precisa só de 2 bits; já uma outra célula, da grade do nível L_3 localizada na mesma extremidade, é identificada por “222”, com 6 bits.

Preservação ou não da orientação no indexador

A Curva-Z é o indexador mais popular por ser fundamentado em algoritmo mais simples, baseados no entrelaçamento de dígitos,[2] em particular do entrelaçamento binário. Com uma escolha adequada da orientação inicial, garante-se que o primeiro bit seja a latitude e o segundo bit a longitude.

Figura-2



A orientação do “Z” ilustrada acima é utilizada por exemplo pelo Bing Tile System.
No caso do Geohash, cuja curva de preenchimento também é classificada como Ordem-Z, segue a rigor a ordem da letra “N” (ver apêndice A1.1), com primeiro quadrante (“00”) na latitude inferior.

No caso da curva de Hilbert, percebe-se que a escolha da orientação inicial não é preservada pelos níveis hierárquicos seguintes, nem é uniforme ao longo da cobertura num mesmo nível hierárquico. Isso não impede a construção de grades uniformes: basta que as células sejam simétricas (ex. quadrados ou losangos), para compensar a sua não-uniformidade dentro de um mesmo nível hierárquico.

Conforme veremos, quando a orientação é preservada a metodologia sugerida neste ensaio é mais facilmente aceita pelo usuário final. Do ponto de vista metodológico, portanto, a preservação da orientação é um critério classificatório da curva de preenchimento.

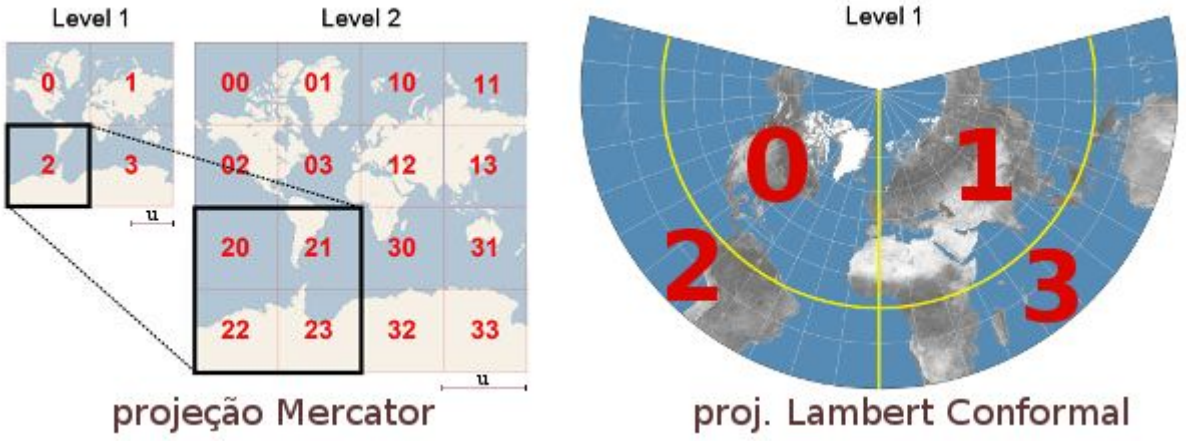
[2] O Geohash como resultado do entrelaçamento binário de latitude e longitude é apresentado de maneira mais didática em https://mapzen.com/blog/geohashes-and-you/

Independência do identificador quanto à base, escala ou projeção

A representação do valor do identificador de célula como geocódigo corresponde à eleição de uma base numérica como padrão. Apesar de ser necessariamente uma dentre várias opções, é interessante que existam opções, para que haja potencial de otimização na escolha, e para que uma mesma infraestrutura de grade seja adotada como fundamento para mais de um padrão de geocódigo ou de prefixo de códigos com outras finalidades. Pode-se considerar que se o número de bits do identificador

representados for conservado então o geocódigo é independente da base e pode por exemplo ser expresso em binário e novamente na base usual sem perda de informação.

Figura-3



Tomando-se como exemplo geocódigos base4 do *Bing Tile System*, a ilustração destaca diversos aspectos discutidos:

- a orientação “Z” (traçado unindo centros de célula na sequência 0 a 3) é preservada;
- a escala do mapa, assim como a hierarquia da grade, não afetam a identificação das células;
- a projeção utilizada no mapa não afeta a identificação das células;
- a união das células de mesmo prefixo (no caso prefixo “2” em “20”, “21”, “22” e “23”) corresponde à célula de nível *L* mais baixo, identificada pelo prefixo.

Com estas restrições em vista, podemos expressar a noção de "independência" como sendo a liberdade de escolha entre as opções disponíveis:

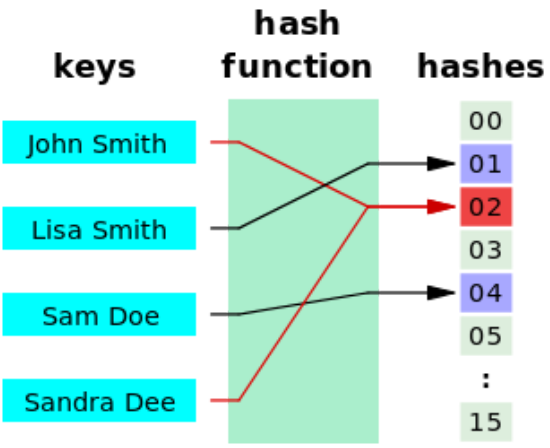
- *Independência de base*: requer um conjunto *B* definindo as opções de base válidas, além da base4. Por exemplo um identificador de 36 bits pode demandar *B*={8, 16, 64}. A conversão de volta para valor binário partindo de qualquer uma das bases de *B* resultará nos mesmos 36 bits.

- *Independência de escala*: a célula identificada pelo geocódigo é sempre a mesma, é indiferente à escala do mapa escolhido para ilustrá-la.

- *Independência de projeção*: a célula identificada pelo geocódigo é indiferente à projeção adotada no mapa escolhido para ilustrá-la.
  Outras propriedades relevantes são também preservadas: a cardinalidade de cada nível hierárquico e a topologia da sua grade, ou seja, a orientação das células e as relações de vizinhança são garantidas, bem como o valor de área ocupada pela célula sobre o terreno ou elipsóide de referência.[3]
  PS: a forma geométrica da célula é afetada pela projeção (ex. um quadrado muda para retângulo), assim como o grau de uniformidade geométrica da grade (ex. muda com a latitude).

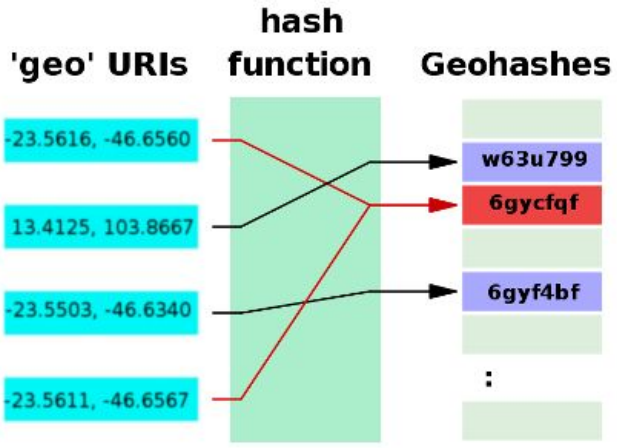
[3] Foi convencionado no início desta seção do ensaio que toda célula possui um centróide e esse ponto tem coordenadas de latitude e longitude usuais. A área é inferida a partir do cone geodésico e sua projeção no elipsóide, tipicamente WGS84.

Algoritmos de *hash*, geohash e grade



O termo *"hash function"* tem sido amplamente utilizado nos anos recentes, em geral mantendo sua conotação matemática, centrada no conceito de função de *hash*: uma caixa preta que transforma "chaves" em identificadores mais compactos e computacionalmente adequados, ditos "hashes".

Na ilustração ao lado, tomada da Wikipedia, a função hash mapeia nomes para inteiros de 0 a 15, que são números inteiros compactos e mais adequados como identificadores por exemplo num banco de dados. Não há garantia de unicidade, pode "dar azar" como nas chaves "John Smih" e "Sandra Dee", que colidiram,[4] resultando no mesmo *hash*.



Em bancos de dados e diversas outras aplicações a chave utilizada é um *metadado*, e justamente o metadado mais importante para a caracterização de entidades geográficas são suas coordenadas, tipicamente coordenadas compatíveis com o protocolo "geo" da internet, conhecido como geoURI. O museu MASP, por exemplo, está em `geo:-23.56162,-46.65600`

Usando como função *hash* o algoritmo Geohash podemos ter algum controle sobre as colisões: serão identificadas por um mesmo *hash* as geoURIs relativas a posições geográficas vizinhas, conforme ilustrado ao lado com a localização dos portões da frente e do fundo do MASP. O mesmo princípio vale para a reversibilidade: quando utilizada para fins de identificação, a função *hash* comporta função inversa, preservando integralmente o metadado; quando utilizada para fins de agrupamento ou classificação,[5] parte da informação relativa à geoURI pode ser descartada.

O termo "geohash" portanto pode ser adotado como substantivo comum, designando um conceito independentemente do nome próprio Geohash: designa "função de *hash* especializada em geoURIs", uma subclasse das funções de *hash*.

Já o nome próprio adotado para batizar a metodologia que se apresenta, Geohash Generalizado, é tanto uma referência ao substantivo comum como uma homenagem ao algoritmo (nome próprio) e a todos os outros algoritmos existentes que compartilham as mesmas propriedades matemáticas, descritas no presente ensaio.

Para evitar confusão e orientar a metodologia de avaliação do "mercado de algoritmos", é suposto que todos possam ser modularizados nos seguintes sub-algoritmos:

- Algoritmo da **grade**: recebe como rótulo o nome da grade (DGG) ou tecnologia de grade adotada. Geohash, S2geometry, PlusCode, Eircode, MapCode, e centenas de outras são exemplos.
- Algortimo para obter o **identificador** de uma célula da grade a partir da geoURI: pode ser separado em duas partes, prefixo (*id0*) e sufixo (*id*). Em sendo uma DGG hierárquica, com divisão recorrente em 4 regiões, é exigido que o sufixo seja um número inteiro com hierarquia expressa em base4.
- Algoritmo de geração do **hash do identificador** da célula: prefixo e sufixo concatenados, com prefixo livre (qualquer função hash) e sufixo expresso em base32, com alfabeto livre.

---

[4] Os conceitos de *colisão* e "resistência à colisão" são formalizado matematicamente como probabilidade na Teoria das Funções Hash, principalmente no arcabouço das funções hash criptográficas.
[5] Neste tipo de aplicação caracteriza-se como "função hash de semelhança", ou "Locality-sensitive hashing".

Elementos da proposta e hipóteses de trabalho

É proposta no presente ensaio uma metodologia que permite o uso da base32 em hierarquias base4. É também proposta a generalização do conceito de *geohash* e das funções de *hash* de comportamento semelhante ao algoritmo Geohash.

Apresentação do problema

A suposta impossibilidade de uso da base32 com indexadores baseados em Curva de Hilbert (ex. S2geometry) e os problemas com o Geohash são revistos e destacados no Apêndice-1.

Grades hierárquicas podem apresentar quatro tipos distintos de problema, simultaneamente ou não, quando fazendo uso da base32:

1. Impossibilidade de representação espacial da célula identificada por número ímpar de dígitos base32.
2. Perda de orientação no indexador quando o geocódigo base32 exibe número de dígitos ímpar.
3. Forma ou proporções geométricas das células individuais mudando entre diferentes níveis hierárquicos.
4. Forma geométrica coletiva das células (ex. orientação relativa) mudando num mesmo nível (quebra de uniformidade) ou entre níveis diferentes, dificultando computacionalmente a plotagem da grade.

Hipóteses de trabalho

A base32 simplesmente não tem representação exata[6] na base4: para “fazer o proibido” é preciso antes torná-lo admissível ou saber quando (em qual contexto) será admitido.

A noção central para o presente ensaio é a possibilidade de uso da base32 independente do tipo de curva indexadora. Conforme será descrito, deixa de ser proibido quando é admitida a construção de uma grade degenerada (mais especificamente “grade nível *L*+2½”) para a visualização espacial dos identificadores base32.

As hipóteses de trabalho, a rigor, são aquelas vinculadas à flexibilização da representação espacial. Considerações gerais:

1. São de maior interesse e relevância as **grades do tipo DGG hierárquica com identificador de célula dado por um número natural** sequencial (começando do 0 ou 1 e sem “saltar” valores).

2. O valor de cada identificador tem seu **número de bits proporcional ao nível hierárquico da célula** que identifica. Conforme o tipo de representação aceito, o nível hierárquico será determinado a cada bit ou cada dois bits. Se for “a cada bit” o geocódigo pode ser expresso em qualquer base. Se for “a cada dois bits” diremos o geocódigo é estruturado na base4.

Hipóteses de trabalho para os identificadores de grades estruturadas na base4:

3. O identificador é um **número sequencial determinado por uma curva de preenchimento**, garantindo que a maioria (70% ou mais) das células espacialmente vizinhas tenham na sua representação base4 um mesmo prefixo. E vice-versa, sempre (100% dos casos) que dois identificadores possuírem mesmo prefixo, estarão identificado células próximas uma da outra.
4. A **forma geométrica das células** de um mesmo nível hierárquico, quando projetadas no plano, é “satisfatoriamente **uniforme**”.
5. A **orientação das células** de um mesmo nível hierárquico não precisa ser a mesma, mas deve ser **consistente**, seguindo uma regra bem definida.
6. A **forma geométrica das células de níveis hierárquicos diferentes** não precisa ser a mesma, mas apresentam “**similaridade** satisfatória”.

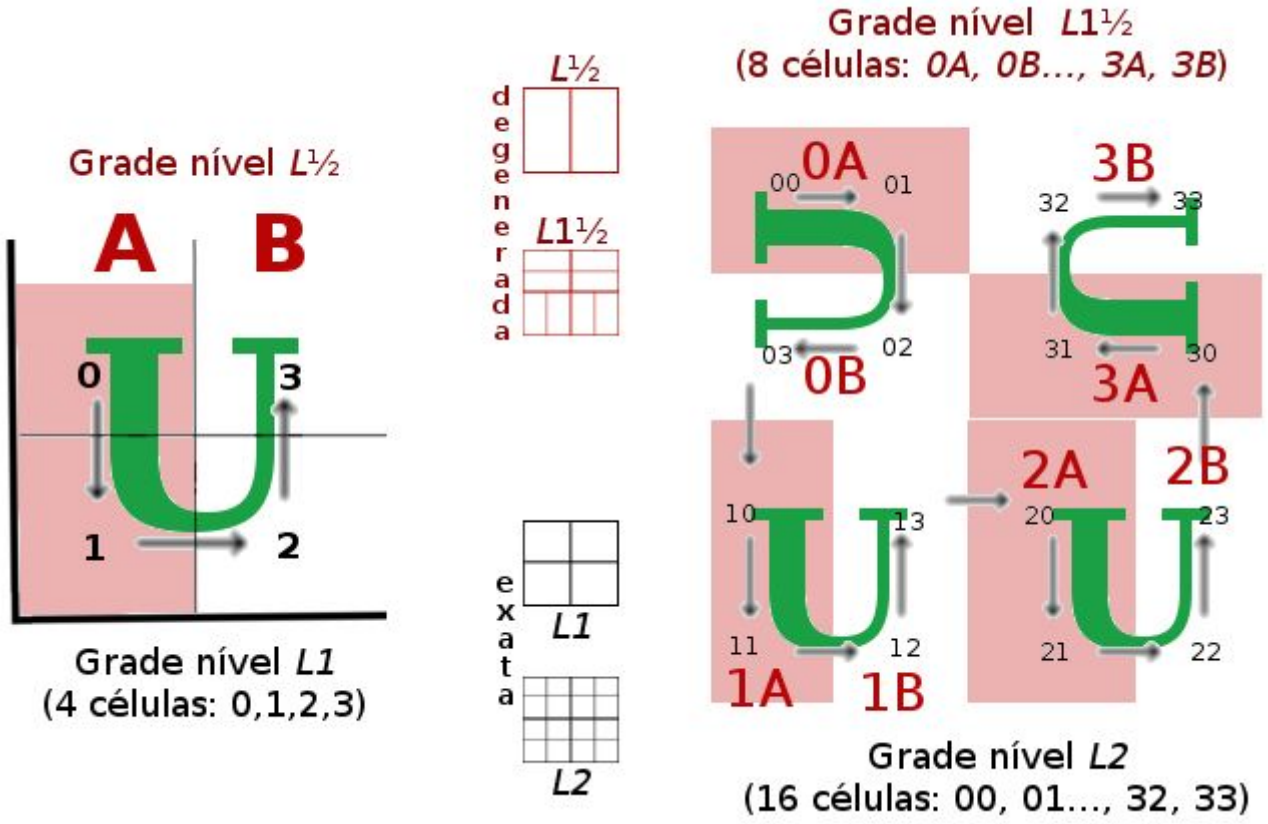
[6] Conforme destacado na tabela-1.

É suposto que existam casos práticos onde as noções de “satisfatório” e “similar” ficam melhor definidas e a base32 torna-se admissível.

Solução da grade degenerada

Pode-se conceber, do ponto de vista espacial, uma hierarquia intermediária entre dois níveis consecutivos, para então dotar códigos incompatíveis com a base4 de uma representação espacial. Para tanto é necessário formalizar o processo de reindexação e a representação espacial dos novos identificadores.

Figura 4



A ilustração acima mostra a formação da grade degenerada na curva de Hilbert:
apenas o último dígito da base4 é submetido ao processo de degenerescência, reduzindo pela metade o número de células da grade, ao unir células vizinhas pelo novo indexador ***j* = *floor*(*i* / 2)**.

A reindexação dá origem ao nível hierárquico intermediário. No exemplo:

- **Nível *L*½**: reindexação do nível *L*1. A união das células “0” e “1” de *L*1 resulta na célula “A” de *L*½; a união das células “2” e “3” de *L*1 resulta na célula “B” de *L*½.
- **Nível *L*1½**: reindexação do nível *L*2. A união das células “00” e “01” de *L*2 resulta na célula “0A” de *L*1½; a união das células “02” e “03” de *L*2 resulta na célula “0B” de *L*1½; e assim por diante. O conjunto completo de células da grade degenerada é {“0A”, “0B”, “1A”, “1B”, “2A”, “2B”, “3A”, “3B”}.

Grades de qualquer outro nível (*L*3, *L*4, etc.) podem ser submetidas ao mesmo procedimento de reindexação e redução do nível em meio para a obtenção da grade degenerada, *L*2½, *L*3½, etc.

Na ilustração as células de nível hierárquico intermediário são rotuladas por dígitos “A” ou “B” (ao invés de 0 ou 1) para destacarem-se da hierarquia base4 depois do “nível meio” (*L*½), onde se misturam num mesmo código.

Outra forma de se especificar a grade degenerada é pela análise do último bit do identificador. As células “0”, “1”, “2” e “3” de *L*1, em binário são respectivamente “00”, “01”, “10”, “11”. A grade degenerada surge da operação lógica *ou* aplicada ao último bit do identificador da célula: “A” é “00” ou “01”; “B” é “10” ou “11”. E exatamente a mesma operação aplicada aos identificadores do nível *L*2 para se obter o nível *L*1½ em binário:
“00A” é “0000” ou “0001”; “00B” é “0010” ou “0011”; “01A” é “0100” ou “0101”; etc.

A redução do comprimento do identificador em 1 bit equivale a dividir o indexador por 2, e portanto reduzir pela metade o total de células na grade. O resultado é equivalente à fórmula *j*=floor(i/2), mas o tratamento de bits torna a implementação do algoritmo mais simples. Expressando em linguagem Javascript, supondo que *j* seja o indexador da grade degenerada e *i0* um identificador qualquer na grade usual, tanto *j* como a escolha da célula *i1* a ser unida a *i0* são obtidos a partir de operações de rotação de bits:
***j* = *i0* >> 1**; ***i0* = *j* << 1**; ***i1* = *i0* + 1**.

Como a Curva de Hilbert não preserva a orientação, o efeito topológico é que células com mesmo final podem se tocar (na ilustração as células “0B” e “1B” assim como “2A” e “3A”), perde-se a garantia de intercalação espacial. Se a célula é assimétrica, o efeito também acarreta perda de uniformidade na grade (os centróides das células deixam de estar uniformemente espaçados como numa grade perfeitamente retangular). Apesar dessa perda, o aspecto da grade de nível maior é ainda aceitável em diversas aplicações como uniforme, quando vista de longe e com células representadas apenas pelos pontos centrais.

A curva de Ordem-Z preserva orientação e portanto a uniformidade, mesmo quando as células são assimétricas: fato amplamente notado no Geohash, e ilustrado no apêndice A1.1.1.

Obtenção da célula degenerada de um ponto qualquer

Dada uma coordenada latitude/longitude, qual a célula correspondente na grade degenerada?

O procedimento é simples, sejam *k* um inteiro e *Lk*½ o nível hierárquico da grade:

1. Obter o identificador da célula pelo método usual, *i0=encode(lat,lon,precisão)* para o nível ***Lk***.
2. Calcular o índice ***j***=*floor*(*i0*/2).
3. Se *floor*((*i0*+1)/2) resulta no mesmo *j*, usar ***i1***=*i0*+1 senão usar ***i1***=*i0*-1.
4. Obter o polígono das células *i0* e *i1*.
5. Obter enfim o polígono da célula *j*, como união geométrica dos polígonos de *i0* e *i1*.

Os passos 2 e 3 podem ser simplificados conforme explicado. Expressando em um só passo com operadores de rotação de bits Javascript:
j=i0>>1; i0 = j<<1; i1 = i0+1.

O passo 4, pode requerer uma biblioteca mais extensa em função do algoritmo utilizado. Algoritmo Geohash por exemplo não requer pois a geometria da célula é obtida da grade latlong, enquanto S2geometry tem geometria específica. Todas elas podem ser expressas como polígono simples convexo em GeoJSON.

O passo 5 requer apenas dissolução das bordas. Como são polígonos simples, convexos e contiguos, pode-se usar um algoritmo mais simples, tal como “Sutherland-Hodgman polygon clipping” ou Greiner-Hormann.

Obtenção do geoURI e GeoJSON de uma célula

Dado um geocódigo, qual a sua coordenada latitude/longitude central? E o GeoJSON da célula?

O Geohash que cobre a cidade de São Paulo por exemplo, "6g", do nivel *L*5, tem a sua representação de cinco dígitos na base4 é exata, "03033". Por outro lado "6gy", do nivel *L*7½, não tem representação exata, mas podemos representar a base4 com um bit a menos como "0303333A".

Para o primeiro caso, “6g”, um código base32 **com número par** de dígitos, o valor das coordenadas deve ser obtido da forma usual já fixada pela biblioteca da DGG utilizada. Por exemplo na biblioteca de referência Geohash é a função “*decode*()” que devolve as coordenadas. Na biblioteca S2geometry (versão Javascript

adaptada), como recodificamos em base4, a função que devolve as coordeadas é “*keyToLatLng*()”.

Para o segundo caso, “6gy”, um código base32 **com número ímpar** de dígitos, é necessário decompor o dígito exótico “A” da representação base4 em seus reais componentes:

"6gy" base32 = “0303333A” = “03033330” ou “03033331”

Novamente voltando à respectiva biblioteca DGG obtemos as células “03033330” e “03033331”, e por fim calculamos o centro geométrico. Sumarizando, o **procedimento** consiste nos seguintes passos:

1. Obter a repreentação em base4 do hash, *hb4*.
2. Se o último dígitio de *hb4* é normal, usar diretamente *LngLat* = *decode*(*hb4*);
   senão (se último dígito de *hb4* é exótico), decompor e obter
   LngLat = *centroide*(*decode*(*hb4_x*), *decode*(*hb4_y*)).

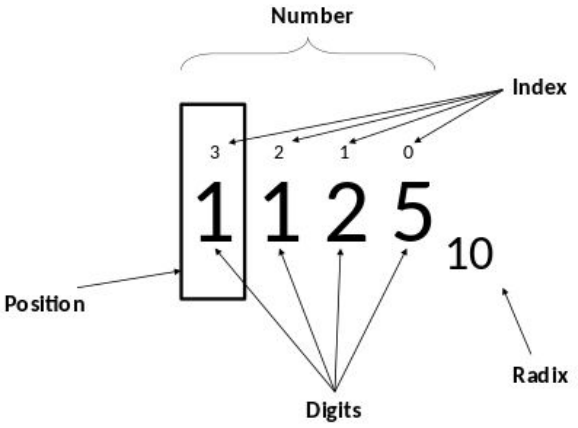
A regra geral de decomposição, supondo “A” e “B” os dígitos exóticos da base4, é dada por *decompor*(“A”)={“0”,”1”}; *decompor*(“B”)={“2”,”3”}. Exemplos:

"6gy" base32 = "0303333A" base4 = “03033330” ou “03033331”
"6gz" base32 = "0303333B" base4 = “03033332” ou “03033333”

Quanto ao polígono GeoJSON relativo a uma célula da DGG, nem sempre é fornecido pela biblioteca DGG. No caso da grade Geohash, que vem ancorada à grade latlong, as bibliotecas fornecem a BBOX da célula, que é facilmente convertida em retângulo. Já bibliotecas tais como S2geometry carecem de meio direto, requerendo adaptações mais complexas, baseadas na posição das células vizinhas.

A união geométrica de duas células expressas como GeoJSON pode ser facilmente calculada, e como já destacado acima, existem formas otimizadas. Por fim, quanto ao centroide de duas células, é em geral será mais simples calcular como média das coordenadas de cada uma delas, ao invés do polígono. A orientação da célula-união, por vezes necessária em representações da célula como ponto e dispersão (ver ilustração do apêndice A1.3.3) também pode ser obtida da linha que une os dois pontos.

Numeração posicional mista *plus-HalfDigit*



Os sistemas de numeração binário, decimal, hexadecimal, etc. que utilizamos tradicionalmente são subclasses da notação posicional de representação numérica. Os conceitos de *dígito*, *posição* (ou índice do dígito) e *base* (ou *radix*) fazem parte das convenções estabelecidas pela notação.

Cada número tem um significado matemático exato, todavia esse significado é extendido conforme contexto de uso. No caso da grade degenerada, há demanda por **acréscimo de um bit**, ou seja, concatenar mais um dígito para formar um novo número, mas o valor desse dígito representará apenas a metada das possibilidades do dígito não-binário (ou conjuntos associados à metade das possibilidades). Exemplo: acrescentando os valores “A” e “B” aos possíveis valores 0-9 usuais do dígito decimal, e associando “A” aos valores 0-4 e “B” aos valores 4-9. Supondo os números 100 até 109 e trocando o dígito da posição zero por “A” ou “B” teremos, nesta nova base batizada de base10**h**,

10A = 100 ou 101 ou 102 ou 103 ou 104
10B = 105 ou 106 ou 107 ou 108 ou 109.

A sintaxe "10+A" e "10+B", inspirada no PlusCode, destaca melhor, podendo ser interpretada como "concatenação ou" dos respectivos conjuntos *10+{0,1,2,3,4}* e *10+{5,6,7,8,9}*. Opcionalmente os símbolos "⬒" (*plus up*) e "⬓" (*plus down*) forneceriam “10⬒” e “10⬓”, evitando confusão com dígitos hexadecimais.

Redução e reconstrução por prefixo

O identificador de célula dentro de um contexto conhecido não precisa ser tão longo quanto o identificador de célula global. Uma célula, por exemplo identificada pelo Geohash “6gy2”, dentro de uma célula maior, por exemplo “6g”, pode ser citada como “célula ‘y2’ dentro da célula ‘6g’”. Essa é a lógica da operação apelidada de ***redução*** *por prefixo*. A operação inversa é a ***reconstrução*** *por prefixo*.

Na prática **o *prefixo* pode pertencer a qualquer nivel hierárquico** (inclusive intermediário), perdendos-se compatibilidade da representação base32 com a grade global na célula reduzida, mas garantindo maior eficiência na escolha da célula envolvente de um polígono arbitrário. A operação portanto é realizada com ambas, célula envolvente e interior, representadas como base4h. O resultado final da redução, ainda em base4h, pode ou não ser passível de representação exata na base32 — restrições de nível máximo e mínimo na célula interior garantem o seu “encaixe” na base32.

Antes de apresentar cada um dos algoritmos, importante destacar a operação utilizada por ambos, que é a conversão entre base4h e base2. Em Javascript sem otimizar por rotação de bits:

```
function base4h_to_bin(xB4h) {
      var good=xB4h.match(/^([0-3]+)([⬒-⬓])?$/u);
      var lastHalfDigit = good[2]? good[2]: '';
      var lastBit =  lastBit? (  (lastBit=='⬒')? '0': '1'  ): '';
      var xB4 = good[1];
      var b4_len = xB4.length;
      var b2_len = b4_len*2 + ((lastBit==='')?0:1);
      var xB2 = (bigInt(xB4,4).toString(2) + lastBit).padStart(b2_len,'0');
      return { b2:xB2, level:b4_len+lastHalfDigit?0.5:0 }
}
```

O corte de prefixo então é realizado pelo seguinte algoritmo, onde p é o prefixo e o retorno [cut,level]:

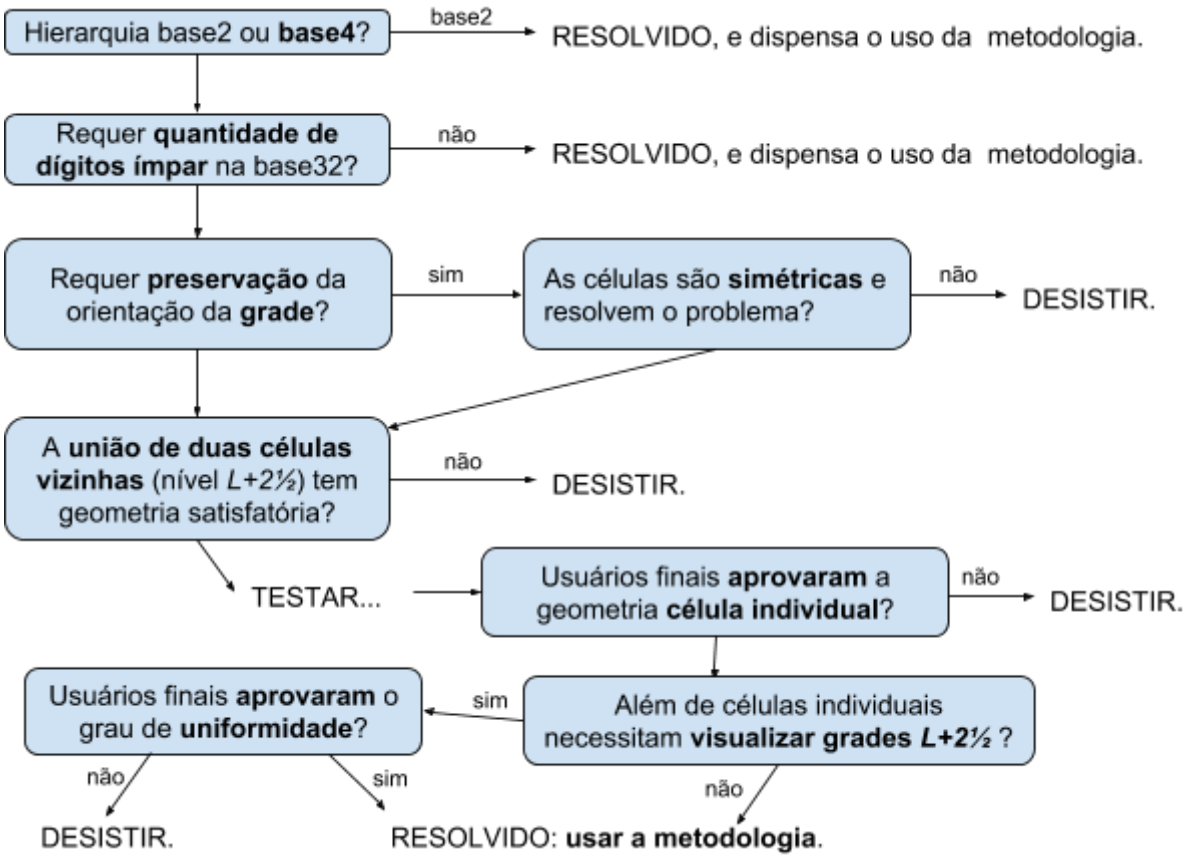
```
function base4h_cutPrefix(p,x) {
      p = base4h_to_bin(p);
      x = base4h_to_bin(x);
      var x_root = x.b2.slice(0,p.b2.length);
      var resB2_len = x.b2.length - p.b2.length;
      var res = x.b2.slice(p.b2.length);
      var resB4h_len = resB2_len/2.0;
      if (Number.isInteger(resB4h_len))
        return [bigInt(res,2).toString(4).padStart(resB4h_len,'0'),resB4h_len];
      else {
        var btoh = {"0":"⬒","1":"⬓"};
        var bit = res.slice(-1);
        var res = res.slice(0,-1);
        var len = Math.floor(resB4h_len);
        return [bigInt(res,2).toString(4).padStart(len,'0') + btoh[bit], resB4h_len];
      }
}
```

O algoritmo de reconstrução faz uso de procedimentos similares.

Método

O fluxograma abaixo resume a heurística de decisão proposta pelo presente ensaio.

Figura 5



Existe hoje mais de uma dezena de opções tecnológicas para a geração de geocódigos hierárquicos. Conforme a combinação de tecnologia (representação numa DGG hierárquica de 4 regiões recursivas) e algoritmo de indexação espacial (curva de preenchimento com idendexador exibindo hierarquia em base2 ou base4), surge um número ainda maior de possibilidades para a decisão de padrão de geocódigo a ser adotado numa data aplicação. O fluxograma ajuda a selecionar e decidir quais cobinações submeter à metodologia proposta.

Na figura-5 a noção de “aprovado pelo usuário final” pode ser também uma verificação de performance. Estima-se, por exemplo, que em aplicações onde haja demanda pela plotagem da grade *L+2½* numa tela inteira, o custo de processamento gráfico pode ficar muito acima do esperado — sem otimização do algoritmo, a representação visual de todas as células simultâneas para a formação do mosaico da grade, requer o mesmo custo de CPU que a plotagem de uma-a-uma das células.[7]

[7] Em geoprocessamento essas aplicações são raras e opcionais (por exemplo opção na ferramenta de edição pontos), de modo que nas aplicações investigadas, como os pontos de endereçamento, não compromete a solução. Do ponto de vista da interface, também é possível editar o ponto com visualização de grade alternativa, mostrando a grade apenas nas vizinhanças do ponto.

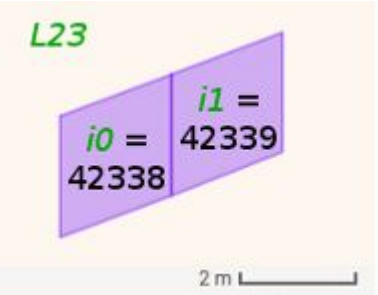
Algoritmos gerais

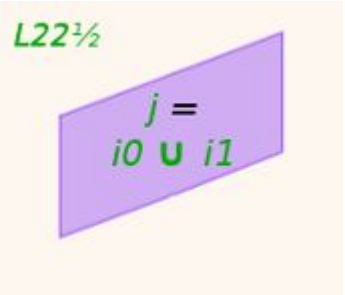
A seguir um resumo passo-a-passo, sumarizando os procedimentos já descritos e justificados.

Obtenção do identificador de célula e sua geometria

Também denominada “codificação”, o algoritmo básico para se desenhar a **célula identificada pelo valor *j*** consiste nos seguintes passos:

1. Quando aceita-se a grade como uma estrutura hierárquica expressa em base2, basta usar a célula usual do nível correspondente.
2. Quando a estrutura hierárquica da grade é expressa em base4:
   - 2.1. Quando o número de dígitos de base32(*j*) **é par**, basta usar a célula usual do nível correspondente. Cada vez que isso ocorre diremos que estamos usando o nível *L+0*.
   - 2.2. Quando o número de dígitos **é ímpar**, o valor *j* deve ser desenhado na grade de nível ***L+2½***, que consiste em fazer uso da grade do **nível *L+3*** da seguinte forma:
     - 2.2.1. Obter os valores ***i0*** e ***i1*** de identificação de células do nível *L+3*, que sejam solução para a equação ***j* = floor (*i* / 2)**
     - 2.2.2. Plotar na grade *L+3* a **união geométrica**[8] das células *i1* e *i2* como uma só célula, a qual será então empregada como **célula *j* do nível *L+2½***.





Exemplificando com os valores da tabela-2: na coluna *base32*, entre as células “KL” e “KLHN”, poderíamos acrescentar a célula identificada por *j*=“KLH”. Ela pertence à grade do nível L22½ e em decimal temos *j*=21169, então os valores
j = floor (*i* / 2)
que resultam neste mesmo *j* são *i0*=42338 e *i1*=42339. A célula *j* pode por fim ser desenhada a partir união das células *i0* e *i1*. Para uma implementação eficiente, de qualquer forma, conforme já explicado anteriormente, o melhor é manter os identificadores em binário e a solução binária para obtenção de *j*, *i0* e *i1*.

O procedimento pode ser repetido para diversos valores de *j*. Quando for repetido para um conjunto específico *C*, dizemos que o desenho é a um “conjunto **cobertura**”. Quando for repetido para o mapa inteiro, dizemos que é uma **grade** do nível *L+2½*.

Algoritmos de construção de grades podem ser eventualmente otimizados, como no caso das grades Geohash, dispensando a operação “célula a célula” descrita acima: o algoritmo proposto deve ser entendido como uma especificação formal, não como método de implementação final.

Decodificação do geocódigo

O algoritmo já foi apresentado na seção “Obtenção do geoURI e GeoJSON de uma célula” e uma das estratégias de implentação é apresentada no Apêndice 2.

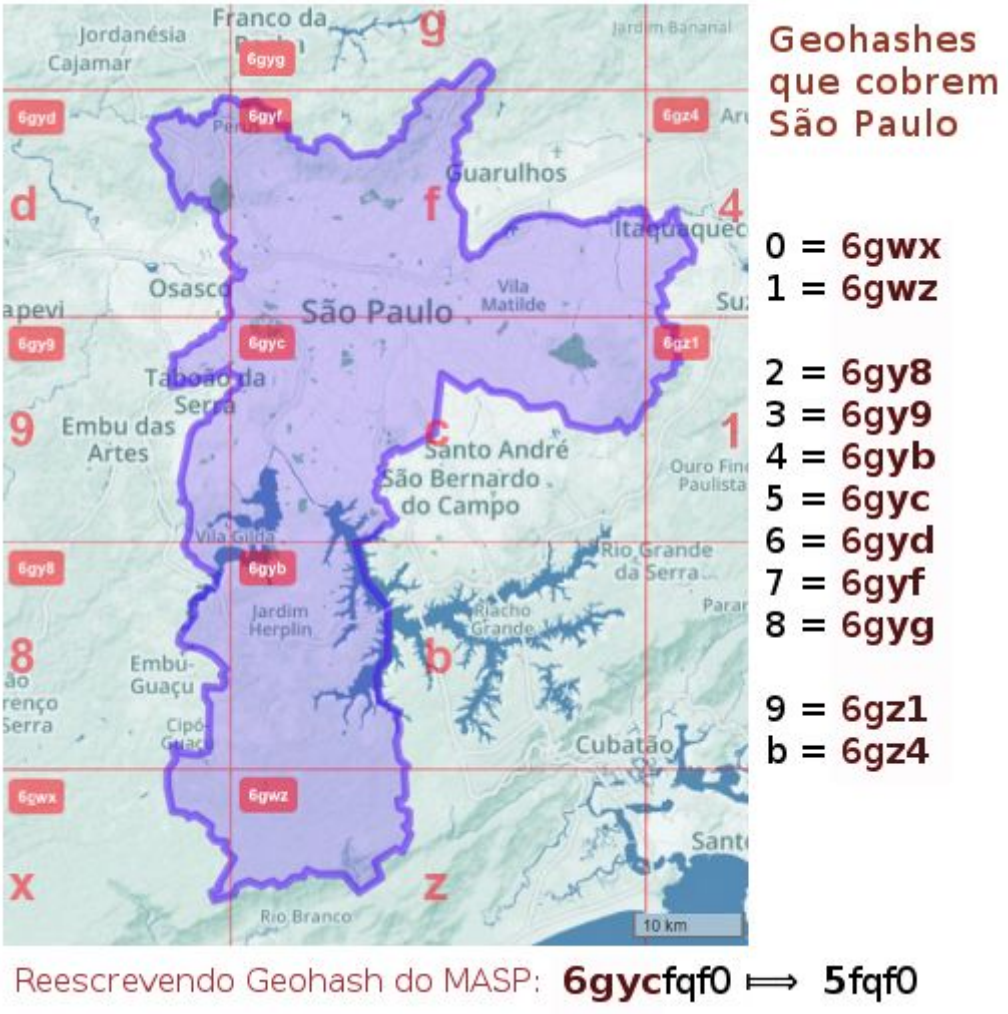
[8] Função ST_Union() conforme especificada pelo padrão OGC/SFA, dissolvendo a fronteira comum às células. PostGIS e Turf-JS fazem uso do padrão. Pode-se também otimizar usando algoritmos mais simples conforme já descrito anteriormente.

Redução com base na cobertura

A convenção mais popular para a formação de geocódigos globais de cobertura parcial[9] é a norma ISO 3166, que define códigos de duas letras para os países, numa norma específica (de jurisdição federada) define códigos de duas ou mais letras dentro do país, para as delimitações administartivas de primeiro nível no seu interior. No caso do Brasil é a ISO 3166-2:BR.

Geocódigos com requisito de serem também mnemônicos (fáceis de se lembrar) portanto podem iniciar por um código ISO. No caso do Brasil são duas letras para cada unidade da federação, por exemplo “BR-SP” designa o Estado de São Paulo. Em seguida viria a subdivisão de segundo nível, que no Brasil são os municípios. Portanto a **principal referência para a formação de coberturas são os polígonos dos municípios**.

A cobertura do município é um conjunto de macrocélulas que define com apenas um dígito a substituição de prefixos de dois ou mais dígitos de um geocódigo completo. Uma vez eleita a grade (DGG) que será adotada, por exemplo grade Geohash (base32), teríamos a seguinte cobertura para o município de São Paulo, com seu respectivo esquema de indexação.



O geocódigo de localização do MASP por exemplo, originalmente “6gycfqf0” (8 dígitos) fica então reduzido a “5fqf0” (5 dígitos). Outros exemplos de localizações dentro da cidade:
“6gycf5q2” □ “5f5q2”, portão 9 do Ibirapuera (troca de “6gyc” pelo índice “5”);
“6gybcsdv5” □ “4csdv5”, entrada do Circo-escola Grajaú (troca de “6gyb” pelo índice “4”).
Na prática a cobertura leverá a dígitos ainda mais curtos se explorar todos os possíveis 32 valores do primeiro dígito, ou seja, ao invés de se restringir a 11 macrocélulas, como na ilustração, devemos fazer uso de 32 macrocélulas. Os algoritmos da seção “Redução e reconstrução por prefixo” garantem maior liberdade para a escolha das macrocélulas.

[9] Quando a cobertura não é total (no caso não contem p. ex. os oceanos) não podemos ter uma grade do tipo DGG.

Conclusões

Com o desenvolvimento apresentado no texto conclui-se que é possível usar a *base32* também com número ímpar de dígitos, desde que seja de interesse a representação do geocódigo em uma grade degenerada.

No trabalho, avaliou-se que a uniformidade da grade degenerada, mesmo na indexação por curva de Hilbert, é razoável. No caso de simples ponto para a localização de um endereço postal, a grade degenerada é tolerada.

A metodologia se mostra satisfatória para as aplicações sugeridas, e sua simplicidade garante que possa ser implementada nas mais diversas bibliotecas de geocodificação, incluindo a S2geometry.

Quanto a uma classificação das aplicações onde a metodologia pode ou não ser utilizada, ainda não existem aplicações de sucesso ou testes suficientes para se propor uma classificação segura, o diagrama da fig.5 é precisaria utilizado em mais casos concretos para se fornecer um parecer mais seguro.

Referências

Padrões:

RFC4648. *“The Base16, Base32, and Base64 Data Encodings”*.
urn:ietf:rfc:4648

RFC5870. *“A Uniform Resource Identifier for Geographic Locations ('geo' URI)”*.
urn:ietf:rfc:5870

Bibliografia de fundamentação:

Goo01. *“Discrete Global Grids for Digital Earth”*, Michael F. Goodchild (2001).
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.9660&rep=rep1&type=pdf

Pet17. *“Discrete Global Grid Systems”*, Perry R. Peterson (2017).
urn:doi:10.1002/9781118786352.wbieg1050

Ppk18. Registro da primeira apresentação dos fundamentos da metodologia, ppkrauss (2018)
https://web.archive.org/web/20181018153701/https://github.com/tammoippen/geohash-hilbert/issues/4

ShXiZh17. *“Encyclopedia of GIS”*, (Eds.) Shashi Shekhar, Hui Xiong, Xun Zhou (2017). Ed. Springer.
urn:isbn:978-3-319-17884-4

Vuk16. *“Hilbert-Geohash, Hashing Geographical Point Data Using the Hilbert Space-Filling Curve”*, Tibor Vukovic (2016).
https://core.ac.uk/download/pdf/154676518.pdf

Nomes próprios de tecnologias, metodologias ou algoritmos controlados, citados ao longo do ensaio:

- **CLP - Código Localizador de Portão**. Proposta de padronização de geocódigos soberanos. Divulgação preliminar em http://openstreetmap.com.br/CLP e repositório em https://github.com/OSMBrasil/CLP
- **Geohash**. https://en.wikipedia.org/w/index.php?title=Geohash&oldid=872879510 , seu serviço de referência, https://web.archive.org/web/20080302043138/http://geohash.org:80 e sua implementação de referência para os algoritmos de *encode* e *decode* em Javascript, https://github.com/chrisveness/latlon-geohash
- **Javascript** (ECMAScript). Referência para representação de base e exemplificação de algoritmos. *“ECMA-262, 9th edition, Language Specification”,* http://www.ecma-international.org/ecma-262/8.0
- **PlusCode**. http://plus.codes e sua implementação de referência em https://github.com/google/open-location-code
- **S2geometry**. https://web.archive.org/web/20180829114528/http://s2geometry.io/ e sua implementação de referência em C++, https://github.com/google/s2geometry . Para ilustrações Javascript, https://github.com/hunterjm/s2-geometry.js

Apêndice 1. Exemplos e discussão de casos

A seguir duas tecnologias de aplicações baseadas em Curva-Z e uma baseada em Curva de Hilbert, que ajudam a ilustrar o uso da metodologia. As “perguntas preliminares” (PP) do diagrama da fig.5 serão referenciadas conforme a tabela abaixo:

PP1. Hierarquia base2 ou base4? **PP2**. Requer quantidade de dígitos ímpar na base32? **PP3**. Requer preservação da orientação da grade? *__PP4__. As células são simétricas e resolvem o problema?	**PP5**. A união de duas células vizinhas ... satisfatória? **PP6**. Usuários finais aprovaram a geometria célula individual? **PP7**. Além de células individuais ... visualizar grades L+2½ ? *__PP8__. Usuários finais aprovaram o grau de uniformidade?

A1.1. DGG do Geohash

O algoritmo Geohash lançado em 2008 já nasceu como aplicação da base32 generalizada. Para ilustrar a metodologia e conceitos do Geohash Generalizado serão destacados a seguir dois tipos de grade Geohash.

O processo completo de construção de cada nível hierárquico do Geohash, ilustrado ao lado, envolve a partição em quatro regiões, identificando cada região por um código de 2 bits (número base4).
O mapa mundi é segmentado em regiões rotuladas em binário como “00”, “01”, “10” e “11” (em base4 seria 0,1, 2 e 3).

Como se percebe, todavia, **a regra de construção do Geohash não precisa se ater à partição da célula original em 4 novas regiões**, pode ser interrompida no primeiro passo, resultando em partição da célula em apenas 2 regiões, portanto com hierarquia binária ao invés de base4.

Existem assim dois **tipos de Geohash**:

- **Assimétrico**, com hierarquia binária e sem garantia de codificação simétrica da latitude e longitude: todas as bases são válidas e possuem representação espacial, não há preocupação com a simetria.
- **Simétrico**, com hierarquia base4: são a rigor permitidas apenas as bases B={8,16,64}.

Há uma sutil diferença entre requerer que o Geohash Simétrico seja flexibilizado, ou requerer adoção do Geohash Assimétrico. Para incorporar a base32 ao Geohash Simétrico aceita-se apenas parcialmente a assimetria — a *grade degenerada* é adotada apenas quando a base32 exibe quantidade ímpar de dígitos.

A1.1.1. Resultados do método aplicado a Geohash

Se adotados Gehash Assimétrico devemos responder “base2” à pergunta PP1 da fig.5, está tudo resolvido.

Supondo o Geohash simétrico, e que a resposta à PP2 foi positiva, desejamos geocódigos com tamanho variável, portanto sempre haverão tamanhos pares e ímpares de geocódigo.

Retomando a ilustração: o ponto vermelho está na célula “00”. Na Figura-3 o mesmo ponto está contido na célula “00110”, que na base32ghs é “6” (decimal 6), mas na base4 não tem representação: corresponderia à união das células “001100” (decimal 12) e “001101” (decimal 13), ou seja, expressando em base4, “030” e “031”. Pela fórmula de reindexação determinada pelo método, e usando decimal nas contas, temos o mesmo

resultado:
6=floor(*12*/2) e *6=floor*(*13*/2).

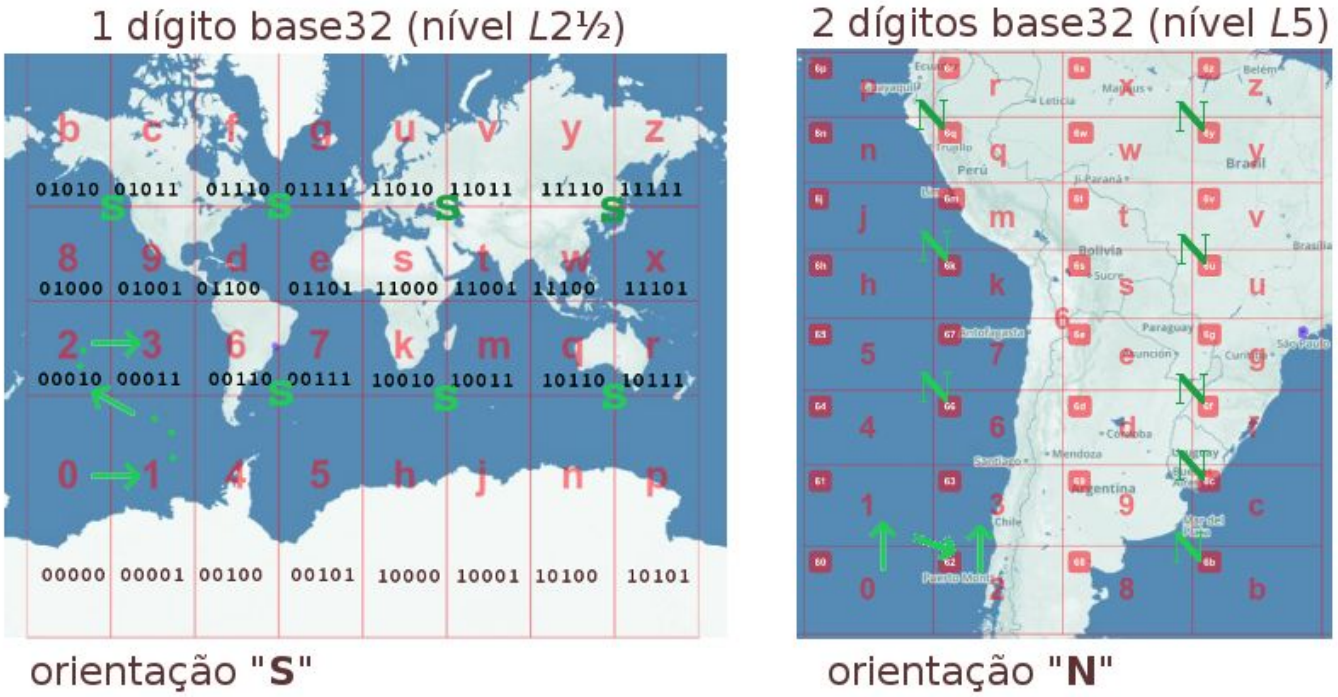
O método proposto portanto reproduz os resultados do padrão Geohash-simétrico. A grade degenerada, para ser aceitável, requer que se aceite também os problemas já expostos na introdução.

Em aplicações orientadas a endereçamento postal teremos as respostas a seguir.
PP3: sim. PP4: são retangulares e satisfatórias, sim, resolvem. PP5: sim, o retangulo não chega a parecer tira achada demais, pode ser usado. PP6: sim, a comunidade Teto Brasil estuda seu uso, foi receptiva. PP7: não é necessário visualizar a grade. PP8: sim, a mesma comunidade em estudo preliminar aprova.

A1.1.2. Problemas típicos da grade Geohash degenerada

Problemas que surgem em outras aplicações e que são de interesse mais teórico do que prático.
Um problema pouco comentado[10] na literatura sobre Geohash é a diferença de **orientação** na curva indexadora quando se passa de um geocódigo com número **par** de dígitos base32 (orientação "N") para um geocódigo com número **ímpar** de dígitos base32 (orientação "S") ou vice-versa.[11]

Figura-3



A orientação canônica, dada pelo nível *L*1, é do tipo "N", como pode ser notado pelos prefixos de 2 bits, "00", "01", "10" e "11", e sua distribuição espacial, com "00" embaixo e "01" em cima.

Outro problema pouco destacado na literatura e pouco notado pelo usuário final é quanto ao **formato geométrico das células**, que muda a cada dígito acrescentado no geocódigo base32gsh. Como se pode notar na visualização das células "6" e "6g" abaixo, não chega a afetar a percepção de uniformidade da grade. Corresponde a um **estiramento vertical** na grade de representação de Geohashes com número **ímpar** de dígitos base32.

[10] Até o momento da publicação não localizamos qualquer artigo de cunho científico ou didático expressando o fato.
[11] Ao longo deste ensaio sempre que nos referimos à base32 do Geohash, é a base32gsh apresentada na introdução.

A mesma observação é válida para outras escalas, por exemplo células menores, a "6gyc" e a "6gycf".

A proporção entre base e altura do retângulo permite classificar os retângulos, caracterizando[12] com precisão as células em dois grupos geometricamente distintos:

- Geohashes com número de dígitos **ímpar**: células “6”, “6gy” e “6gycf” são retângulos verticais, quase quadrados, com **proporção 0,9**.
- Geohashes com número de dígitos **par**: células “6g” e “6gyc” são retângulos horizontais, mais alongados, com **proporção 1,8**.

Como, a rigor, a estrutura hierárquica do Geohash é fixada em base4, podemos tomar o número de dígitos par como referência, e então dizer que a grade espacial associada aos Geohashes com número ímpar de dígitos **é uma grade degenerada**, com proporção 1,8 diversa da canônica (0,9).

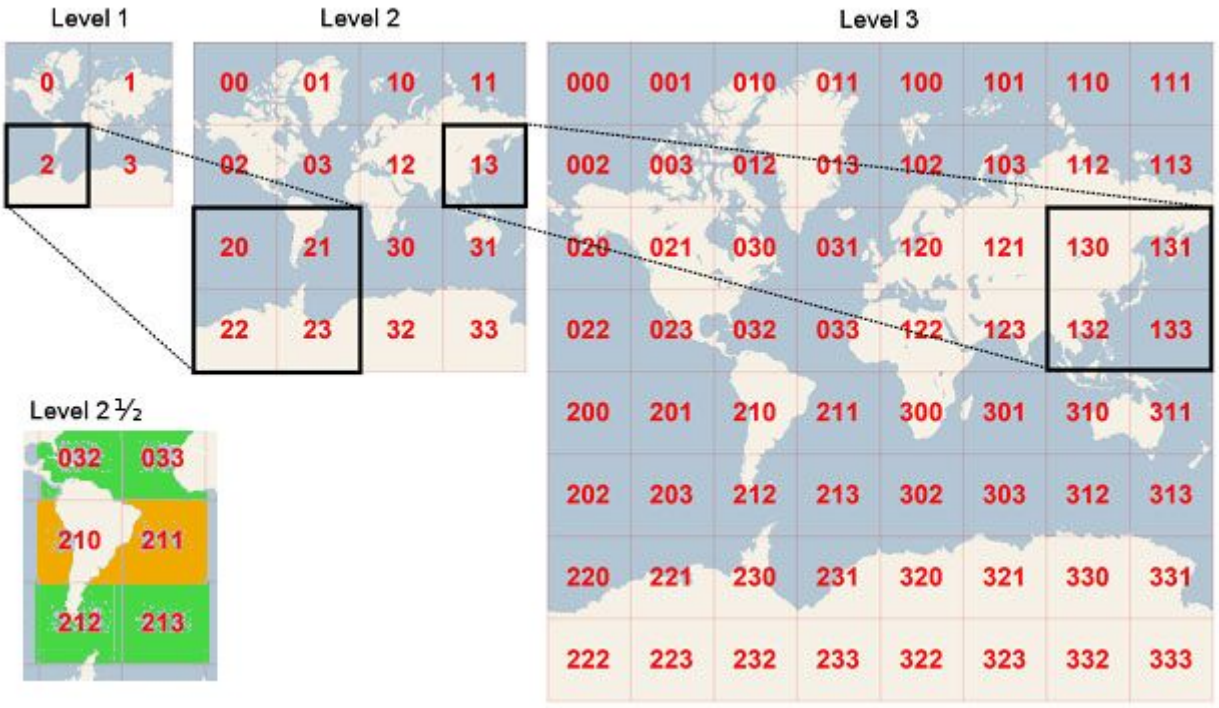
Dizemos “a rigor” pois essa diferença entre proporções não é enfatizado no padrão Geohash, pelo contrário, em geral o usuário não é informado e, curiosamente, não percebe isso como problema: apenas quando se depara com a necessidade de codificar simetricamente latitude e longitude, ou de fazer uso de células com proporção constante.

[12] Deve-se tomar o cuidado para não comparar Geohashes de prefixo diferente: há também uma distorção provocada pela latitude. A caracterização dada pela proporção fica mais precisa com mais dígitos.

A1.2. DGG do Bing Tile System

O Bing Tile System é um exemplo de serviço de mapas raster oferecido pela divisão Bing Maps da Microsoft.

Abaixo a ilustração do indexação base4 realizada pela Curva de Ordem-Z do “sistema de mosaico” (*tile system* ou *tiled web map*) do portal de mapas Bing.



fonte: adaptado de Microsoft (Bing Tile System)

Em destaque ilustrando-se como seriam as células de um nível intermediário entre o *L2* e o *L3*: apesar de possuir representação espacial dada pela união de duas células vizinhas do *L3*, resultando numa grade retangular uniforme, a aplicação (*tile system*) não permite células assimétricas, de modo que sería **inválido** o uso de níveis intermediários tais como o *L2½*.

A exigência de simetria (quadrados) está presente em outros padrões de *tile system* geográfico, tais como o TMS OpenLayers , o *Slippy map tilenames* do OpenStreetMap, ou o *tile system* da API do Google Maps: todos por padrão distribuem imagens quadradas, por exemplo de 256x256 pixels.

Para finalidades ilustrativas, todavia, não haveria maior impedimento: a propriedade CSS *background-repeat* por exemplo aceita imagens retangulares.

Resumindo em termos das perguntas do diagrama da fig.5.
PP1: o algoritmo oferece apenas base4. PP2: sim, quatidade de dígitos variável, portanto impar e par.
PP3: sim. PP4: não, as células obtidas da união de vizinhas perdem a simetria e deixam de ser uteis.

A1.3. DGG da S2geometry

É bem conhecida e já foi sugerida a representação base16 do identificador de célula S2geometry como geocódigo. Para pequenos condomínios ou parques, a base16 pode se mostrar tão boa quanto a base32. No apêndice A2 destacamos que a biblioeca pode fazer uso de um identficador composto, e este é o caso na S2geometry: *id0* representa a face de cubo; *id* representa a chave base4 da hierarquia completa da célula. O identificador base16 funde *id0* com *id*. Para a base32, por outro lado, pode-se ignorar o id0, estaremos sempre operando com o *id* base4 convertido para base32.

Respondendo às primeiras perguntas da fig.5 com aplicações de endereçamento em mente.
PP1: base4, não há opção base2 nativamente disponível como no Geohash.
PP2: sim, tamanho variável, portanto quantidades pares e ímpares.
PP3: apenas o suficiente... Quase “sim”. (usuário decidirão depois no teste de interface).

Antes de seguir nas demais análises e respostas, ilustraremos com mais cuidado os problemas da base32 no S2geometry sem o uso da metodologia.

A1.3.1. Impedimentos no S2geometry

Suponhamos uma célula de ~15×15 metros do S2geometry utilizada como localização do portão do MASP , dada pelo geocódigo “*94ce59c94ac*”, que é um geocódigo expresso em base16 de uma célula do nível *L19*.

Internamente no S2geometry o identificador da célula é composto da face do cubo de referência (no caso de São Paulo a face 4), seguida da hierarquia completa, que pode ser representada em base4 como “*2212130230321022111*”. Para simplificar foram considerados apenas os últimos 5 dígitos, “*22111*” como sendo o código da localização do MASP.

Um conjunto de células da hierarquia-acima e -abaixo, conforme ilustrado, é detalhado na tabela-2 abaixo:

Tabela-2

Nível	Grade	base4	base16	base32rfc	base64
L19	L+0	22111	295	**KL**	KV
L20	L+1	221112	-	-	-
L21	L+2	2211120	2958	-	-
L22	L+3	22111203	-	-	KVJ
L23	L+4	221112031	2958d	-	-
L24	L+5 (L+0)	2211120313	-	**KLHN**	-

L25	L+1	22111203130	2958dc	-	KVJC
L26	L+2	221112031303	-	-	-
L27	L+3	2211120313031	2958dcd	-	-
L28	L+4	22111203130310	-	-	KVJC0
L29	L+5	221112031303103	2958dcd3	**KLHN6J**	-

Percebe-se pelas linhas destacadas na tabela-2, relativas aos níveis *L18*, *L23* e *L28,* que o código da base32 tem alinhamento com a base4 apenas a cada dois dígitos (grade *L+5*), enquanto que a base16 e a base64 alinham-se dígito a dígito.

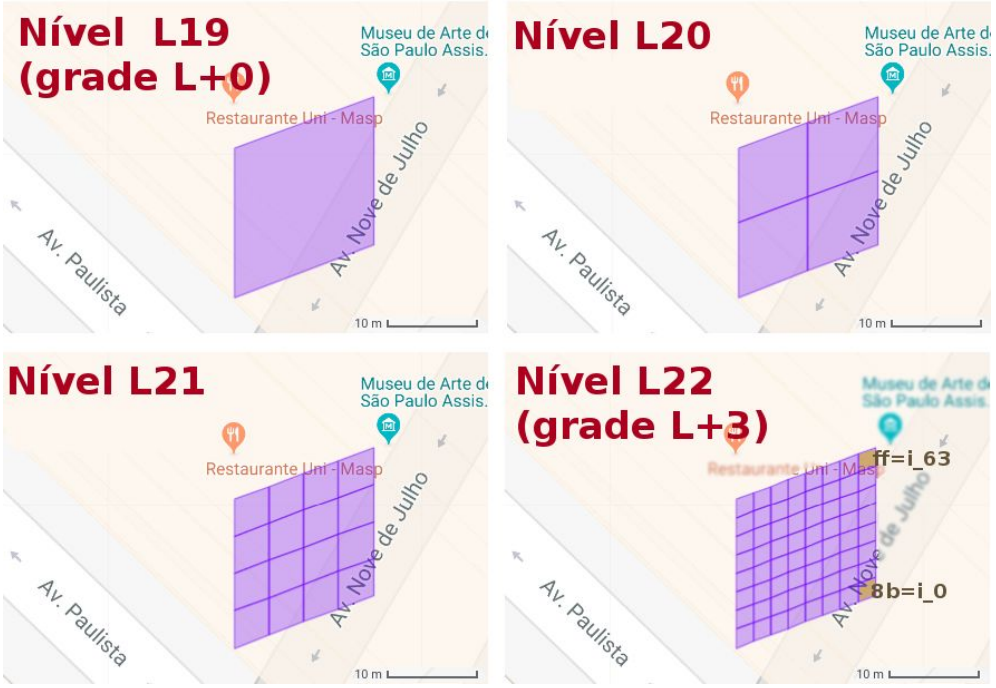
A ausência de uma grade (base4) para representar espacialmente identificadores base32 com qualquer quantidade de dígitos é um **problema** que**,** a princípio, **inviabiliza o uso da base32** com identificadores de célula S2geometry. Usuários e desenvolvedores, inclusive de outros geocódigos baseados em curva de Hilbert, como o Geohash-Hilbert, descartavam qualquer possibilidade de uso.[13]

A noção de viabilidade, entretanto, pode depender do rigor da representação das células no mapa: será viável sempre que o contexto/aplicação admitir uma grade ligeiramente irregular.

A1.3.2. Grade S2geometry degenerada acomodando base32

Voltando ao caso da localização do portão do MASP, para exemplificar, com célula da escala de metros, `94ce59c94ac`, no caso uma célula do nível *L19*.

Figura-3

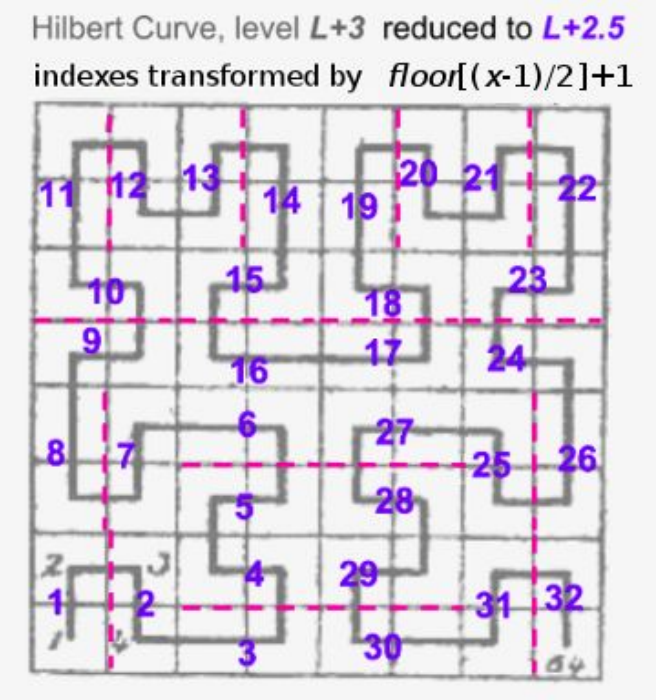


[13] Mesmo com após uma primeira apresentação resumida da metodologia, em outubro de 2018, o posicionamento da comunidade técnica ainda era de resistência à representação em grades degeneradas.

Na ilustração temos uma sequência crescente de níveis hierárquicos:

- *L19*, com a célula de referência e grade de nível ***L+0***.
- *L20* e *L21* sem previsão de representação na base32 nem uso na construção da grade degenerada.
- *L22*, grade ***L+3***, particionando cada uma das 16 anteriores de *L21* em 4, formando uma grade de 4×16=**64** células.

Como apresentado na tabela-2, se a célula de interesse é de nível *L19* e é tomada como grade nível *L+0,* então o geocódigo com mais dois dígitos estará no nível *L+5* ou seja, será uma célula *L24*.



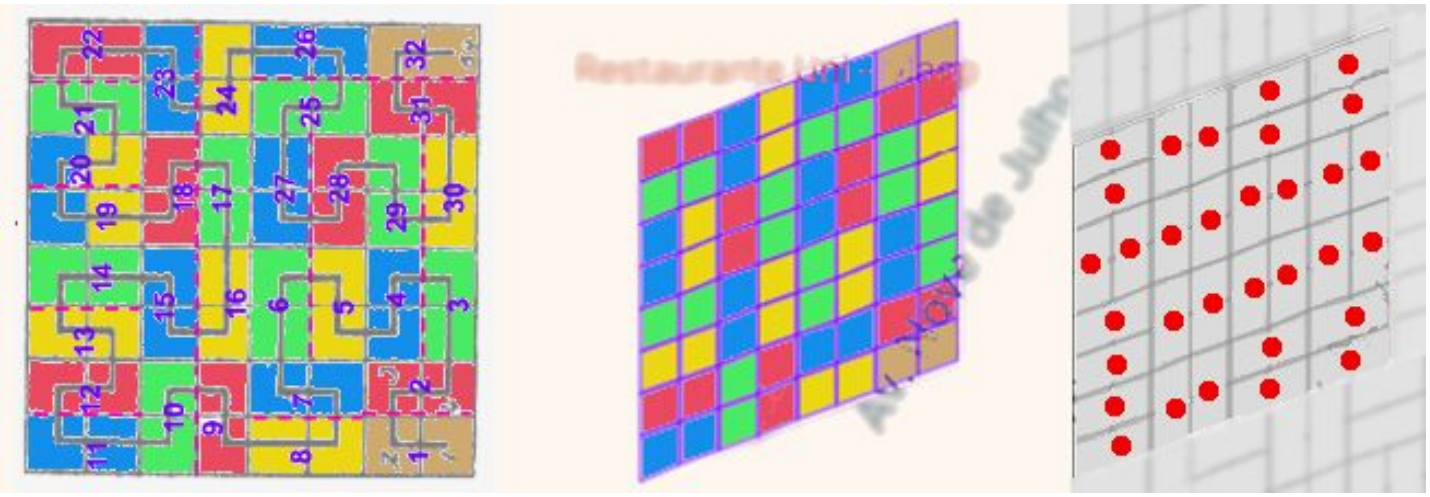
O *L22* é referência da *grade degenerada*. Para mapearem-se os seus 64 índices, *i_0*, *i_1*, *i_2*, etc. em 32 índices utilizaria-se a fórmula proposta:

$$j = \text{floor}\,(i\,/\,2)$$

A fórmula faz sentido, conforme visualmente demonstrado ao lado, tendo ao fundo a ilustração do artigo original de D. Hilbert (1891), onde o índice *x* foi iniciado em 1 ao invés de 0, o que requer mudar o indexador de *i* para *x*=*i*+1.

No sistema S2geometry são as mesmas representações, exceto, na ilustração, por uma rotação de 90° no sentido anti-horário.

Figura-4



Na ilustração acima as novas células foram destacadas por cores diferentes. Como espacialmente as células *i* do *L22* que levam ao mesmo identificador *j* são pares de células vizinhas, as células *j* do nível *L21½* recebem automaticamente uma mesma coloração. À direita os pontos centrais das células *L21½* foram destacados, com a grade degenerada ao fundo.

O artifício para se obter 32 células consistiu em compor uma grade do nível *L+2½*, reduzindo pela metade o número de células da grade do nível *L+3*. O preço é a perda de simetria na forma das células (paralelogramos alongados ao invés de losangos) e na sua distribuição ao longo da grade, com orientação horizontal/vertical irregular... Se o objetivo é uma grade de pontos, o preço não é tão alto.

Resumindo:

> A grade S2geometry do nível **L+2½** é a grade do nível **L+3** com células vizinhas reunidas duas a duas, resultando em 16×4/2=32 células identificadas por **j=floor(i/2)**, tomando-se o índice **i** do nível L+3 como referência. É uma grade irregular, mas suficientemente uniforme para estabelecer localizações pontuais.

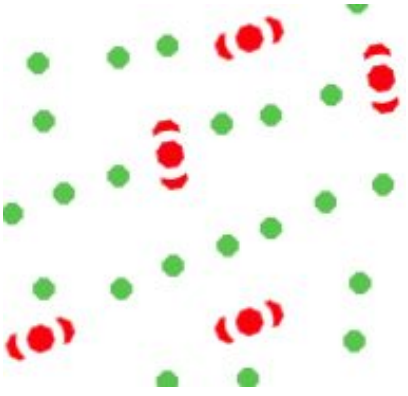
A1.3.3. Resultados em diversos níveis e visualização

A tabela abaixo se baseia na tabela-2. Foram acrescidas linhas para os níveis hierárquicos intermediários, *L21½* e *L26½*, mostrando-se a correspondência entre código base32rfc e o tipo de grade em que pode ser expresso.

Tabela-3

Nível e grade		Células base4 na grade	base32	Regular
L19	*L+0*	`22111`	**KL**	sim
L21½	*L+2½*	`22111202` e `22111203` de *L22*	KLH	não [14]
L24	*L+5* (L+0)	`2211120313`	**KLHN**	sim
L26½	*L+2½*	`2211120313030` e `2211120313031` de *L27*	KLHN6	não
L29	*L+5*	`22111203130310`	**KLHN6J**	sim

Estes resultados também foram testados quando à aprovação da representação espacial pelo usuário final, no contexo da aplicação. Para aplicações associadas ao endereçamento postal, localização de portões urbanos e localização de portões rurais, é eleita uma grade com resolução suficiente para distinguir dois portões vizinhos. No caso de endereçamento rural, de partes e de condomínios, a escala para distinção é da ordem de 15 metros, e no caso urbano da ordem de 3 mestros.



A visualizações neste caso de uso requer apenas o destaque da localização pontual (centro de célula) e da precisão ou erro de posicionamento da própria célula. Interfaces com o recurso ilustrado ao lado, com “pontos vermelhos borrados”, se mostraram satisfatórias para o grupo de usuários consultados nos testes de interface. A grade degenerada (pontos verdes) não precisa ser visualizada pelo usuário, basta o “snap to grid” e a avaliação da precisão dos pontos fixados.

Com isso podemo concluir as respostas às perguntas da fig.5.
PP3: apenas o suficiente... Quase “sim”.
(usuário decidirão depois no teste de interface).
PP4: as células resultantes da união não são simétricas mas parecem aceitáveis.
PP5: sim, mantém parentesco com formato retangular.
PP6: sim, aprovada por pequeno grupo de usuários.
PP7: sem problema, a grade não é visualizada, e no snap-to-grid não se percebe a perda de uniformidade.
PP8: a ilustração acima foi aprovada por pequeno grupo de usuários.

[14] A conversão deve ser feita primeiramente da base4 para binário, removendo-se o bit final esquerdo de *i0*, e só depois convertendo o resultado para base32. Para converter diretamente os dados da tabela, de base4 para base32, é necessário o acréscimo de zeros na base4 e truncamento do resultado. Por exemplo 22111202 seria primeiramente convertido como base4(2211120200)=base32(KLH0), então do truncamento resultaria os esperado “KLH”.

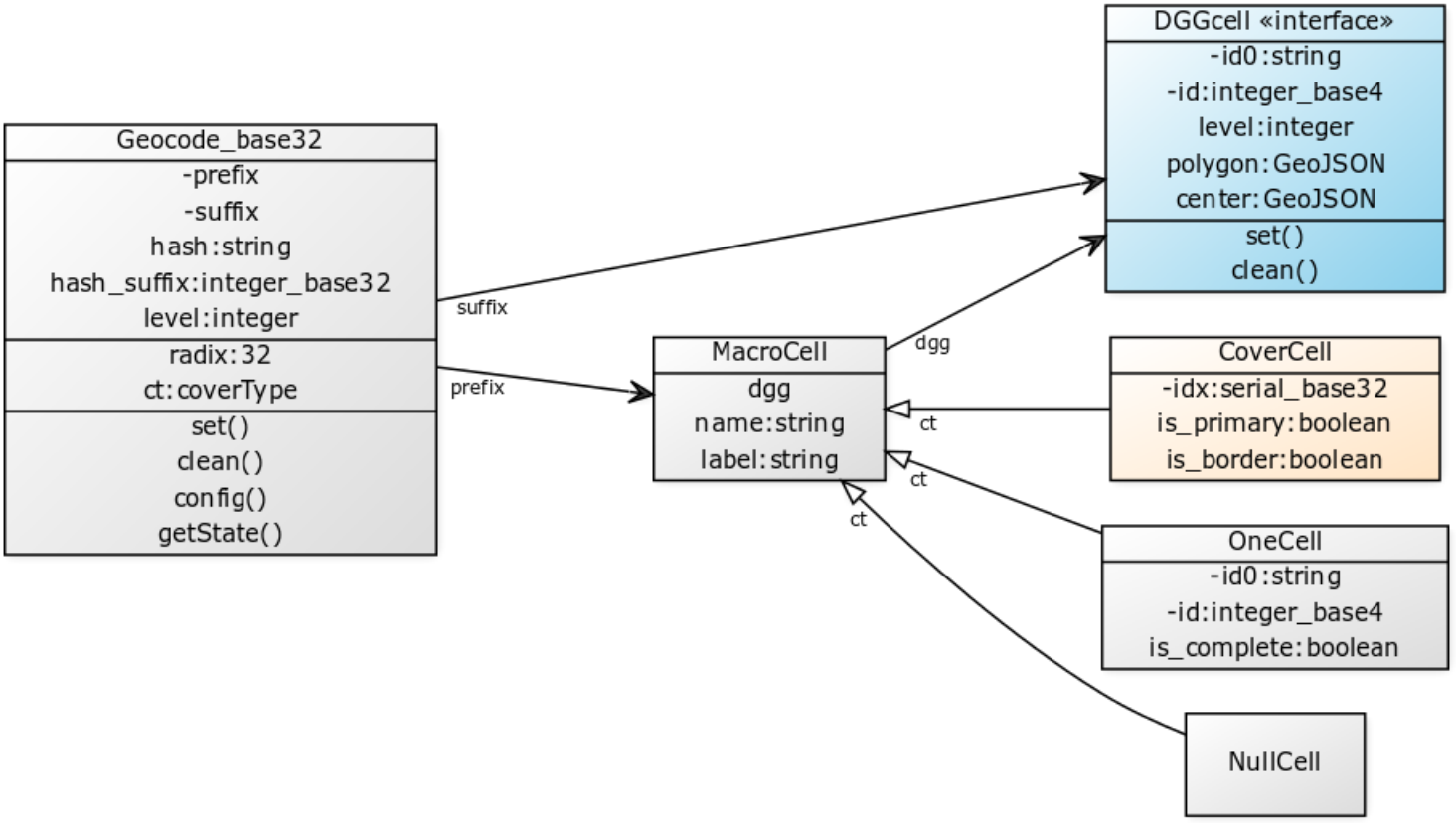
Apêndice 2. Implementação

A prova de conceito e testes da metodologia foram realizados com ajuda de uma implementação Javascript, que serve de referência para a discussão de detalhes não abordados no ensaio ou nas discussões.

A2.1. Diagrama de classes do módulo Geocode

A classe principal, Geocode, delega a formação da célula às classes CoverCell e DGGCell.

A classe DGGcell é um *proxy* ou *adapter* para a implementação de fato, com as funções de biblioteca essenciais da DGG escolhida, eventualmente adaptadas por exemplo para fornecer o polígono da célula em GeoJSON. A classe CoverCell é apenas um dataset para a look-up table da cobertura, quando uma cobertura for definida (Cover é opcional), caso contrário é inerte e pressupõe prefixo nulo no Geocode. O construtor de CoverCell pode fazer uso do método getState() da classe Geocode. O método CoverCell.set() é opcional para o caso de Cover ausente.



O mesmo diagrama expresso em linguagem yUML:

```
[MacroCell|dgg; name:string; label:string]
[DGGcell «interface»|-id0:string; -id:integer_base4; level:integer; polygon:GeoJSON;
center:GeoJSON | set(); clean(){bg:skyblue}]
[Geocode_base32|-prefix; -suffix;hash:string; hash_suffix:integer_base32; level:integer;
|radix:32; ct:coverType | set(); clean(); config(); getState()]
[CoverCell|-idx:serial_base32; is_primary:boolean; is_border:boolean{bg:bisque}]

[MacroCell]^ct--[NullCell]
[MacroCell]^ct--[OneCell|-id0:string; -id:integer_base4; is_complete:boolean]
[MacroCell]^ct--[CoverCell]

[Geocode_base32]suffix-->[DGGcell «interface»]
[Geocode_base32]prefix-->[MacroCell]
[MacroCell]dgg-->[DGGcell «interface»]
```

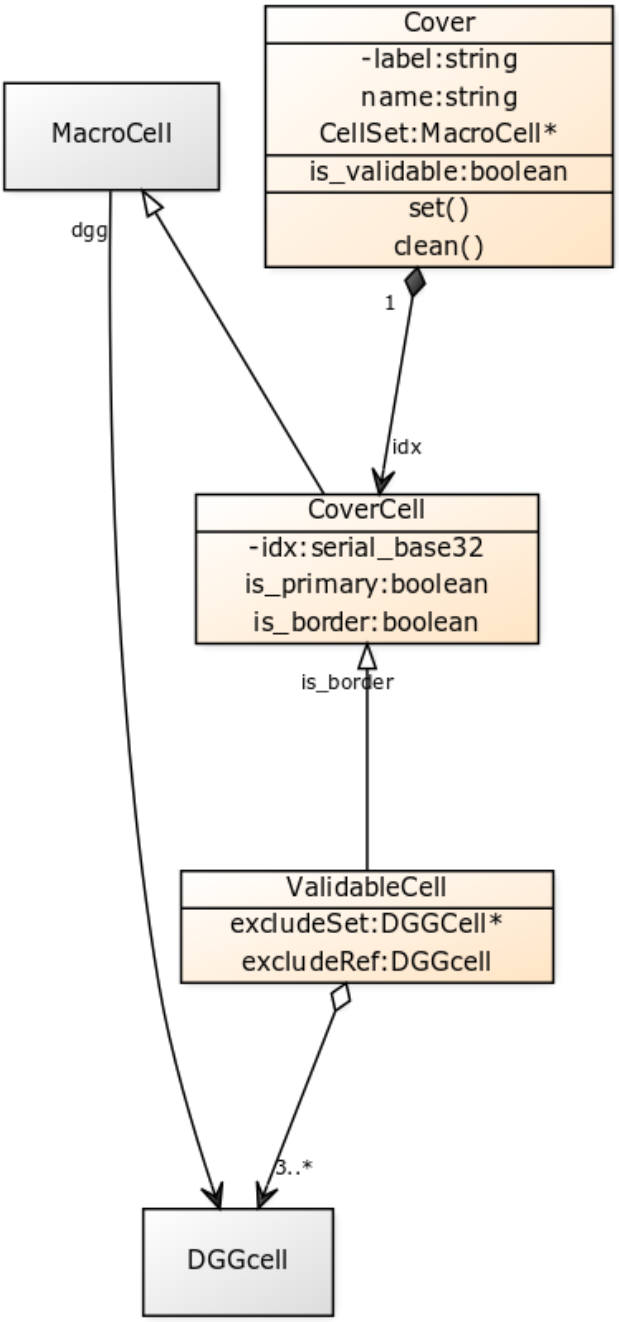
A.2.2. Extensão do módulo para abrigar a classe Cover

Diagrama UML expresso em linguagem yUML e ilustrado em seguida à direita:

```
[MacroCell]dgg-->[DGGcell]
[Cover|-label:string;name:string;CellSet:MacroCell*|is_validable:boolean|
set();clean(){bg:bisque}]
[CoverCell|-idx:serial_base32; is_primary:boolean; is_border:boolean{bg:bisque}]
[Cover]++1---idx>[CoverCell]
[MacroCell]^--[CoverCell]

[CoverCell]^is_border--[ValidableCell|excludeSet:DGGCell*;
excludeRef:DGGcell{bg:bisque}]

[ValidableCell]<>-3..*>[DGGcell]
```



Cada item de uma cobertura com dois ou mais elementos é uma especialização da MacroCell, a **CoverCell**, que recebe um indexador *idx* na cobertura que fará as vezes de dígito (um ou dois primeiros dígitos) do prefixo do geocódigo.

A cobertura como um todo é gerenciada pela classe **Cover**.

A validação de uma cobertura relativa a um polígono arbitrário (ex. decidir se um ponto está ou não no interior de um município) demanda armazenamento da informação relativa ao polígono, o que em geral é feito apenas no lado servidor ou com apenas a cobertura em uso. O *flag* ***is_validable*** em geral indicará implementação do algoritmo no lado server, e demandará uso da especialização **ValidableCell** de CoverCell.

A validação desse tipo de cobertura requer ambos, maior poder computacional e maior capacidade de armazenamento. Uma arquitetura cliente-servidor onde apenas o servidor faz a validação, se for de consulta frequente não será eficiente.

Para reduzir o número de consultas há que se delegar parte da tarefa de validação e, principalmente, o armazenamento do cache. Isso é possível através de algoritmos leves e rápidos baseados em Filtros de Bloom em prefixos de código (limitando a carga e fixando uma referência de precisão para o usuário).