Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Instituto de Matemática e Estatística

Natália Pedroza de Souza

Uma análise dos esquemas de dígitos verificadores usados no Brasil

Rio de Janeiro

2013

Natália Pedroza de Souza

Uma análise dos esquemas de dígitos verificadores usados no Brasil.

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro. Área de concentração: Otimização.

Orientadores: Prof. Dr. Luerbio Faria

Prof. Dr. Paulo Eustáquio Duarte Pinto

Rio de Janeiro

2013

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS/ BIBLIOTECA CTC/A

S729 Souza, Natália Pedroza de.
Uma análise dos esquemas de dígitos verificadores usados no Brasil / Natália Pedroza de Souza. - 2013.
94f.

Orientadores: Luerbio Faria, Paulo Eustáquio Duarte Pinto
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística.

1. Teoria da codificação - Teses. 2. Códigos de controle de erros (Teoria da informação) - Teses. 3. Teoria dos grupos - Teses. I. Faria, Luerbio. II. Pinto, Paulo Eustáquio Duarte. III. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. IV. Título.

CDU 004.415.3

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial deste trabalho, desde que citada a fonte.

Assinatura

Data

Natália Pedroza de Souza

Uma análise dos esquemas de dígitos verificadores usados no Brasil

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro. Área de concentração: Otimização.

Aprovado em 31 de julho de 2013.

Banca Examinadora:

Prof. Dr. Luerbio Faria

Instituto de Matemática e Estatística - UERJ

Prof. Dr. Paulo Eustáquio Duarte Pinto (Orientador)

Instituto de Matemática e Estatística - UERJ

Prof.ª Dra. Christina Fraga Esteves Maciel Waga

Instituto de Matemática e Estatística - UERJ

Prof. Dr. Mitre Costa Dourado

Departamento de Ciência da Computação - UFRJ

Rio de Janeiro

2013

AGRADECIMENTOS

Gostaria de agradecer à minha família por ter me dado acesso a uma boa educação. Ao Aylton Cordeiro, por seu incentivo à minha inscrição no processo seletivo do mestrado. Ao Moysés Sampaio, pelo trabalho inicial sobre o tema desta dissertação e por todo o seu apoio. Ao Pedro Henrique González, por sua colaboração com o editor de texto utilizado. A todos os professores que me ajudaram a traçar o caminho até aqui. E principalmente ao meu orientador Paulo Eustáquio, por ter acreditado em mim aceitando me orientar. Por todo o tempo que me foi dedicado. Por sua preocupação com a minha área de interesse. Por todo elogio e toda cobrança que me motivaram. E por toda a sua energia e paixão em trabalhar que inspiram a todos.

O estudo em geral, a busca da verdade e da beleza são domínios em que nos é consentido ficar crianças toda a vida.

Albert Einsten

RESUMO

SOUZA, Natália Pedroza de. *Uma análise dos esquemas de dígitos verificadores usados no Brasil.* 2013. 85f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

Neste trabalho discutimos vários sistemas de dígitos verificadores utilizados no Brasil, muitos deles semelhantes a esquemas usados mundialmente, e fazemos uma análise da sua capacidade de detectar os diversos tipos de erros que são comuns na entrada de dados em sistemas computacionais. A análise nos mostra que os esquemas escolhidos constituem decisões subotimizadas e quase nunca obtêm a melhor taxa de detecção de erros possível. Os sistemas de dígitos verificadores são baseados em três teorias da álgebra: aritmética modular, teoria de grupos e quasigrupos. Para os sistemas baseados em aritmética modular, apresentamos várias melhorias que podem ser introduzidas. Desenvolvemos um novo esquema ótimo baseado em aritmética modular base 10 com três permutações para identificadores de tamanho maior do que sete. Descrevemos também o esquema Verhoeff, já antigo, mas pouquíssimo utilizado e que também é uma alternativa de melhoria para identificadores de tamanho até sete. Desenvolvemos ainda, esquemas ótimos para qualquer base modular prima que detectam todos os tipos de erros considerados. A dissertação faz uso ainda de elementos da estatística, no estudo das probabilidades de detecção de erros e de algoritmos, na obtenção de esquemas ótimos.

Palavras-chave: Sistemas de dígitos verificadores. Detecção de erros. Aritmética modular.

ABSTRACT

SOUZA, Natália Pedroza de. *An analysis of check digit schemes used in Brazil.* 2013. 85f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

In this paper we present several check digit systems used in Brazil, many of them similar to schemes used worldwide, and we do an analysis of their ability to detect various types of errors that are common in data entry computer systems. This analysis shows that the schemes constitute suboptimal decisions and almost never get the best rate possible error detection. Check digit schemes are based on three algebra theory: modular arithmetic, group theory and quasigroup. For the schemes based on modular arithmetic we present several improvements that can be made. We developed a new optimal scheme based on modular arithmetic base 10 with three permutations for identifiers larger than 7. We also present the Verhoeff scheme, already old but used very little and that is also a good alternative for improvement identifiers for size up to 7. We have also developed, optimum schemes for any modular base prime that detect all types of errors considered. The dissertation also makes use of elements of statistics in the study of the probability of error detection and algorithms to obtain optimal schemes.

Keywords: Error detection. Check digit systems. Modular arithmetic.

SUMÁRIO

INTRODUÇÃO 10

1 **SISTEMAS DE DíGITOS VERIFICADORES** 13

1.1 **Esquemas baseados na Aritmética Modular** 13

1.1.1 Módulo 10 14

1.1.1.1 Módulo 10 com 2 pesos 15

1.1.1.2 Módulo 10 com 3 pesos 16

1.1.1.3 Módulo 10 Verhoeff 19

1.1.2 Módulo 11 20

1.1.3 Módulo 11 com dois dígitos verificadores 23

1.2 **Esquemas mistos** 26

1.2.1 Bases diferentes de 10 e 11 27

1.3 **Esquemas baseados na Teoria dos Grupos: Verhoeff** 28

1.4 **Esquemas baseados em Quadrados Latinos** 30

2 **ASPECTOS MATEMÁTICOS DOS ESQUEMAS DE DíGITOS VERIFICADORES** 34

2.1 **Detecção de erros nos esquemas baseados em aritmética modular** 34

2.1.1 Detecção de erros com Módulo 10 37

2.1.1.1 Detecção de erros individualizados 37

2.1.1.2 Detecção de transposições de dígitos adjacentes 38

2.1.1.3 Detecção de transposições de dígitos alternados 39

2.1.1.4 Detecção de erros gêmeos adjacentes 39

2.1.1.5 Detecção de erros gêmeos alternados 40

2.1.2 Detecção de erros com Módulo 11 e um dígito verificador 41

2.1.2.1 Detecção de erros individualizados 41

2.1.2.2 Detecção de transposições de dígitos adjacentes e alternados 42

2.1.2.3 Detecção de erros gêmeos adjacentes 42

2.1.2.4 Detecção de erros gêmeos alternados 43

2.1.3 Módulo 11 com dois dígitos verificadores 43

2.2 **Detecção de erros no esquema baseado na Teoria dos Grupos** 44
2.3 **Detecção de erros em esquemas baseados em Quadrados Latinos** . 45
2.4 **Resumo e análise de detecção de erros** 47

3 **ESQUEMAS MÓDULO 10 ÓTIMOS** 51
3.1 **Algoritmos para análise da detecção de erros em sistemas Módulo 10** 52
3.1.1 O sistema Módulo 10 de Verhoeff 54
3.2 **Esquema ótimo módulo 10** 58
3.3 **Comparação entre os dois métodos ótimos** 63

4 **MELHORIAS NOS ESQUEMAS DE DíGITOS VERIFICADORES** 64
4.1 **Esquema ótimo para bases modulares primas** 64
4.1.1 Esquema ótimo para módulo 11, com um dígito verificador 67
4.2 **Esquema ótimo para identificadores alfanuméricos com um dígito verificador** 68
4.3 **Esquemas ótimos para módulo 11 com dois dígitos verificadores** .. 68
4.4 **Resumo das melhorias propostas** 69

CONCLUSÃO 70

REFERÊNCIAS 71

ANEXO A – Detecção de erros para Módulo 11 com dois dígitos verificadores 75

ANEXO B – Distribuição de erros gêmeos para o esquema Módulo 10 Ótimo 83

ANEXO C – Média de não-detecção para o Módulo 10 Ótimo 84

INTRODUÇÃO

Os dígitos verificadores (DV's) são fundamentais no universo tecnológico. Estão presentes em incontáveis informações que são armazenadas e enviadas, especialmente no contexto comercial. Sem esses códigos, o número de informações seria muito pouco confiável em decorrência principalmente dos erros humanos na digitação e também erros na leitura de códigos de barra por equipamentos automatizados.

Dígitos verificadores são um ou mais caracteres acrescentados a uma cadeia de caracteres (numérica ou alfanumérica) original, que a certifica e/ou a corrige, dando maior segurança contra fraudes, erros de digitação ou leitura. Esses dígitos são formulados através de algoritmos que podem ser públicos ou não.

Tais dígitos são utilizados em códigos de barras de produtos, números de conta corrente, documentos de identidade, CPF, CNPJ etc. e têm grande importância no mundo comercial, na identificação civil, arrecadação de tributos, instituições financeiras e muitas outras áreas. Existem diversos tipos de erros. Os mais comuns foram categorizados por J. Verhoeff [30], baseado em um estudo de 12.000 erros. A Tabela 1 apresenta os principais erros humanos e suas frequências medidas empiricamente.

Tabela 1 - Tipos de erros e suas frequências na entrada de dados.

Tipo de erro	**Sigla**	**Formato**	**Frequência (%)**
Erros individualizados	IND	..a.. $\rightarrow$ ..b..	79.10
Transposição adjacente	TAD	..ab.. $\rightarrow$ ..ba..	10.20
Transposição alternada	TAL	..abc.. $\rightarrow$ ..cba..	0.80
Gêmeos adjacentes	GEM	..aa.. $\rightarrow$ ..bb..	0.50
Gêmeos alternados	GAL	..aca.. $\rightarrow$ ..bcb..	0.30
Fonéticos	FON	..1a.. $\rightarrow$ ..a0.., a$>$ 1	0.50
Outros	OUT		8.60

Observe que os erros fonéticos estão relacionados às línguas inglesa, holandesa e alemã, onde existem números com pronúncias muito semelhantes. No inglês, por exemplo, temos a semelhança fonética entre *thirty* e *thirteen*. Neste trabalho desconsideraremos este tipo de erro quando estivermos discutindo situações específicas do Brasil. Todos os erros aleatórios distintos dos descritos na tabela são englobados no tipo Outros e não serão analisados.

Para detectar os erros indicados na Tabela 1, normalmente adiciona-se uma redundância aos códigos que consta de um ou mais dígitos verificadores e, na maioria das

vezes, permite apenas sinalizar a ocorrência de erro. Em alguns casos, os dígitos acrescentados podem, inclusive, permitir a correção do erro. O ideal seria que qualquer esquema de dígito verificador detectasse todas as ocorrências de erros individualizados e de transposição de dígitos adjacentes, os dois tipos iniciais da tabela, que correspondem a quase 90% de todos os erros, sendo os demais bastante específicos.

O objetivo deste trabalho é analisar os esquemas de dígitos verificadores utilizados no Brasil, identificando fragilidades nos mesmos e indicando melhorias que possam ser feitas. Poder-se-ia perguntar qual a vantagem de despender esforços na análise de esquemas, na maioria dos casos já muito antigos e não passíveis de modificação em muitas situações. A resposta a essa pergunta é que muitos novos esquemas continuam sendo propostos e parece haver, nesse contexto, a perpetuação de decisões subotimizadas, como será visto no caso do passaporte, do CNH, do título eleitoral, relativamente recentes e do RIC, que ainda está sendo lançado no país. A análise trazida neste trabalho, bem como as propostas apresentadas, podem contribuir para interromper essa trajetória de repetições de erros.

A organização deste trabalho é descrita a seguir. O Capítulo 1 contém um levantamento bibliográfico sobre as técnicas para uso de dígitos verificadores. Contém, também, a descrição dos principais identificadores usados no Brasil que têm dígitos verificadores. O Capítulo 2 embasa a análise posterior de detecção de erros, apresentando um estudo matemático do poder de detecção dos diversos esquemas. Para os esquemas utilizados no Brasil, fizemos um aprofundamento dos erros não detectáveis pelos esquemas utilizados. Neste capítulo é ainda apresentado o resultado de uma extensa experimentação prática relativa aos diversos métodos analisados. Foram rodados inúmeros programas para confirmar todos os aspectos teóricos de detecção estudados. No Capítulo 3 resumimos os resultados encontrados no Capítulo 2, apresentando a tabela com as taxas de não detecção de todos os esquemas descritos. No Capítulo 4 desenvolvemos um esquema de dígito verificador ótimo que utiliza módulo 10. Apresentamos, também, uma alternativa computacional que obtém sistemas equivalentes ao sistema módulo 10 de Verhoeff, que, até o presente, era o melhor método dessa linha. Finalmente no Capítulo 5 apresentamos propostas de melhorias para os demais esquemas baseados em aritmética modular.

Em termos de divulgação de resultados, preparamos artigos para submissão em revistas. Apresentamos também este trabalho em forma de pôster no LAGOS 2013,

realizado em abril, no México. E ainda uma palestra, em maio, no I Simpósio do Programa de Pós-Graduação de Ciências da Computação do IME/UERJ.

1 SISTEMAS DE DÍGITOS VERIFICADORES

Neste capítulo iremos descrever os principais esquemas utilizados para calcular dígitos verificadores. Os esquemas analisados podem ser classificados em três grandes grupos, conforme a base matemática utilizada. Essas três bases são a Aritmética Modular, a Teoria dos Grupos e os Quadrados Latinos. A grande maioria das aplicações práticas em todo o mundo utiliza Aritmética Modular. Sempre que possível ilustraremos os esquemas com identificadores brasileiros detalhados em [1].

1.1 Esquemas baseados na Aritmética Modular

Este é o esquema mais divulgado universalmente [1], [11], [14], [16], [20], [22], [29] e [31]. Nestes esquemas são feitas operações aritméticas nos dígitos dos identificadores e os resultados são considerados em módulo n, para $n \in \mathbb{N}$. Dado um identificador com m dígitos $a_1a_2a_3 \cdots a_m$, o dígito verificador a_m é determinado da seguinte forma: pesos p_i de uma sequência predeterminada são multiplicados da esquerda para a direita aos dígitos iniciais: $a_i,\ 1 \leq i \leq m-1$. Em seguida, somam-se todos os valores obtidos e toma-se o resultado em módulo n. A equação do *esquema direto* é dada por,

$$a_1p_1 + a_2p_2 + a_3p_3 + \cdots + a_{m-1}p_{m-1} \bmod n = a_m. \tag{1}$$

Em alguns casos, para se obter o dígito verificador, devemos fazer o complemento do resultado em relação a n, quando este resultado é diferente de 0. Neste caso a equação do *esquema por complemento* é dada por:

$$a_1p_1 + a_2p_2 + a_3p_3 + \cdots + a_{m-1}p_{m-1} + a_m \equiv 0 \bmod n. \tag{2}$$

De forma mais abrangente, podemos, ao invés de atribuir pesos aos dígitos do identificador, definir funções σ_i, em cada posição $1 \leq i \leq m-1$. Então, a equação do

esquema direto pode ser reescrita da seguinte forma:

$$\sum_{i=1}^{m-1} \sigma_i(a_i) \bmod n = a_m. \tag{3}$$

A maioria dos identificadores sobre os quais os dígitos verificadores são calculados utiliza dígitos decimais e, usualmente, a base modular é 10 ou 11. A seguir descrevemos as principais variações dos métodos que usam essas bases.

1.1.1 Módulo 10

Este é um dos métodos mais naturais quando se trabalha com identificadores compostos por dígitos decimais, pois o dígito verificador também será um elemento desse conjunto. O efeito de multiplicar cada dígito por um peso é equivalente a se aplicar uma função a esse dígito. Por exemplo, ao se multiplicar por 2 os dígitos de 0 a 9 em módulo 10, os resultados obtidos são, respectivamente, 0, 2, 4, 6, 8, 0, 2, 4, 6 e 8. Note que esta escolha acarreta na não detecção de erros individuais, pois a troca entre os dígitos i e $i+5$ para $0 \leq i \leq 4$ não alterara o cálculo do dígito verificador. Portanto, é desejável que esta função seja uma permutação de 0 a 9 para identificadores numéricos. Ao se multiplicar por 3 os dígitos de 0 a 9 em módulo 10, são obtidos os seguintes resultados, respectivamente 0, 3, 6, 9, 2, 5, 8, 1, 4 e 7. Portanto, esta operação gera os mesmos valores que a aplicação da permutação $\sigma : (0\ 3\ 6\ 9\ 2\ 5\ 8\ 1\ 4\ 7)$ nos dígitos de 0 a 9. Note que isto acontece porque $mdc(10, 3) = 1$.

Foi mostrado na Introdução que os erros individualizados são os mais frequentes. Devido a isso, a primeira condição que um sistema modular de base n procura atender é que os pesos escolhidos sejam relativamente primos a n. Desta forma o resultado é o mesmo que a aplicação de uma permutação de 0 a $n-1$ aos dígitos do identificador. Com isso, todos os erros individualizados são detectados.

As variantes mais comuns encontradas utilizam indiretamente permutações através de esquemas de 2 ou 3 pesos. O método também é usado em algumas situações onde os identificadores são alfanuméricos e, neste caso, é necessária a conversão de caracteres não numéricos para dígitos decimais.

1.1.1.1 Módulo 10 com 2 pesos

O primeiro exemplo descrito é o **código de barras de produtos**, usado mundialmente para a identificação comercial de produtos. Esse código apresenta uma versão para leitura óptica através de scanners e outra versão com dígitos para entrada manual. O identificador consiste em 13 dígitos decimais dos quais os três primeiros representam o país do código, no caso do Brasil, são os números 789. Os quatro seguintes são associados à empresa cadastrada e os cinco finais antes do do dígito verificador representam o código do produto em si.

Para o cálculo do dígito verificador, é utilizada a equação por complemento com pesos 1 e 3, aplicados alternadamente. Como exemplo mostraremos como foi calculado o dígito verificador para o código 7 89102 711427 5. Multiplica-se os dígitos alternadamente por 1 e 3:

$$\begin{array}{cccccccccccccc} & 7 & 8 & 9 & 1 & 0 & 2 & 7 & 1 & 1 & 4 & 2 & 7 \\ \times & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 \\ \hline & 7 & 24 & 9 & 3 & 0 & 6 & 7 & 3 & 1 & 12 & 2 & 21 \end{array}$$

Os resultados são somados, obtendo-se $7+24+9+3+0+6+7+3+1+12+2+21 = 95 \equiv 5 \bmod 10$. O valor do dígito verificador é o complemento de 5 para 10, ou seja, 5. Este esquema detecta todos os erros individualizados, muitas transposições de dígitos adjacentes e erros gêmeos, mas nenhuma transposição alternada de dígitos.

Módulo 10 IBM

Uma variante do esquema por complemento módulo 10 foi proposta por Luhn [21], nos anos 60, tendo sido, inclusive, objeto de uma patente. Nesta variante 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 deste produto quando este é maior que 9.

Este esquema é utilizado no documento de **Registro Geral do Estado do Rio de Janeiro**, que consiste em um número decimal com nove dígitos usado na identificação

civil. Para exemplificar, utilizaremos o número de identidade 21883353-1. Inicialmente multiplicamos cada algarismo pelo seu peso correspondente:

$$\begin{array}{rcccccccc} & 2 & 1 & 8 & 8 & 3 & 3 & 5 & 3 \\ \times & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ \hline & 2 & 2 & 8 & 16 & 3 & 6 & 5 & 6 \end{array}$$

Aplicamos os "noves fora" a 16, obtendo 1+6 = 7, e a soma dos resultados é $2+2+8+7+3+6+5+6 = 39 \equiv 9 \mod 10$. Toma-se o dígito verificador como o complemento de 9 para 10, ou seja, 1 e assim, temos satisfeita a equação do esquema.

O Módulo 10 IBM é também utilizado no identificador do **cartão de crédito**. Esse código é adotado em toda a rede bancária mundial. Consiste em um número variando de 14 a 19 dígitos decimais, onde os iniciais identificam o banco e a bandeira do cartão. No Brasil é comum que se utilizem 16 dígitos. Exemplificamos a seguir este esquema para o cartão de número 5549 0630 0976 5723. Inicialmente, multiplicam-se os primeiros 15 valores pelos seus respectivos pesos.

$$\begin{array}{rcccccccccccccc} & 5 & 5 & 4 & 9 & 0 & 6 & 3 & 0 & 0 & 9 & 7 & 6 & 5 & 7 & 2 \\ \times & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ \hline & 10 & 5 & 8 & 9 & 0 & 6 & 6 & 0 & 0 & 9 & 14 & 6 & 10 & 7 & 4 \end{array}$$

É aplicada a operação "noves fora" aos resultados e tudo é somado, obtendo-se $(1+0)+5+8+9+6+6+9+(1+4)+6+(1+0)+7+4 = 67 \equiv 7 \mod 10$. O valor do dígito verificador é o complemento de 7 para 10, ou seja, 3.

Note que o esquema Módulo 10 IBM é equivalente a aplicar alternadamente duas permutações, a identidade e $\sigma : (0\ 2\ 4\ 6\ 8\ 1\ 3\ 5\ 7\ 9)$, nos dígitos do identificador. Isto porque, multiplicando por 2 os dígitos de 0 a 9 e fazendo "noves fora" em cada resultado obtem-se a segunda permutação. Será visto adiante que este esquema detecta todos os erros individualizados, muitas transposições de dígitos adjacentes e erros gêmeos. Entretanto, não detecta nenhuma dos erros alternados, pois os pesos correspondentes a essas posições são iguais.

1.1.1.2 Módulo 10 com 3 pesos

A solução para se detectar erros alternados é utilizar três pesos, que devem estar no conjunto $\{1, 3, 7, 9\}$, ou de forma equivalente, aplicar 3 permutações aos dígitos do identificador.

O número do **passaporte** no Brasil utiliza o padrão internacional MRP (Machine Readable Passport) definido pela OACI (Organização de Aviação Civil Internacional). Esse padrão consiste em duas linhas de informação, cada uma com 44 caracteres, onde a primeira contém somente informações sobre o passageiro. Utilizam-se as letras maiúsculas de A a Z e os dígitos de 0 a 9. No padrão existem campos de livre preenchimento que cada país pode adotar ou não. Além disso, os tamanhos dos identificadores podem variar dentro de certos limites. O símbolo $<$ é usado como espaço vazio para levar em conta essas diferenças.

A primeira linha não possui nenhum esquema de dígitos verificadores. Na segunda linha, os 10 primeiros caracteres representam o identificador do passaporte, que é a informação onde vamos focar. Outros campos nessa linha são associados à nacionalidade do passageiro (BRA, no caso dos brasileiros), data de nascimento, sexo, data de expiração do passaporte, campos livres e mais três dígitos verificadores associados a esses últimos dados descritos.

Para o cálculo do DV, é utilizada a equação direta e os pesos 7, 3 e 1. Sendo que 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 e o caractere $<$ é mapeado para 0. Vamos detalhar apenas a obtenção do dígito verificador do identificador. Vejamos, por exemplo, como foi calculado o dígito verificador para o passaporte F964114F<3. Inicialmente, converte-se F para 15 e $<$ para 0. Em seguida aplicam-se os pesos:

$$\begin{array}{cccccccccc}
 & 15 & 9 & 6 & 4 & 1 & 1 & 4 & 15 & 0 \\
\times & 7 & 3 & 1 & 7 & 3 & 1 & 7 & 3 & 1 \\
\hline
 & 105 & 27 & 6 & 28 & 3 & 1 & 28 & 45 & 0
\end{array}$$

Os resultados são somados, obtendo-se $105+27+6+28+3+1+28+45+0 = 243 \equiv 3 \mod 10$. Portanto, o dígito verificador é 3. No próximo capítulo mostraremos que este esquema deixa a desejar na detecção de erros. Pode-se ver facilmente 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 pode ser detectado. O esquema não possibilita, também, a detecção de algumas trocas de dígitos adjacentes bem como de erros gêmeos.

Foram feitas várias tentativas para melhorar o Módulo 10 IBM, especialmente no que toca à detecção de transposições adjacentes. A seguir apresentamos algumas dessas variantes, descritas com mais detalhes em [11].

Módulo 10 IBM Generalizado

Uma melhoria no código IBM se dá neste esquema onde são utilizadas as potências da permutação $\sigma : (0\ 2\ 4\ 6\ 8\ 1\ 3\ 5\ 7\ 9)$. Isso possibilita a detecção de erros de transposição alternada. A equação do esquema, neste caso, é dada por:

$$\sum_{i=1}^{m-1} \sigma^{i-1}(a_{m-i}) \equiv a_m \bmod 10. \tag{4}$$

Módulo 10 IBM Generalizado Modificado

Nesta variante, a permutação σ do código IBM Generalizado é modificada para $\mu(a) = \sigma(a) + 6 \bmod 10$, sendo $a = 0, 1, \cdots, 9$. Ou seja, $\mu : (6\ 8\ 0\ 2\ 4\ 7\ 9\ 1\ 3\ 5)$.

A detecção de erros individuais, de transposição e gêmeos é a mesma, mas este método é melhor para os outros tipos de erros.

Código Colenbrander e Colenbrander Modificado

Aqui apresentamos duas variações do Código Colenbrander, que se diferem apenas pela permutação utilizada no esquema. A equação do cálculo do dígito verificador é novamente como a do Módulo 10 IBM generalizado, apenas mudando a permutação. O método Colenbrander utiliza a permutação $f : (9\ 2\ 4\ 6\ 8\ 0\ 1\ 3\ 5\ 7)$. Já no Colenbrander Modificado, a permutação é $\lambda : (1\ 3\ 5\ 7\ 9\ 0\ 2\ 4\ 6\ 8)$.

A permutação λ é obtida a partir de f através da equação $\lambda = (f((a+1) \bmod 10) - 1) \bmod 11$. A média de detecção de erros é a mesma em ambos os métodos. Porém no método Colenbrander Modificado os erros fonéticos são detectados em 100% das vezes.

Codigo P.T.T. Alemão

Este método foi criado por um banco alemão. São usadas as seguintes três per-

mutações aplicadas alternadamente aos dígitos do identificador:

$$\sigma_1 : (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 0)$$

$$\sigma_2 : (2\ 4\ 6\ 8\ 0\ 1\ 3\ 5\ 7\ 9)$$

$$\sigma_3 : (3\ 6\ 9\ 1\ 4\ 7\ 0\ 2\ 5\ 8)$$

Estas permutações são encontradas através da fórmula $\sigma_i(a) = ((ia + i) \bmod 11) \bmod 10$.

1.1.1.3 Módulo 10 Verhoeff

Em sua tese de doutorado Verhoeff [30] desenvolveu o esquema descrito a seguir. Procurou permutações $f_i, i = \{0, 1, \cdots, m-1\}$ para serem aplicadas aos dígitos $a_0, a_1, \cdots, a_{m-1}$ obtendo o dígito verificador com o seguinte cálculo $a_m = \sum_{i=0}^{m-1} f_i(a_i) \bmod 10$. As permutações deveriam satisfizer prioritariamente as condições de detecção de erros adjacentes. Posteriormente, as condições de detecção erros alternados. E por fim, as condições de deteção de erros fonéticos. O processo é descrito no Capítulo 3.

Na Tabela 2 a seguir as permutações são dadas.

Tabela 2 - Permutações do esquema de Verhoeff

x	0	1	2	3	4	5	6	7	8	9
$f_0(x)$	5	8	1	3	6	9	7	0	4	2
$f_1(x)$	5	0	6	1	7	2	3	9	8	4
$f_2(x)$	0	1	8	7	5	9	6	4	2	3
$f_3(x)$	1	8	6	9	0	4	2	3	5	7
$f_4(x)$	5	0	4	8	2	9	3	6	1	7
$f_5(x)$	0	1	7	5	6	8	9	4	3	2
$f_6(x)$	5	8	0	9	7	4	2	6	3	1
$f_7(x)$	5	0	9	2	3	8	4	7	1	6
$f_8(x)$	0	1	4	9	6	2	3	5	7	8
$f_9(x)$	1	8	3	4	2	5	7	0	9	6

Vamos aplicar o esquema a um identificador com dígitos iniciais 2194673290. O cálculo do dígito verificador é dado por:

$$a_{10} = f_0(2) + f_1(1) + f_2(9) + f_3(4) + f_4(6) + f_5(7) + f_6(3) + f_7(2) + f_8(9) + f_9(0) \bmod 10. \quad (5)$$

Portanto temos $a_{10} = 1 + 0 + 3 + 8 + 3 + 4 + 9 + 9 + 8 + 1 \bmod 10 = 6$.

Este sistema possui a melhor média de detecção de erros dentre todos os métodos módulo 10 descritos neste trabalho. Entretanto, não encontramos nenhuma utilização prática do mesmo. Este fato é muito intrigante porque se repete no sistema de dígito verificador criado pelo mesmo autor, usando a Teoria de Grupos e descrito adiante. Este outro método é ainda melhor quanto à detecção de erros. A razão da não utilização prática de bons métodos parece estar relacionada com a dificuldade de entendimento e de implementação do método. Baseado nisso é que procuramos desenvolver neste trabalho um esquema módulo 10 que possua uma média de detecção muito próxima a dos métodos desenvolvidos por Verhoeff, mas que seja de compreensão e implementação mais fáceis.

1.1.2 Módulo 11

Quando utilizamos o módulo 11 para o cálculo do dígito verificador, considerando identificadores numéricos, temos uma particularidade. O resto da divisão por 11 pode ser 10, exigindo alguma forma especial para representar essa situação. Para se usar apenas um dígito verificador, são adotadas duas soluções diferentes:

1. *Módulo 11 completo* - Neste caso utiliza-se o caractere X para representar o resto 10.

2. *Módulo 11 restrito* - Esta outra possibilidade utiliza o dígito 0 para representar o resto 10.

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 dificuldade na entrada dos identificadores. Por exemplo, muitas vezes a informação tem que ser dada por telefone. Nesse caso, 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 erros que alteram o cálculo do dígito verificador de 0 para 10, ou vice-versa.

Multiplicar em módulo 11 os dígitos de um identificador por 3 ou 2 é equivalente a aplicar, respectivamente, as funções injetoras $\varphi_1 : (0\ 3\ 6\ 9\ 1\ 4\ 7\ 10\ 2\ 5)$ e

φ_2 : (0 2 4 6 8 10 1 3 5 7). Note que para qualquer peso a função é sempre injetora, pois 11 é um número primo.

Apresentamos a seguir, esquemas baseados em módulo 11 utilizados no Brasil. O primeiro esquema utiliza módulo 11 completo e os demais módulo 11 restrito.

Módulo 11 completo

O número da agência do **Banco do Brasil** é representado por 5 dígitos decimais. Para o cálculo do dígito verificador é utilizado o esquema por complemento, com atribuição dos pesos $\langle 5, 4, 3, 2 \rangle$. Já o número da conta corrente possui 8 dígitos e os pesos utilizados são $\langle 8, 7, 6, 5, 4, 3, 2 \rangle$. Vamos ilustrar, simultaneamente, o cálculo para agência 1642-X e conta corrente 0343488-5.

$$\begin{array}{rcccc} & 1 & 6 & 4 & 2 \\ \times & 5 & 4 & 3 & 2 \\ \hline & 5 & 24 & 12 & 4 \end{array} \qquad \begin{array}{ccccccc} 0 & 3 & 4 & 3 & 4 & 8 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ \hline 0 & 21 & 24 & 15 & 16 & 24 & 16 \end{array}$$

Para a agência temos soma 5+24+12+4 = 45, cujo resto da divisão por 11 é 1. O complemento de 1 em módulo 11 é 10, portanto o dígito verificador é representado por X. No caso da conta corrente, a soma das parcelas é $0 + 21 + 24 + 15 + 16 + 24 + 16 = 116$. O resto da divisão de 116 por 11 é 6 e o dígito verificador é 5.

Este esquema detecta todos os erros individualizados e de transposição de dígitos adjacentes e alternados, e também grande parte dos erros gêmeos.

Uma propriedade adicional do esquema relativo ao módulo 11 completo é que ele permite a correção de erros de um dígito, quando se sabe qual é a posição errada (isso pode ocorrer, por exemplo, se tiver havido alguma rasura no documento fonte). Se o erro foi na posição i, faz-se o cálculo da equação do esquema, considerando a_i da parcela $p_i a_i$ uma incógnita. Aplicando-se a operação modular, obtém-se: $p_i a_i \equiv x \bmod 11$, onde p_i e x são conhecidos. O cálculo de a_i é baseado em uma propriedade da aritmética modular para números primos, como é o caso do 11. A propriedade é a seguinte.

Se n é primo e tomamos $1 \leq a \leq n-1$, os produtos ia, para $0 \leq i \leq n-1$ em módulo n, formam uma permutação dos números de 0 a $n-1$. Descobrimos, desta forma, qual deve ser o valor a multiplicar pelo peso p_i para obter o resultado desejado e, portanto, esse é o valor de a_i.

Módulo 11 restrito

Para números de agências e contas correntes do **Bradesco** é utilizado o esquema por complemento em módulo 11 restrito com os pesos $\langle 5,4,3,2\rangle$ e $\langle 7,6,5,4,3,2\rangle$ respectivamente. Vamos ilustrar o cálculo para agência 6128-0 e conta corrente 055163-5.

$$\begin{array}{rcccc} & 6 & 1 & 2 & 8 \\ \times & 5 & 4 & 3 & 2 \\ \hline & 30 & 4 & 6 & 16 \end{array} \qquad \begin{array}{cccccc} 0 & 5 & 5 & 1 & 6 & 3 \\ 7 & 6 & 5 & 4 & 3 & 2 \\ \hline 0 & 30 & 25 & 4 & 18 & 6 \end{array}$$

Para a agência temos soma $30 + 4 + 6 + 16 = 56$. Para que o resultado seja congruente a 0 em módulo 11, a parcela relativa ao dígito verificador deve ser 10. O esquema transforma o resultado 10 para 0, sendo este o dígito verificador. No caso da conta corrente, a soma das parcelas é $0 + 30 + 25 + 4 + 18 + 6 = 83$. O resto da divisão de 83 por 11 é 6 e, portanto, o dígito verificador é 5.

A transformação de 10 para 0 no esquema de módulo 11 restrito leva a não detecção de muitos erros de tipos variados, neste aspecto, portanto, o esquema é inferior ao módulo 11 completo.

No caso da **Caixa Econômica Federal**, o dígito verificador é feito com o esquema direto para o conjunto de campos envolvendo a agência com 4 dígitos, o identificador de operação com 3 dígitos (001 significa operação com conta corrente) e o número da conta com 8 dígitos. Os multiplicadores são $\langle 8,7,6,5,4,3,2,9,8,7,6,5,4,3\rangle$.

O identificador internacional de livros **ISBN** (International Standard Book Number) é adotado mundialmente, inclusive no Brasil, e sua padronização é feita pela ISO (International Organization for Standardization). Esse padrão sofreu variações ao longo dos anos, as principais são chamadas ISBN10 (usada até 2007), contendo 10 dígitos, e ISBN13 para 13 dígitos. No identificador com padrão ISBN10, os 9 primeiros dígitos identificam a linguagem, o editor e o número serial do livro. O cálculo do dígito verificador é feito com o esquema por complemento em módulo 11 completo, com pesos $\langle 10,9,8,7,6,5,4,3,2\rangle$. Já o padrão ISBN13 adota o mesmo esquema módulo 10 utilizado no código de barras de produtos.

O **PIS** – Programa de Integração Social – é administrado pela Caixa Econômica Federal e voltado aos trabalhadores da iniciativa privada. O **PASEP** - Programa de Formação do Patrimônio do Servidor Público - é administrado pelo Banco do Brasil e voltado aos servidores públicos. Ambos possuem identificador com a estrutura de 11

dígitos numéricos que segue o esquema direto de módulo 11 restrito. Os multiplicadores são dados pelo conjunto $\langle 3, 2, 9, 8, 7, 6, 5, 4, 3, 2 \rangle$.

1.1.3 Módulo 11 com dois dígitos verificadores

Alguns esquemas utilizam dois dígitos verificadores, ao invés de um só, para uma maior detecção de erros. Seja um identificador com m dígitos da forma $a_1 a_2 a_3 \cdots a_{m-1} a_m$, onde a_{m-1} e a_m são os dois dígitos verificadores. As equações do esquema normalmente têm a seguinte forma:

$$\begin{aligned} &a_1 p_1 + a_2 p_2 + a_3 p_3 + \cdots + a_{m-2} p_{m-2} \bmod 11 = a_{m-1} \quad \text{e} \\ &a_1 p_1 + a_2 p_2 + a_3 p_3 + \cdots + a_{m-2} p_{m-2} + a_{m-1} p_{m-1} \bmod 11 = a_m. \end{aligned} \tag{6}$$

Caso o resultado do cálculo seja 10, o dígito verificador é 0.

No Brasil estes esquemas são utilizados em vários identificadores. O primeiro deles é o **CPF** (Cadastro de Pessoas Físicas) que identifica os contribuintes do tipo pessoa física junto à Receita Federal. Este identificador é formado por 11 dígitos decimais, com os pesos $\langle 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$ para a obtenção do primeiro dígito verificador e $\langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$ para o segundo. Vamos ilustrar o caso do CPF 587171594-07. O cálculo do primeiro dígito verificador é dado por:

$$\begin{array}{cccccccccc} & 5 & 8 & 7 & 1 & 7 & 1 & 5 & 9 & 4 \\ \times & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline & 5 & 16 & 21 & 4 & 35 & 6 & 35 & 72 & 36 \end{array}$$

A soma das parcelas é $5 + 16 + 21 + 4 + 35 + 6 + 35 + 72 + 36 = 230 \equiv 10 \bmod 11$. Neste caso, o resultado é transformado em 0, o primeiro dígito verificador. O cálculo do segundo dígito verificador é dado por:

$$\begin{array}{cccccccccc} & 8 & 7 & 1 & 7 & 1 & 5 & 9 & 4 & 0 \\ \times & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline & 8 & 14 & 3 & 28 & 5 & 30 & 63 & 32 & 0 \end{array}$$

A soma das parcelas é $8 + 14 + 3 + 28 + 5 + 30 + 63 + 32 + 0 = 183 \equiv 7 \bmod 11$. Portanto, o segundo dígito é 7.

O uso de dois dígitos verificadores no CPF diminui os casos de falha de detecção em relação a usar apenas um dígito com o módulo 11 restrito mas, como veremos adiante, poderia ter levado a uma maior capacidade de detecção de erros, caso a escolha dos pesos tivesse sido feita de forma mais cuidadosa.

O próximo esquema a ser examinado é o do identificador do **CNPJ** (Cadastro Nacional de Pessoas Jurídicas), que identifica as empresas contribuintes junto à Receita Federal. Este identificador é formado por 14 dígitos decimais, onde os oito dígitos iniciais representam a identificação da empresa e os quatro seguintes, a filial. Os multiplicadores para o cálculo do primeiro dígito são $\langle 6, 7, 8, 9, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$ e, para o segundo, $\langle 5, 6, 7, 8, 9, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$. Vamos ilustrar o cálculo para o CNPJ 03.749.639/0001. O cálculo do primeiro dígito verificador é:

$$\begin{array}{cccccccccccccc}
 & 0 & 3 & 7 & 4 & 9 & 6 & 3 & 9 & 0 & 0 & 0 & 1 \\
\times & 6 & 7 & 8 & 9 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
 & 0 & 21 & 56 & 36 & 18 & 18 & 12 & 45 & 0 & 0 & 0 & 9
\end{array}$$

A soma das parcelas é $0 + 21 + 56 + 36 + 18 + 18 + 12 + 45 + 0 + 0 + 0 + 9 = 215 \equiv 6 \bmod 11$, portanto primeiro DV é 6. O cálculo do segundo dígito verificador é dado por:

$$\begin{array}{ccccccccccccccc}
 & 0 & 3 & 7 & 4 & 9 & 6 & 3 & 9 & 0 & 0 & 0 & 1 & 6 \\
\times & 5 & 6 & 7 & 8 & 9 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
 & 0 & 18 & 49 & 32 & 81 & 12 & 9 & 36 & 0 & 0 & 0 & 8 & 54
\end{array}$$

A soma das parcelas é $0 + 18 + 49 + 32 + 81 + 12 + 9 + 36 + 0 + 0 + 8 + 54 = 299 \equiv 2 \bmod 11$. Portanto, o segundo DV é 2.

Este é o mais bem sucedido esquema que utiliza dois dígitos verificadores no Brasil, falhando em poucos casos de cada tipo de erro.

O terceiro esquema desta classe é o identificador da **CNH** (Carteira Nacional de Habilitação) emitida pelos DETRANs (Departamentos Estaduais de Trânsito) sob a orientação do DENATRAN (Departamento Nacional de Trânsito). Este identificador é constituído de 11 dígitos decimais, os multiplicadores para o cálculo do primeiro DV são $\langle 9, 8, 7, 6, 5, 4, 3, 2, 1 \rangle$ e, do segundo, $\langle 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$, existindo uma particularidade

no cálculo desse segundo dígito, explicada adiante. A equação que define o esquema para o primeiro dígito verificador é:

$$(9a_1 + 8a_2 + \cdots + 2a_8 + a_9) \bmod 11 = a_{10}, \tag{7}$$

lembrando que, quando o resultado é 10, a_{10} é transformado em 0. A equação básica para o cálculo do segundo dígito é a seguinte:

$$(a_1 + 2a_2 + \cdots + 8a_8 + 9a_9) \bmod 11 = r. \tag{8}$$

O valor de a_{11} depende do resultado r e do cálculo inicial de a_{10}.

1. Se $a_{10} \neq 10$ e $r = 10$, então a_{11} é 0.
2. Se $a_{10} \neq 10$ e $r \neq 10$, então $a_{11} = r$.
3. Se $a_{10} = 10$ e $r \leq 2$, então $a_{11} = r$.
4. Se $a_{10} = 10$ e $r > 2$, então $a_{11} = r - 2$.

Vamos ilustrar o cálculo dos dígitos verificadores para a carteira de número 033756375-04. O cálculo do primeiro dígito verificador é:

$$\begin{array}{cccccccccc} & 0 & 3 & 3 & 7 & 5 & 6 & 3 & 7 & 5 \\ \times & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ \hline & 0 & 24 & 21 & 42 & 25 & 24 & 9 & 14 & 5 \end{array}$$

A soma das parcelas é $0 + 24 + 21 + 42 + 25 + 24 + 9 + 14 + 5 = 164 \equiv 10 \bmod 11$. Como o resultado é 10, o primeiro dígito verificador é transformado para 0. O cálculo do segundo dígito verificador é dado por:

$$\begin{array}{cccccccccc} & 0 & 3 & 3 & 7 & 5 & 6 & 3 & 7 & 5 \\ \times & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline & 0 & 6 & 9 & 28 & 25 & 36 & 21 & 56 & 45 \end{array}$$

A soma das parcelas é $0 + 18 + 39 + 32 + 81 + 12 + 9 + 36 + 0 + 0 + 8 + 54 = 226 \equiv 6 \bmod 11$. Como o cálculo inicial do primeiro dígito resultou em 10, então subtrai-se 2 desse resultado para obter o segundo dígito, cujo valor é 4.

Este esquema especial do cálculo trouxe desvantagens na capacidade de detecção de erros, conforme será mostrado adiante.

O identificador do **título eleitoral** é um número de 12 dígitos (13 no caso dos estados SP e MG), emitido pelo TSE. Os 8 primeiros dígitos são um número sequencial, os dois seguintes identificam o estado. Para o cálculo do primeiro dígito verificador são utilizados os pesos $\langle 2,3,4,5,6,7,8,9\rangle$ nos primeiros 8 dígitos do identificador e, para o segundo DV, são utilizados os pesos$\langle 7,8,9\rangle$ nos dígitos a_9, a_{10} e no primeiro DV, a_{11}. Para o título eleitoral de número 0082474803-02, o cálculo do primeiro dígito verificador é dado por:

$$\begin{array}{cccccccccc} & 0 & 0 & 8 & 2 & 4 & 7 & 4 & 8 \\ \times & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline & 0 & 0 & 32 & 10 & 24 & 49 & 32 & 72 \end{array}$$

A soma das parcelas é $0+0+32+10+24+49+32+72 = 219 \equiv 10 \bmod 11$. Como o resultado é 10, o primeiro dígito verificador é transformado em 0. O cálculo do segundo dígito verificador é dado por:

$$\begin{array}{cccc} & 0 & 3 & 0 \\ \times & 7 & 8 & 9 \\ \hline & 0 & 24 & 0 \end{array}$$

A soma das parcelas é $0+24+0 = 24 \equiv 2 \bmod 11$. Portanto, o segundo dígito verificador é 2.

Este esquema não tira partido do fato de usar dois dígitos verificadores e, como será visto adiante, chega a ser inferior a esquemas que utilizam apenas um dígito verificador.

1.2 Esquemas mistos

Nesta seção descrevemos brevemente a utilização de dígitos verificadores em dois tipos de documentos de cobrança bancária, apenas para ressaltar que, nos dois casos, há um enfoque misto utilizando tanto os esquemas módulo 10 quanto módulo 11.

Estes esquemas são utilizados para **arrecadação de tributos e pagamentos de contas relacionadas ao serviço público** e para **boletos de cobrança bancária**. Ambos possuem uma versão de código de barras para a leitura óptica e uma versão para digitação. No primeiro, a versão numérica corresponde a uma sequência de 44 dígitos

decimais, com informações tais como: tipo de serviço, valor, CNPJ da empresa, data de vencimento. No segundo, as informações contidas no documento são a identificação do banco, tipo de moeda, valor, data de vencimento e vários campos de preenchimento livre, de acordo com cada banco

Na versão de código de barras existe um dígito verificador geral e, na versão para digitação, são acrescentados mais quatro dígitos verificadores. Os esquemas utilizados são módulo 11 restrito ou módulo 10 IBM, dependendo do tipo de tributo ou conta.

RIC

Vale ressaltar que está sendo implementado no Brasil o Registro de Identidade Civil (RIC), um cartão com chip que conterá os números do RG, CPF, título eleitoral, e ainda, informações de biotipo. O identificador será um número de 11 dígitos, sendo o último um dígito verificador, que é calculado utilizando o esquema direto com módulo 11 restrito e os seguintes multiplicadores $\langle 9, 8, 7, 6, 5, 4, 3, 2, 9, 8 \rangle$.

1.2.1 Bases diferentes de 10 e 11

Esquemas que utilizam bases diferentes de 10 e 11 são incomuns, mas alguns exemplos podem ser encontrados em [11], [16] e [13], todos usados nos Estados Unidos. O esquema de dígito verificador do identificador de Ordem de Pagamento Postal e dos Cheques de Viagem da Visa utilizam módulo 9. A companhia aérea americana Federal Express e a empresa United Parcel Service utilizam módulo 7. Estes são os esquemas menos eficientes encontrados.

Foi encontrada, ainda, a proposta do uso de dois dígitos verificadores para identificadores numéricos com base modular 97. Neste caso é feito apenas um cálculo, o resultado é representado em dois algarismos, sendo estes considerados os dígitos verificadores. Por exemplo, se o resultado é 7, o dígitos verificadores são 0 e 7, se o resultado é 45, os dígitos verificadores são 4 e 5. Este esquema é capaz de detectar todos os cinco tipos de erros considerados.

No caso de identificadores alfanuméricos são relatados dois tipos de tratamentos. O primeiro é o caso do passaporte, já descrito na seção Módulo 10. Outro esquema possível

seria calcular o dígito verificador usando módulo 37, podendo esse dígito ser uma letra ou o caractere especial <. O esquema seria capaz de detectar todos os cinco tipos de erros que estamos considerando.

1.3 Esquemas baseados na Teoria dos Grupos: Verhoeff

Como será visto posteriormente neste trabalho, nenhum dos métodos que utiliza um dígito verificador decimal detecta todos os seis principais tipos de erros na entrada de identificadores. Os métodos baseados em esquemas módulos 10 detectam o primeiro tipo de erro facilmente, porém, falham no segundo. Os esquemas módulo 11 não conseguem detectar nenhum dos dois tipos de erros com perfeição, devido à limitação que estes sofrem por causa da diferença de base numérica entre o dígito verificador e os algarismos utilizados para representar o código, no caso de identificadores com dígitos decimais.

Uma das razões destes métodos não conseguirem detectar esses tipos de erros com eficiência é a utilização da operação de soma que, no caso da aritmética modular, é comutativa.

Tentativas diferentes foram feitas utilizando a Teoria de Grupos descritas em [30], [5], [17] e [18]. Podemos definir, para o conjunto de entrada de dados $D = \{a_1, a_2, \cdots, a_m\}$, uma operação não comutativa $*$ de forma que $(D, *)$ seja um grupo. A equação do esquema do dígito verificador pode ser escrita como:

$$\sigma_1(a_1) * \sigma_2(a_2) * \sigma_3(a_3) * \cdots * \sigma_{m-1}(a_{m-1}) \bmod n = a_m. \tag{9}$$

Estamos interessados nos casos em que o grupo D pode ser associado ao conjunto dos números de 0 a 9. A seguir descrevemos o esquema de Verhoeff que segue exatamente essas ideias.

Verhoeff em [30] desenvolveu um esquema não comutativo que detecta com perfeição erros individualizados e de transposições adjacentes. Este esquema é baseado no grupo diedral D_5 que representa as simetrias de um pentágono. Este grupo é associado à operação $*$ que simboliza a composição entre as operações de rotação e espelhamento deste pentágono.

Figura 1 - Pentágono

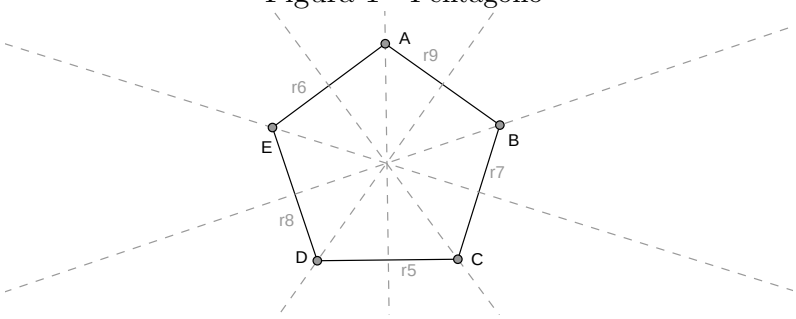


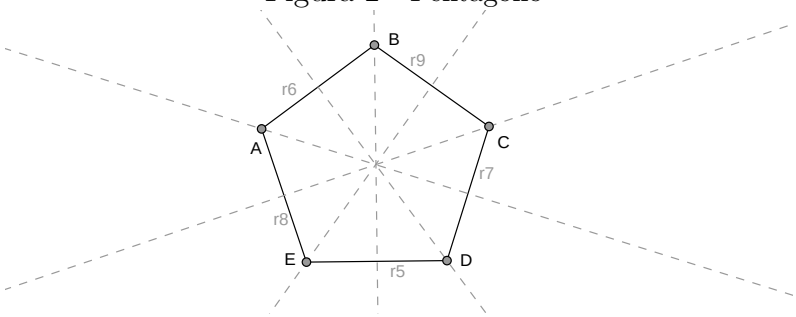
Tabela 3 - Tabela Cayley do grupo diedral D_5

$*$	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	0	6	7	8	9	5
2	2	3	4	0	1	7	8	9	5	6
3	3	4	0	1	2	8	9	5	6	7
4	4	0	1	2	3	9	5	6	7	8
5	5	9	8	7	6	0	4	3	2	1
6	6	5	9	8	7	1	0	4	3	2
7	7	6	5	9	8	2	1	0	4	3
8	8	7	6	5	9	3	2	1	0	4
9	9	8	7	6	5	4	3	2	1	0

A Tabela Cayley deste grupo é mostrada na Tabela 3, onde os elementos de 1 a 4 representam as rotações de $72°, 144°, 216°$ e $288°$, os elementos de 5 a 9 representam um espelhamento no pentágono em relação às retas r_5, r_6, r_7, r_8, r_9 e o 0 representa a identidade, não causando nenhum efeito no pentágono.

Por exemplo, o número 1 representa a rotação de $72°$, como é mostrado na Figura 2.

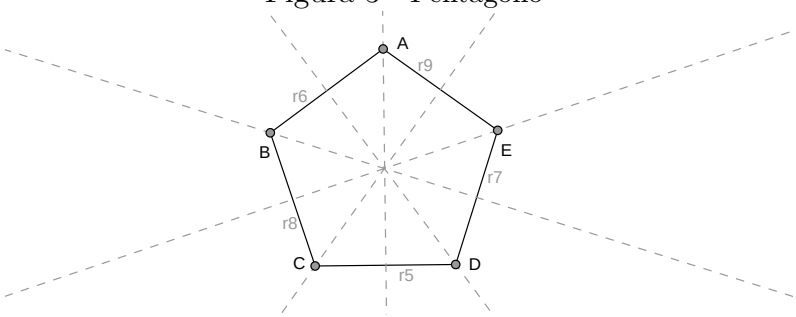
Figura 2 - Pentágono



O número 5 representa o espelhamento em relação ao eixo r_5, como é mostrado na Figura 3.

Seja um identificador de n dígitos. Para que a sequência seja reconhecida como

Figura 3 - Pentágono



verdadeira a seguinte igualdade deve ser satisfeita:

$$\sigma^{n-1}(a_1) * \sigma^{n-2}(a_2) * \cdots * \sigma(a_{n-1}) * a_n = 0, \tag{10}$$

onde a_n é o dígito verificador e σ uma permutação antissimétrica, ou seja, satisfaz $a*\sigma(b) \neq b * \sigma(a)$, para $a \neq b$, como visto em [15], [27] , [28] e [25].

Em nossos exemplos utilizaremos a permutação $\sigma : (0\ 4\ 1\ 3\ 2\ 6\ 7\ 8\ 9\ 5)$. E $*$ é a operação não comutativa representada pela tabela anterior.

A seguir verificaremos a autenticação da sequência 3170092, utilizando a permutação mostrada anteriormente. Temos:

$$\begin{aligned} \sigma^6(3) * \sigma^5(1) * \sigma^4(7) * \sigma^3(0) * \sigma^2(0) * \sigma(9) * 2 &= 3 * 4 * 6 * 0 * 0 * 5 * 2 \\ &= 2 * 1 * 2 \\ &= 3 * 2 = 0 \end{aligned}$$

Logo, a sequência é reconhecida como verdadeira.

O método desenvolvido por Verhoeff garante a detecção de 100% de erros individuas e de transposição adjacente.

1.4 Esquemas baseados em Quadrados Latinos

Existem ainda os esquemas de dígitos verificadores baseados em quadrados latinos [4], [2], [3] e [11]. Um quadrado latino é uma matriz $n \times n$ onde n elementos distintos ocorrem uma única vez em cada linha e em cada coluna. Podemos considerar estes

elementos como sendo os números de 0 a 9 e, neste caso, as linhas e colunas da matriz são permutações de 0 a 9. Como por exemplo, apresentamos um quadrado latino na Figura 4.

Figura 4 - Quadrado latino de ordem 10

$$\begin{bmatrix} 0 & 4 & 8 & 2 & 3 & 9 & 6 & 7 & 1 & 5 \\ 3 & 6 & 2 & 8 & 7 & 1 & 9 & 5 & 0 & 4 \\ 8 & 9 & 3 & 1 & 0 & 6 & 4 & 2 & 5 & 7 \\ 1 & 7 & 6 & 5 & 4 & 8 & 0 & 3 & 2 & 9 \\ 2 & 1 & 9 & 0 & 6 & 7 & 5 & 8 & 4 & 3 \\ 5 & 2 & 7 & 4 & 9 & 3 & 1 & 0 & 8 & 6 \\ 4 & 3 & 0 & 6 & 1 & 5 & 2 & 9 & 7 & 8 \\ 9 & 8 & 5 & 7 & 2 & 0 & 3 & 4 & 6 & 1 \\ 7 & 0 & 1 & 9 & 5 & 4 & 8 & 6 & 3 & 2 \\ 6 & 5 & 4 & 3 & 8 & 2 & 7 & 1 & 9 & 0 \end{bmatrix}$$

Um quadrado latino pode representar a tabela de multiplicação de um quasigrupo. Um quasigrupo $(D, *)$ é definido através de uma operação binária $*$ sobre o conjunto D tal que dados $a, b \in D$, as equações $a * x = b$ e $y * a = b$ têm, cada uma, exatamente uma solução. Por esta razão quadrados latinos podem ser interpretados como quasigrupos. Por exemplo, a seguinte operação:

$$a * b = (ha + kb + l) \bmod n, \tag{11}$$

onde h, k, l são inteiros fixos, com h e k relativamente primos a n, define um quasigrupo em $Z_n = \{0, 1, 2, \cdots, n-1\}$. O cálculo do dígito verificador é dado por $a_n = (\cdots(a_1 * a_2) * \cdots) * a_{n-1}$.

Mais especificamente, escolhamos por exemplo $n = 10, h = 3, k = 9, l = 1$, e portanto a operação $*$ é dado por

$$a * b = (3a + 9b + 1) \bmod 10. \tag{12}$$

Para um identificador com valor 1482, o dígito verificador seria:

$$a_5 = (((a_1 * a_2) * a_3) * a_4) = (((1 * 4) * 8) * 2) = ((0 * 8) * 2) = (3 * 2) = 8. \tag{13}$$

H. Michael Damm [7], [8] demonstrou a existência de esquemas para identificadores

decimais baseados em quadrados latinos que detectam erros individuais e de transposição adjacente com perfeição. Desenvolveu este esquema em sua tese de doutorado [9].

O cálculo do dígito verificador é dado por:

$$a_n = (\cdots(((0 * a_1) * a_2) * a_3) * \cdots) * a_{n-1}. \tag{14}$$

Damm encontrou algumas formas de fazer esta operação $*$. Uma delas [10] está representada na Tabela 4.

Tabela 4 - Quadrado Latino Damm.

$*$	0	1	2	3	4	5	6	7	8	9
0	0	3	1	7	5	9	8	6	4	2
1	7	0	9	2	1	5	4	8	6	3
2	4	2	0	6	8	7	1	3	5	9
3	1	7	5	0	9	8	3	4	2	6
4	6	1	2	3	0	4	5	9	7	8
5	3	6	7	4	2	0	9	5	8	1
6	5	8	6	9	7	2	0	1	3	4
7	8	9	4	5	3	6	2	0	1	7
8	9	4	3	8	6	1	7	2	0	5
9	2	5	8	1	4	3	6	7	9	0

Em [11] é descrito um código para dados alfanuméricos, utilizando quadrados latinos. Esse esquema é capaz de detectar todos os erros individuais e de transposição. Descreveremos no capítulo 4 uma boa alternativa para o tratamento de identificadores alfanuméricos usando aritmética modular com módulo 37, mas o código de Ecker e Poch teria a vantagem de utilizar apenas 36 caracteres do conjunto $Z_{36} = \{0, 1, \cdots, 35\}$ economizando assim um símbolo.

O cálculo do dígito verificador é dado por:

$$a_n = (\cdots(((a_1 * a_2) * a_3) * a_4) * \cdots) * a_{n-1}. \tag{15}$$

Os resultados das operações $a * b$ para todo $a, b \in Z_{36}$ é mostrado no quadrado latino da Tabela 5.

Tabela 5 - Quadrado Latino para o esquema de DV alfanumérico

*	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	8	0	1	2	3	4	5	6	7	17	9	10	11	12	13	14	15	16	26	18	19	20	21	22	23	24	25	35	27	28	29	30	31	32	33	34
2	7	8	0	1	2	3	4	5	6	16	17	9	10	11	12	13	14	15	25	26	18	19	20	21	22	23	24	34	35	27	28	29	30	31	32	33
3	6	7	8	0	1	2	3	4	5	15	16	17	9	10	11	12	13	14	24	25	26	18	19	20	21	22	23	33	34	35	27	28	29	30	31	32
4	5	6	7	8	0	1	2	3	4	14	15	16	17	9	10	11	12	13	23	24	25	26	18	19	20	21	22	32	33	34	35	27	28	29	30	31
5	4	5	6	7	8	0	1	2	3	13	14	15	16	17	9	10	11	12	22	23	24	25	26	18	19	20	21	31	32	33	34	35	27	28	29	30
6	3	4	5	6	7	8	0	1	2	12	13	14	15	16	17	9	10	11	21	22	23	24	25	26	18	19	20	30	31	32	33	34	35	27	28	29
7	2	3	4	5	6	7	8	0	1	11	12	13	14	15	16	17	9	10	20	21	22	23	24	25	26	18	19	29	30	31	32	33	34	35	27	28
8	1	2	3	4	5	6	7	8	0	10	11	12	13	14	15	16	17	9	19	20	21	22	23	24	25	26	18	28	29	30	31	32	33	34	35	27
9	27	28	29	30	31	32	33	34	35	18	19	20	21	22	23	24	25	26	9	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7	8
10	35	27	28	29	30	31	32	33	34	26	18	19	20	21	22	23	24	25	17	9	10	11	12	13	14	15	16	8	0	1	2	3	4	5	6	7
11	34	35	27	28	29	30	31	32	33	25	26	18	19	20	21	22	23	24	16	17	9	10	11	12	13	14	15	7	8	0	1	2	3	4	5	6
12	33	34	35	27	28	29	30	31	32	24	25	26	18	19	20	21	22	23	15	16	17	9	10	11	12	13	14	6	7	8	0	1	2	3	4	5
13	32	33	34	35	27	28	29	30	31	23	24	25	26	18	19	20	21	22	14	15	16	17	9	10	11	12	13	5	6	7	8	0	1	2	3	4
14	31	32	33	34	35	27	28	29	30	22	23	24	25	26	18	19	20	21	13	14	15	16	17	9	10	11	12	4	5	6	7	8	0	1	2	3
15	30	31	32	33	34	35	27	28	29	21	22	23	24	25	26	18	19	20	12	13	14	15	16	17	9	10	11	3	4	5	6	7	8	0	1	2
16	29	30	31	32	33	34	35	27	28	20	21	22	23	24	25	26	18	19	11	12	13	14	15	16	17	9	10	2	3	4	5	6	7	8	0	1
17	28	29	30	31	32	33	34	35	27	19	20	21	22	23	24	25	26	18	10	11	12	13	14	15	16	17	9	1	2	3	4	5	6	7	8	0
18	9	10	11	12	13	14	15	16	17	0	1	2	3	4	5	6	7	8	27	28	29	30	31	32	33	34	35	18	19	20	21	22	23	24	25	26
19	17	9	10	11	12	13	14	15	16	8	0	1	2	3	4	5	6	7	35	27	28	29	30	31	32	33	34	26	18	19	20	21	22	23	24	25
20	16	17	9	10	11	12	13	14	15	7	8	0	1	2	3	4	5	6	34	35	27	28	29	30	31	32	33	25	26	18	19	20	21	22	23	24
21	15	16	17	9	10	11	12	13	14	6	7	8	0	1	2	3	4	5	33	34	35	27	28	29	30	31	32	24	25	26	18	19	20	21	22	23
22	14	15	16	17	9	10	11	12	13	5	6	7	8	0	1	2	3	4	32	33	34	35	27	28	29	30	31	23	24	25	26	18	19	20	21	22
23	13	14	15	16	17	9	10	11	12	4	5	6	7	8	0	1	2	3	31	32	33	34	35	27	28	29	30	22	23	24	25	26	18	19	20	21
24	12	13	14	15	16	17	9	10	11	3	4	5	6	7	8	0	1	2	30	31	32	33	34	35	27	28	29	21	22	23	24	25	26	18	19	20
25	11	12	13	14	15	16	17	9	10	2	3	4	5	6	7	8	0	1	29	30	31	32	33	34	35	27	28	20	21	22	23	24	25	26	18	19
26	10	11	12	13	14	15	16	17	9	1	2	3	4	5	6	7	8	0	28	29	30	31	32	33	34	35	27	19	20	21	22	23	24	25	26	18
27	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
28	26	18	19	20	21	22	23	24	25	35	27	28	29	30	31	32	33	34	8	0	1	2	3	4	5	6	7	17	9	10	11	12	13	14	15	16
29	25	26	18	19	20	21	22	23	24	34	35	27	28	29	30	31	32	33	7	8	0	1	2	3	4	5	6	16	17	9	10	11	12	13	14	15
30	24	25	26	18	19	20	21	22	23	33	34	35	27	28	29	30	31	32	6	7	8	0	1	2	3	4	5	15	16	17	9	10	11	12	13	14
31	23	24	25	26	18	19	20	21	22	32	33	34	35	27	28	29	30	31	5	6	7	8	0	1	2	3	4	14	15	16	17	9	10	11	12	13
32	22	23	24	25	26	18	19	20	21	31	32	33	34	35	27	28	29	30	4	5	6	7	8	0	1	2	3	13	14	15	16	17	9	10	11	12
33	21	22	23	24	25	26	18	19	20	30	31	32	33	34	35	27	28	29	3	4	5	6	7	8	0	1	2	12	13	14	15	16	17	9	10	11
34	20	21	22	23	24	25	26	18	19	29	30	31	32	33	34	35	27	28	2	3	4	5	6	7	8	0	1	11	12	13	14	15	16	17	9	10
35	19	20	21	22	23	24	25	26	18	28	29	30	31	32	33	34	35	27	1	2	3	4	5	6	7	8	0	10	11	12	13	14	15	16	17	9

2 ASPECTOS MATEMÁTICOS DOS ESQUEMAS DE DÍGITOS VERIFICADORES

Neste capítulo, vamos discutir a capacidade de detecção de erros dos métodos descritos, do ponto de vista teórico.

2.1 Detecção de erros nos esquemas baseados em aritmética modular

Os esquemas utilizados no Brasil têm como base teórica a aritmética modular.

Denominaremos sistema de dígito verificador de módulo n ao esquema de atribuição de um dígito verificador a um identificador composto por m dígitos $a_1a_2a_3\cdots a_m$ tal que seja atribuído um peso p_i para cada posição i do identificador e o dígito verificador a_{m+1} é calculado para satisfazer a equação: $\sum(p_ia_i+a_{m+1}) \equiv 0 \bmod n, 1 \leq i \leq m, 0 \leq a_{m+1} < n$.

Observe que ainda não estamos particularizando essa definição para o caso mais comum, onde são usados dígitos decimais.

Uma propriedade da aritmética modular que será utilizada adiante pode ser expressa pelo seguinte teorema:

Teorema 2.1 *Sejam a, b, p, q inteiros, com $a, p, q < b$. Então, para todos os pares p, q, com $0 \leq q < p < b$, tem-se $ap \not\equiv aq \bmod b$ se, e somente se, $mdc(a, b) = 1$.*

Demonstração: $\Rightarrow$ Seja $ap \not\equiv aq \bmod b$, para todos os pares p, q, com $0 \leq q < p < b$. Suponhamos, por absurdo que $mdc(a, b) = d \neq 1$. Podemos escrever $a = du$ e $b = dv$, com $d, u, v < b$. Por hipótese, sabemos que $a(p - q) \not\equiv 0 \bmod b$, ou seja, $du(p - q) \not\equiv 0 \bmod b$. Podemos tomar $q = 0$ e $p = v$, e a equação então torna-se: $duv \not\equiv 0 \bmod b$. Absurdo, pois $dv = b \equiv 0 \bmod b$. Logo, $mdc(a, b) = 1$.

$\Leftarrow$ Seja $mdc(a, b) = 1$. Suponhamos, por absurdo, que $ap \bmod b = aq \bmod b$. Então existem inteiros u e v tal que $ap = ub + x$ e $aq = vb + x$, implicando $a(p - q) = (u - v)b$. Como $q < p$, isso implica que $p - q$ seja múltiplo de b. Absurdo, pois $p - q$ é menor que b e não é zero. Logo, $ap \not\equiv aq \bmod b$. $\square$

Temos dois corolários úteis deste teorema.

Corolário 2.1 *Se tomarmos os inteiros a, b, com $0 < a < b$ e $mdc(a, b) = 1$ e fizermos todos os produtos $ac \bmod b$, com $0 \leq c < b$, esses b resultados constituem uma permutação dos números de 0 a $b - 1$.*

A demonstração é imediata, pois todos os resultados são diferentes entre si e menores que b.

Corolário 2.2 *Seja um dígito verificador dado por $d \equiv \sum_{i=1}^{m} p_i a_i \bmod n$, onde $0 < p_i, a_i < n$, para cada peso p_i fixo. A probabilidade de d ser cada número entre 0 e $n - 1$ é $\frac{1}{n}$.*

Demonstração:

Dados pesos p_1, p_2 fixos não-nulos, e a seguinte equação para dígito verificador $d \equiv p_1 a + p_2 b \bmod n$, para quaisquer $0 \leq a, b < n$. Sabemos, pelo corolário 1, que os produtos $p_1 a$, para todo $0 \leq a < n$, geram uma permutação de 0 a $n - 1$, idem para $p_2 b$. Fixando $k = p_2 b$, para algum b, temos que $k + p_1 a$ gera o conjunto $\{k, k+1, k+2, \cdots, k+n-1\}$ com n números consecutivos distintos. Logo, o conjunto forma uma permutação em módulo n. Como para cada b fixo, $p_1 a + p_2 b$ forma uma permutação, temos que a probabilidade de se obter cada número entre 0 e $n - 1$ como resultado é a mesma. A ideia da demonstração pode ser generalizada para o cálculo do dígito verificador dado por $d \equiv p_1 a_1 + \cdots + p_m a_m \bmod n$, para todo $m > 2$. □

No nosso caso, devido ao fato de a_i assumir valores entre 0 e 9, os corolários são aplicados apenas para módulo 10.

Vamos agora examinar as propriedades das funções σ_i ou dos pesos p_i com relação à detecção dos diversos tipos de erros considerados neste trabalho.

Teorema 2.2 *Um sistema modular de dígitos verificadores é capaz de detectar:*

1. *Erros individualizados, a troca de a com b se, e somente se, a imagem de σ_i é uma permutação de 0 até $n - 1$ ou p_i é relativamente primo a n, para todo i.*

2. *Erros de transposição de dígitos a_i e a_j entre as posições i e j se, e somente se, $(a_i - a_j)(p_i - p_j) \not\equiv 0 \bmod n$ ou $(\sigma_i - \sigma_j)(a_i) \not\equiv (\sigma_i - \sigma_j)(a_j) \bmod n$.*

3. *Erros gêmeos entre a e b, nas posições i e j se, e somente se, $(a - b)(p_i + p_j) \not\equiv 0 \bmod n$ ou $(\sigma_i + \sigma_j)(a) \not\equiv (\sigma_i + \sigma_j)(b) \bmod n$.*

4. *Erros fonéticos nas posições i e $i+1$ se, e somente se, $p_i \not\equiv (p_i - p_j)a \bmod n$ ou $(\sigma_i - \sigma_{i+1})(a) \not\equiv \sigma_{i+1}(0) - \sigma_i(1) \bmod n$, para $a \neq 0, 1$.*

Demonstração: Para um sistema de dígitos verificadores detectar:

1. a troca de a para b, com $a \neq b$, é suficiente que a imagem de σ_i seja uma permutação dos dígitos de 0 a $n-1$, desde que temos n dígitos e é necessário que $\sigma_i(a) \not\equiv \sigma_i(b) \bmod n$. A propriedade em relação aos pesos já foi demonstrada.

2. a transposição de dígitos entre as posições i e j, além de $a_i \neq a_j$, devemos ter

$$\sigma_i(a_i) + \sigma_j(a_j) \not\equiv \sigma_i(a_j) + \sigma_j(a_i) \bmod n. \tag{16}$$

Pela definição de diferença de funções, podemos escrever,

$$(\sigma_i - \sigma_j)(a_i) \not\equiv (\sigma_i - \sigma_j)(a_j) \bmod n. \tag{17}$$

Considerando os pesos, a condição pode ser escrita como:

$$p_i a_i + p_j a_j \not\equiv p_i a_j + p_j a_i \bmod n. \tag{18}$$

ou de maneira equivalente, $(a_i - a_j)(p_i - p_j) \not\equiv 0 \bmod n$.

3. erros gêmeos de a para b nas posições i e j, devemos ter $a \neq b$ e

$$\sigma_i(a) + \sigma_j(a) \not\equiv \sigma_i(b) + \sigma_j(b) \bmod n. \tag{19}$$

Pela definição de soma de funções, podemos escrever,

$$(\sigma_i + \sigma_j)(a) \not\equiv (\sigma_i + \sigma_j)(b) \bmod n. \tag{20}$$

Considerando os pesos, a condição pode ser reescrita como:

$$p_i a + p_j a \not\equiv p_i b + p_j b \bmod n. \tag{21}$$

ou de maneira equivalente, $(a - b)(p_i + p_j) \not\equiv 0 \bmod n$.

4. erros fonéticos nas posições i e $i+1$, devemos ter $a \neq 0, 1$ e

$$\sigma_i(1) + \sigma_j(a) \not\equiv \sigma_i(a) + \sigma_j(0) \bmod n. \tag{22}$$

Pela definição de soma de funções, podemos escrever,

$$(\sigma_j - \sigma_i)(a) \not\equiv \sigma_j(0) - \sigma_i(1) \bmod n. \tag{23}$$

Considerando os pesos, a condição pode ser reescrita como:

$$p_i 1 + p_j a \not\equiv p_i a + p_j 0 \bmod n. \tag{24}$$

ou de maneira equivalente, $p_i \not\equiv (p_i - p_j)a \bmod n$. □

2.1.1 Detecção de erros com Módulo 10

Uma primeira consideração é que o esquema módulo 10 da IBM é o único esquema em que não é feita apenas uma multiplicação em cada dígito. Nesse esquema, além da multiplicação, é feita a operação "noves fora". Para os outros sistemas, basta analisar os pesos utilizados. Aplicando, então, as condições obtidas no teorema 2.2 aos esquemas baseados em módulo 10, temos as possibilidades descritas a seguir.

2.1.1.1 Detecção de erros individualizados

Para a detecção de erros individualizados, cada parcela deve induzir uma permutação dos dígitos de 0 a 9. Isso só é possível quando cada peso é relativamente primo a 10. Ou seja, os pesos têm que estar no conjunto $\{1, 3, 7, 9\}$. Nos casos estudados onde os identificadores contêm apenas dígitos decimais isso ocorre, significando que todos os erros individualizados são detectados.

No esquema módulo 10 IBM, o resultado da multiplicação com a operação "noves fora" também é uma permutação, pois gera as seguintes transformações: $0 \to 0, 1 \to 2, 2 \to 4, 3 \to 6, 4 \to 8, 5 \to 10 \to 1, 6 \to 12 \to 3, 7 \to 14 \to 5, 8 \to 16 \to 7, 9 \to 18 \to 9$. Portanto, todos os erros individualizados também são detectados.

Já quando se usa caracteres alfanuméricos no identificador, como é o caso do passaporte, não é mais possível ter-se uma permutação, pois há mais elementos que a base modular e mais de um elemento de cada parcela vai ser mapeado para os valores de 0 a 9.

No caso do passaporte, as letras entre A e Z são mapeadas respectivamente para os números entre 10 e 35 e o caractere $<$ é mapeado para 0. Então esse alfabeto com 37 carac-

teres é particionado em 10 subconjuntos, onde a contribuição dos elementos de cada conjunto para o cálculo do dígito verificador é a mesma, pois seu mapeamento tem o mesmo resto da divisão por 10. O primeiro subconjunto tem 5 elementos: $\{0, A, K, U, <\}$. Há cinco subconjuntos com 4 elementos: $\{1, B, L, V\}, \{2, C, M, W\}, \{3, D, N, X\}\{4, E, O, Y\}$ e $\{5, F, P, Z\}$ e quatro subconjuntos com 3 elementos: $\{6, G, Q\}, \{7, H, R\}, \{8, I, S\}$ e $\{9, S, T\}$. A troca de um caractere por outro, em cada subconjunto, não é detectada pelo esquema. Como cada um dos 37 caracteres pode ser trocado para 36 outros, o percentual de trocas não detectadas é dado por $(1 \cdot 5 \cdot 4 + 5 \cdot 4 \cdot 3 + 4 \cdot 3 \cdot 2)/(37 \cdot 36)$ que corresponde a 7,81% dos casos, uma alta taxa de erros não detectados, portanto.

2.1.1.2 Detecção de transposições de dígitos adjacentes

Para a detecção de transposição de dígitos adjacentes, devemos ter a condição 2 do teorema 2.2 satisfeita, ou seja, para toda posição i do identificador, devemos ter:

$(a_i - a_{i+1})(p_i - p_{i+1}) \not\equiv 0 \bmod 10.$

Nos esquemas módulo 10 essa desigualdade não pode ser garantida. Vimos que os pesos têm que estar no conjunto $\{1,\ 3,\ 7,\ 9\}$. Como a diferença dos pesos sempre será par, a desigualdade nunca ocorrerá se a diferença entre os dígitos trocados for 5 e esse tipo de erro, portanto, não será detectado. Quando os dígitos do identificador são decimais, temos 90 casos possíveis de troca em cada posição. Desses casos, 10 não são detectados, correspondendo às diferenças iguais a 5. Portanto, o percentual de erros não detectados é 10/90, ou seja, 11,11%. No caso do passaporte, com valores de 0 a 35, as trocas não detectadas ocorrem quando a diferença entre os valores é múltiplo de 5. Aqui formam-se 5 classes cuja troca não é percebida. A primeira classe, com 9 elementos é $\{0, 5, A, F, \cdots, Z, <\}$. As demais classes contêm 7 elementos sendo $\{1, 6, B, \cdots, V\}$, $\{2, 7, C, \cdots, W\}$, $\{3, 8, D, \cdots, X\}$ e $\{4, 9, E, \cdots, Y\}$. O percentual de trocas não detectadas é dado por $(9 \cdot 8 + 4 \cdot 7 \cdot 6)/(37 \cdot 36)$, correspondendo a 18,02% dos erros. No esquema da IBM temos poucos erros não detectados. Uma inspeção cuidadosa mostra que esses casos ocorrem quando há uma troca de 0 por 9, ou vice-versa, correspondendo

ao percentual 2/90, ou seja, 2,2%.

2.1.1.3 Detecção de transposições de dígitos alternados

Para a detecção de transposição de dígitos alternados, devemos ter também a condição 2 do teorema 2.2 satisfeita, ou seja, para todo i, devemos ter:

$(a_i - a_{i+2})(p_i - p_{i+2}) \not\equiv 0 \bmod 10.$

Nos esquemas que utilizam apenas 2 pesos, como são os casos do módulo 10 IBM, Registro Geral do Rio de Janeiro, cartões de crédito e código de barras de produtos, essa desigualdade nunca é satisfeita, e nenhum erro é detectado.

Nos esquemas com 3 pesos a capacidade de detecção é análoga ao caso anterior, ou seja, deixam de ser detectados 11,11% dos casos de erro, quando o identificador é numérico e, no caso do passaporte, 18,02% dos erros.

2.1.1.4 Detecção de erros gêmeos adjacentes

Para a detecção de erros gêmeos adjacentes, devemos ter a condição 3 do teorema 2.2 satisfeita, ou seja, para todo i, devemos ter:

$(a - b)(p_i + p_{i+1}) \not\equiv 0 \bmod 10.$

Aqui a desigualdade não será garantida, pois a soma dos pesos, que estão no conjunto $\{1, 3, 7, 9\}$, sempre é par. Sendo assim, quando a soma dos pesos adjacentes é diferente de 10, não são detectados os erros onde a diferença entre os dígitos trocados é 5, o que acontece, no caso de dígitos numéricos, com uma probabilidade de $10/90 =$ 11,11%. No caso de a soma dos pesos adjacentes ser 10, não são detectadas 100% dessas trocas. Consideremos um identificador com 10 dígitos e pesos $\langle 7, 3, 1, 7, 3, 1, 7, 3, 1 \rangle$. A troca em três das nove colunas não detecta 100% dos erros. Portanto, o percentual

de erros não detectados no caso de identificadores numéricos é dado por $(3 \cdot 100 + 5 \cdot 11,11)/8 = 44,44\%$. No caso do passaporte, esse percentual é um pouco maior e é dado por $(3 \cdot 100 + 5 \cdot 18,02)/8 = 48,76\%$. Em ambos os casos, um percentual muito elevado de não detecção. No caso do módulo 10 IBM, o erro não é detectado quando se troca 2 por 5, 3 por 6 ou 4 por 7, pois as somas resultantes são iguais. Estes casos correspondem a 6/90 do total, ou seja, 6,67% dos casos.

2.1.1.5 Detecção de erros gêmeos alternados

Para a detecção de troca de gêmeos alternados, devemos ter a condição 3 do teorema 2.2 satisfeita, ou seja, para todo i, devemos ter:

$(a - b)(p_i + p_{i+2}) \not\equiv 0 \bmod 10.$

Para os esquemas com 2 pesos (e também para o módulo 10 IBM), a análise é a mesma do caso de erros gêmeos adjacentes. Para os esquemas com 3 pesos o cálculo é análogo ao de erros gêmeos adjacentes só que considerando uma coluna a menos. No identificador considerado anteriormente, nenhuma troca envolvendo os pares de colunas (2,4) e (5,7) são detectados e, nas outras 5 colunas, 11,11% dos casos não são detectados, o que gera um valor proporcional de $(2 \cdot 1 + 5 \cdot 11,11)/7 = 36,50\%$. No caso de 3 pesos com dígitos alfanuméricos o percentual de erros não detectados é dado por $(2 \cdot 100 + 5 \cdot 18,02)/7 = 41,57\%$.

Vimos que nenhum dos casos com uso de módulo 10 consegue detectar todos os erros de transposição e erros gêmeos. Surge, naturalmente, a seguinte questão: será que existe algum esquema baseado na aritmética modular, com base 10 que permita a detecção simultânea de todos os erros individualizados e de transposição de dígitos adjacentes? A resposta para essa questão é negativa e pode ser expressa pelo seguinte teorema.

Teorema 2.3 *Suponha um esquema de dígito verificador módulo n, onde n é par, que detecte todos os erros individualizados. Então existe pelo menos uma troca de dígitos adjacentes que este esquema não detecta.*

Demonstração: Como o sistema deve detectar todos os erros individuais, $\sigma_i(a)$ é uma permutação, para todo i. Para detectar todos os erros de transposições adjacentes, deve-se ter $\sigma_i(a_i) + \sigma_{i+1}(a_{i+1}) \not\equiv \sigma_i(a_{i+1}) + \sigma_{i+1}(a_i) \bmod n$, para todo $a_i \neq a_{i+1}$. Portanto, a diferença $\sigma(a) = \sigma_i(a_i) - \sigma_{i+1}(a_i)$ também deve ser uma permutação. Contudo, a soma dos elementos em $\sigma_i(a)$ em $\mathbb{Z}_n$ é dada por:

$$\sum_{i=0}^{n} i = \frac{n}{2} + (1+n-1) + (2+n-2) + (3+n-3) + \ldots + \left(\frac{n}{2} - 1 + \frac{n}{2} + 1\right) \equiv \frac{n}{2} \bmod n. \quad (25)$$

Isto implica em:

$$\frac{n}{2} \equiv \sum \sigma(a) \equiv \sum (\sigma_i(a_i) - \sigma_{i+1}(a_i)) = \sum \sigma_i(a_i) - \sum \sigma_{i+1}(a_i) \equiv \frac{n}{2} - \frac{n}{2} = 0 \bmod n. \quad (26)$$

De acordo com a última igualdade, $\frac{n}{2} \equiv 0 \bmod n$, isto é uma contradição. □

De maneira análoga, é feita a prova para o caso de erros gêmeos.

2.1.2 Detecção de erros com Módulo 11 e um dígito verificador

Vamos agora analisar os esquemas baseados em módulo 11, inicialmente com um dígito verificador.

2.1.2.1 Detecção de erros individualizados

Nos esquemas módulo 11, não temos problemas com os pesos em relação à detecção de erros individualizados já que, como 11 é primo, todos os pesos de 1 a 10 induzem uma permutação, considerando as operações modulares.

Quando se adota o módulo 11 completo, nenhum problema adicional existe, já que o resto da divisão igual a 10 é transformado em X e todos os erros individualizados são detectados. Entretanto, quando se usa o módulo 11 restrito, onde o resultado 10 é transformado em 0, temos uma fonte de erros. Os erros individualizados que deixam de ser detectados são aqueles onde o valor inicial do dígito verificador era 0 e, com a troca de um

dígito, passou para 10, ou vice-versa. Cada resultado 0 ou 10 ocorre em aproximadamente 1/11 dos identificadores, pois a distribuição de restos é praticamente homogênea. Quando o dígito verificador é 0 ou 10, dada troca de um dígito tem probabilidade de 1/10 de não ser percebida. Logo, os casos de erro correspondem ao percentual $2/(11 \cdot 10)$ de erros, ou seja, 1,82% dos erros.

2.1.2.2 Detecção de transposições de dígitos adjacentes e alternados

Para a detecção de transposição de dígitos adjacentes e de dígitos alternados, devemos ter a condição 2 do teorema 2.2 satisfeita de duas formas:

$$\begin{aligned} (a_i - a_{i+1})(p_i - p_{i+1}) &\not\equiv 0 \bmod 11 \quad \text{e} \\ (a_i - a_{i+2})(p_i - p_{i+2}) &\not\equiv 0 \bmod 11. \end{aligned} \tag{27}$$

Essas desigualdades são sempre satisfeitas por ser 11 um primo. No módulo 11 completo não há considerações adicionais e todos os erros são detectados. Entretanto, quando se usa o módulo 11 restrito, existe o mesmo problema que da detecção de erros individualizados se o resultado inicial for 10 ou 0 e, com a troca o resultado passar para 0 ou 10, respectivamente, o erro não é detectado. Essas situações são em mesmo número do caso de erros individualizados. Portanto 1,82% dos erros não são detectados.

2.1.2.3 Detecção de erros gêmeos adjacentes

Para a detecção de erros gêmeos adjacentes, correspondendo a troca do dígito a para b, além de termos $a \neq b$, devemos ter a condição 3 do teorema 2.2 satisfeita, ou seja,

$(a - b)(p_i + p_{i+1}) \not\equiv 0 \bmod 11.$

Aqui a garantia da desigualdade só depende de a soma dos pesos não ser 11. Entretanto, usualmente usam-se os pares de pesos 5 e 6 vizinhos e algumas vezes o par 9

e 2, nesses casos a desigualdade falha e os erros não são detectados.

Quando se usa o módulo 11 restrito, além desses casos de erro, há também os casos análogos aos itens já estudados, o dígito verificador deveria ser 10 e o resultado indicou 0 ou vice-versa. Isso corresponde ao mesmo percentual já visto, 1,82% dos casos. Sendo assim, para um identificador com m dígitos e uma quantidade q de pesos vizinhos 5 e 6 ou 2 e 9, o percentual de erros não detectados é dado por $\dfrac{q \cdot 100 + (m-2-q) \cdot 1,82}{m-2}$.

Para um identificador de 9 dígitos com pesos de 1 a 8, por exemplo, a probabilidade de não detecção de erros é de $(100 + 6 \cdot 1,82)/7 = 15,85\%$.

2.1.2.4 Detecção de erros gêmeos alternados

Para a detecção de erros gêmeos alternados, correspondendo a troca do dígito a para b, além de termos $a \neq b$, devemos ter a condição 3 do teorema 2.2 satisfeita, ou seja,

$(a-b)(p_i + p_{i+2}) \not\equiv 0 \bmod 11$.

Considerando um identificador numérico com 9 dígitos e pesos 1 a 8, o módulo 11 completo detecta todos os casos de erro. Ao se utilizar o módulo 11 restrito, o caso é análoga ao de erros gêmeos adjacentes, com a diferença que, em um identificador com m dígitos, sendo um dígito verificador, temos $m-3$ colunas alternadas. Considerando uma quantidade q de pesos alternados 5 e 6 ou 2 e 9, temos que o percentual de erros não detectados é $\dfrac{q \cdot 100 + (m-3-q) \cdot 1,82}{m-3}$.

Por exemplo, um identificador com 9 dígitos e pesos $\langle 4,5,6,7,8,9,1,2 \rangle$, não detecta $(100 + 5 \cdot 1,82)/6 = 18,18\%$.

2.1.3 Módulo 11 com dois dígitos verificadores

Os exemplos analisados neste trabalho usando dois dígitos verificadores e módulo 11 são todos ligeiramente distintos entre si. Os cálculos da detecção de erros para CNPJ e CPF e os resultados para CNH e título eleitoral estão detalhados no anexo A.

2.2 Detecção de erros no esquema baseado na Teoria dos Grupos

A única utilização conhecida do esquema de dígito verificador baseado na Teoria dos Grupos desenvolvido por Verhoeff foi na identificação de notas do marco alemão, antes de a Alemanha mudar sua moeda para o euro. Ou seja, esse uso foi descontinuado. Mesmo assim vamos analisar o poder de detecção desse código.

Detecção de erros individualizados

Se a troca de um dígito a_i por um dígito b_i não fosse detectada, teríamos a seguinte igualdade satisfeita:

$$\begin{aligned}\sigma^{n-1}(a_1) * \sigma^{n-2}(a_2) * \cdots * \sigma^{n-i}(a_i) * \cdots * \sigma(a_{n-1}) * a_n = \\ \sigma^{n-1}(a_1) * \sigma^{n-2}(a_2) * \cdots \sigma^{n-i}(b_i) * \cdots * \sigma(a_{n-1}) * a_n \\ \Rightarrow \sigma^{n-i}(a_i) = \sigma^{n-i}(b_i).\end{aligned}$$

Desde que σ é uma permutação, deve-se ter $a_i = b_i$. Portanto todos os erros individualizados são detectados.

Detecção de erros de transposição adjacente

Pode-se verificar que $a * \sigma(b) \neq b * \sigma(a)$, para $a \neq b$, notando-se que dois elementos em posições simétricas da Tabela 6 são sempre distintos.

Observe que $\sigma^i(a) * \sigma^{i+1}(b) \neq \sigma^i(b) * \sigma^{i+1}(a)$ pode ser escrito como $\sigma^i(a) * \sigma^i(\sigma(b)) \neq \sigma^i(b) * \sigma^i(\sigma(a))$. Como $\sigma^i(a) \neq \sigma^i(b)$, temos a generalização da desigualdade. E portanto, todos os erros de transposição adjacente são detectados.

O esquema não detecta nenhum dos demais tipos de erro, mas é o que apresenta a melhor média ponderada de detecção para os cinco tipos de erros que estamos considerando.

Tabela 6 - Operação $a * \sigma(b)$

$\backslash b$	0	1	2	3	4	5	6	7	8	9
$a\backslash\sigma(b)$	0	1	2	3	4	5	6	7	8	9
0	-	4	3	2	1	6	7	8	9	5
1	1	-	4	3	2	7	8	9	5	6
2	2	1	-	4	3	8	9	5	6	7
3	3	2	1	-	4	9	5	6	7	8
4	4	3	2	1	-	5	6	7	8	9
5	5	6	7	8	9	-	4	2	1	0
6	6	7	8	9	5	0	-	3	2	1
7	7	8	9	5	6	1	0	-	3	2
8	8	9	5	6	7	2	1	0	-	3
9	0	5	6	7	8	3	2	1	0	-

2.3 Detecção de erros em esquemas baseados em Quadrados Latinos

Não foram encontradas aplicações práticas para os códigos baseados em quadrados latinos, mas é útil analisar o poder de detecção de erros dos mesmos.

Todos os erros individuais são detectados por códigos baseados em quadrados latinos. Os erros de transposições dependem de que as condições descritas na Tabela 7 sejam satisfeitas:

Tabela 7 - Condições para detecção de erros de transposição

Tipo de erro	Condição
Transposição adjacente	$(i * j) * k \neq (i * k) * j$ $j * k \neq k * j$ $\forall\ i, j, k \in Q, k \neq j$
Transposição alternada	$((i * j) * l) * k \neq ((i * k) * l) * j$ $(j * l) * k \neq (k * l) * j$ $\forall\ i, j, k, l \in Q, k \neq j$

Como dito no Capítulo 1, Damm encontrou quadrados latinos de ordem 10 que satisfazem à condição de transposição adjacente. A análise de deteção dos erros é muito particular, portanto apresentamos apenas as taxas de não detecção dos erros encontradas computacionalmente na Tabela 11.

Vamos descrever a construção do código para dados alfanuméricos, mostrado no Capítulo 1, que também é capaz de detectar todos os erros de dígitos individuais e erros de transposição. Utilizaremos $Z_{36} = \{0, 1, \cdots, 35\}$.

Vamos começar com a seguinte proposição:

Proposição 2.1 *[11] Seja $Z_n = \{0, 1, \cdots, n-1\}$ e assuma $n \geq 2$ e h, k, l inteiros com h e k relativamente primos a n. Então $Q_n = (Z_n, *)$ com $a * b = (ha + kb + l) \bmod n$ é um quasigrupo. As condições para a detecção das transposições adjacentes da Tabela 7 são satisfeitas se $h-1$ e $h-k$ são relativamente primos a n. Transposições alternadas podem ser detectadas, se $h-1, h+1$ e h^2-k são relativamente primos a n.*

Pode ser verificado computacionalmente que as condições da Tabela 7 são satisfeitas nestes casos.

Corolário 2.3 *Seja q um inteiro relativamente primo a n, com $1 \leq q \leq n-2$ $(n \geq 3)$. Se $h = -q, k = 1$ e $l = 0$ todas as transposições adjacentes são detectadas, desde que $q+1$ é relativamente primo a n. Além disso, todas as transposições alternadas também serão detectadas se $q-1$ é relativamente primo a n.*

Utilizando o corolário 2.3, com $q = 1$ e $n = 9$ podemos construir o quadrado latino de ordem 9 que tem a propriedade de detectar todas as transposições adjacentes. Depois tomamos um quadrado latino de ordem 4 que também satisfaça à condição de detecção de transposição adjacente, que será escrito como uma matriz $(\alpha_{i,j})_{i,j=0,1,2,3}$. Como por exemplo o descrito na Figura 5. Substituindo cada elemento $\alpha_{i,j}$ pelo quadrado latino de ordem 9 e adicionando $9\alpha_{i,j}$ para aqueles elementos, temos um quadrado latino de ordem 36 que detecta todas as transposições adjacentes.

Figura 5 - Quadrado latino de ordem 4

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \end{bmatrix}$$

Mais explicitamente, a operação $a * b$ para todo $a, b \in Z_{36}$ é definida por

$$a * b = (-k + l) \bmod 9 + 9\alpha_{i,j}, \tag{28}$$

onde $a = k + 9i,\ b = l + 9j,\ l, k \in \{0, 1, \cdots, 8\}, i, j \in \{0, 1, 2, 3\}$.

Faremos agora um resumo da detecção de erros tanto nos métodos usados no Brasil quanto nos demais métodos encontrados na literatura.

2.4 Resumo e análise de detecção de erros

Para análise e comparação dos diversos sistemas de dígitos verificadores considerados neste trabalho, identificaremos cada sistema através de uma sigla, dada pela Tabela 8. Note que os esquemas que utilizam dois dígitos verificadores no Brasil são tão específicos, que colocamos cada um deles como um método especial.

Tabela 8 - Siglas para os diversos sistemas considerados

Sistema de dígito verificador	**Sigla**
Módulo 10 numérico com 2 pesos	MOD10N2P
Módulo 10 numérico com 3 pesos	MOD10N3P
Módulo 10 alfanumérico com 3 pesos	MOD10A3P
Módulo 10 IBM	MOD10IBM
Módulo 10 IBM Generalizado	MOD10IBMG
Módulo 10 IBM Generalizado Modificado	MOD10IBMGM
Módulo 10 IBM Colembrander Modificado	MOD10COL
Módulo 10 P.T.T	MOD10PTT
Módulo 10 Verhoeff	MOD10VER
Módulo 11 Completo 1 DV	MOD11C1
Módulo 11 Restrito 1 DV	MOD11R1
Módulo 11 Restrito 2 DV - CPF	MOD11CPF
Módulo 11 Restrito 2 DV - CNPJ	MOD11CNPJ
Módulo 11 Restrito 2 DV - CNH	MOD11CNH
Módulo 11 Restrito 2 DV - Título Eleitoral	MOD11TITE
Quadrado Latino	QUADL
Teoria dos Grupos - Verhoeff	GRUPVER

A Tabela 9 resume a classificação dos identificadores brasileiros considerados neste estudo, segundo os diversos sistemas de dígito verificador. Na primeira coluna, o nome do identificador e na segunda a sigla do sistema utilizado, segundo a tabela anterior.

Fizemos uma extensa experimentação prática de detecção de erros dos diversos esquemas estudados. Para identificadores de até 9 dígitos consideramos, exaustivamente, todos os erros possíveis. Para identificadores maiores, tal como o CNPJ, fizemos uma grande amostragem estatística, considerando amostras de 10^9 valores.

A Tabela 10 resume os percentuais de erros não detectados para os casos estudados, considerando apenas identificadores brasileiros. A coluna 1 indica a sigla do método utilizado; as colunas 2 a 5 referem-se ao tipo de erro, segundo a Tabela 1. Note que não consideramos erros fonéticos, uma vez que eles não se aplicam a nosso país. A última

Tabela 9 - Classificação dos identificadores brasileiros

Identificador	**Sigla do Sistema**
Código de barras de produtos	MOD10N2P
I.S.B.N (atual)	MOD10N2P
Passaporte	MOD10A3P
Registro geral do Estado do Rio de Janeiro	MOD10IBM
Cartão de crédito	MOD10IBM
Agência e conta corrente do Banco do Brasil	MOD11C1
I.S.B.N (antigo)	MOD11C1
Agência e conta corrente da Caixa Econ. Federal	MOD11R1
Agência e conta corrente do Bradesco	MOD11R1
PIS-PASEP	MOD11R1
CPF	MOD11CPF
CNPJ	MOD11CNPJ
CNH	MOD11CNH
Título Eleitoral	MOD11TITE

coluna, MED, é um percentual ponderado de erros não detectados, levando em conta os percentuais para cada tipo de erro, também segundo a Tabela 1.

Tabela 10 - Percentuais de erros não detectados nos identificadores brasileiros

Sistema	**IND**	**TAD**	**TAL**	**GEM**	**GAL**	**MED (%)**
MOD10N2P	0	11.11	100.00	11.11	11.11	2.02
MOD10A3P	7.81	18.02	18.02	48.76	41.57	8.53
MOD10IBM	0	2.22	100.00	6.67	11.11	1.09
MOD11C1	0	0	0	14.29	0	0.07
MOD11R1	1.82	1.82	1.82	15.85	18.18	1.77
MOD11CPF	0.24	0.17	0.17	0.45	0.02	0.21
MOD11CNPJ	0.04	0.14	0.14	0.83	0.03	0.05
MOD11CNH	0.15	0.17	0.17	0.48	1.09	0.14
MOD11TITE	1.82	1.59	0.16	12.53	1.36	1.67

Verificamos que nenhum dos esquemas de dígito verificador utilizado no Brasil detecta totalmente os cinco principais tipos de erros que ocorrem na digitação ou leitura de identificadores.

O esquema módulo 10 IBM é o melhor dos esquemas que usam módulo 10. O esquema com 3 pesos e identificadores alfanuméricos, usado no passaporte, é pouco eficiente na detecção de erros, embora seja um padrão usado mundialmente. No próximo capítulo mostraremos que existem alternativas muito melhores da que foi adotada na definição do dígito verificador para esse identificador.

Temos duas soluções módulo 11 usando um dígito verificador. O esquema que

adota o módulo 11 restrito é inferior ao esquema módulo 10 IBM. Já o esquema módulo 11 completo se destaca dentre todos os esquemas de um dígito verificador. Entretanto, ele não chega a ser uma solução totalmente satisfatória, pois é inadequado nas situações que exigem teclados numéricos, que não contêm o caractere X exigido no esquema. É ilustrativo considerar o caso do número de agência e conta corrente do Banco do Brasil. Quando as informações têm que ser digitadas via telefone, o esquema se transforma para o esquema com módulo 11 restrito, pior que alguns dos demais.

Os esquemas módulo 11 com dois dígitos verificadores visam obter uma maior detecção de erro. No caso do Título Eleitoral o esquema deixa a desejar, pois é inferior a alguns esquemas que só utilizam 1 dígito. Se examinarmos o cálculo dos dígitos verificadores para este esquema, observamos que cada cálculo leva em conta dígitos distintos, perdendo-se a oportunidade de obter um efeito cumulativo tal como é feito nos demais esquemas desta classe.

Os três outros esquemas conseguem detecção razoavelmente eficiente. O esquema do CNPJ se destaca dos dois outros, indicando que esses outros poderiam ser mais efetivos pois são, na essência, muito semelhantes.

No caso do CPF, o abandono do primeiro dígito do identificador, no cálculo do segundo dígito verificador, tem relação direta com a detecção mais baixa do que a do CNPJ.

No caso do CNH, o complicado esquema de subtrair 2 do dígito calculado é inferior ao esquema do CNPJ.

Para a comparação entre os diversos métodos, incluindo os não utilizados no Brasil, apresentamos a Tabela 11, análoga à Tabela 10, mas incluindo agora os erros fonéticos. Para representar os sistemas que usam módulo 11 com dois dígitos verificadores, colocamos apenas o esquema referente ao identificador do CNPJ do Brasil, por ser ele o melhor de todos os analisados. Nessa nova tabela acrescentamos a coluna relativa aos erros fonéticos (FON) e agrupamos os métodos em 5 grupos: esquemas para módulo 10, módulo 11 com 1 dígito verificador, módulo 11 com dois dígitos verificadores, quadrados latinos e grupos.

Podemos observar que o melhor de todos os métodos é o que utiliza módulo 11 com dois dígitos verificadores, como seria esperado.

Verificamos que, dentre os métodos que utilizam aritmética modular com base 10, o melhor método é o do Verhoeff. Este enfoque, como veremos no próximo capítulo

Tabela 11 - Percentuais de erros não detectados dos métodos analisados

Sigla do Sistema	**IND**	**TAD**	**TAL**	**GEM**	**GAL**	**FON**	**MED (%)**
MOD10N2P	0	11.11	100.00	11.11	11.11	0	2.02
MOD10N3P	0	11.11	11.11	49.23	40.73	0	1.59
MOD10A3P	7.81	18.02	18.02	48.76	41.57	0	8.53
MOD10IBM	0	2.22	100.00	6.67	11.11	12.50	1.16
MOD10IBMG	0	2.22	17.78	6.67	4.44	11.67	0.47
MOD10IBMGM	0	2.54	4.44	6.67	4.44	11.67	0.40
MOD10COLE	0	2.22	4.44	6.67	4.44	16.67	0.39
MOD10PTT	0	3.49	3.70	5.40	5.19	3.33	0.45
MOD10VER	0	2.22	4.44	2.22	4.44	3.57	0.30
MOD11C1	0	0	0	14.29	0	6.25	0.10
MOD11R1	1.82	1.82	1.82	15.85	1.82	14.61	1.80
MOD11CNPJ	0.04	0.14	0.14	0.83	0.03	2.36	0.06
QUADL	0	0	10.33	8.5	11.32	1.09	0.16
GRUPVER	0	0	5.78	4.44	5.78	0	0.09

quase atingiu a situação ótima em termos dessa linha de métodos. Apresentaremos uma melhoria que consegue atingir a situação ideal.

O módulo 11 completo com um dígito verificador é bastante bom, mas apresenta os inconvenientes já comentados anteriormente.

Constatamos que os métodos mais avançados, ou seja, baseados em quadrados latinos e teoria dos grupos, são superiores aos métodos baseados em aritmética modular, com destaque para o método de Verhoeff que utiliza o enfoque de grupos, quando se usa um dígito verificador. Entretanto é bastante intrigante o fato de que os melhores métodos não sejam utilizados na prática. Partimos do princípio que isso ocorre devido à complexidade desses métodos, o que dificulta sua compreensão e implementação. Dentre as melhorias nos esquemas que serão apresentadas no próximo capítulo, destacamos um novo método, baseado na aritmética modular base 10, usando três permutações, que busca balancear a simplicidade com a eficiência.

3 ESQUEMAS MÓDULO 10 ÓTIMOS

Neste capítulo discutimos os dois esquemas Módulo 10 que apresentam as melhores taxas de detecção de erros para esquemas deste tipo. O primeiro deles é o esquema de Verhoeff [30], descrito de forma resumida e para o qual é apresentada uma nova versão computacional. O segundo é um novo sistema ótimo quando se utilizam apenas três permutações, desenvolvido nesta dissertação.

A utilização de sistemas de dígitos verificadores visa detectar erros de dígitos em identificadores. Portanto, um sistema ótimo é o que deixa de detectar o menor percentual ponderado de erros. Como foi visto no teorema 2.3, nenhum esquema baseado em aritmética modular com base 10 pode detectar, simultaneamente, todos os seis tipos de erros que estamos considerando. A equação para a minimização da média ponderada dos erros não detectados é dada por:

$$\mu = \sum_{i=1}^{n} e_i p_i, \tag{29}$$

onde e_i é o percentual do erro tipo i, considerando todos os n tipos de erros e p_i é a probabilidade de não detecção desse tipo de erro pelo método considerado.

Deixando de lado os erros aleatórios, trabalhamos com os 6 tipos de erros iniciais da Tabela 1, IND, TAD, TAL, GEM, GAL e FON, com probabilidades de não detecção $p_1, p_2, \cdots, p_6$, respectivamente. Considerando os valores numéricos para os percentuais e_i daquela tabela, ficamos com:

$$\mu = 0,791p_1 + 0,102p_2 + 0,008p_3 + 0,005p_4 + 0,003p_5 + 0,005p_6. \tag{30}$$

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, sua detecção é primordial. Quando $k < m-1$, as permutações são aplicadas repetidamente da esquerda para a direita.

Os dois sistemas de dígitos verificadores que serão descritos foram desenvolvidos procurando-se permutações dos dígitos 0 a 9 que satisfizessem as propriedades de detecção de erros necessárias aos sistemas. Começaremos a discutir esses sistemas apresentando

algoritmos envolvidos na manipulação de permutações, para atingir as propriedades buscadas.

3.1 Algoritmos para análise da detecção de erros em sistemas Módulo 10

Consideremos, primeiramente, o cálculo de 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.

Vamos exemplificar com as permutações $p : (1\ 4\ 2\ 5\ 0\ 9\ 3\ 6\ 8\ 7)$ e $q : (3\ 7\ 0\ 8\ 5\ 1\ 4\ 6\ 2\ 9)$. Para encontrar os casos de erros não detectados, deve-se fazer a diferença módulo 10 entre essas permutações: $d = p - q : (8\ 7\ 2\ 7\ 5\ 8\ 9\ 0\ 6\ 8)$. Essa diferença mostra que as transposições entre os números 1 e 3, nas posições em que p e q foram aplicadas, não são detectadas, pois $p(1) - q(1) = 4 - 7 = -3 \equiv p(3) - q(3) = 5 - 8 = -3 \bmod 10$, ou seja, temos o mesmo resultado.

Isso mostra que, para calcularmos todos os erros de transposição não detectados, devemos contar a quantidade s de repetições 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. No exemplo, temos duas repetições do número 7 e três do número 8. O total de erros de transposição não detectados pelo par de permutações considerado é, portanto, $t = 2.1 + 3.2 = 8$. Os 8 pares de trocas não detectadas são (1, 3), (3, 1), (0, 5), (5, 0), (0, 9), (9, 0), (5, 9) e (9, 5).

Essa explicação sugere o algoritmo $ErrosTra$ a seguir, que calcula o número de transposições não detectadas pela aplicação das permutações p e q.

Observe que o algoritmo mostrado serve para calcular a não detecção, tanto de erros de transposição adjacente quanto a alternada. Basta usar o par de permutações adequado.

O algoritmo para calcular os erros gêmeos não detectados é análogo. Ao invés de subtrair as permutações envolvidas, elas têm que ser somadas, pois a não detecção da

Algoritmo 1: ErrosTra(p, q):

inicio
- Zerar vetor S; $t \leftarrow 0$;
- **Para** *i de 0 a 9* **:**
  - $k \leftarrow (p[i] - q[i] + 10) \bmod 10$; $S[k] \leftarrow S[k] + 1$;
- **Para** *i de 0 a 9* **:**
  - $t \leftarrow t + S[i] * (S[i] - 1)$;
- **Retorna** t;

troca de um dígito a nas posições em que p e q são aplicadas, por outro dígito $b \neq a$, ocorre se $p(a) + q(a) \equiv p(b) + q(b) \bmod 10$. O algoritmo *ErrosGem* faz o cálculo procurado.

Algoritmo 2: ErrosGem(p, q)

inicio
- Zerar vetor S; $t \leftarrow 0$; **Para** *i de 0 a 9* **:**
  - $k \leftarrow (p[i] + q[i]) \bmod 10$; $S[k] \leftarrow S[k] + 1$;
- **Para** *i de 0 a 9* **:**
  - $t \leftarrow t + S[i] * (S[i] - 1)$;
- **Retorna** t;

Imprime vetor CG;

Esse algoritmo permite o cálculo tanto para erros gêmeos adjacentes quanto alternados.

Resta considerar o cálculo de erros fonéticos. Observe que 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 $a = 2, 3, \cdots, 9$. De forma equivalente, quando temos, $p(1) - q(0) \equiv p(a) - q(a) \bmod 10$. Sempre que a igualdade ocorre, 2 erros deixam de ser detectados, a troca de $1a$ por $a0$ e vice-versa.

O algoritmo *ErrosFon* faz o cálculo procurado.

Algoritmo 3: ErrosFon(p, q)

inicio
- $t \leftarrow 0$;
- **Para** *i de 2 a 9* **:**
  - **Se** *($(p[1] - q[0] + 10) \bmod 10 = (p[i] - q[i] + 10) \bmod 10$)* **Então**
  - $t \leftarrow t + 2$;
- **Retorna** t;

3.1.1 O sistema Módulo 10 de Verhoeff

O melhor sistema de dígito verificador módulo 10 conhecido até aqui era aquele proposto por Verhoeff [30], cujos resultados foram mostrados no Capítulo 2. Ele desenvolveu um sistema 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 para erros gêmeos quanto de transposição.

4. apenas deixam de ser detectados 2 em 16 erros fonéticos a cada grupo de 6 permutações usadas.

Devido às limitações computacionais da época de seu desenvolvimento, 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. Como veremos adiante, esse método parece ser ótimo apenas para identificadores com até 7 dígitos.

Mostraremos a seguir, uma sequência de algoritmos que obtém um sistema equivalente ao de Verhoeff. Esse sistema consiste de um conjunto L de permutações. O primeiro passo para encontrar L é obter o conjunto S de todas as permutações que minimizam a não detecção de transposições e erros gêmeos em relação à permutação identidade, r : (0 1 2 3 4 5 6 7 8 9), que é a primeira permutação do conjunto. Isso pode ser conseguido através do seguinte algoritmo:

No Algoritmo 4, a função **Proxperm** obtém a próxima permutação em ordem lexicográfica, em relação à última gerada. As funções **ErrosTra(p,q)** e **ErrosGem(p,q)** são as que foram descritas anteriormente e obtêm, respectivamente, a quantidade de erros de transposição adjacente e de erros gêmeos não detectados quando as permutações p e q são aplicadas a duas posições consecutivas de um identificador. Ao rodar esse algoritmo obtemos um conjunto S com $s = 2800$ permutações.

Algoritmo 4:

```
inicio
    r ← (0 1 2 3 4 5 6 7 8 9);   p ← r;   n ← 0;
    Para i de 2 a 10! :
        p ← Proxperm(p);
        Se (ErrosTra(r, p) = 2 e ErrosGem(r, p) = 2) Então
            n ← n + 1;   S[n] ← p;
```

O conjunto S será utilizado para encontrar a cadeia de permutações procurada. Um primeiro requisito entre cada par consecutivo de permutações é ter as propriedades mencionadas relativas à detecção de transposições e erros gêmeos adjacentes. Dada uma permutação p, o conjunto T das permutações que têm essa propriedade em relação a p pode ser obtido a partir do conjunto S mediante um mapeamento simples que transforma r em p. O Lema 3.1 comprova que as propriedades são mantidas por esse mapeamento.

Lema 3.1 *Seja um sistema de dígito verificador baseado em aritmética modular. Dadas três permutações p, q e a identidade r, o par r e q deixa de detectar os mesmos erros que o par p e s, onde s é tal que $s \circ p^{-1} = q$.*

Demonstração: Sejam as permutações r e q tais que

$$r(a) - q(a) \equiv r(b) - q(b) \bmod n, \text{ para } a \neq b. \tag{31}$$

E seja um mapeamento s tal que $s(p^{-1}(x)) = q(x)$.

Como r é a identidade, temos que $r = p \circ p^{-1}$. Substituindo q por $s \circ p^{-1}$ na equação 31, temos

$$p(p^{-1}(a)) - s(p^{-1}(a)) \equiv p(p^{-1}(b)) - s(p^{-1}(b)) \bmod n. \tag{32}$$

Como p^{-1} é uma permutação e $a \neq b$, temos que $p^{-1}(a) = c \neq d = p^{-1}(b)$. Portanto, podemos reescrever a equação 32 como:

$$p(c) - s(c) \equiv p(d) - s(d) \bmod n \text{ para } c \neq d. \tag{33}$$

Concluímos assim que, se as transposições entre a, b, nas posições em que r, q são aplicadas, não são detectadas, então as transposições entre c, d, nas posições em que p, s são aplicadas também não são detectadas.

As demonstrações de que os casos de detecção são mantidos e para o caso de erros gêmeos são análogas. □

Esse lema justifica o algoritmo $MapPerm(p, q)$ que transforma uma permutação q, do conjunto S em uma permutação do conjunto T, que tem, em relação a p, as mesmas propriedades que q tem em relação a r.

Algoritmo 5: MapPerm(p, q)

```
inicio
    Para i de 0 a 9 :
        pinv[p[i]] = i;
    Para i de 0 a 9 :
        qt[pinv[i]] = q[i];
    Retorna qt;
```

Um segundo requisito na busca das permutações é a minimização secundária das não detecções de transposições e erros gêmeos alternados, de forma que cada valor seja igual a 4, que é o mínimo possível.

Finalmente, o terceiro requisito é, que ao mesmo tempo que se obtém uma permutação na sequência procurada, seja verificado que todos os erros fonéticos são detectados.

Esses requisitos são considerados no algoritmo de backtracking a seguir, que encontra conjuntos L com 6 permutações.

Algoritmo 6:

```
Config();
Para i de 1 a s :
    q ← MapPerm(S[i], L[l]);   e ← ErrosFon(q, L[l]);
    Se (l > 1) Então
        e ← e + ErrosGem(q, L[l − 1]) + ErrosTra(q, L[l − 1])
    Senão e ← e + 8;
    Se (e = 8) Então
        l ← l + 1;   L[l] ← q;
        Se (l = 6) Então
            Imprime
        Senão Config();
        l ← l − 1;
inicio
    Obter S;   L[1] ← r;   l ← 1; Config();
```

No Algoritmo 6, s e l referem-se aos tamanhos de S e L respectivamente. O procedimento $Config$ preenche, sucessivamente, conjuntos L com 6 permutações. A função

$MapPerm$ faz o mapeamento de uma permutação de S, tal que a nova permutação tenha as propriedades desejadas em relação às permutações já colocadas em L.

A variável e conta os casos de não detecção de erros fonéticos e alternados. Na geração da segunda permutação a contagem de erros alternados não se aplica, então soma-se artificialmente 8 a e. Quando temos $e = 8$, permutação é colocada em L.

Quando se tenta ampliar esse algoritmo para obter um conjunto L com mais de 6 permutações, observa-se experimentalmente um fato que Verhoeff tinha constatado em seu sistema. Não é possível acrescentar a sétima permutação ao conjunto L mantendo todas as regras do Algoritmo 6. Para que a geração continue após cada grupo de 6 permutações é necessário admitir-se 2 erros fonéticos. Essa mudança está refletida no Algoritmo 7, o qual pode gerar conjuntos L de quaisquer tamanhos.

Algoritmo 7:

```
Config()
Para i de 1 a s :
    q ← MapPerm(S[i], L[l]);   e ← ErrosFon(q, L[l]);
    Se (l mod 6 = 0) Então e ← e − 2;
    Se (l > 1) Então
        e ← e + ErrosGem(q, L[l − 1]) + ErrosTra(q, L[l − 1])
    Senão e ← e + 8;
    Se (e = 8) Então
        l ← l + 1;   L[l] ← q;
        Se (l = m) Então
            ec ← ErrosTra(L[m], r) + ErrosGem(L[m], R);
            ec ← ec + ErrosTra(L[m − 1], r) + ErrosGem(L[m − 1], r);
            ec ← ec + ErrosFon(L[m], r);
            Se (m mod 6 = 0) Então ec ← ec − 2;
            Se (ec = 12) Então Imprime;
        Senão Config();
        l ← l − 1;
inicio
    Obter S;   Definir m;   L[1] ← r;   l ← 1;   Config();
```

O Algoritmo 7 mostrou-se capaz de gerar conjuntos L de qualquer tamanho. A Tabela 12 mostra um exemplo de um conjunto L obtido através do Algoritmo 7 para $m = 18$.

Embora o sistema de Verhoeff tenha sido o melhor sistema módulo 10 desde 1969, não encontramos qualquer referência sobre sua utilização prática. Supomos que isso se

Tabela 12 - Conjunto L com 18 permutações

1	0123456789	7	3456710298	13	6752498310
2	9460571382	8	8359472106	14	8310795264
3	3620845197	9	0397568421	15	2931568704
4	1290857436	10	0189723564	16	6582741903
5	8340971265	11	0562184739	17	0541823976
6	8142630579	12	4175908236	18	4189053627

deve a uma barreira psicológica que considera a implementação do sistema complicada e passível de erros, por envolver um grande número de permutações.

3.2 Esquema ótimo módulo 10

Durante este trabalho foi idealizado e desenvolvido um esquema de dígito verificador módulo 10 ótimo, mediante a restrição de somente utilizar três permutações. A ideia foi buscar um código bastante simples que pudesse ser usado na prática, contornando as dificuldades que parecem existir no uso de esquemas eficientes, porém complexos.

Note que três permutações é o mínimo necessário para que os erros alternados sejam detectáveis. A primeira dessas permutações é a identidade: $\sigma_1 : (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$. Essas três permutações são aplicadas, alternadamente, aos dígitos do identificador.

Vamos a seguir mostrar as etapas da construção desse sistema e provar sua otimalidade. Mostraremos adiante que existem 48 triplas que satisfazem às condições que desejamos. Inicialmente analisamos o poder de detecção de erros de um sistema formado por uma dessas 48 triplas: $\sigma_1 : (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$, $\sigma_2 : (0\ 8\ 6\ 4\ 2\ 7\ 9\ 1\ 3\ 5)$ e $\sigma_3 : (1\ 6\ 3\ 2\ 8\ 7\ 4\ 0\ 5\ 9)$.

Lema 3.2 *O sistema de dígito verificador módulo 10 que utiliza as três permutações* $\sigma_1 : (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$, $\sigma_2 : (0\ 8\ 6\ 4\ 2\ 7\ 9\ 1\ 3\ 5)$ *e* $\sigma_3 : (1\ 6\ 3\ 2\ 8\ 7\ 4\ 0\ 5\ 9)$ *não detecta:*

1. *0 erros individuais;*
2. *2/90 dos erros de transposição;*
3. *16/270 dos erros gêmeos;*
4. *0 erros fonéticos.*

Sendo assim, possui uma taxa de não detecção de erros de $\mu = 0,102(2/90)+0,008(2/90)+0,005(16/270)+0,003(16/270) = 0,29\%$.

Demonstração: O sistema detecta todos os erros individuais, pois utiliza permutações. Um sistema de dígito verificador não detecta erros de transposição nas posições i e j quando $(\sigma_i - \sigma_j)(a_i) \equiv (\sigma_i - \sigma_j)(a_j) \bmod 10$; erros gêmeos nas posições i e j quando $(\sigma_i + \sigma_j)(a) \equiv (\sigma_i + \sigma_j)(b) \bmod 10$ e erros fonéticos quando $\sigma_i(1) - \sigma_{i+1}(0) \equiv (\sigma_i - \sigma_{i+1})(a) \bmod 10$, para $a \neq 0, 1$. Mostramos na Tabela 13, os casos em que as igualdades são satisfeitas para cada tipo de erro.

Tabela 13 - Posições de erros não detectados

1	$\sigma_1 - \sigma_2$	0 3 6 9 2 8 7 6 5 4
2	$\sigma_2 - \sigma_3$	9 2 3 2 4 0 5 1 8 6
3	$\sigma_3 - \sigma_1$	1 5 1 9 4 2 8 3 7 0
4	$\sigma_1 + \sigma_2$	0 9 8 7 6 2 5 8 1 4
5	$\sigma_2 + \sigma_3$	1 4 9 6 0 4 3 1 8 4
6	$\sigma_1 + \sigma_3$	1 7 5 5 2 2 0 7 3 8
7	$\sigma_1(1) - \sigma_2(0)$	1
8	$\sigma_2(1) - \sigma_3(0)$	7
9	$\sigma_3(1) - \sigma_1(0)$	6

Note que para analisar os erros adjacentes deve-se relacionar as permutações σ_1 com σ_2, σ_2 com σ_3 e σ_3 com σ_1. Para os erros alternados relaciona-se σ_1 com σ_3, σ_2 com σ_1 e σ_3 com σ_2. No caso geral de transposições e gêmeos, basta analisar as combinações dois a dois das três permutações.

As linhas de 1 a 3 da Tabela 13 mostram os casos de não detecção de erros de transposição. Todo par de valores cujas imagens de $\sigma_i - \sigma_j$ são iguais não são detectadas. Portanto, $A_{2,2} = 2$ transposições em 90 possíveis não são detectadas. As linhas de 4 a 5 mostram os casos de não detecção de erros gêmeos em geral. Na linha 4, ocorrem $A_{2,2} = 2$ casos de não detecção. Na linha 5, à troca entre os elementos cujas imagens de $\sigma_2 + \sigma_3$ são iguais a 1 correspondem $A_{2,2} = 2$ erros e à troca entre os elementos cujas imagens são iguais a 4 correspondem $A_{3,2} = 6$ erros, portanto somam um total de 8 erros não detectados. Na linha 6, à troca entre os elementos cujas imagens $\sigma_1 + \sigma_3$ são iguais a 2, 5 e 7, correspondem $3A_{2,2} = 6$ erros.

As linhas de 7 a 9, mostram que nenhum erro fonético deixa de ser detectado, pois $\sigma_i(1) - \sigma_{i+1}(0) \not\equiv (\sigma_i - \sigma_{i+1})(a) \bmod 10, \forall\ 0 \leq a \leq 9$. Esta equação pode ser generalizada para qualquer valor de a, pois nos casos de $a = 0$ ou 1, a equação corresponde a erros

individuais. □

Lema 3.3 *Um sistema de dígito verificador módulo 10 com três permutações ótimo deixa de detectar 2/90 dos erros de transposição.*

Demonstração: Suponha um sistema ótimo no qual todos os erros fonéticos são detectados e todos os pares de permutações deixam de detectar 2/90 dos erros gêmeos (o melhor possível), dois dos 3 pares deixam de detectar 2/90 erros de transposição e apenas um par deixa de detectar 4/90 (o próximo menor valor possível). Então, o percentual de não detecção $\mu = 0,102(8/270) + 0,008(8/270) + 0,005(2/90) + 0,003(2/90) = 0,0034$ é superior ao do sistema do Lema 3.2. Isto é uma contradição. □

Como consequência dos fatos mostrados, os erros individuais devem ser 100% detectados e exatamente 2/90 erros de transposição não serão detectados, temos o corolário a seguir:

Corolário 3.1 *Um sistema de dígito verificador módulo 10 ótimo deve minimizar a não detecção de erros gêmeos e erros fonéticos.*

Mostraremos na sequência que a detecção de erros gêmeos é independente da de erros fonéticos. Como passo inicial, temos um algoritmo para a obtenção do número mínimo de não detecção de erros gêmeos.

Algoritmo 8:

```
r ← (0 1 2 3 4 5 6 7 8 9);   p ← r;   m ← 0;   Zerar CG[ ];
inicio
    Para i de 2 a 10! :
        p ← Proxperm(p);
        Se (ErrosTra(r, p) = 2) Então m ← m + 1;   S[m] ← p;
    Para i de 1 a m − 1 :
        Para j de i + 1 a m :
            Se (ErrosTra(S[i], S[j]) = 2) Então
                k ←
                ErrosGem(r, S[i]) + ErrosGem(S[i], S[j]) + ErrosGem(S[j], r);
                CG[k] ← CG[k] + 1;
    Imprime vetor CG;
```

O Algoritmo 8 gera um vetor S com as permutações p tais que $ErrosTra(r, p) = 2$, ou seja, visando a minimização de não detecção de transposições. Experimentalmente

verifica-se que S contém 46.400 permutações. No algoritmo são examinadas todas as triplas (r, p, q), onde $p, q \in S$, tal que $ErrosTra(p, q) = 2$. O vetor CG conta, para cada valor k, com $0 \leq k \leq 90$, a quantidade total de erros gêmeos não detectados pelos pares de permutações de cada tripla. Obteve-se experimentalmente que o total de triplas onde todos os pares de permutações deixam de detectar 2 erros de transposição é de 12.654.000. O menor índice não zerado do vetor CG é $k = 16$, que é então o número mínimo de erros gêmeos não detectados. No Anexo B pode-se ver a distribuição desses erros. O mínimo de 16 erros é relativo ao total de 270 erros, já que temos três pares distintos em cada tripla e o número de erros refere-se às três permutações em conjunto.

Uma pequena modificação no Algoritmo 8 permite analisar a estrutura dos 16 erros gêmeos para as triplas com erros mínimos. Ela mostra que as distribuições de erros gêmeos não detectados pelos três pares de permutações é sempre uma permutação do conjunto $\{2, 6, 8\}$.

Como já dito, a minimização de erros fonéticos pode ser tratada de forma independente da de erros gêmeos pois existem triplas mínimas capazes de detectar todos os erros fonéticos, como será mostrado pelo resultado do algoritmo a seguir. Uma modificação no algoritmo anterior permite obter, então, os sistemas que minimizam erros gêmeos e fonéticos.

Algoritmo 9:

```
r ← (0 1 2 3 4 5 6 7 8 9);   p ← r;   m ← 0;
inicio
    Para i de 2 a 10! :
        p ← Proxperm(p);
        Se (ErrosTra(r, p) = 2) Então m ← m + 1;   S[m] ← p;
    Para i de 1 a m − 1 :
        Para j de i + 1 a m :
            Se (ErrosTra(S[i], S[j]) = 2) Então
                k ←
                ErrosGem(r, S[i]) + ErrosGem(S[i], S[j]) + ErrosGem(S[j], r);
                f ←
                ErrosFon(r, S[i]) + ErrosFon(S[i], S[j]) + ErrosFon(S[j], r);
                Se ((k = 16) e (f = 0)) Então Imprime r, S[i], S[j];
```

O Algoritmo 9 obtém 100 triplas com as características desejadas. Pode-se observar experimentalmente, agora, que a distribuição de não detecção de erros gêmeos entre os

pares de permutações é apenas de dois tipos: $\langle 2, 8, 6\rangle$ e $\langle 6, 8, 2\rangle$. As triplas do primeiro tipo são preferíveis, pois minimizam a não detecção para identificadores de quaisquer tamanhos, como será visto no Anexo C. Com essa última observação, pode-se ainda modificar o Algoritmo 9 para que a não detecção de erro gêmeos para o primeiro par de cada tripla seja igual a 2. Isso deixa um total de 48 triplas ótimas. A Tabela 14 mostra as duas últimas permutações das 48 triplas, que têm a identidade como permutação inicial obtidas pelo uso desse algoritmo.

Tabela 14 - Sistemas ótimos módulo 10

1	0864279135	1632874059	25	2086491357	9410652837
2	0864279135	2743985160	26	3197502468	3854096271
3	0864279135	3854096271	27	3197502468	4965107382
4	0864279135	4965107382	28	3197502468	5076218493
5	0864279135	5076218493	29	3197502468	6187329504
6	0864279135	6187329504	30	3197502468	7298430615
7	0864279135	7298430615	31	3197502468	8309541726
8	0864279135	8309541726	32	3197502468	9410652837
9	0864279135	9410652837	33	4208613579	4965107382
10	1975380246	2743985160	34	4208613579	5076218493
11	1975380246	3854096271	35	4208613579	6187329504
12	1975380246	4965107382	36	4208613579	7298430615
13	1975380246	5076218493	37	4208613579	8309541726
14	1975380246	6187329504	38	4208613579	9410652837
15	1975380246	7298430615	39	5319724680	6187329504
16	1975380246	8309541726	40	5319724680	7298430615
17	1975380246	9410652837	41	5319724680	8309541726
18	2086491357	2743985160	42	5319724680	9410652837
19	2086491357	3854096271	43	6420835791	7298430615
20	2086491357	4965107382	44	6420835791	8309541726
21	2086491357	5076218493	45	6420835791	9410652837
22	2086491357	6187329504	46	7531946802	8309541726
23	2086491357	7298430615	47	7531946802	9410652837
24	2086491357	8309541726	48	8642057913	9410652837

Lema 3.4 *Um sistema de dígito verificador módulo 10 ótimo com três permutações deve detectar os mesmos erros que o sistema do Lema 3.2.*

Esse resultado é provado experimentalmente pelos Algoritmo 8 e Algoritmo 9.

3.3 Comparação entre os dois métodos ótimos

O sistema de Verhoeff continua sendo o melhor sistema Módulo 10 conhecido para identificadores com até 7 dígitos, porque até esse tamanho consegue detectar todos os erros fonéticos. Entretanto não está provado que ele é um sistema ótimo. Até onde experimentamos, parece ser, mas a demonstração da otimalidade não foi feita nem pelo próprio autor. O novo sistema desenvolvido neste trabalho passa a ser superior ao de Verhoeff para identificadores de tamanhos maiores que 7, que são os mais usuais. Além disso, ele traz uma vantagem adicional por ser de utilização mais simples, pois envolve apenas três permutações. Considerando essa classe de sistemas, foi provado que o sistema apresentado é ótimo. Ele apresenta a menor taxa média de não detecção de erros dentre todos os citados na literatura, conforme pode ser visto na Tabela 15. Porém continua em aberto a questão de se existe algum outro sistema ótimo, superior ao apresentado, utilizando mais permutações, para identificadores de mais de 7 dígitos.

Tabela 15 - Esquemas Módulo 10

Esquema	**IND**	**TAD**	**TAL**	**GEM**	**GAL**	**FON**	**MED**
2 pesos	0	11.11	100.00	11.11	11.11	0	2.02
3 pesos	0	11.11	11.11	49.23	40.73	0	1.59
IBM	0	2.22	100.00	6.67	11.11	12.50	1.16
IBM Generalizado	0	2.22	17.78	6.67	4.44	11.67	0.47
PTT	0	3.49	3.70	5.40	5.19	3.33	0.45
IBM Modificado	0	2.54	4.44	6.67	4.44	11.67	0.40
Colenbrander	0	2.22	4.44	6.67	4.44	16.67	0.39
Vehroeff	0	2.22	4.44	2.22	4.44	3.57	0.30
Mod 10 3P Ótimo	0	2.22	2.22	5.93	5.93	0	0.29

4 MELHORIAS NOS ESQUEMAS DE DÍGITOS VERIFICADORES

Neste capítulo discutimos e apresentamos melhorias para os demais esquemas de dígito verificador utilizados no Brasil. Propomos também sistemas para bases modulares diferentes de 10 e 11.

4.1 Esquema ótimo para bases modulares primas

Desenvolveremos, a seguir, um esquema geral ótimo para bases modulares primas. Estamos interessados, particularmente, nas bases 11, adequada para identificadores formados por dígitos decimais e 37, para dígitos alfanuméricos.

Vimos que, quando a base modular n é um número primo, podemos utilizar os esquemas de pesos variando de 1 a $n-1$ pois garantimos a detecção de erros individuais. Para detectar transposições basta 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 levar em conta os erros gêmeos, é necessário que a soma de pesos consecutivos ou alternados não seja igual a n. Para a detecção de erros fonéticos, é necessário que:

$$p_i \not\equiv (p_i - p_{i+1})a \bmod n, \quad \text{para } 0 \leq a \leq 9. \tag{34}$$

Note que os produtos $(p_i - p_{i+1})a$ para $0 \leq a < n$ e quaisquer valores distintos de p_i e p_{i+1} geram uma permutação dos números de 0 a $n-1$ em módulo n. Portanto, para todo $a > 9$ teremos a desigualdade satisfeita, determinando assim uma relação que p_i e p_{i+1} devem satisfazer.

No caso do módulo 11, a desigualdade é satisfeita apenas para $a = 10$. Substituindo esse valor na equação, temos:

$$p_i \equiv 10(p_i - p_{i+1}) \bmod 11 \Longrightarrow p_{i+1} \equiv 2p_i \bmod 11. \tag{35}$$

Portanto, escolhendo-se cada peso como o dobro do anterior em módulo 11 satisfazemos às condições para detecção de erros fonéticos. Ou seja, deve-se aplicar alterna-

damente 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, utilizaremos uma abordagem diferente para gerar estas várias combinações. Iremos criar 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. Note que a relação não é comutativa, pois o par (u, v) pode detectar todos os erros fonéticos, enquanto o par (v, u) pode não ter essa propriedade. Na criação desse digrafo, para evitar erros gêmeos, desconsideramos os pares onde temos $u + v = n$. O Algoritmo 10 faz essa criação, onde E é a matriz de adjacências do digrafo.

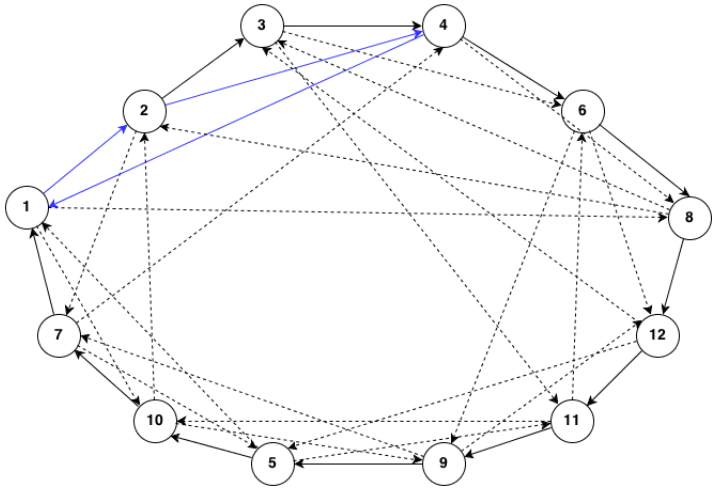
Algoritmo 10:

```
CriaDigrafo();
Zera E;
Para i de 1 a n − 1 :
    Para j de 1 a n − 1 :
        Se (i ≠ j) e ((i + j) ≠ n) Então
            t ← 0;
            Para m ← 2 a 9 :
                Se i + jm ≡ im mod n Então t ← 1;
            Se t = 0 Então E[i, j] ← 1;
```

Após criar o digrafo, procuramos ciclos no mesmo, do tamanho desejado, tal que, para cada dois elementos alternados do ciclo, sua soma nunca seja n. O digrafo para $n = 13$ é mostrado na Figura 6. Destacamos, como exemplos, o ciclo de tamanho 3, em azul $\langle 1, 2, 4\rangle$; e o ciclo de tamanho 12, em preto $\langle 1, 2, 3, 4, 6, 8, 12, 11, 9, 5, 10, 7\rangle$.

Para obter esses ciclos poderíamos utilizar o algoritmo de Dijkstra. Porém, como temos o teste especial para gêmeos alternados e também como, em geral, estamos procurando ciclos de tamanho pequeno, podemos utilizar o algoritmo de backtracking mostrado na sequência. Nesse algoritmo, L é o conjunto de pesos procurados e $k > 2$ o tamanho do ciclo desejado. T é um vetor auxiliar para indicar os elementos que já entraram no ciclo. A variável *pode* é verdadeira quando se pode acrescentar um novo elemento ao ciclo. Para isso, verifica-se se o novo elemento a acrescentar ainda não entrou no ciclo, se a soma de elemmentos alternados é diferente de n e se o ciclo fecha apropriadamente quando se vai colocar o k-ésimo elemento nesse ciclo.

Figura 6 - Digrafo para $n = 13$



Algoritmo 11:

```
Config();
Para i de 1 a n − 1 :
    pode ← True;
    Se (not T[i]) Então pode ← False;
    Senão Se (l > 0) e (E[L[l], i] = 0) Então pode ← False;
    Senão Se (l > 1) e ((L[l − 1] + i) = n) Então pode ← False;
    Senão Se (l = (k − 1)) e ((E(i, L[1]) = 0) ou ((L[l] + L[1]) = n)) Então
        pode ← False;
    Se (pode) Então
        l ← l + 1;   L[l] ← i;   T[i] ← False;
        Se (l = k) Então Imprime;
        Senão Config();
        l ← l − 1;   T[i] ← True;
inicio
    l ← 0;   T[∗] ← True;   Config();
```

Mostramos na Tabela 16 algumas saídas do algoritmo para quatro bases primas: 11, 13, 37 e 97 e valores de k: 3, 6 e 10.

A partir do algoritmo dado verificamos, experimentalmente, a propriedade já vista com o tratamento algébrico, a de que para $n = 11$, o ciclo de tamanho 10 encontrado é o único que satisfaz à condição de detecção de erros fonéticos. Portanto, para termos sistemas ótimos de base 11, sendo m o tamanho do identificador, é preciso usar $m - 1$ pesos para identificadores com $m \leq 10$ e, para valores maiores, usar repetidamente os pesos do ciclo mostrado. Nos demais casos, temos sistemas ótimos com o uso de apenas 3 pesos, como é mostrado na proposição a seguir.

Tabela 16 - Exemplos de conjuntos com k pesos para bases primas n

$n \backslash k$	3	6	10
11	-	-	1 2 4 8 5 10 9 7 3 6
13	1 2 4	1 2 3 6 9 5	1 2 3 4 6 8 12 11 9 5
	3 4 8	1 8 3 6 12 5	1 2 3 4 6 12 5 10 9 7
	5 10 7	2 3 6 9 5 10	1 2 3 4 6 12 11 9 7 5
37	1 2 3	1 2 3 4 5 6	1 2 3 4 5 6 7 8 9 10
	1 2 8	1 2 3 4 5 8	1 2 3 4 5 6 7 8 9 11
	1 2 12	1 2 3 4 5 12	1 2 3 4 5 6 7 8 9 12
97	1 2 3	1 2 3 4 5 6	1 2 3 4 5 6 7 8 9 10
	1 2 5	1 2 3 4 5 9	1 2 3 4 5 6 7 8 9 12
	1 2 12	1 2 3 4 5 14	1 2 3 4 5 6 7 8 9 14

Proposição 4.1 *Para todo número primo $n \geq 17$, o esquema de dígito verificador que utiliza base n e o ciclo de pesos $\langle 1, 2, 3 \rangle$ detecta todos os erros fonéticos.*

Demonstração: A equação 34 é satisfeita de três formas:

$$1 + a \not\equiv 0 \bmod n, \quad \text{para } 0 \leq a \leq 9. \tag{36}$$

$$2 + a \not\equiv 0 \bmod n, \quad \text{para } 0 \leq a \leq 9. \tag{37}$$

$$3 - 2a \not\equiv 0 \bmod n, \quad \text{para } 0 \leq a \leq 9. \tag{38}$$

Esta última equação é sempre verificada para $n \geq 17$, pois os possíveis valores do lado esquerdo são 3, 1, -1, -6, -5, -7, -9, -11, -13 e -15. Portanto todos os erros fonéticos são detectados. □

Particularizamos, a seguir, os casos relacionados a identificadores utilizados no Brasil.

4.1.1 Esquema ótimo para módulo 11, com um dígito verificador

No caso de Módulo 11 completo, basta utilizar a sequência de pesos $\langle 1, 2, 4, 8, 5, 10, 9, 7, 3, 6 \rangle$ e temos a detecção de todos os seis tipos de erros considerados neste trabalho.

No caso do módulo 11 restrito, os erros não detectados correspondem a 1,82% dos casos de erro de cada tipo, devido às situações onde inicialmente o cálculo indica 10 e, com o erro, passa para 0, ou vice-versa, ou seja, 2/110 dos casos de erros. Com isso, o percentual ponderado de erros não detectados fica em 1,65%.

Nos dois casos, o esquema ótimo representa uma melhoria em relação à situação atual.

4.2 Esquema ótimo para identificadores alfanuméricos com um dígito verificador

No caso de identificadores alfanuméricos, usando um dígito verificador, o ideal é que esse dígito seja também alfanumérico. Neste caso, podemos usar o esquema geral mostrado, usando base modular 37. Como temos 36 dígitos alfanuméricos no alfabeto latino, uma solução natural quando se usa esse alfabeto é considerar um caractere adicional. No caso do passaporte esse caractere corresponde a $<$. Além disso, os caracteres não numéricos, ou seja, pertencentes ao conjunto $\{A, B, \cdots, Z, <\}$ seriam mapeados para o conjunto numérico $\{10, 11, 12, \cdots, 36\}$. O conjunto de pesos mais simples a ser usado consiste de 3 pesos $\langle 1, 2, 3\rangle$ usados de forma repetida.

No caso de outros alfabetos tais como o alemão e o escandinavo, números primos adequados teriam que ser utilizados.

Na solução restrita, usariam-se apenas identificadores do conjunto $\{0, 1, 2, \cdots, 9, A, B, C, \cdots, Z\}$, mapeados para números de 0 a 35, e dígito o verificador no mesmo conjunto inicial. O cálculo que indicasse o dígito correspondente a 36 seria transformado para 0. Então os erros não detectados seriam análogos ao caso do módulo 11 restrito, correspondendo a $2/(37 \cdot 36)$, ou seja, 0,15% de não detecção para cada tipo de erro. Exceto para erros fonéticos, cujo valor de não detecção corresponderia a $56 \cdot 2/(37 \cdot 36) = 0,06\%$. Sendo assim, a média ponderada de não detecção de erros corresponde a 0,14%.

Qualquer das duas opções corresponde a uma melhoria muito grande em relação ao esquema utilizado atualmente.

4.3 Esquemas ótimos para módulo 11 com dois dígitos verificadores

No caso de dois dígitos verificadores usando módulo 11, o esquema ótimo consiste

em usar o esquema geral, tal como descrito no caso de um dígito. Além disso, pode-se fazer apenas um cálculo e deve-se representar o resultado com dois dígitos. Ou seja, resultados 0 a 9 seriam representados com um 0 no primeiro dígito verificador e o resultado propriamente dito no segundo. O resultado 10 seria representado normalmente com dois dígitos.

Este esquema é muito mais simples que os diversos casos de dois dígitos verificadores mostrados e mais eficiente, pois detecta todas as situações de erro.

Sua vantagem em relação ao módulo 11 com apenas um dígito é evitar o uso do caractere não numérico X, trabalhando-se sempre apenas com os dígitos decimais Porém tem a desvantagem de se utilizar um dígito a mais em cada identificador.

4.4 Resumo das melhorias propostas

A Tabela 17 a seguir resume os percentuais de erros não detectados para essas melhorias consideradas. A estrutura dessa tabela é análoga à da Tabela 10.

Tabela 17 - Porcentagem de erros não detectados para os sistemas propostos

Esquemas	IND	TAD	TAL	GEM	GAL	FON	MED %
Mod 10 ótimo	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

CONCLUSÃO

Verificamos que todos os esquemas de dígitos verificadores utilizados no Brasil podem ser melhorados através das propostas apresentadas neste trabalho, considerando os 6 tipos de erros mais comuns na entrada de dados e suas probabilidades de ocorrência.

No caso do módulo 10, foi desenvolvido um método ótimo usando três permutações, superior a todas as alternativas módulo 10 encontradas na literatura. Apresentamos também uma nova versão computacional para o sistema módulo 10 Verhoeff, que era o melhor sistema módulo 10 conhecido até este trabalho e que continua sendo superior ao método apresentado, para identificadores pequenos. Além disso, o esquema é também melhor que o módulo 11 restrito ótimo. Nos demais casos as melhorias são alcançadas com pequenos cuidados derivados de uma análise detalhada das causas de não detecção de erros pelos diversos esquemas.

O esquema para situações alfanuméricas melhora consideravelmente o uso de dígitos verificadores nas situações análogas às do uso no passaporte.

Mostramos também neste trabalho que existem sistemas de dígito verificador baseados na Teoria dos Grupos e em Quadrados Latinos que são melhores que os sistemas baseados em aritmética modular. Entretanto, parece existir uma barreira psicológica intransponível, pois não foi encontrada qualquer aplicação prática desses sistemas, embora alguns deles sejam bastante antigos. A hipótese dessa barreira é que guiou o estudo das diversas melhorias nos métodos de aritmética modular, todas elas de implementação muito simples.

Em se tratando de identificadores apenas com dígitos numéricos, pode-se constatar a superioridade do esquema de Verhoeff nas situações em que se utiliza apenas 1 dígito verificador, inclusive no método módulo 10 ótimo apresentado. Entretanto, este último método tem a vantagem de poder ser implementado de forma mais fácil.

Nos casos em que se podem utilizar dois dígitos verificadores, a alternativa apresentada neste trabalho, apesar de ser de grande simplicidade, é superior a todos os casos estudados.

Esperamos que as análises e propostas aqui apresentadas possam contribuir para que novas criações de dígitos verificadores parem de se espelhar no passado e utilizem as técnicas melhores existentes hoje pouco consideradas.

Uma última observação é que, de maneira geral, a situação no restante do mundo é análoga à do Brasil. Mostramos pelo menos três exemplos onde o uso de esquemas subotimizados ocorre: código de barras de produtos, passaporte e numeração internacional de livros (ISBN), este último estabelecido recentemente.

Para comodidade do leitor, encontram-se disponíveis na página `http://magnum.ime.uerj.br/~pauloedp/MEST/TESES/NATALIA/Pagina.html`, três programas Pascal que implementam os três principais algoritmos apresentados neste trabalho, a saber:

1. ALG8.pas: implementação do Algoritmo 8 para obter sistemas módulo 10 equivalentes ao de Verhoeff.

2. AL9.pas: implementação do Algoritmo 9 para obter sistemas módulo 10 ótimos com três permutações.

3. ALG11.pas: implementação do Algoritmo 11 para obter sistemas modulares ótimos de base prima.

REFERÊNCIAS

[1] ARGENTA, D.; AMORIM, R. *Estudo e Implementação de Dígitos Verificadores.* 2012. 52 f. Monografia (Graduação em Ciência da Computação). Universidade do Estado do Rio de Janeiro, 2012.

[2] BELYAVSKAYA, G. B.; ISBASH V. I.; SHCHERBACOV, V. A. Check character systems over quasigroups and loops. *Quasigroups and related systems*, v. 10, n. 10, p 1-28, 2003.

[3] BELYAVSKAYA, G. B.; ISBASH V. I.; MULLEN, G. L. Check Character Systems Using Quasigroups: I. *Des. Codes Cryptography*, v. 37, n.2, p. 215-227, 2004.

[4] BELYAVSKAYA, G. B.; ISBASH V. I.; MULLEN, G. L. Check Character Systems Using Quasigroups: II. *Des. Codes Cryptography*, v. 37, n.3, p. 405-419, 2005.

[5] BLACK, W. L. Error detection in decimal numbers. *Proc. IEEE* v. 60, n.3, p. 331-332, 1972.

[6] BROECKER, C.; SCHULZ, R-H.; STROTH, G. Check Character Systems Using Chevalley Groups. *Designs, Codes and Cryptography*, v. 10, n.2, p. 137-143, 1997.

[7] DAMM, H. M. Check digit systems over groups and anti-symmetric mappings. *Archiv Der Mathematik*, v. 75, n.6, p. 413-421, 2000.

[8] DAMM, H. M. On the Existence of Totally Anti-symmetric Quasigroups of Order $4k + 2$. *Computing*, v. 70, n. 4, p. 349-357, 2003.

[9] DAMM, H. M. *Total anti-symmetrische Quasigruppen.* 125 f. Tese (Doutorado em Ciências Naturais) - Philipps-Universität Marburg, Marburg/Lahn, 2004.

[10] DAMM, H. M. Totally anti-symmetric quasigroups for all orders $n \neq 2, 6$. *Discrete mathematics*, v. 307, n. 6, p. 715–729 , 2007.

[11] ECKER, A.; POCH, G. Check Character Systems. *Computing*, v. 37, n. 4, p. 277–301, 1984.

[12] GALLIAN, J. A. *Contemporary Abstract Algebra*, 8th ed., Cengage Learning, 2012.

[13] GALLIAN, J. A. Error detection methods. *ACM Computing Surveys (CSUR)*, v. 28, n. 3, p. 504-517, 1996.

[14] GALLIAN, J. A. *Contemporary Abstract Algebra*. 70. ed. Estados Unidos: Brooks/Cole, 1974.

[15] GALLIAN, J. A.; MULLIN, M. D. Groups with anti-symmetric mappings. *Arch Math*, v. 65, n. 4, p. 273-280, 1995.

[16] GALLIAN, J. A. WINTERS, S. Modular Arithmetic in the Marketplace. *The American Mathematical Monthly*, v. 95, n. 6, p. 548-551, 1988.

[17] GUMM, P. A new class of check digit methods for arbitrary number systems. *Information Theory, IEEE Transactions on* v. 31, n. 1, p. 102-105, 1985.

[18] GUMM, P. Encoding of numbers to detect typing errors. *Inter. J. Applied Eng. Educ.*, v. 2, n. 1, p. 61-65, 1986.

[19] JIANNAN, C. The NP-completeness of Completing Partial anti-symmetric Latin squares. *Computer School, National University of Defense Technology*, Changsha, China, 2009.

[20] KIRTLAND, J. *Identification Numbers and Check Digit Systems*. 1. ed. Estados Unidos: The Mathematical Association of America, 2001.

[21] LUHN, H. P. *Computer for verifying numbers*. US n. 2.950.048, 6 January 1954, 23 August 1960.

[22] MALEK, M. *Coding Theory*, Decimal Code. Disponível em `http://www.mcs.csueastbay.edu/ malek/TeX/Decimal.pdf`, 2009.

[23] PUTTER, P. WAGNER, N. R. Error detecting decimal digits. *Communications of the ACM*, v. 32, n. 1, p. 106-110, 1989.

[24] SATORU, A.; YASUSHI, M.; MASAHIRO, U. Error correction and error detection techniques for wireless ATM systems. *Wirel. Netw.*, v. 3, n. 4, p. 285–290, 1997.

[25] SCHULZ, R.-H. Check Character Systems and Anti-symmetric Mappings. *Computational Discrete Mathematics, Advanced Lectures*, p. 136-147, 2011.

[26] SCHULZ, R.-H. Check Character Systems Using Chevalley Groups. *Designs, Codes and Cryptography*, v. 10, n. 2, p. 137-143, 1997.

[27] SCHULZ, R.-H. Equivalence of Check Digit Systems Over the Dicyclic Groups of Order 8 and 12. *Mathematikdidaktik aus Begeisterung fur die Mathematik*, p. 227-237, 1999.

[28] SCHULZ, R.-H. On Check Digit Systems using Antisymmetric Mappings. *Mathematik und Informatik, Freie Universät Berlin*, 1999.

[29] SELMER, E.S. Registration Number in Norway: Some Applied Number Theory and Psychology. *Journal of the Royal Statistical Society, Series A (General)*, v. 130, n. 2, p. 225-231, 1967.

[30] VERHOEFF, J. *Error Detecting Decimal Codes*. 122 f. Tese, Mathematical Centre Tract 29, The Mathematical Centre, 1969.

[31] WOOD, E. F. Self-Checking Codes - An Application of Modular Arithmetic. *The Mathematics Teacher*, v. 80, n. 4, p. 312-316, 1987.

ANEXO A – Detecção de erros para Módulo 11 com dois dígitos verificadores

CNPJ

Detecção de erros individualizados

No caso do CNPJ, os erros individualizados ocorrem quando o cálculo inicial para ambos os dígitos resulta em 0 ou 10 e a troca de um dígito por outro em dada posição, altera o cálculo para 10 ou 0, respectivamente. Vamos calcular a probabilidade disto ocorrer, lembrando que se a troca do dígito a por um dígito b transforma ambos os cálculos dos DVs de 0 para 10, por exemplo, a troca deste dígito b pelo a transforma ambos os DVs de 10 para 0, portanto só é necessário analisar um dos casos.

1. Troca de dígito em posição diferente de 5, onde o 1º e o 2º DV são alterados de 0 para 10.

   Neste caso, devem ser satisfeitas as seguintes equações:

$$\begin{cases} pa + x \equiv 0 \bmod 11 & \text{e} \quad pb + x \equiv 10 \bmod 11 \\ (p-1)a + y \equiv 0 \bmod 11 & \text{e} \quad (p-1)b + y \equiv 10 \bmod 11, \end{cases}$$

   logo, $p(a-b) \equiv -10 \bmod 11$ $(p-1)(a-b) \equiv -10 \bmod 11$.

   Essa situação não existe, pois o sistema nunca é satisfeito.

2. Troca de dígito em posição diferente de 5, onde o 1º DV é alterado de 10 para 0 e o 2º DV de 0 para 10.

   Devem ser satisfeitas as equações:

$$\begin{cases} p(a-b) \equiv 10 \bmod 11 & \Rightarrow a - b \equiv 10p - 1 \bmod 11 \\ (p-1)(a-b) \equiv -10 \bmod 11 & \Rightarrow a - b \equiv (p-1) - 1 \bmod 11. \end{cases}$$

   ou seja, $10p - 1 \equiv (p-1) - 1 \bmod 11 \Rightarrow p = 6$.

   Este caso só pode ocorrer quando o peso no cálculo do 1º DV é 6. Portanto, devemos ter $a - b = 9 \bmod 11$.

   Não são detectados então, 9 pares (a, b) nesta situação. Claramente, nas situações em que altera-se o 1º DV de 0 para 10 e o 2º DV de 10 para 0, há também 9 pares

possíveis.

A probabilidade de não detecção é o produto das probabilidades de o 1º DV ser 10, o 2º DV ser 0, o dígito a ser trocado estar na posição 1 ou 9 e ser um dos 9 pares citados e que o dígito seja trocado por outro com 9 unidades a mais. Multiplica-se ainda essa probabilidade por 2, devido à situação contrária. Portanto a probabilidade aqui é dada por $2 \cdot (1/11) \cdot (1/11) \cdot (2/12) \cdot (9/10) \cdot (1/9)$.

3. Troca de dígito na posição 5, onde o 1º e o 2º DV são alterados de 0 para 10.

   Devem ser satisfeitas as equações:

$$\begin{cases} 2a + x \equiv 0 \bmod 11 \quad \text{e} \quad 2b + x \equiv 10 \bmod 11 \\ 9a + y \equiv 0 \bmod 11 \quad \text{e} \quad 9b + y \equiv 10 \bmod 11, \end{cases}$$

   de modo equivalente,

$$\begin{cases} 2(a - b) \equiv -10 \bmod 11 \\ 9(a - b) \equiv -10 \bmod 11. \end{cases}$$

   Portanto esta situação nunca ocorre.

4. Troca de dígito na posição 5, onde o 1º DV é alterado de 10 para 0 e o 2º DV de 0 para 10.

   Devem ser satisfeitas as equações:

$$\begin{cases} 2(a - b) \equiv -10 \bmod 11 \\ 9(a - b) \equiv 10 \bmod 11. \end{cases}$$

   De forma equivalente, $a - b \equiv 6 \bmod 11$.

   Não são detectados então, 9 pares (a, b) nesta situação. Claramente, nas situações em que altera-se o 1º DV de 0 para 10 e o 2º DV de 10 para 0, há também 9 pares possíveis. A probabilidade de não detecção é dada por $(1/11) \cdot (1/11) \cdot (1/12) \cdot (9/10) \cdot (1/9) \cdot 2$.

Somando todas as parcelas de todos os itens, temos a probabilidade total de $6/(10 \cdot 11^2 \cdot 12) = 0,04\%$.

Detecção de transposição de dígitos adjacentes ou alternados

Os erros de transposição de dígitos adjacentes ou alternados ocorrem quando o cálculo inicial para ambos os dígitos resulta em 0 ou 10 e a troca de dígitos adjacentes altera o cálculo para 10 ou 0, respectivamente. Os casos em que esta situação ocorre são os seguintes:

1. Transposição de dígitos em posições adjacentes diferentes de (4,5) e (5,6), onde o 1º e o 2º DV são alterados de 0 para 10:

   Caso o 1º dígito verificador seja 0 e a transposição transforme o cálculo para 10, as seguintes equações são satisfeitas:

   $$\begin{cases} pa_i + (p+1)a_{i+1} + x \equiv 0 \bmod 11 \\ pa_{i+1} + (p+1)a_i + x \equiv 10 \bmod 11 \end{cases}$$

   que é equivalente a $-a_i + a_{i+1} \equiv -10 \bmod 11$. Sendo o 2º dígito verificador também 0, temos $(p-1)a_i + pa_{i+1} + x \equiv 0 \bmod 11$

   e com isso, $(p-1)a_{i+1} + pa_i + x \equiv a_i - a_{i+1} = 10 \bmod 11$.

   Logo, nesse caso, 100% dos erros não são detectados.

2. Transposição de dígitos em posições adjacentes diferentes de (4,5) e (5,6), onde o 1º DV é alterado de 0 para 10 e o 2º DV de 10 para 0.

   Neste caso temos o seguinte sistema de equações:

   $$\begin{cases} pa_i + (p+1)a_{i+1} + x \equiv 0 \bmod 11 \quad \text{e} \quad pa_{i+1} + (p+1)a_i + x \equiv 10 \bmod 11 \\ (p-1)a_i + pa_{i+1} + x \equiv 10 \bmod 11 \quad \text{e} \quad (p-1)a_{i+1} + pa_i + x \equiv 0 \bmod 11, \end{cases}$$

   que é equivalente a:

   $$\begin{cases} -a_i + a_{i+1} \equiv -10 \bmod 11 \\ -a_i + a_{i+1} \equiv 10 \bmod 11, \end{cases}$$

   o que é impossível para $a_i \neq a_{i+1}$.

3. Transposição de dígitos nas posições (4,5).

Para que o erro não fosse detectado nesses casos, deveria se ter:

$$\begin{cases} 9a_4 + 2a_5 + x \equiv 0 \bmod 11 \quad \text{e} \quad 9a_5 + 2a_4 + x \equiv 10 \bmod 11 \\ 8a_4 + 9a_5 + y \equiv 0 \bmod 11 \quad \text{e} \quad 8a_5 + 9a_4 + y \equiv 10 \bmod 11 \end{cases}$$

e, de forma equivalente,

$$\begin{cases} 7a_4 - 7a_5 \equiv -10 \bmod 11 \Rightarrow a_4 - a_5 \equiv 8 \bmod 11 \\ -a_4 + a_5 \equiv -10 \bmod 11 \Rightarrow a_4 - a_5 \equiv 10 \bmod 11, \end{cases}$$

o que é impossível.

Pode ser provado de maneira análoga que os outros casos para essas posições também são impossíveis.

4. Transposição de dígitos nas posições (5,6).

   Os erros são sempre detectados, da mesma forma que no caso anterior.

Logo a probabilidade de que um erro de transposição adjacente não seja detectado é de $(9/11) \cdot (1/11) \cdot (1/11) \cdot (1/10) \cdot 2 = 0,14\%$.

O caso das transposições de dígitos adjacentes é análogo.

Detecção de erros gêmeos adjacentes

Como já visto, para que sejam detectados os erros gêmeos adjacentes, correspondendo à troca do dígito a para b, com $a \neq b$, devemos ter

$$(a - b)(p_i + p_{i+1}) \not\equiv 0 \bmod 11.$$

No caso do CNPJ, não garantimos que a equação seja satisfeita, pois temos os pesos (5, 6) e (9, 2) adjacentes.

Os erros não são detectados então, quando ocorrem em posições correspondentes aos pesos (5,6) ou (9,2) na equação do cálculo de um DV e alteram o cálculo do outro DV de 0 para 10 ou vice-versa. Num total de 11 colunas adjacentes, no 1º DV existem 2 pares com esses pesos. E no 2º DV, 3 pares. Logo, a probabilidade é dada por $[(2/11) + (3/11)] \cdot (2/11) \cdot (1/10) = 0,83\%$.

De maneira análoga às anteriores, pode-se provar que não existem casos em que a troca de dígitos em posições não correspondentes aos pesos (5,6) e (9,2) no cálculo dos DVs que alteram o cálculo de ambos de 0 ou 10 para 10 ou 0.

Detecção de erros gêmeos alternados

Novamente vamos dividir os casos.

1. Posições diferentes de (3,5), (4,6) e (5,7), onde o 1º e o 2º DV são alterados de 0 para 10.

   As seguintes equações deveriam ser satisfeitas:

$$\begin{cases} pa + (p+2)a + x \equiv 0 \bmod 11 & \text{e} \quad pb + (p+2)b + x \equiv 10 \bmod 11 \\ (p-1)a + (p+1)a + y \equiv 0 \bmod 11 & \text{e} \quad (p-1)b + (p+1)b + y \equiv 10 \bmod 11. \end{cases}$$

   De forma equivalente,

$$\begin{cases} (2p+2)(a-b) \equiv -10 \bmod 11 \\ 2p(a-b) \equiv -10 \bmod 11. \end{cases}$$

   Como as equações não podem ser satisfeitas, todos os erros são detectados.

2. Posições diferentes de (3,5), (4,6) e (5,7), onde o 1º DV é alterado de 0 para 10 e o 2º DV de 10 para 0.

   As seguintes equações deveriam ser satisfeitas:

$$\begin{cases} pa + (p+2)a + x \equiv 0 \bmod 11 & \text{e} \quad pb + (p+2)b + x \equiv 10 \bmod 11 \\ (p-1)a + (p+1)a + y \equiv 10 \bmod 11 & \text{e} \quad (p-1)b + (p+1)b + y \equiv 0 \bmod 11. \end{cases}$$

   De forma equivalente,

$$\begin{cases} (2p+2)(a-b) \equiv -10 \bmod 11 \\ 2p(a-b) \equiv 10 \bmod 11. \end{cases} \Rightarrow a - b \equiv 1 \bmod 11.$$

   Voltando a última equação, tem-se $2p \equiv 10 \bmod 11 \Rightarrow p \equiv 5 \bmod 11$. Portanto os

erros não são detectados apenas nas posições (8,10). E a probabilidade é dada por $2 \cdot (1/10) \cdot (1/11) \cdot (1/11) \cdot (1/10) = 0,017\%$.

3. Posições (4,6), onde o 1º DV é alterado de 0 para 10 e o 2º DV de 10 para 0.

   As seguintes equações deveriam ser satisfeitas:

$$\begin{cases} 8a + 2a + x \equiv 0 \bmod 11 & \text{e} \quad 8b + 2b + x \equiv 10 \bmod 11 \\ 7a + 9a + y \equiv 10 \bmod 11 & \text{e} \quad 7b + 9b + y \equiv 0 \bmod 11. \end{cases}$$

   De forma equivalente,

$$\begin{cases} 10(a - b) \equiv -10 \bmod 11 \\ a - b \equiv 10 \bmod 11 \end{cases} \Rightarrow 10(a - b) \equiv -10 \bmod 11.$$

   Essas equações são sempre satisfeitas e, portanto, nenhum erro é detectado. A probabilidade de não detecção desses erros é dada por $2 \cdot (1/10) \cdot (1/11) \cdot (1/11) \cdot (1/10) = 0,017\%$.

Em nenhum dos outros casos os erros deixam de ser detectados. Acumulando-se os casos, a probabilidade é dada por $2 \cdot 0,017 = 0,03\%$.

CPF

Detecção de erros individualizados

O cálculo dos dígitos verificadores do CPF é análogo ao do CNPJ, mas há uma diferença crucial que leva a uma situação de pior detecção de erros. Trata-se do uso do peso 0 no cálculo do segundo dígito verificador (o que é equivalente a dizer que o primeiro dígito do identificador não é considerado no cálculo). Desta forma, temos que distinguir os erros do primeiro dígito para os demais. Os erros no primeiro dígito ocorrem quando o cálculo inicial é 0 ou 10 e, depois da troca, passa para 10 ou 0, respectivamente. O erro no primeiro dígito corresponde a 1/9 dos casos de erro; o resultado 0 ou 10 é obtido em 2/11 dos casos e a troca citada em 1/10 dos casos. A não detecção de erros no primeiro dígito é de $(1/9) \cdot (2/11) \cdot (1/10)$ dos casos, equivalente a 0,2% dos casos de erro.

O cálculo para os outros dígitos é o mesmo do caso 2 dos erros individualizados do CNPJ, ou seja, os erros não são detectados quando o peso no cálculo do 1º DV é 6, o que ocorre apenas uma vez no CPF. Portanto, os erros não detectados nesta situação

equivalem a um percentual de $2 \cdot (1/9) \cdot (1/11) \cdot (1/10) \cdot (1/11)$, ou seja, 0,02%. O total de não detecção de erros é, portanto, 0,22%.

Detecção de erros de transposição adjacente e alternada

Aqui, o cálculo para determinar a probabilidade de não detecção é o mesmo que no caso 2 dos erros de tranposições do CNPJ, exceto para as transposições nas posições (1,2). A análise deste caso é a que segue:

1. A transposição altera o 1º e o 2º DV de 0 para 10.

   As seguintes equações deveriam ser satisfeitas:

$$\begin{cases} a_i + 2a_{i+1} + x \equiv 0 \bmod 11 & \text{e} \quad a_{i+1} + 2a_i + x \equiv 10 \bmod 11 \\ \Rightarrow a_i - a_{i+1} \equiv 10 \bmod 11 & \\ a_{i+1} + y \equiv 0 \bmod 11 & \text{e} \quad a_i + y \equiv 10 \bmod 11 \\ \Rightarrow a_i - a_{i+1} \equiv -10 \bmod 11. & \end{cases}$$

   Portanto todos os erros são detectados.

2. A transposição altera o 1º DV de 0 para 10 e o 2º DV de 10 para 0.

   As seguintes equações deveriam ser satisfeitas:

$$\begin{cases} a_i + 2a_{i+1} + x \equiv 0 \bmod 11 & \text{e} \quad a_{i+1} + 2a_i + x \equiv 10 \bmod 11 \\ \Rightarrow a_i - a_{i+1} \equiv 10 \bmod 11 & \\ a_{i+1} + y \equiv 10 \bmod 11 & \text{e} \quad a_i + y \equiv 0 \bmod 11 \\ \Rightarrow a_i - a_{i+1} \equiv 10 \bmod 11. & \end{cases}$$

   Portanto nenhum erro é detectado.

   A probabilidade total é dada por $(7/8 + 1/8) \cdot 2 \cdot (1/11) \cdot (1/11) \cdot (1/10) = 0,17\%$.

Detecção de gêmeos

A situação é análoga ao da detecção de erros gêmeos do CNPJ. Sendo os erros não detectados os que ocorrem nas posições correspondentes aos pesos (5,6) no cálculo de um

DV e alteram o cálculo do outro DV de 0 para 10 ou vice-versa. Num total de 8 colunas adjacentes, existe um par de pesos (5,6) em cada cálculo de DV. Logo, a probabilidade é dada por $2 \cdot (1/8) \cdot (2/11) \cdot (1/10) = 0,45\%$.

Pode ser mostrado que os erros nas posições não correspondentes aos pesos (5,6) no cálculo dos DVs e que alteram o cálculo de ambos de 0 ou 10 para 10 ou 0 são sempre detectados.

Detecção de gêmeos alternados

O caso é análogo ao do CNPJ, sendo os erros não detectados apenas nas posições correspondentes a (5,7), portanto a probabilidade é dada por $2 \cdot (1/9) \cdot (1/11) \cdot (1/11) \cdot (1/10) = 0,02\%$.

CNH e Título Eleitoral

Para os demais casos de dois dígitos verificadores considerados, ou seja, o CNH e o Título Eleitoral, as análises são análogas àquelas feitas para o CNPJ e CPF e os resultados são dados na Tabela 17 da próxima seção.

ANEXO B – Distribuição de erros gêmeos para o esquema Módulo 10 Ótimo

Seja S o conjunto das 12.654.000 triplas que deixam de detectar 2 erros transposição. A Tabela 18 mostra a quantidade de pares de permutações de S que deixam de detectar uma quantidade k de erros gêmeos.

Tabela 18 - Quantidade de Erros Gêmeos.

k	Nº de pares
16	132000
18	156000
20	372000
22	636000
24	1440000
26	1296000
28	1464000
30	1386000
32	1338000
34	1134000
36	1080000
38	720000
40	360000
42	276000
44	312000
46	234000
48	180000
50	78000
52	36000
56	12000
58	12000

ANEXO C – Média de não-detecção para o Módulo 10 Ótimo

No esquema ótimo módulo 10 proposto nesta dissertação, a taxa de não detecção de erros gêmeos e gêmeos alternados depende das posições em que estes erros ocorrem.

Vamos analisar a taxa de não deteção quando se utilizam triplas de permutações que deixam de detectar 2, 8 e 6 erros gêmeos a cada duas permutações utilizadas e consecutivamente, 6, 2 e 8 erros gêmeos alternados a cada três permutações utilizadas.

Se para o cálculo do dígito verificador são utilizadas $i+1$ permutações, então as médias de não detecção de erros são dadas a seguir:

1. para $i = 3k$:

$$\mu_0 = 0,11 \cdot \frac{2}{90} + 0,005 \cdot \frac{16}{270} + 0,003 \cdot \frac{16(i/3-1)+8}{90(i-1)}. \tag{39}$$

2. para $i = 3k+1$:

$$\mu_1 = 0,11 \cdot \frac{2}{90} + 0,005 \cdot \frac{16\lfloor i/3 \rfloor + 2}{90i} + 0,003 \cdot \frac{16\lfloor i/3 \rfloor}{90(i-1)}. \tag{40}$$

3. para $i = 3k+2$:

$$\mu_2 = 0,11 \cdot \frac{2}{90} + 0,005 \cdot \frac{16\lfloor i/3 \rfloor + 10}{90i} + 0,003 \cdot \frac{16(\lfloor i/3 \rfloor + 6)+8}{90(i-1)}. \tag{41}$$

Pode se comprovar que μ_0, μ_1, μ_2 é sempre menor que

$$\mu = 0,11 \cdot \frac{2}{90} + 0,008 \cdot \frac{16}{270}. \tag{42}$$

Portanto, utilizamos esta taxa de não detecção μ na demonstração da otimalidade do módulo 10 ótimo.

No esquema módulo 10 de Verhoeff, a média de não detecção também depende do tamanho do identificador. A cada 7 permutações utilizadas, um par deixa de detectar 2 entre 8 erros fonéticos. Portanto se são utilizadas i permutações, a média de não detecção é dada por:

$$\mu = 0,102 \cdot \frac{2}{90} + 0,008 \cdot \frac{4}{90} + 0,005 \cdot \frac{2}{90} + 0,003 \cdot \frac{4}{90} + 0,005 \cdot \frac{\lfloor i/7 \rfloor}{(i-1)8}. \tag{43}$$

Toda permutação circular das triplas encontradas no esquema ótimo podem ser utilizadas no cálculo do dígito verificador. A Tabela 19 mostra a média de não detecção com a utilização da melhor ordem das permutações para identificadores de tamanho até 14 do esquema proposto e do Verhoeff.

Tabela 19 - Média de erros não detectados.

Nº de permutações usadas	**Proposto**	**Verhoeff**
2	0,237777	0,237777
3	0,290000	0,286666
4	0,287407	0,286666
5	0,287222	0,286666
6	0,291111	0,286666
7	0,290074	0,297083
8	0,289206	0,295595
9	0,291270	0,294479
10	0,290741	0,293611
11	0,290000	0,292917
12	0,291394	0,292348
13	0,291044	0,297083