Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 5, N. 1, 2017.

Trabalho apresentado no CNMAC, Gramado - RS, 2016.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Sistemas de Dígitos Verificadores Ótimos baseados em Aritmética Modular

Natália Pedroza de Souza[1]
Departamento de Informática e Ciências da Computação - IME/UERJ, Rio de Janeiro, RJ
Luerbio Faria[2]
Departamento de Informática e Ciências da Computação - IME/UERJ, Rio de Janeiro, RJ
Paulo Eustáquio Duarte Pinto[3]
Departamento de Informática e Ciências da Computação - IME/UERJ, Rio de Janeiro, RJ
ias da Computação - IME/

Resumo. Neste artigo discutimos os sistemas de dígitos verificadores baseados em aritmética modular, utilizados mundialmente. Dígitos verificadores são usados para eliminar a maioria dos erros na entrada de dados em sistemas computacionais. Embora antigos, não é encontrada na literatura uma discussão sobre a otimalidade dos esquemas utilizados. Descrevemos os principais esquemas existentes e destacamos aqueles adotados no Brasil. Apresentamos as melhorias necessárias para tornar os diversos esquemas ótimos. Propomos, também, um novo esquema ótimo com 3 permutações para sistemas com base modular 10.

Palavras-chave. Dígitos Verificadores, Detecção de Erros, Aritmética Modular

1 Introdução

Dígitos verificadores (DV's) são um ou mais caracteres, numéricos ou alfanuméricos, acrescentados a uma cadeia de caracteres original, com o objetivo de permitir a detecção de erros na entrada de dados em sistemas computacionais. Exemplos de dígitos verificadores são os dois últimos dígitos do CPF e do CNPJ no Brasil e dígitos acrescentados nos códigos de barra de produtos. Os principais erros são erros humanos, que foram categorizados por J. Verhoeff [7] e são apresentados juntamente com suas frequências na Tabela 1.

Erros fonéticos estão relacionados a algumas línguas, tais como inglês, holandês e alemão, onde existem números com pronúncias muito semelhantes. Em inglês, por exemplo, é difícil distinguir *thirty* de *thirteen*. Erros distintos dos descritos na Tabela 1 são englobados no tipo Outros e não serão analisados.

Consideraremos como ótimo um esquema que permita detectar, em termos ponderados, o máximo possível de erros das 6 primeiras classes de erros da Tabela 1.

Os sistemas de dígitos verificadores encontrados na literatura são baseados em três ramos da Matemática: Aritmética Modular, Teoria dos Grupos e Quadrados Latinos.

[1]npsnatalia@gmail.com
[2]luerbio@cos.ufrj.br
[3]pauloedp@ime.uerj.br

Tabela 1: Tipos de erros e suas frequências de ocorrência.

Tipo de erro	**Sigla**	**Formato**	**Frequência (%)**
Erros individuais	IND	..a.. → ..b..	79.10
Transposição adjacente	TAD	..ab.. → ..ba..	10.20
Transposição alternada	TAL	..abc.. → ..cba..	0.80
Gêmeos adjacentes	GAD	..aa.. → ..bb..	0.50
Gêmeos alternados	GAL	..aca.. → ..bcb..	0.30
Fonéticos	FON	..1a.. → ..0a.., a$\geq$ 2	0.50
Outros	OUT		8.60

Entretanto, somente são utilizados na prática sistemas baseados em Aritmética Modular. Para essa classe de códigos não foram encontrados na literatura artigos que discutam a otimalidade dos mesmos. Este artigo tem como objetivo fazer essa discussão. São caracterizadas as condições para a otimalidade dos métodos em uso, notadamente os métodos com base modular prima e com base 10.

2 Esquemas de DV's baseados na Aritmética Modular

São os mais divulgados universalmente [3, 5]. Nestes esquemas são feitas operações aritméticas nos dígitos dos identificadores e os resultados são considerados em módulo $n \in \mathbb{N}$. Dado um identificador com m dígitos $a_1a_2a_3\cdots a_m$ e uma sequência de pesos p_i, o dígito verificador a_m é determinado através da equação do *esquema direto* dada por:

$$a_m = \sum_{i=1}^{m-1} a_i p_i \bmod n. \tag{1}$$

De forma mais abrangente, podemos definir funções σ_i, para cada posição $1 \leq i \leq m-1$ do identificador e usar:

$$a_m = \sum_{i=1}^{m-1} \sigma_i(a_i) \bmod n. \tag{2}$$

A seguir descrevemos métodos que usam base 10 ou 11, que são as mais usuais.

Módulo 10 - Um dos métodos mais naturais quando se trabalha com identificadores decimais é o que utiliza a base modular 10. As variantes mais comuns encontradas usam indiretamente permutações através de esquemas de 2 ou 3 pesos.

Um exemplo é o **código de barras de produtos**, usado mundialmente para identificar produtos no comércio. O identificador consiste em 13 dígitos decimais. Para o cálculo do DV, é utilizada a equação por complemento com pesos 1 e 3, aplicados alternadamente. Como exemplo mostraremos o cálculo do DV do código 7 89102 711427 5:

	7	8	9	1	0	2	7	1	1	4	2	7
$\times$	1	3	1	3	1	3	1	3	1	3	1	3
	7	24	9	3	0	6	7	3	1	12	2	21

Os resultados são somados, obtendo-se 7+24+9+3+0+6+7+3+1+12+2+21 = 95. O valor do dígito verificador é o complemento de 95 mod 10 para 10, ou seja, 5.

Na variante Módulo 10 IBM, após os algarismos do identificador serem multiplicados pelos pesos 1 e 2 alternadamente, em cada produto, deve ser feita a operação chamada "noves fora", que consiste em somar os algarismos desse produto quando este é maior que 9. Este esquema é utilizado no documento de **Registro Geral do Estado do RJ**.

O número do **passaporte** no Brasil utiliza as letras maiúsculas de A a Z, os dígitos de 0 a 9 e o símbolo $<$. São utilizados os pesos 7, 3 e 1. Os caracteres que podem ser alfanuméricos devem ser convertidos para o cálculo, através de um mapeamento das letras para números. A letra A é mapeada para 10, B para 11 e assim sucessivamente. O caractere $<$ é mapeado para 0. Pode-se ver que não é possível detectar todos os erros individualizados, pois o mapeamento de vários caracteres para um mesmo dígito decimal impede a detecção da troca entre esses caracteres. Por exemplo, $0, A, K, U, <$ são todos mapeados para múltiplos de 10. Portanto, qualquer intercâmbio entre eles não é detectado.

Módulo 11 - Quando utilizamos o módulo 11 para o cálculo do dígito verificador, considerando identificadores numéricos, o resto da divisão por 11 pode ser 10. Para se usar apenas um dígito verificador, são adotadas duas soluções: módulo 11 completo e módulo 11 restrito, onde utiliza-se para representar o resto 10, o caractere X ou o dígito 0, respectivamente.

Cada uma dessas soluções apresenta inconveniências no seu uso. No caso do módulo 11 completo, introduz-se um caractere não numérico, o que pode trazer dificuldades. Como exemplo, na informação de dados via telefone, o X tem que ser mapeado para um dígito decimal, o que transforma a situação em módulo 11 restrito. Quando se adota o módulo 11 restrito, não conseguimos detectar as situações de erro que alteram o cálculo do DV de 0 para 10, ou vice-versa.

Módulo 11 com dois dígitos verificadores - Alguns esquemas utilizam dois dígitos verificadores para uma maior detecção de erros. Neste caso é feito um cálculo para cada DV. Caso o resultado de um dos cálculos seja 10, este é transformado para 0.

Esquemas de DV's usados no Brasil e sua detecção de erros

A Tabela 2 mostra diversos esquemas de DV's usados no Brasil e os percentuais de erros de cada tipo não detectados pelos esquemas. Esses percentuais foram obtidos teoricamente e comprovados em termos práticos em [6]. A última coluna da tabela, MED, exibe um percentual ponderado de erros não detectados. Os sistemas que utilizam 2 DV's estão representados apenas pelo CNPJ, pois é o que apresenta a melhor taxa de detecção.

O esquemas mostrados são de uso mundial, com exceção daquele relativo ao CNPJ, particular do Brasil. Uma pergunta natural é a de se esses sistemas são ótimos do ponto de vista de detecção de erros. A resposta é negativa e chega a ser surpreendente constatar a ineficiência na detecção de erros do esquema utilizado nos passaportes. Na próxima seção discutimos alterações que tornam ótimos os esquemas apresentados.

DOI: 10.5540/03.2017.005.01.0463

Tabela 2: Percentuais de erros não detectados

Esquema de DV	**IND**	**TAD**	**TAL**	**GAD**	**GAL**	**FON**	**MED (%)**
10 - 2 pesos	0	11.11	100.00	11.11	11.11	0	2.02
10 - 3 pesos	0	11.11	11.11	49.23	40.73	0	1.59
10 alfanum. 3 pesos	7.81	18.02	18.02	48.76	41.57	0	8.53
10 IBM	0	2.22	100.00	6.67	11.11	12.50	1.16
11 completo	0	0	0	14.29	0	6.25	0.10
11 restrito	1.82	1.82	1.82	15.85	1.82	14.61	1.80
11 CNPJ	0.04	0.14	0.14	0.83	0.03	2.36	0.06

3 Esquemas modulares ótimos

Um sistema de DV ótimo é o que deixa de detectar o menor percentual ponderado de erros. A equação para a minimização da média ponderada dos erros não detectados, baseada na Tabela 1, é dada por:

$$\mu = 0,791p_1 + 0,102p_2 + 0,008p_3 + 0,005p_4 + 0,003p_5 + 0,005p_6, \tag{3}$$

onde p_i é a probabilidade de não detecção dos erros IND, TAD, TAL, GEM, GAL e FON, respectivamente. Vamos mostrar que todos os sistemas em uso da Tabela 2 podem ser otimizados, alguns deles detectando todos os erros possíveis. Na discussão a seguir consideramos os sistemas com base modular prima e aqueles com base modular 10.

3.1 Esquema ótimo para bases modulares primas

Mostramos, a seguir, um esquema geral ótimo para bases modulares primas. Estamos interessados, particularmente, nas bases 11, para identificadores com dígitos decimais e 37, para dígitos alfanuméricos.

Quando a base modular n é um número primo, podemos utilizar pesos no intervalo 1 a $n-1$. Com isto garantimos a detecção de todos erros individuais (IND). Para detectar todas as transposições (TAD e TAL) é necessário que tenhamos pelo menos três pesos distintos, aplicados repetidamente, já que a diferença entre esses pesos sempre será um número relativamente primo a n. Para encontrar os erros gêmeos (GEM e GAL), a soma de pesos consecutivos ou alternados não pode ser igual a n. Para a detecção de erros fonéticos, é necessário que os pesos satisfaçam a:

$$p_i \not\equiv (p_i - p_{i+1})a \bmod n, \quad \text{para } 0 \leq a \leq 9. \tag{4}$$

Note que os produtos $(p_i - p_{i+1})q$ para $0 \leq q < n$ e quaisquer valores fixos distintos de p_i e p_{i+1} geram uma permutação dos números de 0 a $n-1$ em módulo n, onde n é primo. Portanto, todo número p_i será congruente a $(p_i - p_{i+1})q$ para algum q em $\mathbb{Z}_n$.

No caso do módulo 11, a única possibilidade de p_i ser diferente de $(p_i - p_{i+1})a$ para $0 \leq a \leq 9$ em módulo n é fazendo:

$$p_i \equiv 10(p_i - p_{i+1}) \bmod 11 \Longrightarrow -9p_i \equiv -10p_{i+1} \bmod 11 \Longrightarrow p_{i+1} \equiv 2p_i \bmod 11. \tag{5}$$

DOI: 10.5540/03.2017.005.01.0463

Assim, escolhendo cada peso como o dobro do anterior em módulo 11, satisfazemos às condições para detecção de erros fonéticos. Ou seja, deve-se aplicar alternadamente qualquer permutação circular dos pesos $\langle 1, 2, 4, 8, 5, 10, 9, 7, 3, 6\rangle$.

Para $n > 11$, existem muitas outras relações possíveis entre dois pesos consecutivos. Portanto, para este caso, criamos um digrafo com $n-1$ vértices, onde existe aresta entre os vértices u e v, $1 \leq u \neq v < n$, se os pesos u e v aplicados a duas posições consecutivas do identificador conseguem detectar todos os erros fonéticos. Após criar o digrafo, procuramos ciclos de tamanho pelo menos 3, tal que, para cada 2 elementos alternados do ciclo, sua soma nunca seja n. Um algoritmo para encontrar os ciclos é mostrado em [6]. Para a base 13, um conjunto cíclico possível é $\langle 1, 2, 4\rangle$ e para os primos maiores que 13, $\langle 1, 2, 3\rangle$.

Com a utilização de bases primas todos os erros considerados neste trabalho poderiam ser detectados. No caso do esquema de passaporte mostrado, seria suficiente que a base utilizada fosse 37 e não 10, correspondendo aos dígitos 0 a 9, caracteres A a Z e mais o caractere especial $<$, já utilizados na representação do identificador. Os complicados esquemas com dois dígitos verificadores e módulo 11 usados no Brasil poderiam detectar também todos os erros, bastando que o cálculo do dígito verificador (0 a 10) fosse representado em dois dígitos. O esquema de módulo 11 restrito é o único caso em que os erros não podem ser zerados, devido à substituição do dígito verificador X pelo dígito 0. A solução ótima, com essa restrição, consiste em deixar de detectar 2 em cada 110 erros de cada tipo $(1, 82\%)$.

3.2 Esquemas ótimos para base modular 10

Como pode ser visto em [4], nenhum esquema baseado em aritmética modular com base 10 pode detectar, simultaneamente, todos os erros individuais e de transposição. Descreveremos dois esquemas Módulo 10 ótimos, mediante certas restrições. O primeiro deles é o esquema de Verhoeff [7] e o segundo é um novo sistema ótimo quando se utilizam apenas três permutações, descrito em [6].

Lembremos que um sistema de dígito verificador consiste em aplicar k funções aos $m-1$ dígitos inicias de um identificador de m dígitos, para obter o último dígito. Essas funções devem ser permutações pois, devido à alta frequência dos erros individualizados, a detecção de todos os erros deste tipo é necessária.

As propriedades das permutações utilizadas são deduzidas a seguir.

Consideremos os erros de transposição não detectados por duas permutações p e q aplicadas aos dígitos a_i e a_j de um identificador. Um erro de transposição não é detectado se $p(a_i) + q(a_j) \equiv p(a_j) + q(a_i) \bmod 10$ ou, de forma equivalente, $p(a_i) - q(a_i) \equiv p(a_j) - q(a_j) \bmod 10$. Cada igualdade na equação corresponde a 2 erros não detectados: a troca de a_i com a_j e vice-versa. Dadas p e q, para calcular todos os erros de transposição não detectados, devemos contar a quantidade s de cada resultado na função diferença e, para cada s distinto, somar o número de arranjos $A_{s,2} = s(s-1)$ ao total procurado.

Para calcular os erros gêmeos não detectados as permutações envolvidas têm que ser somadas, pois a não detecção da troca de um dígito a nas posições em que p e q são aplicadas, por outro dígito $b \neq a$, ocorre se $p(a) + q(a) \equiv p(b) + q(b) \bmod 10$.

Por fim, um erro fonético, que ocorre nas posições consecutivas em que p e q são

aplicadas, não é detectado quando $p(1) + q(a) \equiv p(a) + q(0) \bmod 10$, para $a = 2, 3, \cdots, 9$, o que é equivalente a termos: $p(1) - q(0) \equiv p(a) - q(a) \bmod 10$. Sempre que a igualdade ocorre, 2 erros deixam de ser detectados: a troca de $1a$ por $a0$ e vice-versa.

3.2.1 O sistema Módulo 10 de Verhoeff

O melhor sistema de dígito verificador módulo 10 conhecido até aqui era aquele proposto por Verhoeff [7], com as seguintes características:

1. utiliza $m - 1$ permutações distintas, uma para cada dígito do identificador com m dígitos.
2. cada par de permutações consecutivas deixa de detectar o mínimo de erros gêmeos e erros de transposição (2 casos em 90 de cada tipo).
3. cada par de permutações alternadas deixa de detectar 4 erros em 90 possíveis, tanto no caso de erros gêmeos quanto de transposição.
4. deixam de ser detectados 2 em 16 erros fonéticos a cada grupo de 6 permutações usadas.

Esse sistema foi encontrado através de um processo misto onde várias propriedades matemáticas do sistema foram deduzidas e muitos resultados foram obtidos computacionalmente. Verhoeff não provou a otimalidade do método. Recentemente os autores desenvolveram um programa, disponível em http://www.ime.uerj.br/∼pauloedp/MEST/TESES/NATALIA/ComprovaVerhoeff.cpp que procura, exaustivamente, conjuntos de até 6 permutações que sejam melhores, em termos de detecção de erros, que o sistema de Verhoeff. O programa não encontrou tais conjuntos, comprovando a otimalidade do sistema de Verhoeff para identificadores de até 6 dígitos.

3.2.2 Esquema ótimo para dígito verificador módulo 10 com 3 permutações

Inúmeros autores procuraram encontrar um sistema ótimo com o uso de apenas 3 permutações, usadas repetidamente [1, 2]. Tal resultado foi finalmente conseguido em 2013 e está descrito em [6]. As características do sistema encontrado, que se baseia na minimização de erros alternados, são as seguintes:

1. utiliza 3 permutações distintas, sendo a primeira delas a identidade, aplicadas repetidamente aos $m - 1$ dígitos do identificador.
2. cada par de permutações deixa de detectar o mínimo de erros de transposição (2 casos em 90 de cada tipo).
3. cada par de permutações deixa de detectar, alternadamente, 2, 6 ou 8 erros gêmeos (média de não detecção de 16 casos em 270 de cada tipo).
4. todos os erros fonéticos são detectados.

Em [6] foram encontrados, de forma exaustiva, 48 conjuntos de 3 permutações com a característica descrita e foi feita a demonstração de que tais sistemas são ótimos, considerando a restrição de apenas utilizar 3 permutações. Um exemplo desse sistema é o conjunto das permutações σ_1 : (0 1 2 3 4 5 6 7 8 9), σ_2 : (0 8 6 4 2 7 9 1 3 5) e σ_3 : (1 6 3 2 8 7 4 0 5 9).

DOI: 10.5540/03.2017.005.01.0463

4 Conclusões

Dígitos verificadores são ferramentas importantes para melhorar a qualidades de dados de entrada nos sistemas computacionais. Tradicionalmente, os esquemas usados para obter esses dígitos são baseados em Aritmética Modular e não utilizam as melhores práticas.

Neste trabalho apresentamos esquemas ótimos para todos os exemplos discutidos. Para os sistemas que utilizam bases primas, as melhorias devem ser feitas em relação às sequências de pesos utilizadas. Para os sistemas que utilizam base 10, evidenciamos que o sistema de Verhoeff é ótimo para identificadores de até 6 dígitos. Apresentamos um novo esquema ótimo quando se usa apenas 3 permutações. Tal sistema, conjectura-se, é o sistema ótimo para identificadores com mais de 6 dígitos. A Tabela 3 resume os percentuais de erros não detectados para os sistemas ótimos considerados.

Tabela 3: Porcentagem de erros não detectados para os sistemas propostos.

Esquemas	**IND**	**TAD**	**TAL**	**GEM**	**GAL**	**FON**	**MED %**
Vehroeff (≤ 6)	0	2.22	4.44	2.22	4.44	0	0.28
Vehroeff (> 6)	0	2.22	4.44	2.22	4.44	3.57	0.30
Mod 10 3 Permutações	0	2.22	2.22	5.93	5.93	0	0.29
Mod 11 completo 1 DV	0	0	0	0	0	0	0
Mod 11 restrito 1 DV	1.82	1.82	1.82	1.82	1.82	1.82	1.66
Mod 37 ótimo	0	0	0	0	0	0	0
Mod 11 ótimo 2 DV's	0	0	0	0	0	0	0

As análises e propostas aqui apresentadas podem contribuir para que novas criações de dígitos verificadores utilizem métodos ótimos, diferentemente do que tem ocorrido até hoje.

Referências

[1] A. Ecker and G. Poch. Check character systems. *Computing*, 37:277–301, 1986.

[2] J. A. Gallian. Error detection methods. *ACM Comput. Surv.*, 28(3):504–517, sep 1996.

[3] J. A. Gallian. *Contemporary Abstract Algebra*. CengageBrain. com, 70 edition, 2010.

[4] H. Gumm. A new class of check-digit methods for arbitrary number systems. *IEEE Trans. Inf. Theor.*, 31(1):102–105, sep 1985.

[5] M. Malek. Decimal code, coding theory. Disponível em www.mcs.csueastbay.edu/ malek/TeX/Decimal.pdf, 2009.

[6] N. Pedroza. Uma análise dos esquemas de dígitos verificadores usados no Brasil. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro, 2013.

[7] J. Verhoeff. *Error Detecting Decimal Codes*. PhD thesis, Mathematical Centre Tract, 1969.

