Busca de Matrizes MDS de Baixo Custo para Implementações de Algoritmos Criptográficos

Tópicos em Otimização Combinatória

Equipe 8

Giovana Kerche Bonás (RA:216832) e
Henrique Finger Zimerman (RA:217771)

14 de Junho, 2024

MO824

1 Resumo

Neste trabalho, focamos na busca por matrizes Maximum Distance Separable (MDS) de baixo custo computacional, vitais para algoritmos criptográficos como o AES (FIPS, 2001), estabelecido como padrão pelo NIST em 2001. Essas matrizes asseguram que pequenas mudanças no texto de entrada causem grandes alterações no texto cifrado, conceito chamado de "difusão"proposto por Claude Shannon em 1949, fundamental para a segurança de algoritmos criptográficos. (Shannon, 1949). Um trabalho similar de busca por matrizes MDS foi realizado por Serpa, 2023, nosso trabalho o expande.

O principal desafio na busca de matrizes MDS é a alta complexidade computacional na geração e validação da propriedade, pois requer que todas as submatrizes sejam invertíveis, o que se torna computacionalmente custoso à medida que as dimensões da matriz aumentam, já que o número de submatrizes cresce exponencialmente. Numa tentativa de encontrar matrizes MDS de custos baixos, aplicamos o algoritmo genético, analisando a compatibilidade entre o método e o problema e criando uma função-objetivo que permite a representação do problema no contexto de otimização ocombinatória. A função-objetivo desenvolvida mescla custos computacionais e proximidade com a condição MDS, facilitando a seleção e evolução de matrizes eficientes e adequadas para algoritmos criptográficos. Para fins de comparação, foi também implementado um algoritmo de busca local que se utiliza da mesma função-objetivo. Foi notado que o desempenho da busca local foi igual ou melhor ao desempenho do algoritmo genético para todas as dimensões testadas. Porém, o desempenho da função-objetivo foi satisfatório e levou a resultados superiores aos obtidos anteriormente por Serpa, 2023.

O trabalho desenvolvido permite uma busca eficaz por matrizes que atendam ou se aproximem dos critérios MDS, maximizando a segurança e diminuindo a carga computacional, contribuindo assim para avanços neste campo da criptografia.

Palavras-chave: Matriz MDS, otimização, algoritmo genético, criptografia, busca local.

2 Introdução

As matrizes Maximum Distance Separable (MDS) constituem uma classe especial de matrizes amplamente utilizadas na criptografia e na teoria de códigos corretores de erros. Sua principal característica é maximizar a distância mínima de Hamming entre as linhas, fazendo com que essas matrizes sejam extremamente eficazes na difusão de alterações nos dados de entrada. Isso é essencial para aumentar a segurança em cifras de bloco, onde pequenas alterações em um bloco de texto não devem apenas alterar significativamente o texto cifrado,

mas também se distribuir de maneira uniforme para aumentar a imprevisibilidade do texto cifrado. Alguns trabalhos que exploram essa característica são SHARK Rijmen et al., 1996, SQUARE Daemen et al., 1997, BKSQ Quisquater e Schneier, 2006, KHAZAD P. Barreto e Rijmen, 2000, ANUBIS P. S. Barreto, 2000, Hierocrypt-3 Ohkuma et al., 2000, Rijndael (AES) Daemen e Rijmen, 2002 e Curupira P. Barreto e Simplicio, 2007.

A propriedade definidora de uma matriz ser MDS é que todas as suas submatrizes possíveis devem ser invertíveis, isto é, o determinante de qualquer submatriz extraída da matriz principal não pode ser zero. Esta propriedade assegura que a matriz possa atingir a capacidade máxima de correção de erros permitida pela distância de Hamming, tornando-a uma ferramenta para a codificação de informações onde a integridade e a correção de erros são cruciais. No entanto, determinar se uma matriz é MDS envolve avaliar a não-singularidade de todas as submatrizes que podem ser formadas, o que apresenta uma complexidade combinatória significativa. À medida que o tamanho da matriz aumenta, o número de submatrizes cresce exponencialmente, tornando o processo computacionalmente intensivo. Isso implica que, para matrizes de grandes dimensões, os métodos convencionais de verificação podem se tornar impraticáveis, pois encontrar matrizes MDS revela uma complexidade combinatória significativa. Para uma matriz de tamanho $n \times n$, não estamos apenas considerando a matriz completa, mas também todas as combinações possíveis resultantes da remoção de x linhas e x colunas, onde x varia de 0 até *n-1*.

Esse problema tem sido abordado na literatura de diferentes formas. Em Roth e Lempel, 1989 foi introduzida uma abordagem eficiente para a construção de códigos MDS através de matrizes de Cauchy. Esse método reduz as demandas computacionais tipicamente associadas às matrizes MDS, pois a invertibilidade de submatrizes é intrínseca às matrizes de Cauchy. Além disso, em Serpa, 2023 é introduzida uma metodologia que encontra matrizes MDS utilizando algorítmos genéticos. Esses trabalhos estabeleceram fundamentos sólidos ao integrar preocupações em relação a eficiência computacional na obtenção de matrizes MDS, porém ambos utilizam da propriedade MDS de forma binária: uma matriz é ou não é MDS.

O projeto proposto expande essas ideias ao usar algoritmos genéticos, introduzindo uma medida quantitativa do quão MDS uma matriz é, avaliando o "grau de MDS" de uma matriz.

2.1 Formulação matemática

Consideramos uma matriz A de tamanho $n \times n$ com elementos no corpo finito GF(2^m). Um corpo de Galois, ou corpo finito GF(2^m), é uma estrutura algébrica onde as operações de adição, subtração, multiplicação e divisão (exceto por zero) são possíveis e fechadas dentro do conjunto de elementos. Cada elemento a_{ij} da matriz A é uma variável de decisão. O objetivo

é configurar essas variáveis de forma a encontrar matrizes que satisfaçam a propriedade MDS e minimizar o custo computacional associado.

Variáveis de decisão:

- a_{ij}, $1 \leq i, j \leq n$: Elementos da matriz A.

Função objetivo: Um dos principais objetivos do projeto é encontrar uma função objetivo que favorece o surgimento de matrizes MDS eficientes. Dessa forma, múltiplas funções objetivos foram propostas, sendo as seguintes as que se mostraram mais promissoras:

- **Formulação 1:** Minimizar c e d, onde c representa o custo computacional total, composto por operações XOR e XTIME (explicados em 3.3.2), d é a proporção de submatrizes com determinante nulos de uma matriz, representando a distância de A para ser uma matriz MDS. Ambos c e d serão explicados na seção 3.3. k é uma constante positiva.

$$\mathrm{f} = \frac{k}{c \cdot (d+1)}$$

  Nesta configuração, valores elevados do parâmetro d (matrizes que se afastam da característica MDS) e valores elevados de c (custo computacional associado) são prejudicados na minimização da função. No entanto, o impacto da variação de d e c são equilibrados linearmente, não oferecendo um bom gradiente na escolha das matrizes. Portanto apesar de ter sido inicialmente testada, essa formulação foi deixada de lado por não dar resultados experimentais interessantes.

- **Formulação 2:** Minimizar c e d de maneira similar à anterior. k_1, k_2 e k_3 são constantes positivas, com $k_1 \geq k_2$.

$$\mathrm{f} = \begin{cases} k_1 / (1 + c + k_3 \cdot d) & d > 0 \\ k_2 / (1 + c) & d = 0 \end{cases}$$

  Essa formulação, apesar de muito similar a anterior, possui o diferencial de possibilitar uma maior flexibilização com o fator multiplicativo k_3, permitindo ponderar a importância dada ao valor de d na minimização. Além disso, como $k_2 \geq k_1$, as matrizes MDS são mais bonificadas nessa configuração. Essa formulação apresentou melhores resultados e foi utilizada extensivamente no desenvolvimento do trabalho.

3 Metodologia

Este trabalho explorou a aplicação de algoritmos genéticos no contexto da busca de matrizes MDS de baixo custo, focando especificamente em melhorar seu desempenho computacional para aplicações em criptografia. A metodologia adotada envolve diversas etapas de diversificação e intensificação dessa busca, especificadas nas seções abaixo.

Inicialmente foram testados métodos que geram a matriz completa através do algoritmo genético, chamada de *Configuração 1* nos resultados. Porém, após alguns experimentos, ficou claro que gerar matrizes circulantes à esquerda tornava o problema mais simples e resultava em matrizes MDS de custo menor. Uma matriz circulante é uma matrix onde cada linha é idêntica à linha anterior, mas rotacionada em uma posição. Neste trabalho, optamos por utilizar matrizes circulantes, tendo como padrão circulantes à esquerda. Essa configuração é a *Configuração 2*.

Ao utilizar matrizes circulantes, foi possível reduzir o cromossomo a apenas a primeira linha de cada matriz. Isso apresenta múltiplas vantagens, como uma facilidade maior ao realizar crossovers, mutações e passos de busca local.

Estes resultados empíricos são compatíveis com as tentativas ao longo do histórico de implementações de algoritmos criptográficos, onde diversos algoritmos, sendo o mais famoso o AES, empregam matrizes circulantes. É possível que construir matrizes MDS se torne mais fácil ao usar matrizes circulantes, por exemplo, por diminuir a quantidade de elementos que precisam ser escolhidos. Além disso, no catálogo de Serpa, 2023, é possível observar uma quantidade significativa desse tipo de matriz, sendo as mais antigas circulantes à direita e as mais recentes circulantes à esquerda. Há outras vantagens para o uso desse tipo de matriz, como por exemplo armazenar menos elementos na memória na implementação de um algoritmo criptográfico. Em outras palavras, a facilidade que vimos em nossos experimentos de encontrar matrizes MDS de baixo custo ao restringir o tipo a circulantes é compatível com a história e o uso delas presente na literatura.

Para manter a brevidade, as descrições de metodologia a seguir são respectivas à configuração de matrizes circulantes.

3.1 População inicial

Cada indivíduo na população será gerado de forma aleatória. No nosso contexto de matrizes, isso significa que cada elemento da primeira linha da matriz será preenchido com um valor aleatório dentro de um limite superior definido. Este limite será calculado com base no corpo de Galois (GF) em que a matriz será usada. Por exemplo, se o corpo de Galois é $\mathrm{GF}(2^m)$, o limite superior será de $2^m - 1$, significando que os elementos da matriz podem

variar de 0 a $2^m - 1$. No entanto, a presença de um valor 0 em uma matriz automaticamente a classifica como não-MDS. Assim, determinamos que nenhuma posição de nenhuma matriz, tanto na população inicial quanto após uma mutação, será 0.

Após a realização de testes preliminares, foi realizada uma segunda modificação, onde os valores iniciais devem também ser menores a um parâmetro s, que foi definido empiricamente como sendo 15. Essa mudança permite que o algoritmo encontre respostas de custo baixo mais rapidamente, uma vez que a função-objetivo se mostrou boa em encontrar matrizes MDS na maioria dos casos, sendo a maior preocupação então diminuir seus custos. O corpo de Galois escolhido para a realização dos experimentos foi GF(2^8).

3.2 Representação dos Cromossomos

A representação cromossômica desenvolvida utilizou a primeira linha da matriz como o cromossomo. Uma vez que optamos por utilizar matrizes circulantes à esquerda, a matriz completa é gerada a partir da primeira linha. Essas matrizes serão caracterizadas por coeficientes inteiros, junto com informações adicionais necessárias no desenvolvimento do método, como a verificação de se eles satisfazem a propriedade MDS e os custos associados às operações de XOR e XTIME.

3.3 Função-Objetivo

As funções-objetivo avaliam a adequação de soluções potenciais em um espaço de busca para o problema investigado, ajudando a guiar o processo de seleção e evolução das soluções. Na construção da nossa função-objetivo, levamos em consideração três importantes fatores na busca por matrizes MDS de baixo custo: a contagem de XTIME, a contagem de XOR e o quanto uma matriz está próxima de ser uma matriz MDS. Detalharemos os componentes a seguir.

3.3.1 Cálculo de d_k para quantificar propriedade MDS em matrizes

Uma matriz A de tamanho $n \times n$ é considerada MDS (Maximum Distance Separable) se todas as submatrizes quadradas de tamanho $k \times k$ têm determinante diferente de zero, onde $k \leq n$.

Definimos d como uma medida de quão longe uma matriz está de satisfazer a propriedade MDS. Ou seja, para quantificar o quão próxima uma matriz está de ser MDS, iremos contar quantas dessas submatrizes $k \times k$ não têm determinante zero. Sendo assim, uma matriz com $d = 0$ é MDS.

3.3.2 Cálculo da relação usada entre XTIME e XOR

No contexto de algoritmos criptográficos o custo computacional de operações matriciais em termos das operações elementares necessárias para realizá-las pode ser avaliado através de duas dessas operações fundamentais: XTIME e XOR.

XOR (Exclusive OR) é uma operação lógica que compara dois bits. O resultado é verdadeiro apenas se os bits comparados forem diferentes. Em termos de processamento, XOR é uma operação básica e leve.

XTIME é uma operação usada em algoritmos de criptografia que operam em corpos finitos. Esta operação é mais complexa e computacionalmente mais cara do que uma simples operação XOR, pois envolve mais lógica para manipulação de bits e aritmética de corpos finitos.

A equação do custo computacional usada pode ser expressa como:

$$c = 3 \cdot \text{XTIME} + 1 \cdot \text{XOR}$$

onde c representa o custo total, XTIME é a contagem de operações XTIME e XOR é a contagem de XOR. Essa relação foi escolhida por refletir a ideia de que a operação XTIME é mais custosa do que a operação XOR, inspirando-se em Satoh et al., 2001. Estima-se que uma operação XTIME tenha aproximadamente o mesmo custo que três operações XOR. Portanto, ao avaliar o custo total de uma matriz em termos de seu uso em algoritmos criptográficos, esse peso diferenciado ajuda a dar uma estimativa mais precisa do impacto computacional real. Esta mesma equação para o custo foi usada por Serpa, 2023, então usá-la permite uma comparação direta dos nossos resultados.

3.3.3 Função-Objetivo

Usando a formulação já estabelecida, a função-objetivo foi definida como:

$$\text{f} = \begin{cases} k_1 / \left(1 + c + k_3 \cdot d\right) & d > 0 \\ k_2 / \left(1 + c\right) & d = 0 \end{cases}$$

3.4 Método de seleção

O método de seleção privilegia as soluções mais aptas levando em consideração o valor do fitness. Quanto maior a aptidão, maior a chance de ser selecionado. Para o projeto, dois métodos de seleção serão avaliados.

3.4.1 Seleção por Torneio

Na seleção por torneio há uma seleção aleatória de um número fixo de indivíduos ("tamanho do torneio") da população para competir em um torneio. A cada torneio, o indivíduo com a maior aptidão (o vencedor) é selecionado como pai. Este processo é repetido até que o número desejado de pais seja alcançado.

3.5 Método de Crossover

O método de crossover lida com a combinação de elementos de duas matrizes ($parent_1$ e $parent_2$), para criar uma nova matriz *child*. A essência é fundir segmentos específicos de $parent_1$ e $parent_2$ para gerar uma matriz que herda características de ambas. Isso é feito por meio da escolha aleatória de dois pontos dentro das matrizes, que definem o intervalo de troca. Inicialmente, *child* é uma cópia exata de $parent_1$. Entre os dois pontos aleatórios escolhidos, os elementos de *child* são substituídos pelos elementos correspondentes de $parent_2$. Assim, a matriz resultante mantém a estrutura geral de $parent_1$, mas com uma seção intercalada diretamente de $parent_2$.

3.6 Métodos de Mutação

Foram testados dois métodos de mutação e um método de manutenção de variedade, descritos nas seções abaixo.

3.6.1 Substituição Aleatória

Neste método, o valor de uma posição aleatória do cromossomo é substituído por outro valor aleatório válido no corpo finito utilizado. Este método apresentou certas limitações, uma vez que o novo valor tinha uma chance alta de ter um valor muito acima dos demais, prejudicando muito o custo da matriz. Assim, a probabilidade de uma mutação boa surgir era muito baixa.

3.6.2 Método de Relaxação

Neste método, as mutações ocorrem de maneira mais limitada. Para cada posição do cromossomo, o mesmo tem 50% de chance de mudar, sendo essa mudança sempre um incremento ou decremento em 1, mantendo os valores dentro dos limites do corpo finito. Após essa etapa, duas posições do cromossomo são trocadas, permitindo assim que permutações das soluções sejam avaliadas e exploradas. Esse método foi utilizado mais extensivamente.

3.6.3 Manutenção de Variedade

Além dos métodos de mutação, foi empregado também um método de manutenção de variedade. Esse método tem como objetivo impedir que a população se torne muito homogênea. Para isso, foi necessário efetuar uma medida do quão variada está uma população.

A variedade da população é medida através das somas da quantidade de ocorrência da moda de cada posição do cromossomo. Por exemplo, se todos os elementos de cada posição dos cromossomos da população forem iguais, a medida f é máxima, sendo $f = p \cdot n$, onde p é o tamanho da população e n o tamanho do lado da matriz. Por outro lado, se todos os valores para cada posição dos cromossomos da população forem diferentes entre si, a medida f é mínima, sendo $f = n$, uma vez que , para cada posição, a moda aparece apenas uma vez.

Assim, o coeficiente de homogeneidade f é definido como:

$$h = \frac{f - n}{p \cdot n - n}$$

Onde $h = 0$ indica uma população completamente heterogênea e $h = 1$ uma população completamente homogênea.

A probabilidade de mutação é então definida a partir do coeficiente h da seguinte forma:

$$p_{mut} = p_{min} + (p_{max} - p_{min}) \cdot h^2$$

O coeficiente h é elevado ao quadrado de forma a limitar a probabilidade de mutação quando o sistema não é homogêneo, crescendo rapidamente quando a população se torna muito homogênea.

Os valores de p_{min} e p_{max} foram definidos empiricamente, onde $p_{min} = 0.05$ e $p_{max} = 0.7$ obteram um bom desempenho e foram utilizados amplamente.

3.7 Funções de Busca Local

Em conjunto com o algoritmo genético, foi utilizada uma implementação de busca local *first-improving* com algumas variações.

A ideia central do método de busca local é escolher uma posição de um cromossomo e substituir o valor por um valor menor, checando se a fitness do indivíduo aumenta com a substituição, e retornando o primeiro que melhora. Esse método também limita os valores testados para cada posição a um valor máximo de forma a diminuir a quantidade de operações.

Variações desse algoritmo incluem:

- *mini local search*: Apenas uma posição é testada, sendo essa escolhida de maneira aleatória ou a posição com o maior valor.
- *local search*: Todas as posições são testadas, com a ordem de testes podendo ser aleatória ou da posição com maior valor para a posição com menor valor. Se nenhuma melhoria for encontrada, o algoritmo faz uma permutação aleatória da solução original e tenta novamente uma vez.
- *full local search*: Chamadas repetidas ao *local search* até que não seja possível melhorar a fitness.

3.8 Estratégias Híbridas

Além das variações nos métodos de seleção, mutação e crossover, outras ideias foram testadas para criar um algoritmo genético mais bem adaptado ao problema, essas variações se diferenciam principalmente na presença e utilização da busca local.

3.8.1 Método Padrão

Este método foi testado inicialmente e é o mais simples, ele depende apenas dos mecanismos intrínsecos ao algoritmo genético para obter as soluções. No entanto, esse método apresentou a pior performance, encontrando matrizes de custo muito alto e tendo dificuldade em reduzir os mesmos, além de apresentar maiores tempos computacionais.

3.8.2 Método com Mutação Guiada na Solução Incumbente

Este método é uma evolução do anterior, onde uma função *diminish mutation* é aplicada à solução incumbente e adicionada à próxima geração. A função *diminish mutation* escolhe um subconjunto aleatório de posições do cromossomo e substitui os valores de cada posição por um valor aleatório menor válido.

O método obteve sucesso em diminuir os custos da população ao longo das gerações, porém com um nível alto de aleatoriedade, sem seguir um gradiente específico.

3.8.3 Método de Busca Local na Solução Incumbente

Neste método, a função de *local search* é aplicada à solução incumbente sempre que uma nova é obtida, incluindo o resultado na nova população. Esse método apresentou o melhor desempenho em matrizes de alta dimensão (7x7 e 8x8), pois requer proporcionalmente menos operações do que os outros métodos que também utilizam busca local. No entanto,

a melhora dos resultados é quase que completamente baseada nesse passo de busca local, sendo discutível o desempenho do componente de algoritmo genético.

3.8.4 Método de Mini Busca Local nos Filhos

Neste método, a solução incumbente não é necessariamente inclusa na nova população. A função *mini local search* é aplicada após a geração de cada nova criança, sendo o índice a ser testado aleatório. Este método se mostrou promissor em explorar soluções além do mínimo local encontrado pela busca local, porém a grande quantidade de passos simples de busca local executados significa que o mesmo não teve desempenho satisfatório para matrizes de alta dimensão.

3.8.5 Método de Busca Exaustiva na População Inicial

Este método, foi testado em conjunto com os dois anteriores, o conjunto inicial passa pelo processo de *full local search* após ser gerado, criando assim uma população inicial de mínimos locais.

3.8.6 Método de Busca Local Puro

Para fins de comparação, foi implementado um programa alternativo que apenas executa a função *full local search* numa população inicial aleatória, retornando o melhor indivíduo.

4 Avaliação dos Resultados

Para avaliar a eficácia da metodologia proposta, nos inspiramos no estudo de Serpa, 2023, utilizando um conjunto de matrizes catalogadas que já possuem as contagens de XOR e XTIME registradas. Além disso, utilizamos a tabela de baselines de cada dimensão das matrizes, ou seja, os menores custos do catálogo completo. Focamos exclusivamente no corpo finito $\mathrm{GF}(2^8)$, comparando nossos resultados com os baselines de Serpa, 2023.

A Tabela 1 contém os custos das matrizes mais baratas encontradas na literatura até 2021, por dimensão, e foi retirada de Serpa, 2023. Ela foi utilizada como referência em nossas tentativas de obter matrizes melhores.

Utilizamos também a Tabela 2, que contém os resultados de Serpa, 2023 com seus algoritmos genéticos, com o intuito de comparar o desempenho da nossa implementação e das nossas estratégias com as de Serpa, 2023.

Durante o trabalho, testamos diferentes metodologias e configurações:

Dim	#xtime	#xor	Custo
2	2	2	8
3	3	6	15
4	8	16	40
5	30	30	120
6	59	59	236
7	96	96	384
8	72	80	296
16	1248	800	4544
32	5440	3712	20032

Tabela 1: Baselines

Dim	#xor	#xtime	Baseline Diff	Corpo finito
2	3	2	1	$GF(2^4)$
3	24	34	21	$GF(2^8)$
4	26	32	24	$GF(2^4)$
5	89	126	96	$GF(2^8)$
6	84	120	210	$GF(2^4)$
6	54	60	0	$GF(2^4)$
7	96	104	24	$GF(2^4)$
7	112	214	370	$GF(2^8)$

Tabela 2: Resultados de Serpa, 2023 - a coluna baseline diff diz o quão mais cara que o baseline uma matriz obtida está

4.1 *Configuração 1*

A *configuração 1* gera e testa matrizes completas, usando método de seleção de torneio ou ranking emutação por substituição aleatória, além de métodos de intensificação.

4.1.1 Resultados obtidos com a *configuração 1*

Observa-se na Tabela 3 que, para as dimensões 2, 3 e 5, foi possível encontrar matrizes de custo igual ou inferior ao baseline rapidamente, mas, para as dimensões 4 e 6, apenas matrizes mais caras foram encontradas, e o tempo de execução aumentou bastante. Os resultados obtidos com a aplicação da configuração 1 em matrizes de dimensão superior revelaram desafios significativos. Embora tenha demonstrado eficácia em dimensões menores,

Dim	Custo	Baseline	Tempo computacional (s)
2	5	8	153
3	15	15	415
4	49	40	1993
5	116	120	38164
6	388	236	37222

Tabela 3: Resultados para a configuração 1

a metodologia se mostrou impraticável para escalas maiores, o que evidenciou a necessidade da investigação de outras configurações.

4.2 *Configuração 2*

A configuração 2 gera apenas a primeira linha da matriz, sendo cada linha seguinte o resultado da rotação da anterior à esquerda, conceito que corresponde à definição de *matrizes circulantes à esquerda*, presente em Serpa, 2023.

4.2.1 Resultados obtidos com a *configuração 2*

Abaixo se encontram os melhores resultados obtidos a partir do algoritmo genético e da busca local pura:

Dim	Matriz	#xtime	#xor	Custo	Custo Baseline
4x4	(1)	8	16	40	40
5x5	(5)	15	25	70	120
6x6	(17)	30	48	138	236
7x7	(39)	42	56	182	384
8x8	(47)	88	96	360	296

Tabela 4: Resultados algoritmo genético configuração 2

Dim	Matriz	#xtime	#xor	Custo	Custo Baseline
6x6	(22)	30	48	138	236
7x7	(33)	42	56	182	384
8x8	(42)	88	88	352	296

Tabela 5: Resultados busca local configuração 2.

Obs: alguns dos valores da Tabela 5, presentes na tabela 6, foram obtidos na Geração 0 do algoritmo genético com *full local search* aplicado ao conjunto inicial, o que os configura como resultado de busca local pura.

Os resultados obtidos com a *configuração 2* foram significativamente superiores aos encontrados pela *configuração 1*. No entanto, é possível notar claramente que os resultados da busca local pura são iguais ou melhores do que os resultados do algoritmo genético. Especificamente as melhores matrizes 6x6 e 7x7 são equivalentes em ambos os casos, apenas rotacionadas. Isso pode significar que a metaheurística de algoritmos genéticos não é compatível com o problema, ou que a utilização da mesma está equivocada.

4.2.2 Discussão dos resultados da *configuração 2*

Durante o desenvolvimento do projeto, foi notado que para uma matriz MDS, não existe uma presença forte de subestruturas transferíveis entre matrizes, com a propriedade MDS dependendo da relação dos valores como um todo. Isso significa que as operações de crossover acabam não obtendo o resultado esperado, sendo pouco efetivas em combinar soluções boas, e isso era notável ao observar que, nas execuções, a grande maioria das soluções incumbentes novas eram resultado dos passos de busca local, e não do processo de crossover. Já o processo de mutação se demonstrou mais relevante para o problema, uma vez que permitia ao os passos de *mini local search* first-improving explorar outros membros da vizinhança fora do mínimo local.

Na realização dos testes, as configurações que utilizavam o *full local search* para gerar a população inicial otimizada apresentavam uma grande dificuldade em obter resultados melhores do que os presentes na geração inicial, o que levou ao estudo de soluções obtidas a partir da busca local pura, que se mostraram equivalentes ou melhores aos obtidos a partir do algoritmo genético.

5 Conclusões e Trabalhos Futuros

Embora o desempenho do algoritmo genético tenha sido inferior ao esperado, mesmo em relação a uma busca local, os resultados quanto à nova função-objetivo foram promissores. Ao adicionar o gradiente de distância para a propriedade de MDS, o programa demonstrou uma grande facilidade em encontrar matrizes que cumprem os requisitos da propriedade, sendo o maior desafio a diminuição do custo das mesmas.

A utilização de matrizes circulantes foi também de grande ajuda na tentativa de otimizar o processo, apresentando uma convergência muito mais rápida do que matrizes completas.

É curiosa, porém, a aparente facilidade com a qual encontramos matrizes de custo mais baixo do que as encontradas na literatura. Embora o processo de busca seja relativamente custoso, nenhuma de nossas execuções demorou mais de 24 horas. Assim, especulamos que existam outros fatores além de custo que determinam se uma matriz MDS é interessante para utilização em um algoritmo criptográfico, por exemplo, particularidades do algoritmo e detalhes de sua aritmética. Outro ponto a ser observado é que as matrizes encontradas de custo inferior às presentes na literatura são de dimensões pouco exploradas, como 5x5, 6x6 e 7x7, enquanto matrizes de tamanho 4x4 e 8x8 são amplamente utilizadas em algoritmos criptográficos e foram exploradas extensivamente.

A principal barreira no desenvolvimento do projeto foi o custo exponencial do cálculo da propriedade MDS para uma matriz com o aumento de suas dimensões. No trabaho Malakhov, 2021, foi proposta uma nova maneira eficiente de testar essa propriedade para matrizes circulantes, o que potencialmente torna o processo de obter novas matrizes muito mais rápido, viabilizando métodos que dependem muito do cálculo rápido da função-objetivo, como é o caso do algoritmo genético.

6 Apêndices

Nesta seção, incluimos outros detalhes do processo experimental, que gostaríamos de manter separados do texto principal para respeitar o limite de páginas, mas podem ser interessantes ou relevantes para replicabilidade dos experimentos e compreensao melhor dos mesmos. Todas as matrizes encontradas neste trabalho estao tambem listadas aqui, na secao 6.2.

6.1 Detalhes da implementação

O código base que lida com as partes mais matemáticas de aritmética de corpos finitos e criptografia nos foi fornecido pela autora de Serpa, 2023, sendo a parte de otimização combinatória e algoritmos genéticos implementada pelos membros do time, assim como algumas modificações do código original para melhor servir nossos experimentos.

No código fonte original, foi criado um arquivo `readonly.py` com funções auxiliares para lidar com os conceitos de criptografia, como por exemplo calcular as contagens de XOR e XTIME de uma matriz, calcular se uma matriz é MDS, retornar o limite superior para coeficientes de uma matriz dado o expoente que define o corpo finito, entre outros. A proposta deste arquivo era não ser modificado pelos membros do grupo durante os experimentos, uma vez que o mesmo foi validado como correto pela autora de Serpa, 2023, de forma a evitar

resultados inconsistentes, dadas as várias nuances e detalhes do problema. A proposta foi apenas chamar as funções deste arquivo e usá-las para construir o algoritmo genético. Por exemplo, chamar `validUpper` para controlar os coeficientes das matrizes, e `isMDS` para calcular se a matriz é MDS.

No arquivo `customization.py`, eventuais customizações foram feitas pelo time, como por exemplo um MDS checker que computa o gradiente MDS em vez de retornar apenas `True` ou `False`. O algoritmo genético em si foi implementado nos arquivos `main.py` para matrizes tradicionais e `main_rot.py` para matrizes circulantes. Outros arquivos auxiliares também foram criados, como 'details.py' que, dada uma matriz e seu parâmetro de corpo finito, retorna seus detalhes como valores de XOR, XTIME, custo combinado e se é MDS; e `local_search.py` que executa a busca local exaustiva mencionado em 3.8.6.

O código está disponível no repositório https://github.com/AnaClaraZoppiSerpa/GAMDS, e instruções de utilização das funções auxiliares estão no README. O código mais recente se encontra na branch `oldSearch`.

6.2 Lista de matrizes

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

$$\begin{bmatrix} 5 & 5 & 1 & 2 & 1 \\ 5 & 1 & 2 & 1 & 5 \\ 1 & 2 & 1 & 5 & 5 \\ 2 & 1 & 5 & 5 & 1 \\ 1 & 5 & 5 & 1 & 2 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} 1 & 8 & 3 & 2 & 1 \\ 8 & 3 & 2 & 1 & 1 \\ 3 & 2 & 1 & 1 & 8 \\ 2 & 1 & 1 & 8 & 3 \\ 1 & 1 & 8 & 3 & 2 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} 1 & 5 & 3 & 2 & 1 \\ 5 & 3 & 2 & 1 & 1 \\ 3 & 2 & 1 & 1 & 5 \\ 2 & 1 & 1 & 5 & 3 \\ 1 & 1 & 5 & 3 & 2 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 2 & 1 & 1 \\ 3 & 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 & 2 \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} 1 & 2 & 1 & 3 & 4 & 65 \\ 2 & 1 & 3 & 4 & 65 & 1 \\ 1 & 3 & 4 & 65 & 1 & 2 \\ 3 & 4 & 65 & 1 & 2 & 1 \\ 4 & 65 & 1 & 2 & 1 & 3 \\ 65 & 1 & 2 & 1 & 3 & 4 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} 1 & 2 & 1 & 3 & 4 & 32 \\ 2 & 1 & 3 & 4 & 32 & 1 \\ 1 & 3 & 4 & 32 & 1 & 2 \\ 3 & 4 & 32 & 1 & 2 & 1 \\ 4 & 32 & 1 & 2 & 1 & 3 \\ 32 & 1 & 2 & 1 & 3 & 4 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} 1 & 3 & 9 & 5 & 1 & 2 \\ 3 & 9 & 5 & 1 & 2 & 1 \\ 9 & 5 & 1 & 2 & 1 & 3 \\ 5 & 1 & 2 & 1 & 3 & 9 \\ 1 & 2 & 1 & 3 & 9 & 5 \\ 2 & 1 & 3 & 9 & 5 & 1 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} 1 & 1 & 12 & 3 & 2 & 3 \\ 1 & 12 & 3 & 2 & 3 & 1 \\ 12 & 3 & 2 & 3 & 1 & 1 \\ 3 & 2 & 3 & 1 & 1 & 12 \\ 2 & 3 & 1 & 1 & 12 & 3 \\ 3 & 1 & 1 & 12 & 3 & 2 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} 1 & 1 & 8 & 3 & 2 & 3 \\ 1 & 8 & 3 & 2 & 3 & 1 \\ 8 & 3 & 2 & 3 & 1 & 1 \\ 3 & 2 & 3 & 1 & 1 & 8 \\ 2 & 3 & 1 & 1 & 8 & 3 \\ 3 & 1 & 1 & 8 & 3 & 2 \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} 3 & 1 & 3 & 2 & 2 & 7 \\ 1 & 3 & 2 & 2 & 7 & 3 \\ 3 & 2 & 2 & 7 & 3 & 1 \\ 2 & 2 & 7 & 3 & 1 & 3 \\ 2 & 7 & 3 & 1 & 3 & 2 \\ 7 & 3 & 1 & 3 & 2 & 2 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} 4 & 1 & 1 & 5 & 6 & 2 \\ 1 & 1 & 5 & 6 & 2 & 4 \\ 1 & 5 & 6 & 2 & 4 & 1 \\ 5 & 6 & 2 & 4 & 1 & 1 \\ 6 & 2 & 4 & 1 & 1 & 5 \\ 2 & 4 & 1 & 1 & 5 & 6 \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} 3 & 2 & 1 & 10 & 1 & 3 \\ 2 & 1 & 10 & 1 & 3 & 3 \\ 1 & 10 & 1 & 3 & 3 & 2 \\ 10 & 1 & 3 & 3 & 2 & 1 \\ 1 & 3 & 3 & 2 & 1 & 10 \\ 3 & 3 & 2 & 1 & 10 & 1 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} 3 & 2 & 1 & 8 & 1 & 3 \\ 2 & 1 & 8 & 1 & 3 & 3 \\ 1 & 8 & 1 & 3 & 3 & 2 \\ 8 & 1 & 3 & 3 & 2 & 1 \\ 1 & 3 & 3 & 2 & 1 & 8 \\ 3 & 3 & 2 & 1 & 8 & 1 \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} 2 & 2 & 1 & 8 & 1 & 3 \\ 2 & 1 & 8 & 1 & 3 & 2 \\ 1 & 8 & 1 & 3 & 2 & 2 \\ 8 & 1 & 3 & 2 & 2 & 1 \\ 1 & 3 & 2 & 2 & 1 & 8 \\ 3 & 2 & 2 & 1 & 8 & 1 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} 2 & 3 & 5 & 7 & 1 & 1 \\ 3 & 5 & 7 & 1 & 1 & 2 \\ 5 & 7 & 1 & 1 & 2 & 3 \\ 7 & 1 & 1 & 2 & 3 & 5 \\ 1 & 1 & 2 & 3 & 5 & 7 \\ 1 & 2 & 3 & 5 & 7 & 1 \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} 1 & 2 & 7 & 2 & 3 & 1 \\ 2 & 7 & 2 & 3 & 1 & 1 \\ 7 & 2 & 3 & 1 & 1 & 2 \\ 2 & 3 & 1 & 1 & 2 & 7 \\ 3 & 1 & 1 & 2 & 7 & 2 \\ 1 & 1 & 2 & 7 & 2 & 3 \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} 6 & 4 & 2 & 4 & 6 & 1 & 1 \\ 4 & 2 & 4 & 6 & 1 & 1 & 6 \\ 2 & 4 & 6 & 1 & 1 & 6 & 4 \\ 4 & 6 & 1 & 1 & 6 & 4 & 2 \\ 6 & 1 & 1 & 6 & 4 & 2 & 4 \\ 1 & 1 & 6 & 4 & 2 & 4 & 6 \\ 1 & 6 & 4 & 2 & 4 & 6 & 1 \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} 1 & 4 & 1 & 2 & 6 & 5 & 1 \\ 4 & 1 & 2 & 6 & 5 & 1 & 1 \\ 1 & 2 & 6 & 5 & 1 & 1 & 4 \\ 2 & 6 & 5 & 1 & 1 & 4 & 1 \\ 6 & 5 & 1 & 1 & 4 & 1 & 2 \\ 5 & 1 & 1 & 4 & 1 & 2 & 6 \\ 1 & 1 & 4 & 1 & 2 & 6 & 5 \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} 1 & 4 & 1 & 2 & 6 & 5 & 1 \\ 4 & 1 & 2 & 6 & 5 & 1 & 1 \\ 1 & 2 & 6 & 5 & 1 & 1 & 4 \\ 2 & 6 & 5 & 1 & 1 & 4 & 1 \\ 6 & 5 & 1 & 1 & 4 & 1 & 2 \\ 5 & 1 & 1 & 4 & 1 & 2 & 6 \\ 1 & 1 & 4 & 1 & 2 & 6 & 5 \end{bmatrix} \tag{20}$$

$$\begin{bmatrix} 2 & 1 & 3 & 4 & 4 & 1 \\ 1 & 3 & 4 & 4 & 1 & 2 \\ 3 & 4 & 4 & 1 & 2 & 1 \\ 4 & 4 & 1 & 2 & 1 & 3 \\ 4 & 1 & 2 & 1 & 3 & 4 \\ 1 & 2 & 1 & 3 & 4 & 4 \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} 1 & 2 & 7 & 2 & 3 & 1 \\ 2 & 7 & 2 & 3 & 1 & 1 \\ 7 & 2 & 3 & 1 & 1 & 2 \\ 2 & 3 & 1 & 1 & 2 & 7 \\ 3 & 1 & 1 & 2 & 7 & 2 \\ 1 & 1 & 2 & 7 & 2 & 3 \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} 2 & 1 & 1 & 3 & 2 & 7 \\ 1 & 1 & 3 & 2 & 7 & 2 \\ 1 & 3 & 2 & 7 & 2 & 1 \\ 3 & 2 & 7 & 2 & 1 & 1 \\ 2 & 7 & 2 & 1 & 1 & 3 \\ 7 & 2 & 1 & 1 & 3 & 2 \end{bmatrix} \tag{23}$$

$$\begin{bmatrix} 2 & 11 & 4 & 6 & 11 & 1 \\ 11 & 4 & 6 & 11 & 1 & 2 \\ 4 & 6 & 11 & 1 & 2 & 11 \\ 6 & 11 & 1 & 2 & 11 & 4 \\ 11 & 1 & 2 & 11 & 4 & 6 \\ 1 & 2 & 11 & 4 & 6 & 11 \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} 2 & 3 & 4 & 12 & 5 & 1 \\ 3 & 4 & 12 & 5 & 1 & 2 \\ 4 & 12 & 5 & 1 & 2 & 3 \\ 12 & 5 & 1 & 2 & 3 & 4 \\ 5 & 1 & 2 & 3 & 4 & 12 \\ 1 & 2 & 3 & 4 & 12 & 5 \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} 1 & 2 & 3 & 3 & 10 & 2 \\ 2 & 3 & 3 & 10 & 2 & 1 \\ 3 & 3 & 10 & 2 & 1 & 2 \\ 3 & 10 & 2 & 1 & 2 & 3 \\ 10 & 2 & 1 & 2 & 3 & 3 \\ 2 & 1 & 2 & 3 & 3 & 10 \end{bmatrix} \tag{26}$$

$$\begin{bmatrix} 1 & 3 & 3 & 11 & 1 & 2 \\ 3 & 3 & 11 & 1 & 2 & 1 \\ 3 & 11 & 1 & 2 & 1 & 3 \\ 11 & 1 & 2 & 1 & 3 & 3 \\ 1 & 2 & 1 & 3 & 3 & 11 \\ 2 & 1 & 3 & 3 & 11 & 1 \end{bmatrix} \tag{27}$$

$$\begin{bmatrix} 3 & 1 & 1 & 8 & 3 & 2 \\ 1 & 1 & 8 & 3 & 2 & 3 \\ 1 & 8 & 3 & 2 & 3 & 1 \\ 8 & 3 & 2 & 3 & 1 & 1 \\ 3 & 2 & 3 & 1 & 1 & 8 \\ 2 & 3 & 1 & 1 & 8 & 3 \end{bmatrix} \tag{28}$$

$$\begin{bmatrix} 2 & 8 & 2 & 3 & 1 & 1 \\ 8 & 2 & 3 & 1 & 1 & 2 \\ 2 & 3 & 1 & 1 & 2 & 8 \\ 3 & 1 & 1 & 2 & 8 & 2 \\ 1 & 1 & 2 & 8 & 2 & 3 \\ 1 & 2 & 8 & 2 & 3 & 1 \end{bmatrix} \tag{29}$$

$$\begin{bmatrix} 2 & 7 & 2 & 3 & 1 & 1 \\ 7 & 2 & 3 & 1 & 1 & 2 \\ 2 & 3 & 1 & 1 & 2 & 7 \\ 3 & 1 & 1 & 2 & 7 & 2 \\ 1 & 1 & 2 & 7 & 2 & 3 \\ 1 & 2 & 7 & 2 & 3 & 1 \end{bmatrix} \tag{30}$$

$$\begin{bmatrix} 11 & 11 & 5 & 12 & 1 & 4 \\ 11 & 5 & 12 & 1 & 4 & 11 \\ 5 & 12 & 1 & 4 & 11 & 11 \\ 12 & 1 & 4 & 11 & 11 & 5 \\ 1 & 4 & 11 & 11 & 5 & 12 \\ 4 & 11 & 11 & 5 & 12 & 1 \end{bmatrix} \tag{31}$$

$$\begin{bmatrix} 1 & 5 & 4 & 2 & 6 & 1 \\ 5 & 4 & 2 & 6 & 1 & 1 \\ 4 & 2 & 6 & 1 & 1 & 5 \\ 2 & 6 & 1 & 1 & 5 & 4 \\ 6 & 1 & 1 & 5 & 4 & 2 \\ 1 & 1 & 5 & 4 & 2 & 6 \end{bmatrix} \tag{32}$$

$$\begin{bmatrix} 1 & 3 & 2 & 4 & 2 & 3 & 1 \\ 3 & 2 & 4 & 2 & 3 & 1 & 1 \\ 2 & 4 & 2 & 3 & 1 & 1 & 3 \\ 4 & 2 & 3 & 1 & 1 & 3 & 2 \\ 2 & 3 & 1 & 1 & 3 & 2 & 4 \\ 3 & 1 & 1 & 3 & 2 & 4 & 2 \\ 1 & 1 & 3 & 2 & 4 & 2 & 3 \end{bmatrix} \tag{33}$$

$$\begin{bmatrix} 10 & 2 & 1 & 8 & 8 & 9 & 12 \\ 2 & 1 & 8 & 8 & 9 & 12 & 10 \\ 1 & 8 & 8 & 9 & 12 & 10 & 2 \\ 8 & 8 & 9 & 12 & 10 & 2 & 1 \\ 8 & 9 & 12 & 10 & 2 & 1 & 8 \\ 9 & 12 & 10 & 2 & 1 & 8 & 8 \\ 12 & 10 & 2 & 1 & 8 & 8 & 9 \end{bmatrix} \tag{34}$$

$$\begin{bmatrix} 3 & 3 & 1 & 6 & 10 & 7 & 2 \\ 3 & 1 & 6 & 10 & 7 & 2 & 3 \\ 1 & 6 & 10 & 7 & 2 & 3 & 3 \\ 6 & 10 & 7 & 2 & 3 & 3 & 1 \\ 10 & 7 & 2 & 3 & 3 & 1 & 6 \\ 7 & 2 & 3 & 3 & 1 & 6 & 10 \\ 2 & 3 & 3 & 1 & 6 & 10 & 7 \end{bmatrix} \tag{35}$$

$$\begin{bmatrix} 1 & 11 & 1 & 1 & 5 & 12 & 2 \\ 11 & 1 & 1 & 5 & 12 & 2 & 1 \\ 1 & 1 & 5 & 12 & 2 & 1 & 11 \\ 1 & 5 & 12 & 2 & 1 & 11 & 1 \\ 5 & 12 & 2 & 1 & 11 & 1 & 1 \\ 12 & 2 & 1 & 11 & 1 & 1 & 5 \\ 2 & 1 & 11 & 1 & 1 & 5 & 12 \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} 1 & 3 & 1 & 1 & 2 & 12 & 9 \\ 3 & 1 & 1 & 2 & 12 & 9 & 1 \\ 1 & 1 & 2 & 12 & 9 & 1 & 3 \\ 1 & 2 & 12 & 9 & 1 & 3 & 1 \\ 2 & 12 & 9 & 1 & 3 & 1 & 1 \\ 12 & 9 & 1 & 3 & 1 & 1 & 2 \\ 9 & 1 & 3 & 1 & 1 & 2 & 12 \end{bmatrix} \tag{37}$$

$$\begin{bmatrix} 2 & 3 & 1 & 1 & 3 & 2 & 9 \\ 3 & 1 & 1 & 3 & 2 & 9 & 2 \\ 1 & 1 & 3 & 2 & 9 & 2 & 3 \\ 1 & 3 & 2 & 9 & 2 & 3 & 1 \\ 3 & 2 & 9 & 2 & 3 & 1 & 1 \\ 2 & 9 & 2 & 3 & 1 & 1 & 3 \\ 9 & 2 & 3 & 1 & 1 & 3 & 2 \end{bmatrix} \tag{38}$$

$$\begin{bmatrix} 2 & 3 & 1 & 1 & 3 & 2 & 4 \\ 3 & 1 & 1 & 3 & 2 & 4 & 2 \\ 1 & 1 & 3 & 2 & 4 & 2 & 3 \\ 1 & 3 & 2 & 4 & 2 & 3 & 1 \\ 3 & 2 & 4 & 2 & 3 & 1 & 1 \\ 2 & 4 & 2 & 3 & 1 & 1 & 3 \\ 4 & 2 & 3 & 1 & 1 & 3 & 2 \end{bmatrix} \tag{39}$$

$$\begin{bmatrix} 1 & 2 & 13 & 8 & 4 & 3 & 10 & 1 \\ 2 & 13 & 8 & 4 & 3 & 10 & 1 & 1 \\ 13 & 8 & 4 & 3 & 10 & 1 & 1 & 2 \\ 8 & 4 & 3 & 10 & 1 & 1 & 2 & 13 \\ 4 & 3 & 10 & 1 & 1 & 2 & 13 & 8 \\ 3 & 10 & 1 & 1 & 2 & 13 & 8 & 4 \\ 10 & 1 & 1 & 2 & 13 & 8 & 4 & 3 \\ 1 & 1 & 2 & 13 & 8 & 4 & 3 & 10 \end{bmatrix} \tag{40}$$

$$\begin{bmatrix} 4 & 11 & 13 & 1 & 1 & 5 & 1 & 3 \\ 11 & 13 & 1 & 1 & 5 & 1 & 3 & 4 \\ 13 & 1 & 1 & 5 & 1 & 3 & 4 & 11 \\ 1 & 1 & 5 & 1 & 3 & 4 & 11 & 13 \\ 1 & 5 & 1 & 3 & 4 & 11 & 13 & 1 \\ 5 & 1 & 3 & 4 & 11 & 13 & 1 & 1 \\ 1 & 3 & 4 & 11 & 13 & 1 & 1 & 5 \\ 3 & 4 & 11 & 13 & 1 & 1 & 5 & 1 \end{bmatrix} \tag{41}$$

$$\begin{bmatrix} 1 & 7 & 1 & 1 & 2 & 6 & 8 & 9 \\ 7 & 1 & 1 & 2 & 6 & 8 & 9 & 1 \\ 1 & 1 & 2 & 6 & 8 & 9 & 1 & 7 \\ 1 & 2 & 6 & 8 & 9 & 1 & 7 & 1 \\ 2 & 6 & 8 & 9 & 1 & 7 & 1 & 1 \\ 6 & 8 & 9 & 1 & 7 & 1 & 1 & 2 \\ 8 & 9 & 1 & 7 & 1 & 1 & 2 & 6 \\ 9 & 1 & 7 & 1 & 1 & 2 & 6 & 8 \end{bmatrix} \tag{42}$$

$$\begin{bmatrix} 3 & 1 & 6 & 5 & 8 & 2 & 2 & 9 \\ 1 & 6 & 5 & 8 & 2 & 2 & 9 & 3 \\ 6 & 5 & 8 & 2 & 2 & 9 & 3 & 1 \\ 5 & 8 & 2 & 2 & 9 & 3 & 1 & 6 \\ 8 & 2 & 2 & 9 & 3 & 1 & 6 & 5 \\ 2 & 2 & 9 & 3 & 1 & 6 & 5 & 8 \\ 2 & 9 & 3 & 1 & 6 & 5 & 8 & 2 \\ 9 & 3 & 1 & 6 & 5 & 8 & 2 & 2 \end{bmatrix} \tag{43}$$

$$\begin{bmatrix} 10 & 2 & 14 & 13 & 6 & 1 & 1 & 2 \\ 2 & 14 & 13 & 6 & 1 & 1 & 2 & 10 \\ 14 & 13 & 6 & 1 & 1 & 2 & 10 & 2 \\ 13 & 6 & 1 & 1 & 2 & 10 & 2 & 14 \\ 6 & 1 & 1 & 2 & 10 & 2 & 14 & 13 \\ 1 & 1 & 2 & 10 & 2 & 14 & 13 & 6 \\ 1 & 2 & 10 & 2 & 14 & 13 & 6 & 1 \\ 2 & 10 & 2 & 14 & 13 & 6 & 1 & 1 \end{bmatrix} \tag{44}$$

$$\begin{bmatrix} 1 & 9 & 3 & 15 & 12 & 2 & 1 & 3 \\ 9 & 3 & 15 & 12 & 2 & 1 & 3 & 1 \\ 3 & 15 & 12 & 2 & 1 & 3 & 1 & 9 \\ 15 & 12 & 2 & 1 & 3 & 1 & 9 & 3 \\ 12 & 2 & 1 & 3 & 1 & 9 & 3 & 15 \\ 2 & 1 & 3 & 1 & 9 & 3 & 15 & 12 \\ 1 & 3 & 1 & 9 & 3 & 15 & 12 & 2 \\ 3 & 1 & 9 & 3 & 15 & 12 & 2 & 1 \end{bmatrix} \tag{45}$$

$$\begin{bmatrix} 15 & 2 & 12 & 3 & 1 & 1 & 3 & 8 \\ 2 & 12 & 3 & 1 & 1 & 3 & 8 & 15 \\ 12 & 3 & 1 & 1 & 3 & 8 & 15 & 2 \\ 3 & 1 & 1 & 3 & 8 & 15 & 2 & 12 \\ 1 & 1 & 3 & 8 & 15 & 2 & 12 & 3 \\ 1 & 3 & 8 & 15 & 2 & 12 & 3 & 1 \\ 3 & 8 & 15 & 2 & 12 & 3 & 1 & 1 \\ 8 & 15 & 2 & 12 & 3 & 1 & 1 & 3 \end{bmatrix} \tag{46}$$

$$\begin{bmatrix} 8 & 3 & 15 & 1 & 12 & 1 & 1 & 2 \\ 3 & 15 & 1 & 12 & 1 & 1 & 2 & 8 \\ 15 & 1 & 12 & 1 & 1 & 2 & 8 & 3 \\ 1 & 12 & 1 & 1 & 2 & 8 & 3 & 15 \\ 12 & 1 & 1 & 2 & 8 & 3 & 15 & 1 \\ 1 & 1 & 2 & 8 & 3 & 15 & 1 & 12 \\ 1 & 2 & 8 & 3 & 15 & 1 & 12 & 1 \\ 2 & 8 & 3 & 15 & 1 & 12 & 1 & 1 \end{bmatrix} \tag{47}$$

6.3 Sequência de instâncias (execuções) e aplicação dos métodos da *configuração 2*

A coluna **Inst** se refere a uma execução do código, tambem chamada de instância. **Dim** se refere à dimensão da matriz, k_1, k_2, k_3 são os parâmetros para a função-objetivo, p_{min} e p_{max} são as taxas minima e máxima de mutação. **Ref** é a referência da matriz na lista de matrizes (Seção 6.2) para facilitar a identificação, **Gen** é geração da instância do algoritmo genetico. Algumas células da tabela apresentam um campo de observações (**Obs**). Quando preenchido com -, significa que o dado não foi anotado apesar do conceito se aplicar (por exemplo, na instancia 17, não foi anotada a geração na qual a matriz foi obtida). Quando preenchido com n/a, significa que aquele dado nao se aplica àquela execução (por exemplo, execuções somente de busca local pura não possuem p_{min}, p_{max} ou gerações).

Métodos	mutação guiada					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
1	4x4	1000	2000	10	0.05	0.3
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	

sim	40	16	8	1	-
Obs	mesmos coeficientes da matriz do AES, em posições diferentes (circulantes à esquerda), e mesmo custo. Provavelmente a mais barata 4x4 existente.				
Métodos	mutação guiada				
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
2	5x5	1000	2000	10	0.05	0.3
Matrizes					
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**
sim	105	30	25	2	36
Métodos	mutação guiada				
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
3	5x5	1000	2000	10	0.05	0.3
Matrizes					
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**
sim	100	25	25	3	13
sim	90	30	20	4	20
sim	70	25	15	5	33
Métodos	mutação guiada				
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
4	6x6	1000	2000	10	0.05	0.3
Matrizes					
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**
sim	222	42	60	6	10
sim	198	36	54	7	>10
Obs	Não anotamos a geração na qual essa melhoria ocorreu, mas foi depois da geração 10				
Métodos	mutação guiada + busca local				
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
5	6x6	1000	2000	10	0.05	0.3
Matrizes					
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**

sim	174	48	42	8		6
Métodos	mutação guiada + busca local					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
6	6x6	1000	2000	10	0.05	0.3
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	156	48	36	9		5
sim	150	42	36	10		6
Métodos	mutação guiada + busca local + método de mutação de relaxação					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
7	6x6	1000	2000	10	0.05	0.3
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	162	54	36	11		9
Métodos	mutação guiada + busca local + método de mutação de relaxação com troca					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
8	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	168	42	42	12		8
Métodos	mutação guiada + busca local com shuffle + método de mutação de relaxação com troca					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
9	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	156	48	36	13		6
sim	150	42	36	14		7
sim	144	36	36	15		8
Métodos	mutação guiada + busca local com shuffle + método de mutação de relaxação com troca					

Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
10	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	162	54	36	16		6
sim	138	48	30	17		7
Métodos	mutação guiada + busca local com shuffle + método de mutação de relaxação com troca					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
11	7x7	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	245	56	63	18		6
sim	203	56	49	19		36
sim	203	56	49	20		195
Obs	Não melhorou a partir da Gen 36					
Métodos	igual inst 11					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
12	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	144	36	36	21		24
Métodos	igual inst 12 + população inicial com busca local exaustiva					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
13	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	138	48	30	22		0
Métodos	Não inclui incumbente na próxima geração + mini busca local nos filhos					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
14	6x6	1000	2000	10	0.05	0.7

Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
não	258	60	66	24		0
sim	210	48	54	25		1
sim	174	48	42	26		2
sim	162	54	36	27		3
sim	150	42	36	28		5
sim	144	36	36	29		6
sim	138	48	30	30		7
Métodos	igual inst 14 + busca local limitada a valores próximos, pode ir acima do valor atual					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
15	6x6	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	300	66	78	31		0
sim	168	42	42	32		1
Métodos	busca local pura sem GA					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
A	7x7	1000	2000	10	n/a	n/a
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	182	56	42	33	n/a	
Obs	1h de busca local, ou seja, não escalou bem pra dimensões altas					
Métodos	igual inst 14					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
16	7x7	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
Não	399	63	112	34		0
sim	294	84	70	35		1
sim	259	70	63	36		2

sim	231	63	56	37		4
sim	210	63	49	38		7
sim	182	56	42	39		11
Métodos	igual inst 16					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
17	8x8	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	400	88	104	40	-	
Métodos	igual inst 12, mas sem mutação guiada no incumbente, apenas busca local					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
18	8x8	1000	2000	10	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	368	104	88	41	8x8	
Métodos	método de busca local puro					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
B	8x8	1000	2000	10	-	-
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
sim	352	88	88	42	-	
Métodos	igual inst 18					
Inst	**Dim**	k_1	k_2	k_3	p_{min}	p_{max}
19	8x8	1000	2000	100	0.05	0.7
Matrizes						
MDS	**Custo**	**XOR**	**XTIME**	**Ref**	**Gen**	
não	400	88	104	43		0
sim	416	104	104	44		8
sim	400	112	96	45		24
sim	392	104	96	46		28

sim	360	96	88	47	42

Tabela 6: Resultados e parâmetros para diferentes instâncias e métodos

6.4 Especificações das Máquinas Utilizadas

Máquina 1 (usada para a *Configuração 1*):

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  12
  On-line CPU(s) list:   0-11
Vendor ID:               GenuineIntel
  Model name:            13th Gen Intel(R) Core(TM) i7-1355U
    CPU family:          6
    Model:               186
    Thread(s) per core:  2
    Core(s) per socket:  10
    Socket(s):           1
    Stepping:            3
    CPU max MHz:         5000,0000
    CPU min MHz:         400,0000
    BogoMIPS:            5222.40
```

Máquina 2 (usada para a *Configuração 2*):

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  12
  On-line CPU(s) list:   0-11
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
```

```
    CPU family:          6
    Model:               165
    Thread(s) per core:  2
    Core(s) per socket:  6
    Socket(s):           1
    Stepping:            2
    CPU max MHz:         5000.0000
    CPU min MHz:         800.0000
    BogoMIPS:            5199.98
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
                         a cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss
                         ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art
                          arch_perfmon pebs bts rep_good nopl xtopology nonstop_
                         tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cp
                         l vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
                          sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsav
                         e avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault
                          epb invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced
                         tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase t
                         sc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed a
                         dx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsa
                         ves dtherm ida arat pln pts hwp hwp_notify hwp_act_wind
                         ow hwp_epp pku ospke md_clear flush_l1d arch_capabiliti
                         es
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   192 KiB (6 instances)
  L1i:                   192 KiB (6 instances)
  L2:                    1.5 MiB (6 instances)
  L3:                    12 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-11
Vulnerabilities:
  Gather data sampling:  Mitigation; Microcode
```

```
Itlb multihit:          KVM: Mitigation: VMX disabled
L1tf:                   Not affected
Mds:                    Not affected
Meltdown:               Not affected
Mmio stale data:        Mitigation; Clear CPU buffers; SMT vulnerable
Retbleed:               Mitigation; Enhanced IBRS
Spec rstack overflow:   Not affected
Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl
                         and seccomp
Spectre v1:             Mitigation; usercopy/swapgs barriers and __user pointer
                         sanitization
Spectre v2:             Mitigation; Enhanced IBRS; IBPB conditional; RSB fillin
                        g; PBRSB-eIBRS SW sequence; BHI Syscall hardening, KVM
                        SW loop
Srbds:                  Mitigation; Microcode
Tsx async abort:        Not affected
```

Referências

Barreto P. S. (2000). «The Anubis block cipher». *NESSIE*.

Barreto P. e Rijmen V. (2000). «The Khazad legacy-level block cipher». *Primitive submitted to NESSIE* 97.106.

Barreto P. e Simplicio M. (2007). «CURUPIRA, a block cipher for constrained platforms». *Anais do 25o Simpsio Brasileiro de Redes de Computadores e Sistemas Distribudos-SBRC* 1, pp. 61–74.

Daemen J., Knudsen L. e Rijmen V. (1997). «The block cipher Square». *Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4*. Springer, pp. 149–165.

Daemen J. e Rijmen V. (2002). *The design of Rijndael. Information security and cryptography*.

FIPS X. (2001). «Advanced encryption standard (AES)». *National Institute for Standards and Technology (NIST)* 197.1.

Malakhov S. S. (2021). «Construction of a set of circulant matrix submatrices for faster MDS matrix verification». *arXiv preprint arXiv:2110.13325*.

Ohkuma K., Muratani H., Sano F. e Kawamura S. (2000). «The block cipher Hierocrypt». *International workshop on selected areas in cryptography*. Springer, pp. 72–88.

Quisquater J.-J. e Schneier B. (2006). *Smart Card. Research and Applications: Third International Conference, CARDIS'98 Louvain-la-Neuve, Belgium, September 14-16, 1998 Proceedings*. Springer.

Rijmen V., Daemen J., Preneel B., Bosselaers A. e De Win E. (1996). «The cipher SHARK». *Fast Software Encryption: Third International Workshop Cambridge, UK, February 21–23 1996 Proceedings 3*. Springer, pp. 99–111.

Roth R. M. e Lempel A. (1989). «On MDS codes via Cauchy matrices». *IEEE transactions on information theory* 35.6, pp. 1314–1319.

Satoh A., Morioka S., Takano K. e Munetoh S. (2001). «A compact Rijndael hardware architecture with S-box optimization». *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, pp. 239–254.

Serpa A. C. Z. (2023). «Study of diffusion and MDS matrices in symmetric block ciphers= Estudo de difusão e matrizes MDS em cifras de bloco simétricas». Tese de mestrado. [sn].

Shannon C. E. (1949). «Communication theory of secrecy systems». *The Bell system technical journal* 28.4, pp. 656–715.