ALGORITMOS DE ICA EM ALFABETOS FINITOS: UM ESTUDO COMPARATIVO

MATEUS MARCUZZO DA ROSA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ALGORITMOS DE ICA EM ALFABETOS FINITOS: UM ESTUDO COMPARATIVO

MATEUS MARCUZZO DA ROSA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

DANIEL GUERREIRO E SILVA, Dr., ENE/UNB
(PRESIDENTE DA COMISSÃO/ORIENTADOR)

FRANCISCO ASSIS DE OLIVEIRA NASCIMENTO, Dr., ENE/UNB
(EXAMINADOR INTERNO)

DENIS GUSTAVO FANTINATO, Dr., CMCC/UFABC
(EXAMINADOR EXTERNO)

Brasília, 02 de dezembro de 2019.

FICHA CATALOGRÁFICA

DA ROSA, MATEUS MARCUZZO

Algoritmos de ICA em alfabetos finitos: um estudo comparativo [Distrito Federal] 2019.

xv, 58p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1.Análise de Componentes Independentes (ICA)	2.Alfabetos finitos
3.Separação Cega de Fontes (BSS)	4.Corpos de Galois
5. Algoritmos lineares	6. Algoritmos não-lineares
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

DA ROSA, M. M. (2019). Algoritmos de ICA em alfabetos finitos: um estudo comparativo. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEE.DM - 730/19. Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 58p.

Dedicatória

Dedico este trabalho às pessoas que acreditam no potencial transformador da investigação científica.

Agradecimentos

Diversas pessoas me auxiliaram, direta ou indiretamente, para que este trabalho pudesse ser concluído. Primeiramente, a minha família, cujo apoio nesta etapa da minha vida se demonstra imprescindível. Meu orientador, Prof. Daniel, sempre atencioso e dedicado, que me apresentou áreas do conhecimento nunca antes vistas por mim, que me deram novas perspectivas de interesse e investigação.

Diversos amigos vivenciaram ou testemunharam minhas dificuldades nesta etapa; especialmente: Leonardo Lourenço, Vinícius Colli, Felipe Pessoa, João Victor Lisboa, Ronaldo Júnior, Renan Prado, Hugo Tadashi, Letícia Ferreira, Arthur Rodrigues Melo, Juliana Silva de Deus, Gesse Roure, Rafael Alves, Fabiana Toledo, Leonardo Ramalho e Thiago Sales. Obrigado por me darem apoio e atenção neste momento tão delicado.

Agradeço à CAPES, pelo fornecimento de auxílio por bolsa para execução deste mestrado.

Peço perdão se por ventura me esqueci de alguém. Todos vocês fazem parte deste trabalho.

"What is the use of a new-born child ?"
- Benjamin Franklin

Resumo

Recentemente, algoritmos de Análise de Componentes Independentes (ICA) em alfabetos finitos foram propostos. Tendo em vista cenários não-testados e a replicação de resultados anteriores, desejamos comparar estes algoritmos, bem como verificar como os algoritmos mais generalistas desempenham em relação aos que assumem corpos finitos.

Desta maneira, nesta dissertação avaliamos, através de simulações, o desempenho de algoritmos de ICA linear como aplicação ao problema de Separação Cega de Fontes (BSS) em corpos finitos. Duas métricas foram consideradas: tempo de execução e Separação Total das Fontes, uma métrica mais pessimista de separação. Apesar dos algoritmos AMERICA, SA4ICA e GLICA convergirem para 100% de Separação Total ao crescermos a quantidade de amostras observadas, o algoritmo SA4ICA apresenta comportamento que rompe este padrão, o que não foi reportado anteriormente. Adicionalmente, implementamos o algoritmo GLICA. Este último apresentou desempenho de separação praticamente igual em relação ao algoritmo AMERICA, apesar de seu tempo de execução se apresentar superior.

Além disso, realizou-se um experimento tendo em vista a aplicação de ICA por si mesma, i.e, a minimização da informação mútua. Os algoritmos lineares previamente aplicados no caso de BSS estão, agora, inseridos em um contexto cuja geração das amostras observadas não se dá por uma mistura linear, o que a princípio não privilegiaria estes algoritmos em relação a um algoritmo não-linear.

Porém, ao serem comparados ao algoritmo não-linear QICA, cuja premissa envolve lidar com esses modelos geradores mais genéricos, este mesmo algoritmo detém desempenho inferior na maioria dos cenários em relação aos lineares, que inclusive demonstraram resultados, entre eles mesmos, praticamente iguais em todos os cenários. Ademais, o algoritmo QICA toma muito tempo para ser executado em relação à todos os outros algoritmos.

Desta forma, os resultados sugerem que os algoritmos lineares detiveram vantagem tanto temporal quanto em desempenho em relação a este novo algoritmo. Para isto, contamos com o uso de inferência estatística em ambos os experimentos para validação de nossos resultados.

Palavras-chave: ICA, BSS, alfabetos finitos, corpos de Galois, linear, não-linear

Abstract

In recent years, Independent Component Analysis (ICA) algorithms over finite alphabets, as well as the particular case of these: finite fields, have been proposed. Given the untested scenarios and the replication of previous results, we want to compare these algorithms with each other, as well as verify how the more generalist algorithms perform compared to those that assume a finite field structure.

Thus, in this dissertation we evaluated, through stochastic simulations, the performance of linear ICA algorithms as an application to the Blind Source Separation (BSS) problem over finite fields. Two metrics were considered: execution time and Total Source Separation, a more pessimistic separation metric. Although the AMERICA, SA4ICA and GLICA algorithms converge at 100 % Total Separation as we grow the number of samples observed, the SA4ICA algorithm has anomalous behavior that breaks this pattern, which was not previously reported. Additionally, we implemented the GLICA algorithm. The latter presented a practically equal separation performance in relation to the AMERICA algorithm, although its execution time is superior to this more consolidated technique.

In addition, we conducted an experiment to apply ICA by itself, that is, to minimize mutual information. The linear algorithms previously applied in the case of BSS are now inserted in a context whose generation of the observed samples is not by a linear mixture, which in principle would not privilege these algorithms over a nonlinear algorithm.

However, when compared to the nonlinear QICA algorithm, whose premise involves dealing with these more generic generator models, this same algorithm has lower performance in most scenarios than the linear ones, which even showed results, among themselves, practically the same in all scenarios. Moreover, the QICA algorithm takes a long time to execute relative to all other linear algorithms.

Thus, the results suggest that linear algorithms had both temporal and performance advantage over this new algorithm. For this, we rely on the use of statistical inference in both experiments to validate our results.

Keywords: ICA, BSS, finite alphabets, Galois fields, linear, non-linear

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AMERICA *Ascending Minimization of EntRopies for ICA*.

BSS *Blind Source Separation*, em português Separação Cega de Fontes.

GLICA *Greedy Linear ICA*.

ICA *Independent Component Analysis*, em português Análise de Componentes Independentes.

MEXICO *Minimizing Entropies by Exchanging in Couples*.

MI *Mutual Information*, em português Informação Mútua.

PCA *Principal Component Analysis*, em português Análise de Componentes Principais.

PMF *Probability Mass Function*, em português Função Massa de Probabilidade.

SA *Simulated Annealing*.

TC *Total Correlation*, em português Correlação Total.

XOR *Exclusive-or*, em português ou-exclusivo.

1 Introdução

Suscita grande discussão o conceito de *processar sinais*, dada a abrangência da área e sua importância [Moura 2009]. Contudo, de maneira intuitiva, *sinais* são medições de interesse de um fenômeno que nos transmitem alguma informação, tal como a intensidade das batidas do coração, o pulso elétrico através sinapse de um neurônio, entre outros. Assim, *processar sinais* envolve uma pletora de atividades com os mesmos, normalmente inescapável de um arcabouço matemático. Considerando este universo, podemos trabalhar problemas envolvendo sinais que se misturam.

Afinal, o termo *ruído* designa algo que atrapalha o sinal original, limitando a quantidade de informação que este sinal, misturado ao ruído, nos traz [Cover e Thomas 2012]. Apesar de, no contexto prático, a presença de ruído em sinais ser inevitável, podemos estar envolvidos em um problema no qual sinais fidedignos (ou com um ruído suficientemente baixo) acabam por se misturar. É de interesse prático e teórico saber como separar estes sinais e em quais circunstâncias.

Em um curso de graduação sobre telecomunicações, aprende-se que, para separar sinais, é possível verificar a representação destes no domínio da frequência, utilizando a transformada de Fourier, para assim aplicar filtros que selecionem apenas as frequências desejadas, onde se encontram os sinais de interesse [Lathi 2012]; ou se soubermos que estes sinais fidedignos seguem um protocolo em que a transmissão, ou recebimento, se dá em espaços temporais específicos, chamados *slots*, saberemos qual sinal estamos recebendo. Ambas as técnicas podem ser empregadas ao mesmo tempo.

Embora isso seja útil e aplicável no dia-a-dia, existem fenômenos que, *a priori*, são decorrentes de algum processo de mistura tal que sua sobreposição espectral, bem como temporal, se mostra irrevogável. Desta forma, esse tipo de separação demonstra pouca adaptabilidade e designa um cenário que não utiliza *aprendizagem*, por se tratarem de métodos fixos.

Para fins de exemplo de uma situação que envolva sinais fidedignos que estão sempre misturados: medir atividade de partes do cérebro. Como cada medição de uma dada região do cérebro é a soma de possíveis regiões próximas, ou distantes, da região medida, nenhuma região medida é um composta por ativação de partes do cérebro isoladamente,

além de medições do potencial elétrico que não estão relacionadas à atividade neural [Comon 2010].

Sob este tipo de cenário de *Separação Cega*, necessitamos de alguma técnica ou premissa, para que possamos separar estes sinais de maneira não-supervisionada, isto é, sem que tenhamos uma "etiqueta" do quanto acertamos ou não nossas predições. Resta-nos fazer algumas suposições *a priori* para que o problema possa ser solucionado.

O trabalho pioneiro quanto a *Separação Cega das Fontes* (BSS, do inglês *Blind Source Separation*) propôs uma arquitetura neuromimética (uma rede neural) para separar sinais "compostos" [Hérault, Jutten e Ans 1985]. Este trabalho apresenta um sistema separador de múltiplas-entradas e múltiplas-saídas. Esta nomenclatura hoje é conhecida como MIMO (do inglês: *Multiple Input Multiple Output*) [Lozano 2019, Lathi 2012], uma técnica que trouxe grandes avanços para as telecomunicações e está cada vez mais presente.

Mais tarde, em 1987, surge a Análise de Componentes Independentes (ICA, do inglês: *Independent Component Analysis*), com um trabalho do mesmo autor [Jutten 1987], em que os sinais da fonte são modelados como variáveis aleatórias mutuamente independentes. Como quase todo arcabouço, este não nasceu com formalização matemática consolidada e se apresenta somente como uma técnica para separação cega de fontes.

Assim, anos depois, Comon formaliza matematicamente ICA em [Comon 1994]. Neste trabalho, enfatiza-se que ICA é um esquema não-supervisionado que pode atender muitos propósitos, como: análise de dados, compressão, localização das fontes, etc. Também traz algumas primeiras contribuições quanto às condições necessárias e suficientes para identificabilidade (separabilidade) utilizando ICA. Apesar disso, a fama de ICA quanto a ser uma técnica que resolve o problema de BSS foi tão grande que os termos eram tratados de maneira intercambiável, apesar de serem atividades e problemas distintos [Romano et al. 2011].

A partir daí, diversos métodos para a realização de ICA foram propostos: baseados em uma função-constraste/função custo a ser minimizada/maximizada. Alguns utilizavam a não-gaussianidade como valor a ser maximizado. Assim, temos a Negentropia e Curtose [Girolami e Fyfe 1996] e estatísticas de ordem superior [Jutten e Herault 1991]. O critério Infomax também fora proposto, capaz de reduzir a Informação Mútua na saída do processo [Bell e Sejnowski 1995]. Há também um método por Máxima Verossimilhança [Belouchrani, Cardoso et al. 1995]. Desenvolveu-se também um método guloso [Delfosse e Loubaton 1995], capaz de extrair uma fonte de cada vez. E, por fim, um dos algoritmos mais populares de ICA: o FastICA [Hyvärinen e Oja 1997], que está presente em diversas bibliotecas de software [Pedregosa et al. 2011, Zito et al. 2009, Tůma, Cristea, Ottosson e Piątyszek, Marchini, Heaton e Ripley]. Destas diversas abordagens possíveis, os métodos baseados em métricas da teoria da informação, como o Infomax,

apresentam o problema quanto a estimação de distribuições, o que normalmente é uma tarefa muito custosa. Por esta razão, métodos que não dependessem deste tipo de estimativa costumam ser amplamente usados no lugar destes.

Mais adiante, em 2007, Yeredor propõe pioneiramente a metodologia de ICA em corpos finitos, um caso particular de ICA em alfabetos finitos, em [Yeredor 2007]. Aqui, os sinais não são números reais ou complexos, mas sim valores inteiros com estrutura cíclica. A princípio, fenômenos da natureza costumam ser números reais, complexos ou simplesmente inteiros. Desta forma, esse ramo de ICA nasceu com proposta puramente teórica e naturalmente abre-se a perspectiva da aplicação deste arcabouço para o problema de BSS em corpos finitos.

Neste contexto, as condições de separação em BSS não são as mesmas do caso clássico, e estatísticas como média, variância, curtose, não exprimem sentido algum [Yeredor 2011]. Desta maneira, os métodos de ICA cabíveis aqui são baseados na teoria da informação ou envolvem trabalhar diretamente nas probabilidades dos símbolos.

De lá para os anos recentes, diversos trabalhos foram desenvolvidos no âmbito de BSS e ICA no contexto de alfabetos finitos [Yeredor 2011, Gutch, Gruber e Theis 2010, Gutch et al. 2012], estabelecendo base teórica para ICA em corpos de potência de números primos, além dos de cardinalidade prima. Três abordagens iniciais se consolidaram, uma baseada em extração/deflação, uma estratégia mais simplificada que envolve a comparação duas a duas de entropias dos sinais, e outra que trabalha diretamente nas probabilidades dos símbolos, sem fazer menção direta à entropia ou outras medidas da teoria da informação.

Desde então, novos algoritmos de ICA foram propostos, alguns destes inspirados em meta-heurísticas imunológicas como [Silva et al. 2011, Silva et al. 2014]. Mais recentemente, as primeiras possibilidades de aplicação de tais algoritmos surgiram como: compressão de dados [Painsky, Rosset e Feder 2015] e *network coding* [Nemoianu et al. 2014, Nemoianu et al. 2013].

Considerando estes novos algoritmos e aplicações, a análise teórica destes é, por vezes, muito custosa. Desta maneira, realizam-se testes empíricos sob diversos cenários, a fim de obter maior entendimento dos algoritmos e aplicações avaliadas.

Sob cenários simulados, podemos verificar quais algoritmos alcançam melhores resultados quanto ao tempo de execução e solução da tarefa. No caso de BSS, algumas métricas de desempenho podem ser consideradas, como: separação parcial e/ou separação total das fontes. Na separação total contamos somente como eventos de sucesso ao recuperarmos *todas* as fontes. Enquanto que na separação parcial contamos a proporção das fontes recuperadas pela quantidade de fontes. Desta maneira, os trabalhos citados anteriormente consideram, em sua maioria, a separação parcial das fontes como métrica de desempenho.

Tendo em vista os algoritmos recentes propostos e os algoritmos referência na área, como o AMERICA, é de caráter elucidante apresentar como esses algoritmos desempenham sob uma métrica mais pessimista.

Não obstante, a avaliação do tempo de execução dos algoritmos é de caráter primordial, seja pela sua análise assintótica ou por valores absolutos, para estipular a validade do uso desses algoritmos em situações práticas. A depender da aplicação, não é desejável um algoritmo que resolve um problema de maneira eficaz, mas que toma um tempo muito longo. Desta maneira, buscamos algoritmos que realizem sua tarefa com uma barganha temporal adequada.

Quanto a estratégias para medir a eficácia da tarefa de ICA, isto é, o problema de se obter um mapeamento que encontre uma maximização da independência entre os componentes, podemos medir a Correlação Total (também chamada de Informação Mútua) do resultado produzido por cada algoritmo. A Correlação Total, ao ser minimizada, implica na maior independência das componentes. Portanto, essa métrica permite medir explicitamente os objetivos desses algoritmos.

Em um problema de busca, o tamanho do espaço de estados e como varrê-lo pode implicar em um tempo maior de execução da tarefa [Russell e Norvig 2009]. Desta forma, tratando-se de alfabetos finitos, algoritmos não-lineares de ICA são, teoricamente, capazes de [Painsky, Rosset e Feder 2018] alcançar menores valores de Informação Mútua que os algoritmos lineares, ainda que isso possa custar mais tempo, por varrer um espaço de estados maior.

Contudo, mesmo considerando a capacidade teórica, o caso prático pode não confirmar a viabilidade desse tipo de procedimento; o compromisso entre melhor solução e menor tempo pode não ser das mais adequadas, esta situação pode envolver uma quantidade de amostras observadas que demore muito tempo a ser coletada, e é inevitável o tratamento com estimativas, isto é, abarcar o caso estatístico, ao invés do probabilístico.

Este trabalho apresenta simulações probabilísticas para fins de teste de algoritmos recentemente propostos, de ICA linear e ICA não-linear, tanto observando-os como solução do problema de BSS ou quanto a capacidade de minimização da Informação Mútua, no caso da tarefa de ICA propriamente dita. Desta maneira, desejamos avaliar estes algoritmos, em caráter comparativo com um algoritmo consolidado, como o AMERICA, e também compará-los a um algoritmo não-linear recentemente proposto, o QICA. Esperar-se-ia que o algoritmo não-linear fosse melhor na minimização da Informação Mútua que os demais, pela própria proposta generalista deste. Contudo, nos cenários simulados, será possível observar que, apenas para alguns cenários específicos pôde ser obtido um desempenho melhor que dos demais métodos.

1.1 Objetivos e proposta

O primeiro objetivo desta dissertação é avaliar empiricamente algoritmos recentemente propostos de ICA linear, que se dá no contexto de corpos finitos, quanto à capacidade de separação total no problema de BSS, bem como avaliar o tempo de execução destes algoritmos, verificando cenários antes não testados. Contribuímos também com a implementação do algoritmo GLICA, que não tinha, até então, código disponibilizado pelo autor.

Como segundo grande objetivo, junto destes algoritmos lineares, colocamos outro algoritmo recentemente proposto, que realiza ICA não-linear, para uma análise quanto à capacidade de minimização da Informação Mútua, isto é, a qualidade dos métodos quanto à tarefa de ICA per se. Avaliamos o problema considerando diversos cenários para os dados de entrada, que não são necessariamente gerados por modelos lineares, o que consequentemente não privilegiaria nenhum algoritmo linear. Assim, seria esperado que um método mais generalista desempenhasse melhor em um cenário não previsto para os métodos lineares, mesmo que pudesse tomar mais tempo em sua execução.

1.2 Organização da Dissertação

No capítulo 2, trazemos os fundamentos de ICA em alfabetos finitos e a formulação do problema de BSS, bem como apresentamos o conceito de ICA não-linear em alfabetos finitos. Também apresentamos os algoritmos utilizados neste trabalho e seus resultados anteriores, abordando também vantagens e desvantagens.

No capítulo 3, analisamos os resultados para os dois ensaios realizados, um para aplicação de ICA em BSS, outro para ICA propriamente dita. Neste primeiro ensaio avaliamos somente algoritmos lineares e verificamos seu desempenho quanto à separação total e tempo de execução da tarefa. No segundo ensaio avaliamos seu desempenho quanto à minimização da Informação Mútua sobre diversos cenários.

Por fim, no capítulo 4, concluímos esta dissertação e trazemos perspectivas de trabalho futuras.

1.3 Trabalho aceito para publicação

da Rosa, Mateus & Silva, D.G. (2019). Um estudo comparativo de algoritmos de ICA em alfabetos finitos sob modelos lineares e não-lineares. XXXVII Simpósio Brasileiro de Telecomunicações (SBrT 2019) [da Rosa e Silva 2019].

2 ICA em alfabetos finitos

Neste capítulo, introduzimos o conceito de ICA adotado neste trabalho. Bem como apresentamos o problema de BSS sobre uma mistura linear em alfabetos finitos e a aplicação de ICA em corpos finitos para solucionar este problema.

Também apresentamos os algoritmos utilizados nas simulações deste trabalho e fundamentamos teoricamente as métricas utilizadas.

2.1 A tarefa de ICA

Independent Component Analysis ICA pode ser considerada uma tarefa de processamento de sinais tanto como aprendizagem não-supervisionada, por estender a Atividade de Componentes Principais (PCA, do inglês *Principal Component Analysis*) [Alpaydin 2014, Comon 1994], visto que independência estatística implica em descorrelação, mas não necessariamente o sentido oposto.

Seu objetivo é, dado um vetor aleatório, encontrar um mapeamento, de dimensão igual ou menor que ao vetor aleatório original, que maximize a independência estatística entre as componentes mapeadas. ICA pode ser descrita no contexto probabilístico ou estatístico. Em uma visão, temos estimação perfeita das variáveis aleatórias, na outra devemos estimar médias, componentes, probabilidades de variáveis aleatórias desconhecidas ou outras métricas que interessem para realização da tarefa. Desta maneira, o caso probabilístico confere a um caso limite do caso estatístico, quando temos estimação perfeita das métricas de interesse.

Este trabalho, em particular, vislumbra ICA como uma aplicação estatística. Os casos em que a visão probabilística se fizer necessária, tomaremos nota. Desta maneira, valores estatísticos de estimativas apresentarão um chapéu sobre o parâmetro estimado, por exemplo: $\widehat{X}$, que representa, por exemplo, a média amostral desta variável aleatória.

Suponha termos um vetor aleatório $\mathcal{X}$, com K componentes. Portanto, temos K variáveis aleatórias, $\mathcal{X}_i$, onde i toma valores de 1 até K:

$$\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{K-1}, \mathcal{X}_K]^T \tag{2.1}$$

Não sabemos, *a priori*, as distribuições dos símbolos de cada variável aleatória em $\mathcal{X}$. Para tanto, montamos uma matriz $\mathbf{X}$, cujas colunas são realizações independentes do vetor aleatório $\mathcal{X}$. A i-ésima realização é denotada por $\mathbf{x}_i$. Se desejarmos enfatizar cada componente k de cada i-ésima realização, fazemos: $(x_i)_k$. Assim, temos T amostras coletadas para o problema. Quanto mais amostras, melhor podemos estimar valores de probabilidades. Cabe salientar que, como nosso vetor aleatório não depende do tempo, podemos tratar estes sinais como meros dados amostrados, não interessando a ordem que chegaram, somente suas distribuições de probabilidade.

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T-1}, \mathbf{x}_T] \tag{2.2}$$

ou

$$\mathbf{X} = \begin{bmatrix} (x_1)_1 & (x_2)_1 & \ldots & (x_{T-1})_1 & (x_T)_1 \\ (x_1)_2 & (x_2)_2 & \ldots & (x_{T-1})_2 & (x_T)_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (x_1)_{K-1} & (x_2)_{K-1} & \ldots & (x_{T-1})_{K-1} & (x_T)_{K-1} \\ (x_1)_K & (x_2)_K & \ldots & (x_{T-1})_K & (x_T)_K \end{bmatrix}_{K \times T} \tag{2.3}$$

Gostaríamos de transformar $\mathcal{X}$ em um novo vetor aleatório $\mathcal{Y}$, tal que suas componentes sejam o mais próximos de independentes entre si quanto possível. Para isto, necessitamos encontrar um mapeamento invertível $\mathcal{F}(\cdot)$. Como não conhecemos $\mathcal{X}$, utilizaremos a própria matriz $\mathbf{X}$ das realizações, para se transforma em uma nova matriz $\mathbf{Y}$. Suponha, então, que $\mathbf{x}$ seja alguma coluna de $\mathbf{X}$ e que feita a transformação desta, $\mathbf{y}$ seja alguma coluna de $\mathbf{Y}$.

O conjunto de equações (2.4) apresenta uma coluna arbitrária de $\mathbf{X}$ e suas características. Mostra-se que cada variável aleatória componente tem suporte em um alfabeto finito $\mathcal{A}$. Deste alfabeto, escolhemos K símbolos quaisquer, representados por $a_1, a_2, ..., a_K$, para o cálculo da probabilidade conjunta de todas as variáveis aleatórias serem iguais ao vetor $[a_1, a_2, ..., a_K]^T$, e, à direita do cálculo da probabilidade conjunta, a multiplicação de cada probabilidade marginal correspondente ao i-ésimo símbolo escolhido. Inicialmente, as componentes não são independentes entre si. Caso fossem, o algoritmo encontraria outro mapeamento equivalente ao atual.

$$
\begin{gathered}
\mathbf{x} \in \mathbf{X} \\
\mathbf{x} = [x_1, x_2, \ldots, x_K]^T \\
suporte(x_j) = \mathcal{A}, \forall j \in \{1, 2, \ldots, K\} \\
\forall m \in \{1, 2, \ldots, \|\mathcal{A}\|\}, a_m \in \mathcal{A} \\
Pr(\mathbf{x} = [x_1 = a_1, x_2 = a_2, \ldots, x_k = a_K]^T) \neq Pr(x_1 = a_1) Pr(x_2 = a_2) \ldots Pr(x_k = a_K)
\end{gathered} \tag{2.4}
$$

Assumiremos que as dimensões de $\mathcal{Y}$ são as mesmas de $\mathcal{X}$. Após o mapeamento, ou seja, realizando-se ICA, encontramos uma nova matriz $\mathbf{Y}$, transformada. Extraindo-se uma coluna qualquer desta matriz, esta deverá manter o mesmo alfabeto no suporte das variáveis aleatórias das componentes e, por fim, terá valores muito próximos quanto ao produto das probabilidades marginais ser igual ao das probabilidades conjuntas, indicando independência entre as componentes. Este processo é ilustrado no conjunto de equações (2.5).

$$
\begin{gathered}
\mathcal{F}(\mathbf{X}) = \mathbf{Y} \\
\mathbf{y} = [y_1, y_2, ..., y_K]^T \\
suporte(y_j) = \mathcal{A}, \forall j \in \{1, 2, \ldots, K\} \\
\forall m \in \{1, 2, \ldots, \|\mathcal{A}\|\}, a_m \in \mathcal{A} \\
Pr(\mathbf{y} = [y_1 = a_1, y_2 = a_2, \ldots]^T) \approx Pr(y_1 = a_1) Pr(y_2 = a_2) \ldots Pr(y_k = a_K)
\end{gathered} \tag{2.5}
$$

Nas equações 2.4 e 2.5, temos o conjunto $\mathcal{A}$; um alfabeto finito. ICA também permite alfabetos infinitos, como o próprio $\mathbb{R}$ [Comon 1994, Hyvärinen 2013, Comon 2010], que é um dos cenários de ICA clássico. Entretanto, o objetivo de tornar as componentes independentes não muda com o tipo do alfabeto. A formulação das equações 2.4 e 2.5 para ICA será mantida por todo o trabalho, salvo menção em contrário.

Para determinarmos o mapeamento que produz componentes o mais estatisticamente independentes quanto possível, necessitamos de uma função objetivo. Uma possível função objetivo para este problema é a Correlação Total (TC, do inglês *Total Correlation*), que é uma medida de Informação Mútua,(MI, do inglês *Mutual Information*) generalizada [Watanabe 1960, Cover e Thomas 2012].

Primeiro, esclareceremos a notação empregada para em seguida definirmos a entropia de uma variável aleatória X, cujos elementos se encontram sob um alfabeto finito $\mathcal{A}$ de tamanho $\|\mathcal{A}\|$, para então definirmos a TC, que necessita da definição de entropia.

A equação 2.6 apresenta a notação para a probabilidade da variável aleatória X assumir

um valor i. Bem como a esperança matemática de uma função $g(\cdot)$ desta mesma variável aleatória.

$$\begin{aligned} p_X(i) &= Pr(X = i) \\ E_X[g(X)] &= \sum_{i \in \mathcal{A}} p_X(i) g(i) \end{aligned} \tag{2.6}$$

Desta maneira, a equação 2.7 é a definição de entropia de uma variável aleatória também conhecida como entropia de Shannon [Shannon 1948].

$$H(X) = -\sum_{i \in \mathcal{A}} p_X(i) \log_2 p_X(i) \tag{2.7}$$

Esta função trabalha diretamente sobre a distribuição[1] da variável aleatória X avaliada, não interessando a natureza dos seus símbolos. É correspondente à expectativa mínima de bits que poderia representar a variável aleatória X, isto é: $E_X[-log_2 p_X]$ [Cover e Thomas 2012]. Esta equação é equivalente à esperança matemática de uma função da variável aleatória X: $-log_2(p_X)$, em que X tem distribuição p_X e alfabeto $\mathcal{A}$.

Cabe notar que se pode utilizar qualquer outra base válida para o logaritmo. Inclusive, se escolhermos uma base de valor coincidente com a cardinalidade (ou quantidade de elementos) do alfabeto, confinamos os valores entre 0 e 1 [Yeredor 2011].

Também podemos definir a entropia de um vetor aleatório. Seja $\mathbf{X}$ um vetor aleatório com K componentes $X_1, X_2, \ldots, X_K$. Cada componente assume valores nos respectivos alfabetos $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_K$. Tome $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_K$ como o produto cartesiano destes alfabetos. A entropia deste vetor aleatório é dada pela equação 2.8. Note que as equações 2.7 e 2.8 são muito similares. A única mudança é a disposição do alfabeto, em que tratamos um vetor aleatório como uma mera variável aleatória unidimensional.

$$H(\mathbf{X}) = -\sum_{\mathbf{i} \in \mathcal{A}} p_{\mathbf{X}}(\mathbf{i}) \log_2 p_{\mathbf{X}}(\mathbf{i}) \tag{2.8}$$

A TC de $\mathbf{X}$ é dada pela equação 2.9. Esta é uma medida generalizada de Informação Mútua.

$$I(\mathbf{X}) = \left(\sum_{i=0}^{K} H(X_i) \right) - H(\mathbf{X}) \tag{2.9}$$

A Informação Mútua de duas variáveis aleatórias X e Y nos informa a redução da incerteza de X, dado nosso conhecimento de Y [Cover e Thomas 2012]. Em teoria de comunicações, desejamos que essa associação/atrelamento entre variáveis aleatórias, que representam a entrada e a saída de um canal de comunicação, não seja perdida. Desta

[1] também chamada de Função Massa de Probabilidade (PMF, do inglês *Probability Mass Function*) no caso discreto

maneira, ao tentarmos maximizar este valor com respeito à distribuição de entrada, temos a *Capacidade do Canal*; visto que o sinal com ruído tem de trazer semelhança ao sinal original no destinatário [Shannon 1948, Cover e Thomas 2012].

Para quantidades arbitrárias de variáveis aleatórias, esta medida de correlação nos dará o valor zero quando todas forem mutuamente independentes entre si [Watanabe 1960].

Recomenda-se para o leitor habituado com a noção de correlação usual, isto é, o coeficiente de correlação de Pearson [Casella e Berger 2019], que mede a relação linear entre duas variáveis aleatórias; que não interprete o termo "correlação" utilizado a seguir desta mesma maneira, e sim como uma noção de similaridade.

No artigo em que a TC é apresentada, o autor sugere interpretações intuitivas como "perda de informação", ou "redundância"; dado que quando as variáveis aleatórias não são independentes, elas estão de alguma maneira atreladas (ou correlacionadas) umas às outras, aumentando este valor. A tarefa de ICA, portanto, pretende "quebrar" essa correlação entre as variáveis aleatórias. Isto sugere a possibilidade de tarefas como, por exemplo: compressão de dados, que buscamos diminuir a redundância [Sayood 2012]; criptografia, em que o texto criptografado não deve trazer de maneira explícita nenhum atrelamento ao texto original [Pfleeger e Pfleeger 2007, Ferguson, Schneier e Kohno 2010] (confusão e difusão); códigos corretores de erro [Moon 2005], em que utilizamos de uma redundância intencional de maneira que possamos recuperar os dados de maneira confiável.

A formulação das equações 2.6 e 2.7 assumem um contexto probabilístico. A forma que a entropia é calculada neste trabalho segue a equação 2.10, onde $\widehat{p_X}(i)$ é a frequência relativa do i-ésimo símbolo do alfabeto. Quando $\mathbf{X}$ é uma matriz com as realizações nas colunas, a frequência relativa é calculada pela contagem dos símbolos nas respectivas colunas, para, em seguida, após término desta contagem, dividirmos suas frequências pela quantidade de amostras, dada pela quantidade de colunas. Desta maneira, podemos calcular a estimativa de entropia diretamente da matriz das realizações.

$$\widehat{H}(\mathbf{X}) = -\sum_{i \in \mathcal{A}} \widehat{p_{\mathbf{X}}}(i) \log_2 \widehat{p_{\mathbf{X}}}(i) \tag{2.10}$$

A TC compreende valores não-negativos e encontramos a independência das componentes se, e somente se, assumir o valor 0 (zero). Ou seja, ao executarmos ICA, devemos minimizar a Correlação Total de $\mathbf{Y}$. Desta maneira, também podemos avaliar quais algoritmos estão ao final maximizando melhor a independência entre as componentes.

Da mesma maneira que definimos a entropia considerando que temos estimação perfeita das probabilidades de uma variável aleatória para em seguida trazermos o estimador, façamos o mesmo para a Correlação Total, apresentada na equação 2.11.

$$\widehat{I}(\mathbf{X}) = \left(\sum_{i=0}^{K} \widehat{H}(X_i)\right) - \widehat{H}(\mathbf{X}) \tag{2.11}$$

Pode-se separar os mapeamentos de ICA em dois tipos: um tipo de mapeamento é dito linear se a forma de $\mathcal{F}(\cdot)$ é dada por combinações lineares das realizações das componentes da entrada. Caso contrário, o mapeamento é simplesmente chamado de não-linear. No contexto de alfabetos finitos, a execução de ICA linear se dá sob a teoria de corpos finitos, que apresentaremos a seguir.

2.2 Corpos Finitos

Também chamados de corpos de Galois, em virtude de sua colaboração no estudo dessas estruturas, os corpos finitos detém algumas propriedades que as tornam desejáveis em manipulações algébricas. Suponha agora um alfabeto finito $\mathcal{A}$, com cardinalidade (ou quantidade de elementos) $\|\mathcal{A}\|$. Tome os elementos de $\mathcal{A}$ como: $\{a_0, a_1, a_2, \ldots, a_{\|\mathcal{A}\|-1}\}$. Se $\mathcal{A}$ é munido de uma operação soma $+$ e multiplicação $\cdot$, $(\mathcal{A}, +, \cdot)$, tais que:

$$\forall a, b \in \mathcal{A} \rightarrow a \cdot b, a + b \in \mathcal{A} \tag{2.12}$$

$$\forall a, b \in \mathcal{A} \mid a + b = b + a \ , \ a \cdot b = b \cdot a \tag{2.13}$$

$$\forall a, b, c \in \mathcal{A} \mid a + (b + c) = (a + b) + c \ , \ a \cdot (b \cdot c) = (a \cdot b) \cdot c \tag{2.14}$$

$$\exists\ 0 \in \mathcal{A}\ , \exists\ 1 \in \mathcal{A}\ (1 \neq 0) \mid 0 + c = c \ , \ 1 \cdot c = c \ , \ \forall c \in \mathcal{A} \tag{2.15}$$

$$\begin{gathered} \forall a \in \mathcal{A} \ , \ \exists b \in \mathcal{A} \mid a + b = 0 \\ Se\ a \neq 0, \ \exists c \in \mathcal{A} \mid a \cdot c = 1 \end{gathered} \tag{2.16}$$

$$\forall a, b, c \in \mathcal{A} \mid \ \ a \cdot (b + c) = a \cdot b + a \cdot c \tag{2.17}$$

Ou seja, para que $(\mathcal{A}, +, \cdot)$ seja um corpo, suas operações são fechadas ao próprio corpo (2.12), deve ser comutativo (2.13), associativo (2.14), deve ter um elemento neutro da soma e da multiplicação (2.15), um inverso aditivo (oposto) e um inverso multiplicativo para elementos não-nulos (2.16) e, por fim, deve apresentar distributividade (2.17). Estas propriedades não são estranhas, visto que $\mathbb{R}$ é um corpo.

Pode-se provar que para que esse conjunto $\mathcal{A}$ finito seja um corpo, sua cardinalidade deve ser um número primo ou potência de um número primo [Lang 2002].

Suponha agora que $\|\mathcal{A}\| = P$, onde P é um número primo. Os elementos de $\mathcal{A}$ são todos os restos não-negativos menores que P de números inteiros quando divididos por P. Ou seja, utilizam-se somente os valores de 0 até $P-1$ quanto aos valores possíveis. Dado a, tal que: $a \in \{0, 1, 2, \ldots, P-1\}$ elemento do corpo finito módulo P, sua representação é dada pela equação 2.18.

$$a \ mod \ P \tag{2.18}$$

Lê-se "a módulo P" ou simplesmente: a, quando não é necessário enfatizar qual o número divisor para tomar o resto.

Na equação 2.18, nota-se que $(a+P) \ mod \ P$ representaria o mesmo valor que $a \ mod \ P$, por deter o mesmo resto não-negativo na divisão. Desta maneira, os valores possíveis são cíclicos e confinados num determinado intervalo de números inteiros, que adotaremos como os representantes de classe [Lang 2002].

Não trabalharemos com corpos finitos com cardinalidade de potência de números primos neste trabalho. Corpos finitos em potências de números primos são representados por polinômios e as operações são feitas considerando divisões polinomiais, devido a se tratarem de anéis quocientes (todo corpo é um anel, mas nem todo anel um corpo) e intensiva computação, em relação ao caso de um número primo, somente.

Com vista à reprodutibilidade e o acesso aos códigos deste trabalho [Marcuzzo 2019], era interessante também ferramentas para implementação que fossem de *código aberto* e, para isto, utilizamos a linguagem Octave, para adaptações dos códigos originalmente em MATLAB; também utilizamos Python para organização dos dados gerados. Algumas destas ferramentas, não tem, ainda, implementação para operações com anéis de polinômios e seus conjuntos quocientes. A questão do desempenho destes algoritmos no caso de potências de primos será deixada para trabalhos futuros. Representamos um corpo finito de cardinalidade prima por $\mathbb{GF}(P)$.

Os números em um dado corpo finito $\mathbb{GF}(P)$ detém algumas propriedades úteis: dois números inteiros a e b são ditos congruentes (ou simplesmente iguais, dentro do corpo finito) módulo P se, e somente se, sua diferença $a-b$ é divisível por P, isto é:

$$\begin{gathered} \exists m \in \mathbb{Z} \mid P \cdot m = a - b \\ a = b \ mod \ P \end{gathered} \tag{2.19}$$

Desta definição, seguem as seguintes propriedades:

$$\begin{gathered}\forall a \in \mathbb{Z} \\ a = a \ mod \ P\end{gathered} \tag{2.20}$$

$$\begin{gathered}\forall a, b \in \mathbb{Z} \\ (a = b \ mod \ P) \rightarrow (b = a \ mod \ P)\end{gathered} \tag{2.21}$$

$$\begin{gathered}\forall a, b, c \in \mathbb{Z} \\ (a = b \ mod \ P), (b = c \ mod \ P) \rightarrow (a = c \ mod \ P)\end{gathered} \tag{2.22}$$

Isto nos diz que um número inteiro qualquer é congruente (sempre módulo P) a ele mesmo (2.20), que se a é congruente a b, então b é congruente a a (2.21) e que vale a transitividade (2.22).

2.2.1 A tarefa de ICA em GF(P)

Revisitando as equações 2.4 e 2.5, e assumindo agora que o suporte destas componentes sejam elementos de corpos finitos, pretendemos encontrar um mapeamento $g(\cdot) = \mathbf{W}$ capaz de minimizar a Informação Mútua de $\mathbf{X}$. O mapeamento $\mathbf{W}$ permite a execução de ICA linear.

Como uma transformação linear pode ser representada por uma matriz invertível $\mathbf{W}$ [Strang 2006], o mapeamento linear resultante é dado pela equação 2.23.

$$\mathbf{Y} = \mathbf{WX} \tag{2.23}$$

Por exemplo, tomemos o caso de $GF(2)$, com 2 componentes. A matriz $\mathbf{X}$, das observações, terá em suas colunas as seguintes possibilidades: $(0,0), (0,1), (1,0), (1,1)$. A matriz $\mathbf{W}$ realizará combinações lineares destas componentes para montar novos mapeamentos. Note que, como o caso é linear, o vetor nulo, representado aqui por $(0,0)$, sempre corresponderá a si mesmo em $\mathbf{Y}$. Enquanto que o símbolo $(1,0)$ pode vir a ser representado no mapeamento por $(0,1)$, permutando a posição das componentes originais.

No caso de $GF(2)$ não temos gradação de escalas, contudo, $\mathbf{W}$ com elementos em $GF(3)$ poderia adicionar uma escala aos componentes, além de potencialmente permutar a posição destas, assim, o vetor em $GF(3)$, $(0,1,2)$, com sua segunda componente multiplicada por 2, resultaria em: $(0,2,2)$. Cada componente pode estar multiplicada por um escalar no mapeamento, o que também ocorre no caso de ICA clássico (em alfabetos infinitos como $\mathbb{R}$ ou os números complexos) [Comon 2010].

2.3 O problema da Separação Cega de Fontes

Tomando conhecimento de ICA e apresentados os corpos finitos, podemos mostrar a sua aplicação ao problema de Separação Cega de Fontes (BSS) em corpos finitos.

Seja $\mathbf{S}$ uma matriz $K \times T$, onde K indica o número componentes/fontes e T a quantidade de amostras do problema. As colunas desta matriz são as amostras geradas por um vetor aleatório, e os valores gerados em cada linha é decorrente de uma variável aleatória. Uma coluna genérica de $\mathbf{S}$ é denotada por $\mathbf{s}$. Contudo, $\mathbf{S}$ ao sofrer uma mistura linear invertível resulta em $\mathbf{X}$, representada pela equação 2.24.

$$\mathbf{X} = \mathbf{AS} \tag{2.24}$$

Em que $\mathbf{A}$ é uma matriz quadrada $K \times K$ invertível e $\mathbf{X}$ uma matriz com as mesmas dimensões de $\mathbf{S}$. $\mathbf{X}$ é a chamada matriz das amostras observadas enquanto que $\mathbf{S}$ é a matriz dos sinais originais ou matriz das fontes.

Ao sermos capazes de observar somente as realizações em $\mathbf{X}$, e sabermos somente que $\mathbf{A}$ é invertível, devemos ser capazes de recuperar $\mathbf{S}$ na medida do possível. É disso que consiste o problema de BSS.

Contudo, é possível utilizar a técnica de ICA para resolver esta situação assumindo algumas premissas básicas. Primeiramente, as fontes de $\mathbf{S}$ devem ser estatisticamente independentes entre si. Em segundo lugar, devem ser assumidas dois tipos de ambiguidades: permutação e escala. Estas ambiguidades já eram admitidas no paradigma de ICA contínua ou clássica. Por fim, no caso de ICA em corpos finitos, nenhuma das componentes de $\mathbf{S}$ deve ter distribuição uniforme ou distribuição degenerada[2], visto que torna impossível encontrar uma solução para o problema [Gutch et al. 2012, Yeredor 2011], da mesma maneira que mais de uma fonte gaussiana impedia no caso de ICA clássica [Comon 2010].

O caso de ICA clássica (em $\mathbb{R}$ ou $\mathbb{C}$) permite que a soma de duas variáveis aleatórias independentes não-degeneradas tenha sempre entropia estritamente maior que as entropias de cada variável aleatória individualmente, não importando a distribuição [Yeredor 2011]. No caso de $\mathbb{GF}(P)$, basta que somente uma delas seja uniforme, para que a entropia alcance igualdade antes ou depois de misturar, como ilustrado na equação 2.25 [Yeredor 2011]. A entropia da distribuição uniforme é a máxima em $\mathbb{GF}(P)$. Desta forma, não temos capacidade de distinção do caso não misturado e do caso observado.

$$\begin{aligned} w &= u + v \\ &\Rightarrow H(w) \geq H(u) \\ H(w) = H(u) &\Leftrightarrow u \sim Uniform \end{aligned} \tag{2.25}$$

[2]determinística; um único símbolo detém probabilidade 1

Assim, ao aplicar **ICA**, encontrar-se-á uma matriz **W** de separação, que idealmente inverteria a matriz de mistura **A**; mas dadas a matriz de permutação **P** e a matriz de ambiguidade de escala **D** [Yeredor 2011, Gutch et al. 2012] tem-se a evolução da equação 2.24, representada nas equações em 2.26.

$$\begin{aligned} \mathbf{X} &= \mathbf{AS} \\ \mathbf{WX} &= \mathbf{WAS} \\ \mathbf{Y} = (\mathbf{PDA^{-1}})\mathbf{X} &= (\mathbf{PD})\mathbf{S} \\ \mathbf{Y} = \mathbf{WX} &= \mathbf{US} \end{aligned} \tag{2.26}$$

Portanto, se achamos corretamente uma solução, **W**, para o nosso problema de BSS, então ao realizarmos **WA**, teríamos uma matriz diagonal com suas linhas permutadas; esta matriz é chamada matriz de contaminação e é denotada por **U**. Isto nos permite checar, ao realizarmos simulações, se determinado algoritmo é capaz de resolver o problema. **Y** é a matriz das fontes recuperadas. O problema de BSS é ilustrado pela Figura 2.1. Nesta Figura, observamos uma caixa preta que indica não-acesso ao conteúdo desta caixa, apesar de sabermos que existem K fontes e que elas sofrem um processo de mistura em uma matriz **A**. Na saída desta caixa preta, podemos amostrar as observações para então inferir uma matriz de separação **W**, que irá recuperar as fontes originais, a menos de escala e permutação, como informado anteriormente.

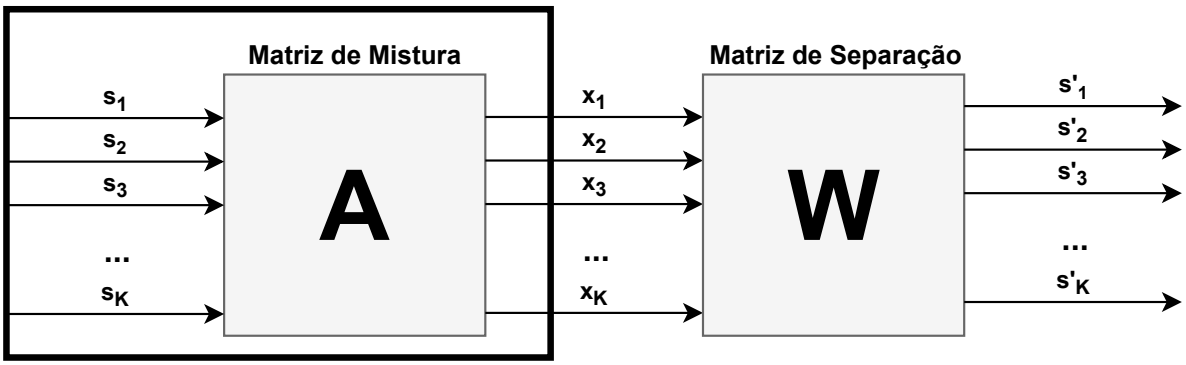

Figura 2.1: Ilustração do problema de BSS. Podendo apenas observar a saída: **x**, encontrar a matriz de separação **W**. A matriz **A** é desconhecida, também não sabemos *a priori*, as fontes originais em **S** .

A seguir, apresentaremos os quatro algoritmos utilizados neste trabalho, em que os três primeiros são lineares e o último não-linear. Reservamos ao contexto do último algoritmo os fundamentos de ICA não-linear e em seguida descrevemos as características do algoritmo.

2.4 O algoritmo AMERICA

A ideia-chave deste algoritmo foi proposta, sem nome, pela primeira vez em [Yeredor 2007]. Yeredor se inspira no trabalho [Delfosse e Loubaton 1995] que utiliza uma técnica de busca gulosa chamada extração/deflação, em que se remove uma fonte de cada vez do problema. A aplicação disto ao contexto de ICA em misturas XOR dá início à área de ICA em corpos finitos. No trabalho [Gutch, Gruber e Theis 2010] temos o *Ascending Minimization of EntRopies for ICA* (AMERICA) sob o nome de "*Algorithm* A", que se baseia em extração/deflação das fontes baseada na entropia, como uma generalização do algoritmo proposto em [Yeredor 2007].

O passo principal deste algoritmo se encontra na equação 2.27. Primeiro contextualizemos os termos da equação. $\mathbf{X}$ é a matriz das observações, $\mathbb{GF}(P)$ designa um corpo finito de ordem P. Suponha K como a quantidade de fontes de um problema de BSS. Então o algoritmo consiste em procurar os vetores $\mathbf{v}_0$ tais que, a cada vez que encontremos um desses (até completar K desses), estes novos vetores não sejam combinações lineares dos encontrados anteriormente (em $\mathbb{GF}(P)$). Portanto:

$$\mathbf{v}_0 = \underset{\mathbf{v} \in \mathbb{GF}(P)^K}{\arg\min} \widehat{H}(\mathbf{v}^T\mathbf{X}) \tag{2.27}$$

Para cada $\mathbf{v}_0$ encontrado, nas K iterações do algoritmo, guardamos a sua transposta e colocamos como uma linha da matriz de separação $\mathbf{W}$.

A ideia desta técnica passa fundamentalmente pela propriedade vista na equação 2.25. Como combinações de variáveis aleatórias independentes no corpo finito não decrescem a entropia, buscamos aquelas combinações que façam a entropia ser recuperada para seu estado original, ou seja, as combinações que minimizam a entropia.

No trabalho [Yeredor 2011], Yeredor aprimora ainda mais este algoritmo e o batiza de AMERICA. A diferença, agora, é que o método trabalha com um tensor de probabilidades, que guardam as probabilidades de cada símbolo/tupla e calcula cada uma das possibilidades para as funções massa de probabilidade dos $\mathbf{v}^T\mathbf{X}$ da equação 2.27[3] passando, antes, por um processo que utilizada a transformada rápida de Fourier K-dimensional com adaptações. Este processo realiza um cálculo equivalente a encontrar todos os $\mathbf{v}^T\mathbf{X} : \mathbf{v} \in \mathbb{GF}(P)^K$ necessários para o processo de otimização, restando apenas calcular as entropias associadas a cada $\mathbf{v}$ com as PMFs encontradas ao fim deste processo. Este mesmo processo nos traz uma diminuição no tempo de execução em relação a calcular diretamente nas amostras $\mathbf{X}$. Vale salientar que nos códigos em MATLAB disponibilizados pelo próprio autor [Yeredor 2011] é esta a versão do algoritmo disponibilizada.

[3]Como visto neste capítulo, podemos calcular a entropia direto das amostras $\mathbf{X}$. O que interessa, na verdade, são as probabilidades dos símbolos. Assim, pode-se simplesmente inserir as probabilidades do símbolos na equação da entropia, aos quais são retornados ao final do processo.

Contudo, em outros trabalhos [Gutch et al. 2012, Gutch, Gruber e Theis 2010], o algoritmo trabalha diretamente com a a matriz das observações $\mathbf{X}$ e não com um tensor de probabilidades estimado.

A seguir, disponibilizamos um pseudo-código do algoritmo utilizado neste trabalho. Como desejamos varrer todas as entropias das combinações lineares, criamos um vetor $\mathbf{C}$, em que guardamos as distribuições estimadas ao sofrerem o processamento da FFT K-dimensional com adaptações, que será aqui tratada como um processo caixa-preta (FFT-K), devido à alta complexidade em sua descrição, mais detalhes em [Yeredor 2011, Yeredor 2011]. Em seguida, sabendo que cada linha tem uma tupla correspondente, calculamos as entropias e as guardamos.

Para cada linha da matriz $\mathbf{W}$, selecionamos a tupla não usada cuja entropia é a menor, marcamos aquela tupla como usada, e todas as combinações lineares das tuplas em $\mathbf{W}$ também. Repete-se esse processo até que $\mathbf{W}$ tenha *rank* (O número de linhas linearmente independentes) completo, i.e, K linhas e colunas.

1. $PMFs \leftarrow$ FFT-K-adaptada(Tensor de Probabilidades oriundo de $\mathbf{X}$)(Equivalente a encontrar todos os $\mathbf{v}^T\mathbf{X}$ e contar a frequência relativa dos símbolos nas colunas)
2. Para cada linha i de $PMFs$, que sabemos a tupla associada, faça:
   (a) $\mathbf{C\_entropias}_i \leftarrow \widehat{H}(PMFs_i)$
3. Marque a entropia associada à tupla nula como usada.
4. Faça até que $\mathbf{W}$ tenha rank completo:
   (a) Encontre o índice (tupla correspondente) de entropia mínima daqueles marcados como não-usados
   (b) Adicione a tupla em uma linha de $\mathbf{W}$
   (c) Marque como usadas as entropias em **C_entropias** que sejam entropias das combinações lineares das tuplas em $\mathbf{W}$
5. Retorne $\mathbf{W}$

Por ser um dos primeiros algoritmos de ICA em alfabetos finitos desenvolvidos e com vasta análise teórica, o AMERICA é um dos algoritmos referência da área, que apresenta maior acurácia que o método MEXICO e o CANADA [Gutch et al. 2012] além de ser mais rápido que o MEXICO, exceto quando $K \gg P$ e a matriz de mistura se assumir esparsa [Yeredor 2011]. Contudo, considerar números primos muito grandes e quantidade de fontes muito grandes torna a execução da técnica passível de estouro de memória,

por precisar alocar tensores de probabilidades muito grandes. O custo computacional do algoritmo é aproximadamente $\mathcal{O}(P^K \times (K^2 + KP + K \log P + P \log P + P))$ [Yeredor 2011].

Como este algoritmo utiliza um tensor de probabilidades, para guardar a PMF conjunta dos símbolos possíveis, o tempo consumido para populacionar esse tensor não é contabilizado no tempo do algoritmo, enquanto as amostras são coletadas. Portanto, este ônus temporal é dispensado do algoritmo. Por fim, enfatizamos que este trabalho adota o algoritmo apresentado em [Yeredor 2011]. O algoritmo não tem hiper-parâmetros para serem definidos.

2.5 O algoritmo SA4ICA

Este algoritmo utiliza a meta-heurística *Simulated Annealing* (SA) para realizar ICA sob corpos finitos [Kirkpatrick, Gelatt e Vecchi 1983], baseada na minimização da entropia entre pares de variáveis aleatórias. Quando $P = 2$ ou $P = 3$, a independência dois-a-dois das variáveis aleatórias garante a independência delas todas entre si [Yeredor 2011]. Contudo, para outros números primos, tal resultado não é garantido. Assim, não seriam necessários varrer todas as combinações possíveis das componentes da entrada como no caso do AMERICA.

Pares de variáveis aleatórias traz em mente o algoritmo *Minimizing Entropies by Exchaging in Couples* (MEXICO), que executa verificação em pares das entropias das fontes a serem extraídas. [Yeredor 2011, Gutch, Gruber e Theis 2010, Gutch et al. 2012, Yeredor 2007, Yeredor 2011] [4]. Por essa similaridade com o MEXICO, o algoritmo SA4ICA é proposto e comparado com esse em [Silva e Attux 2018].

Este trabalho nos indica que, apesar do algoritmo SA4ICA ter custo computacional elevado, este tem desempenho, quanto à separação parcial das fontes, superior ao do MEXICO para cenários similares. Como o custo computacional do MEXICO aumenta exponencialmente com a quantidade de fontes do problema, a partir de determinado K o algoritmo SA4ICA compensará mais o uso, pelo menos para $GF(2)$ e $GF(3)$, pois apresenta comportamento praticamente constante quanto ao aumento de K. Além disso, o MEXICO apresenta decaimento quanto à recuperação das fontes antes do próprio SA4ICA, ao longo da variação de K. Quanto à ICA propriamente dita, isto é, a Correlação Total ao final do processo, o SA4ICA apresenta consistentemente, em $GF(2)$, comparado ao algoritmo não-linear BICA [Painsky, Rosset e Feder 2015] (um caso particular do algoritmo que apresentaremos mais adiante), uma Correlação Total menor. Além de que o BICA, também testado no problema de BSS em [Silva e Attux 2018], começa a decair

[4] Inicialmente com o acrônimo: *Minimizing Entropies of Xored Independent COmponents* [Yeredor 2007].

rapidamente, em relação ao aumento de K e 10000 amostras, a capacidade de separação no problema BSS. Vale lembrar que a métrica de desempenho deste trabalho é dada pela separação parcial e não pela separação total, como utilizada nesta dissertação.

Para fins de contextualização, apresentaremos os termos utilizados na meta-heurística. A meta-heurística é inspirada em um processo da metalurgia chamado *Annealing* ou têmpera, o que traz referências a termos da termodinâmica. O objeto que passa por esse processo é arrefecido e detém uma temperatura T até atingir as características desejadas, dependendo do quão rápido ou lentamente ele resfriou. Quanto mais lento o resfriamento, melhor o processo. E quanto mais rápido, maior é a probabilidade do resultado final não ser tão adequado quanto desejado. Há aqui, portanto, uma barganha entre tempo utilizado e qualidade do objeto temperado [Glover e Sörensen 2015, Kirkpatrick, Gelatt e Vecchi 1983, Russell e Norvig 2009].

A fim de minimizar uma função custo $f(\cdot)$, o SA é normalmente implementado como os passos a seguir. Primeiro, temos uma solução candidata, dada por s_0. Baseada numa temperatura inicial T, o processo de resfriamento dura enquanto esta temperatura for positiva. Repita k vezes os passos seguintes: procuramos na vizinhança da melhor solução atual encontrada até o momento (no início é a solução inicial), uma solução vizinha representada por s_{nova}. Iremos consultar se esse vizinho apresenta aumento de melhoria em relação à solução atual com: $\Delta E = f(s_{nova}) - f(s_0)$. A s_{nova} é melhor se ΔE for negativa, o que indica que atualizaremos a melhor solução atual pela nova encontrada, caso isto não seja verdade, podemos adotar com determinada probabilidade esta vizinha como solução nova, para varrermos o espaço de estado e evitarmos mínimos locais: $rand(0,1) < e^{(-\Delta E/T)}$. Esta etapa é cada vez menos frequente quanto menor for o valor da temperatura e menor for a diferença entre as funções objetivo da atual e da nova solução candidata. O valor β estipula a taxa de resfriamento, atualizamos a temperatura baseado neste valor.

1. Inicialize a solução inicial (candidata) s_0
2. Inicializa a temperatura T
3. Enquanto T for positiva
   (a) Repita k vezes
      i. Calcule a solução atual para uma vizinha baseada em s_0 : s_{nova}
      ii. Calcule $\Delta E = f(s_{nova}) - f(s_0)$
      iii. se $\Delta E < 0$ ou $rand(0,1) < e^{(-\Delta E/T)}$:
      $s_0 \leftarrow s_{nova}$
   (b) $T \leftarrow \beta \cdot T$
4. Retorne s_0

De maneira intuitiva, enquanto o processo está quente, há maiores chances das soluções saltarem pelo espaço de estados inteiro, enquanto que ao encontrarmos soluções melhores, as soluções comecem a oscilar menos e muito próximas umas das outras quanto mais o sistema esfria.

Para o contexto do algoritmo proposto, em $\mathbb{GF}(P)$, P primo, e K a quantidade de componentes do problema, as etapas do algoritmo SA4ICA são:

1. Solução inicial: A matriz identidade I, $K \times K$, que se tornará a matriz de separação
2. Operação de modificação da solução:
   o vizinho da solução candidata $\mathbf{B}$ é:

$$\mathbf{B}_{nova} = \mathbf{I}_{(ij)}(c)\mathbf{B} \tag{2.28}$$

   onde $\mathbf{I}_{(ij)}(c)$ indica uma matriz identidade $K \times K$ tal que na linha i e coluna j, $(i \neq j)$, temos um elemento c do corpo finito $\mathbb{GF}(P)$ escolhido aleatoriamente.

3. Função objetivo:

$$f(\mathbf{B}) = \widehat{H}(b_i^T\mathbf{X}) \tag{2.29}$$

   onde $b_i^T\mathbf{x}$ representação a extração de uma única componente, a i-ésima. b_i^T é a i-ésima linha de $\mathbf{B}$.

O operador modificação da solução muda somente a i-ésima linha de $\mathbf{B}$, que corresponderia a i-ésima fonte do problema, no caso de BSS. A linha modificada é multiplicada por c. Isto nos lembra o operador de comparação empregado pelo MEXICO [Yeredor 2011, Gutch et al. 2012] que não é capaz de escapar de mínimos locais. Porém, a meta-heurística emprega saltos aleatórios, dada a qualidade da solução atual, o que contorna este problema do operador.

Assim como o AMERICA, utiliza-se um tensor de probabilidades para guardarmos as estimativas da PMF conjunta e, desta forma, ao preenchermos esse tensor, não contamos esse ônus temporal do algoritmo.

Os hiper-parâmetros dos algoritmos são T, β e k.

2.5.1 O algoritmo MEXICO

Como comentamos do MEXICO ao apresentarmos o SA4ICA, por servir de inspiração para este, descrevemos aqui, brevemente, o funcionamento deste algoritmo.

O MEXICO foi apresentado primeiramente em [Yeredor 2007] sob o acrônimo *Minimizing Entropies of Xored Independent COmponents*, que mais tarde tomaria o acrônimo *Minimizing Entropies by Exchaging in Couples*, como informado anteriormente.

Em [Gutch, Gruber e Theis 2010], toma o nome "Algorithm B" como uma técnica de "desmistura" baseada em entropia. A ideia é tomar duas das fontes, digamos, X_i e X_j (Linhas i e j da nossa matriz de observações) tais que:

$$\widehat{H}(X_i + c \cdot X_j) < \widehat{H}(X_i) \tag{2.30}$$

Em que c é um escalar não-nulo, para $i \neq j$. Este processo nos recorda um dos passos do AMERICA, pois analisa a entropia da combinação das fontes, exceto que aqui é somente duas-a-duas, ao invés de todas as combinações possíveis entre as fontes. Este processo tem um espaço de busca menor e é suscetível a mínimos locais.

2.6 O algoritmo GLICA

O algoritmo GLICA [Painsky, Rosset e Feder 2018] fora proposto inicialmente para tratar do problema de ICA linear binário e pode ser estendido para cenários P-ários. Consideremos primeiro o caso binário, para depois generalizarmos. Suponha um problema K-dimensional, i.e com K componentes, em que desejamos executar ICA para encontrar as fontes originais.

Tomemos inicialmente uma matriz chamada de $\mathbf{V}$, ela contém 2^K linhas, onde estas linhas representam cada realização possível com K bits. Tome também as X_i, cada i-ésima linha da matriz $\mathbf{X}$. Queremos testar todas as entropias associadas a cada um dos produtos das linhas de $\mathbf{V}$ por $\mathbf{X}$. Para, em seguida, ordenar de maneira ascendente estas entropias, tomando cuidado de ordenar as linhas de $\mathbf{V}$, para saber a qual tupla a entropia está associada. Assim:

1. $\mathbf{C} \leftarrow \mathbf{V}^T\mathbf{X}$
2. Para cada linha i de $\mathbf{C}$
   (a) $\mathbf{C\_entropias}_i \leftarrow H(\mathbf{C}_i)$
3. $\mathbf{C\_entropias} \leftarrow sort(\mathbf{C\_entropias})$,
4. Guarde também a nova ordem dos índices, sendo eles originalmente de 1 até 2^K
5. desconsidere o índice 1, caso originalmente se decidiu gerar os números de $\mathbf{V}$ da tupla nula $[0, 0, 0, \ldots, 0]$ até o último elemento $[1, 1, 1, \ldots, 1]$, ou desconsidere o índice 2^K, caso tenha feito a geração de todas as possibilidades na ordenação oposta.
6. Enquanto a matriz de separação não tiver K linhas preenchidas:

(a) Extraia de **V** a tupla associada à de menor entropia (verifique se esta não é combinação linear de nenhuma outra escolhida anteriormente), esta, na primeira iteração, corresponderá ao primeiro elemento de **C_entropias** e a coloque uma linha abaixo na matriz **W** sendo montada.

(b) Caso sejam combinação linear das linhas de **W**, passe ao próximo índice de **C_entropias**

7. Retorne a matriz **W** encontrada

O mesmo algoritmo pode ser adaptado para casos P-ários, basta gerar as P^K possibilidades e realizar os demais procedimentos do método.

É importante notar que esse algoritmo assemelha-se em alto grau com o método AMERICA apresentado em [Gutch et al. 2012, Gutch, Gruber e Theis 2010, Yeredor 2007]. Contudo, ele difere da versão do algoritmo AMERICA proposta em [Yeredor 2011] por trabalhar diretamente nas amostras **X**, e não com um tensor de probabilidades. Tampouco utiliza a manipulação com transformada rápida de Fourier K-dimensional para otimizar os cálculos associados a $\mathbf{v}^T\mathbf{X}$, como o algoritmo AMERICA faz.

O autor do algoritmo não disponibilizou, até o presente momento, a implementação deste. Por esta razão, o algoritmo fora implementado seguindo as instruções do artigo original e pode ser encontrado em [Marcuzzo 2019] para conferência.

Ainda sobre a distinção entre GLICA e AMERICA: ambos os métodos funcionam sob um esquema de extração/deflação, que é uma estratégia gulosa, na extração individual das componentes, na minimização da entropia e nos testes com as componentes, baseando-se em combinações lineares das já extraídas anteriormente. O AMERICA assume que há fontes independentes e que **X** é oriundo de um modelo gerador linear, enquanto que o método GLICA não assume cenário específico algum [Painsky, Rosset e Feder 2018], ainda que, na prática, busque o mesmo objetivo de extração. O algoritmo GLICA tem custo computacional de aproximadamente $\mathcal{O}(KP^K)$ cálculos de entropia e, no trabalho que é apresentado, fora testado quanto à minimização da Correlação Total com o AMERICA para diversas quantidades de componentes em um problema de BSS, tendo o algoritmo AMERICA obtido valores menores ou iguais ao do método GLICA de acordo com as figuras apresentadas em [Painsky, Rosset e Feder 2018]. Ainda neste mesmo artigo, o autor decidiu não utilizar o algoritmo AMERICA para comparação em um cenário de ICA pura, cuja distribuição geradora da PMF é uma distribuição Zipf (que apresentaremos posteriormente) com representação escolhida aleatoriamente. Mais a frente, testaremos tanto o SA4ICA quanto o AMERICA sob cenário similar ainda que, supostamente, violem a configuração do problema, por assumirem um modelo gerador linear das observações.

Como citado, ao processar a matriz $\mathbf{X}$, diferentemente do AMERICA e do SA4ICA, o GLICA assume um ônus temporal, que depende diretamente da quantidade de amostras observadas, pois é uma das dimensões da matriz $\mathbf{X}$. Assim, é esperado que o tempo de execução do algoritmo aumente quando as dimensões de $\mathbf{X}$ aumentam também. Não há necessidade de se configurar hiper-parâmetros para executar este algoritmo.

Como o código do algoritmo não fora disponibilizado pelo autor, disponibilizamos a nossa implementação do algoritmo em [Marcuzzo 2019].

2.7 *Piecewise Linear Relaxation Algorithm* - QICA

Por fim, temos até então o único algoritmo que realiza ICA não-linear, de maneira genérica, em alfabetos finitos. ICA, como enfatizado anteriormente, visa buscar um mapeamento invertível das amostras observadas $\mathbf{X}$ tal que as amostras geradas $\mathbf{Y}$, tenham, agora, cada componente o mais independentes entre si quanto possível. Antes, presentaremos como se dá ICA não-linear, que se distingue de ICA em corpos finitos, para então prosseguirmos com o algoritmo desta seção.

2.7.1 ICA não-linear

Como informado anteriomente, ICA visa encontrar um mapeamento invertível, ilustrado pela equação 2.31. Seja $\mathcal{F}(\cdot)$ a função invertível que realiza o mapeamento:

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}) \tag{2.31}$$

Este mapeamento, para realização de ICA, não precisa se restringir a transformações lineares. Quando o este mapeamento não é uma transformação linear, não podemos representá-lo, sempre, como uma multiplicação matricial. Apesar disso, podemos enxergá-lo como uma permutação dos vetores-amostra. Por exemplo: suponha um problema em $\mathbb{GF}(P)$ tal que P é o primo 3. Temos como dimensão (no caso de BSS a quantidade de fontes/componentes) $K = 3$. Assim temos P^K vetores possíveis de se observar como amostras em $\mathbf{X}$:

$$
\begin{aligned}
&[0,0,0]^T \\
&[0,0,1]^T \\
&[0,0,2]^T \\
&[0,1,0]^T \\
&[0,1,1]^T \\
&\vdots \\
&[2,2,0]^T \\
&[2,2,1]^T \\
&[2,2,2]^T
\end{aligned}
\tag{2.32}
$$

Nesta disposição, o "primeiro" vetor é $[0,0,0]$ e o "último" é $[2,2,2]$. Vamos batizá-los, respectivamente, de elemento-vazio e elemento-cheio. Poderíamos ver estes vetores como números Q-ários, onde Q, aqui, é o número primo escolhido e K a quantidade de algarismos expostos. Como podemos fazer uma correspondência biunívoca de 1 até $P^K = 27$, a partir do elemento-vazio ao elemento-cheio, podemos representar cada vetor desse por um escalar de 1 até 27.

Assim, se dispusermos esses elementos nesta mesma ordem que aparecem numa lista ordenada, temos:

$$[1,2,3,4,5,6,\ldots,26,27] \tag{2.33}$$

Esta disposição representa a permutação trivial das tuplas possíveis. Ou seja, se realizarmos um mapeamento, cada elemento corresponderá a si mesmo após sofrerem a transformação. Se, por acaso, o primeiro elemento agora fosse se transformar no segundo e vice-versa, a permutação seria representada pela seguinte lista:

$$[2,1,3,4,5,6,\ldots,26,27] \tag{2.34}$$

Esta lista nos indica que o símbolo 1 foi mapeado ao símbolo 2 e o símbolo 2 no símbolo 1. Deixamos os demais elementos nos seus locais originais. Observe que, dessa forma, ICA não-linear permite encontrar mais mapeamentos do que o caso linear. Pode-se inclusive conjecturar que seria possível minimizar melhor a MI em $\mathbf{Y}$, principalmente sobre a premissa de que os dados originais em $\mathbf{X}$ não sejam explicáveis por meio de um processo misturador linear de fontes independentes. Para fins de ilustração, tomemos um exemplo simples em $GF(2)$ com $K = 2$.

A Tabela 2.1 ilustra probabilidades conjuntas de duas variáveis aleatórias, X_1 e X_2. As

somas nas linhas são correspondentes às probabilidades marginais de X_2 e nas colunas as probabilidades marginais de X_1. Note que essas variáveis aleatórias não são independentes pois: $(P(X_2 = 0) = 0.46) \times (P(X_1 = 1) = 0.3) = 0.138 \neq (0.18 = P((X_1, X_2) = (1, 0))$. Contudo, se permutarmos os símbolos $(0, 0)$ e $(0, 1)$ temos a tabela 2.2, que tem as componentes independentes entre si. Note que esse mapeamento é não-linear, pois permutamos o elemento-vazio com um elemento não-nulo. Em ICA linear, o elemento-vazio nunca trocaria de posição.

Como estes mapeamentos por permutação incluem, inclusive, os lineares, se os algoritmos lineares não forem capazes de encontrar um mapeamento que torne as componentes independentes, ICA não-linear teria supostamente a capacidade de encontrá-los.

Tabela 2.1: PMF conjunta de duas variáveis aleatórias não independentes entre si.

P_{X_2} \ P_{X_1}	0	1	
0	0.28	0.18	0.46
1	0.42	0.12	0.54
	0.7	0.3	

Tabela 2.2: PMF conjunta de duas variáveis aleatórias independentes entre si.

P_{X_2} \ P_{X_1}	0	1	
0	0.42	0.18	0.6
1	0.28	0.12	0.4
	0.7	0.3	

É esta lista, ou outra maneira equivalente de representar a permutação, que um algoritmo de ICA não-linear irá retornar ao final de sua execução. Note que, para o exemplo da equação 2.32, temos $27! = 10888869450418352160768000000 \approx 1.08 * 10^{28}$ listas possíveis, o que nos dá um número muito grande para varrermos, de maneira exaustiva, todas as possibilidades em tempo cabível, de maneira serial.

Para fins de ilustração desta dificuldade, suponha que testemos cada uma dessas listas deste problema em um tempo de $0.001ms = 1\mu s$, isto é, $1 \cdot 10^{-6}s$ por teste. Realizando a multiplicação do tempo por lista pela quantidade de listas, temos aproximadamente o tempo total: $1.08 \cdot 10^{22}s = 3.10^{18}$ horas $= 1.25 \cdot 10^{17}$ dias $\approx 3,4 \cdot 10^{14}$ anos. Se tomássemos $1ns$ por teste teríamos $\approx 3,4 \cdot 10^{11}$ anos. Se tomássemos $1ps$ teríamos, ainda, $\approx 3,4 \cdot 10^{8}$ anos.

Para comparar com o caso exaustivo dos algoritmos de ICA linear, cujo objetivo é encontrar uma matriz $\mathbf{W}$, de elementos num corpo $\mathbb{GF}(P)$ com P primo, $K \times K$ que

seja invertível, temos, para $K = 4$, e $P = 3$, um problema de dimensão apenas em uma unidade maior:

$$\mathbf{W} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}_{4\times 4} \tag{2.35}$$

Existem P^{K^2} matrizes possíveis, e não são todas elas que são invertíveis em $\mathbb{GF}(P)$. Veremos a seguir o valor exato para a quantidade de matrizes invertíveis em $\mathbb{GF}(P)$. Uma matriz em $\mathbb{GF}(P)$ é invertível se, e somente se, seu determinante é diferente de zero (assim como no caso de matrizes reais). Para calcular seu determinante, podemos calcular como se os elementos fossem números inteiros e em seguida aplicar *mod* P. A propriedade de multiplicação dos determinantes de matrizes continua valendo, isto é:

$$det(\mathbf{M1}) \cdot det(\mathbf{M2}) = det(\mathbf{M1M2}) \tag{2.36}$$

Isto ocorre devido a operação soma e multiplicação nos números inteiros ser fechada. A aplicação *mod* P é distributiva [Lang 2002]: $(a \ mod \ P) \ * (b \ mod \ P) = (a * b) \ mod \ P$ onde $*$ pode ser tanto o operador soma quanto multiplicação. Assim, podemos fazê-la ao final do processo ou toda vez que realizarmos somas e multiplicações entre inteiros.

Em [Waterhouse 1987] é apresentado a formulação exata para a quantidade de matrizes invertíveis em um corpo finito módulo primo, dada pela equação 2.37.

$$\prod_{i=0}^{K-1} (P^K - P^i) \tag{2.37}$$

Para ilustração de um caso particular: a matriz 2.35 tem quatro linhas. A linha toda preenchida com zeros é proibida, pois levaria a um determinante nulo. Dentre todas as tuplas possíveis P^K, excluímos uma e podemos preencher a primeira linha. A segunda linha não pode ser combinação linear da primeira. Como os escalares possíveis são de 0 até $P - 1$, temos agora $P^K - P$ possíveis tuplas para a segunda linha. A terceira linha não pode ser combinação linear de nenhuma das duas anteriores. Então restam $P^K - P^2$ possibilidades para a terceira linha. E, por fim, para a quarta, somente $P^K - P^3$. O total de matrizes invertíveis, neste contexto, é dado por:

$$
\begin{aligned}
&(P^K - 1)(P^K - P)(P^K - P^2)(P^K - P^3) \\
&= (3^4 - 1)(3^4 - 3)(3^4 - 9)(3^4 - 27) \\
&= (80)(78)(72)(54) \\
&= 24261120
\end{aligned}
\tag{2.38}
$$

Se testássemos exaustivamente todas essas matrizes, cada teste por $1\mu s$ como no caso não-linear, teríamos, para o cenário de comparação com $K = 4$ e $P = 3$:

$$
\begin{aligned}
&24261120 \cdot 10^{-6} s \\
&= 24s
\end{aligned}
\tag{2.39}
$$

O que nos dá no máximo $24s$ para testar exaustivamente todas as possibilidades no mesmo problema, desde que consideremos um mapeamento linear.

2.7.2 O algoritmo QICA

O algoritmo QICA, que é uma generalização do algoritmo BICA para alfabetos Q-ários [Painsky, Rosset e Feder 2016, Painsky, Rosset e Feder 2015] visa buscar a permutação dos símbolos que minimize a TC. O algoritmo adota como função custo a soma das entropias marginais referentes ao vetor de saída, no entanto aproxima a função de entropia, a qual tem $Q-1$ parâmetros, através de uma função linear por partes com k regiões. Através dessas k aproximações por funções lineares, o algoritmo testará diversas configurações que minimizem a soma das entropias marginais.[5]

Assim como no caso do GLICA, descreveremos este algoritmo para o caso binário, dado pelo BICA [Painsky, Rosset e Feder 2015]. Assuma, inicialmente uma função objetivo linear, dada pela equação 2.40. $\mathbf{Y}$ é o mapeamento na saída, e é, portanto, uma variável aleatória binária, com K dimensões. O alfabeto deste vetor aleatório é dada pelo produto cartesiano do alfabeto das marginais: $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_K$, de tal maneira que se fizermos o ordenamento do elemento-vazio ao elemento-cheio de 1 até P^K, o i-ésimo índice, para a PMF conjunta, indicará o i-ésimo símbolo.

$$
\begin{aligned}
L(\mathbf{Y}) &= \sum_{i=1}^{K} a_i Pr(Y_i = 0) + b_i \\
&= \sum_{i=1}^{2^K} c_i Pr(\mathbf{Y} = i\text{-ésimo}) + d_i
\end{aligned}
\tag{2.40}
$$

[5] Assim como ocorre para o SA4ICA, minimizar a soma das entropias marginais é equivalente a minimizar a TC pois, como a $\mathcal{F}(\cdot)$ é invertível, então: $H(\mathbf{X}) = H(\mathbf{Y})$.

Para minimizar essa função custo, basta ordenar, do maior para o menor, as probabilidades conjuntas, e dispor os coeficientes c_i de tal maneira que as probabilidades maiores tenham os menores valores e assim por diante. Contudo, como estamos tratando agora do problema de ICA não-linear, a minimização das entropias marginais é côncava, gostaríamos, para tanto, limitar superiormente, através de aproximações lineares, com k partes. No mesmo artigo, os autores propõem que as aproximações alcançam o problema *hard* de permutação para ICA não-linear o quanto se queira e sugerem o trabalho [Gavrilović 1975] sobre aproximações lineares para maior entendimento da validade das construções a seguir.

Chamemos as expressões de $Pr(Y_i = 0)$ de π_i e que $\sum_i \pi_i = 1, \pi_i \leq 0.5$, tal como no artigo original. Note que os valores de π_i sempre podem ser trocados de tal maneira que a soma destes não muda, então eles podem ser considerados "equivalentes". Desta maneira, podemos resolver o problema linear ao varrermos todas as possíveis combinações na disposição de K variáveis π_i nas k diferentes regiões da função linear por partes. Para cada uma dessas combinações, devemos resolver um problema linear com restrições, tal que restringimos os possíveis valores de π_i para dadas regiões (Para a entropia de uma variável aleatória binária, temos: $h(p) = -(p \log p + (1-p)\log(1-p))$, e dividimos em k regiões para p de tamanho $0.5/k$ distintas. Aqui: $0 \leq p \leq 0.5$.

Temos $\binom{K+(k-1)}{K}$ maneiras de dispor os valores de π_i nestas regiões. Basta agora resolver um problema linear sem restrições para cada uma destas maneiras de dispor. Verifique se a solução está dentro das restrições iniciais. Se não estiver, assume-se que o valor de π_i estar naquela região da restrição é falsa. Contudo, se os valores de π_i atendem à restrição suposta inicialmente, esta disposição é candidata para ótimo global. A complexidade assintótica do algoritmo quando $K \to \infty$ é $\mathcal{O}(K^k \times 2^K)$ [Painsky, Rosset e Feder 2015].

Generalizando para alfabetos Q-ários (daí o nome QICA), devemos nos atentar que as distribuições marginais Y_i seguem distribuições multinomiais, e para isso necessitamos de $Q-1$ parâmetros (no caso binário, somente 1, a probabilidade de ser zero), tal que todos eles não sejam maiores que 0.5. Assim, as regiões que antes se davam no eixo p de $h(p)$, se tornam regiões chamadas *células*. Nem todas as células são visitadas, por não obedecerem as restrições do problema e a quantidade de células visitadas é dada por $C = min(k^{Q-1}, \mathcal{O}(Q^k))$. A complexidade geral do algoritmo é dada por: $\mathcal{O}(K^C \times Q^K)$.

É possível perceber que para alfabetos muito grandes ou de dimensionalidade mais elevada, este algoritmo é inviável. Para tanto, o autor disponibilizou outro algoritmo, heurístico, baseado em gradiente descendente. A seguir apresentamos o seu pseudo-código, inspirado no artigo original [Painsky, Rosset e Feder 2015].

1. Necessita de:

(a) PMF do vetor aleatório $\mathbf{X}$

(b) K número de componentes/a dimensão de $\mathbf{X}$

(c) I = número de iterações para reinício do gradiente

(d) k número de células que estabelecem uma cota superior para o objetivo.

2. min_soma_marg = ∞

3. celula_atual = 0 (a primeira é a zero)

4. temp_solution = ∞ (atual solução do problema linear)

5. index = 1, Solution = ∞

6. Enquanto index $\leq I$, faça:

   (a) celula_atual $\leftarrow$ seleção aleatória de uma combinação de forma a comportar K componentes em k células.

   (b) temp_solution $\leftarrow$ Solução do Problema Linear sem restrição em celula_atual.

   (c) se temp_solution cair dentro dos limites da celula_atual E H(temp_solution) $<$ min_soma_marg então:
   min_soma_marg $\leftarrow H$(temp_solution)
   Solution $\leftarrow$ temp_solution

   (d) index $\leftarrow$ index+1

   retorne Solution

Disponibilizamos o código adaptado do autor em [Marcuzzo 2019], mas os códigos originais do BICA e do QICA podem ser encontradas no site pessoal do autor [Painsky].

O algoritmo consta de duas versões, uma exaustiva e uma que utiliza gradiente descendente. Optamos pela segunda, por ser factível quanto ao tempo em relação ao caso exaustivo. Os hiper-parâmetros para execução no código são o número de regiões/células k mínima, denotada por min_k, que será testada até um k máximo, denotado por max_k e, por fim, uma quantidade de iterações para reinício do gradiente, denotado por I.

2.8 Conclusões

Neste capítulo, apresentamos os algoritmos a serem avaliados neste trabalho. Três deles são recentes, enquanto o método AMERICA é mais antigo e conta, no entanto, com vasta análise teórica. O AMERICA utiliza de um princípio de extração, isto é, recupera uma fonte por vez, baseada nas combinações lineares de menor entropia. O algoritmo GLICA

também emprega mecanismo similar. O SA4ICA é inspirado em uma meta-heurística de busca, a qual que pode valer a pena quando aumentamos a dimensão do problema, além de deter desempenho superior ao do algoritmo que o inspirou, o MEXICO. Por fim, o algoritmo não-linear, QICA, divide em k regiões suas probabilidades marginais e, ao resolver diversos problemas lineares, toma as melhores soluções que encontra. Por ter custo computacional muito elevado, é empregado aqui como uma heurística utilizando gradiente descendente.

No capítulo a seguir, descrevemos os experimentos realizados com as técnicas descritas neste capítulo, quanto à aplicação de ICA em BSS e ICA propriamente dita; bem como as configurações de parâmetros de simulação e a análise do resultado destas simulações.

3 Ensaios Experimentais

Neste capítulo apresentamos os ensaios experimentais que visam comparar os algoritmos simulados tanto em termos de tempo de execução dos algoritmos quanto em taxa de separação perfeita, no caso de BSS. Também avaliamos a tarefa de ICA por si só, medindo o tempo de execução total dos algoritmos e a Informação Mútua das componentes extraídas, dado um modelo gerador para cada símbolo que não é necessariamente linear.

3.1 Experimento de aplicação de ICA em BSS

O experimento em BSS consiste em simulações estocásticas para estimação das médias de desempenho, que são: tempo de execução total para estimar a matriz de separação $\mathbf{W}$ e se o algoritmo foi capaz de recuperar *totalmente* as fontes. Consideramos os três algoritmos lineares apresentados neste trabalho para a preparação do experimento: AMERICA, SA4ICA e GLICA. Não utilizamos o algoritmo não-linear QICA por ter obtido desempenho muito abaixo dos demais algoritmos em ensaios preliminares, apesar de que em seu artigo, o autor propõe BSS como possível aplicação [Painsky, Rosset e Feder 2015] na seção V. Especificamente, testamos no cenário mais básico com $P = 2, K = 2$ e, com o aumento no valor de K, mesmo para quantidade de amostras elevadas, os resultados demonstravam nenhuma mudança. Reparamos, também, que o método mapeava com frequência o vetor de elemento-vazio no elemento-cheio. Isto, no caso de BSS, já garante que não se iria recuperar corretamente todas as fontes. Por se tratar de uma mistura linear, o elemento-vazio deve sempre mapear no elemento vazio.

Para cada fonte em uma determinada simulação, que necessitam ser independentes entre si, escolhemos uma Função Massa de Probabilidade (PMF) que não seja muito próxima da distribuição uniforme. A medida usada para isto é a divergência de Kullback-Leibler (KL). Define-se a divergência de Kullback-Leibler de duas PMFs, p e q tal como a equação 3.1:

$$D_{KL}(p;q) = \sum_{u} p(u) log\left(\frac{p(u)}{q(u)}\right) \tag{3.1}$$

Considera-se $0log\frac{0}{0} = 0$ ou mesmo $0log(0) = 0$.

Tomamos a divergência de Kullback-Leibler de p em relação à uniforme e, se esta for maior que 0.2, aceitamos para cenário de simulação. Isto é:

$$D_{KL}(p; Uniforme) = \sum_u p(u) log\left(\frac{p(u)}{Uniforme(u)}\right) > 0.2 \tag{3.2}$$

Como estamos trabalhando com corpos finitos, temos que escolher números primos com os quais executaremos nossas simulações. Escolhemos de 2 até 7. Para a quantidade de fontes, escolhemos de 2 até 8, com a exceção de que não executamos para o parâmetro do número primo = 7 as simulações com uma quantidade de fontes maiores que 5. Estas simulações não eram viáveis para execução no *hardware* adotado. Além disso, quando conseguíamos resultados preliminares para poucas realizações, o tempo era demasiado longo. Decidiu-se por priorizar outros cenários ou parâmetros. A quantidade de amostras geradas/observadas tomam-se os valores $2^8, 2^9, 2^{10}, 2^{11}, 2^{12}$. A Tabela 3.1 mostra cada um dos parâmetros utilizados na simulação de BSS.

Tabela 3.1: Tabela das configurações do experimento de BSS.

primos (P)	2, 3, 5, 7
quantidade de fontes (K)	2, 3, 4, 5, 6, 7,8
quantidade de amostras (T)	$2^8, 2^9, 2^{10}, 2^{11}, 2^{12}$
algoritmos	AMERICA, SA4ICA, GLICA
quantidade de realizações	50

50 execuções foram realizadas e contou-se o percentual de realizações em que houve separação total e o tempo médio para execução dos algoritmos em linguagem Octave [Eaton et al. 2019]. Alguns destes algoritmos foram confeccionados pelos respectivos autores originalmente para MATLAB, no caso as devidas adaptações, principalmente de questões sintáticas, foram feitas para se executar no ambiente Octave. Quanto aos hiper-parâmetros do SA4ICA, adotamos $T = 1$, $\beta = 0.995$ e $k = 5$, pois são valores já utilizados em trabalho anterior [Silva e Attux 2018], não realizamos *fine-tuning*[1] destes hiper-parâmetros e mantemos o uso por todo o trabalho.

Assim, executamos 50 vezes cada um dos $(4 \times 7 \times 5 \times 3) - (1 \times 3 \times 5 \times 3) = 375$ cenários possíveis, onde um cenário pode ser representado por uma tupla: (número primo escolhido, quantidade de fontes do problema, quantidade de amostras observadas, algoritmo escolhido). Esta estimativa é importante para planejamento das simulações e refinamento dos valores, caso se queira. O tempo total destas simulações foram cerca de 92 horas em um computador com processador Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz e 8GB de memória. Os códigos das simulações podem ser encontrados em [Marcuzzo 2019].

[1] *Fine Tuning* é uma busca dos parâmetros ótimos para a execução de um algoritmo

Quanto à escolha da matriz de mistura **A**, como em 2.24, aos quais as amostras observadas são geradas, utilizamos o procedimento a seguir. Tome **B** com as mesmas dimensões de **W**, da equação 2.35. Cada elemento seu é sorteado aleatoriamente em $\mathbb{GF}(P)$. Em seguida, crie duas matrizes a partir de **B**: **U**, uma matriz triangular superior, cujos elementos da diagonal são 1; e **L**, uma matriz triangular inferior, cujos elementos da diagonal tomam valores não-nulos. Da matriz **B**, para cada elemento da diagonal que for zero, re-sorteie este elemento até que não seja zero, ou, adicione 1 ao valor nulo para que não sobrem zeros na diagonal. Assim, multiplicando **U** por **L**, ambas com determinantes não-nulos, obtemos uma nova matriz **A**, cujo determinante também é não-nulo, como ilustra a equação 3.3. A matriz de mistura **A** é, por fim, construída aleatoriamente pelos elementos de **B**.

$$\begin{pmatrix} 1 & b_{12} & b_{13} & b_{14} \\ 0 & 1 & b_{23} & b_{24} \\ 0 & 0 & 1 & b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \tag{3.3}$$

3.1.1 Resultados

Sumariamente, os resultados deste primeiro experimento indicaram que todos os algoritmos apresentam aumento sistemático da taxa de separação ao aumentarmos a quantidade de amostras, como podemos ver nas Figuras 3.1 e 3.2.

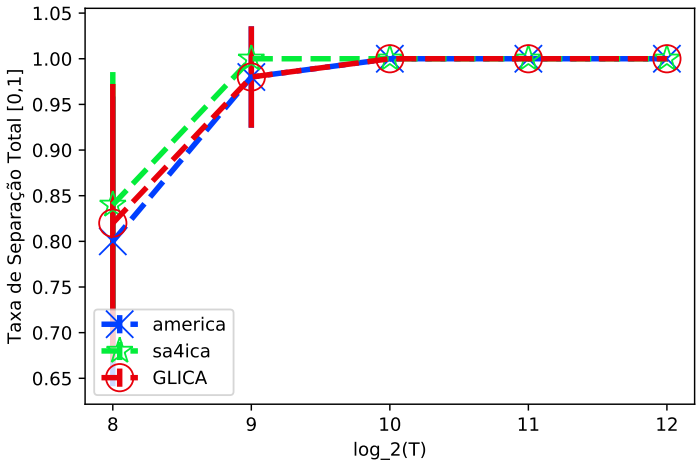

Figura 3.1: Taxa de separação total média ao final de cada técnica, para $P = 2$, $K = 8$, em função de T. Confiança de 95%.

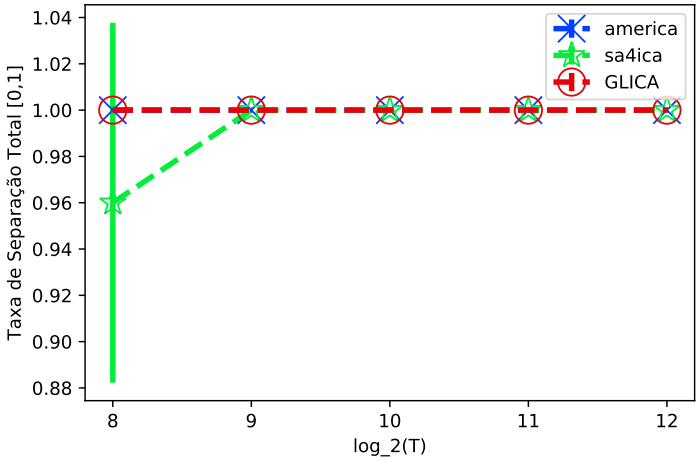

Figura 3.2: Taxa de separação total média ao final de cada técnica, para $P = 3$, $K = 8$, em função de T. Confiança de 95%.

Note que nestas figuras existem barras verticais. Uma barra vertical centrada na média amostral é o intervalo de confiança. Este intervalo correspondente a uma confiança de 95% na estimativa da média amostral, que explicaremos mais adiante. O SA4ICA apresentou, nestes cenários, desempenho inferior quanto à Taxa de Separação em relação aos demais algoritmos; quase sempre minorando o desempenho do GLICA ou do AMERICA.

Mudando para a análise da questão do desempenho ao crescermos o número de componentes (K), percebe-se um comportamento anômalo com o algoritmo SA4ICA. Este algoritmo decresce em desempenho quanto à taxa de separação quando $P = 5$ e K é maior que 6. Esta situação é apresentada na Figura 3.3.

Isto muito provavelmente se deve à dificuldade de varrer o espaço de estados do problema, que se torna muito grande, próprio do funcionamento da meta-heurística. Escolhemos a quantidade de amostras 4096, por melhor representar todos os outros casos para quantidades de amostras fixas.

O intervalo de confiança é composto pela chamada estimativa do erro padrão e pelo coeficiente intervalar [Morettin e Bussab 2010]. Como estamos estimando a média da variável aleatória "Separação Total", o intervalo de confiança apresenta-se como uma barra vertical, simétrica, em torno da média amostral.

Houve diversos cenários em que nenhum dos algoritmos, em nenhuma das realizações, falhou na separação. Assim, é esperado que a estimativa do erro padrão seja zero nestes casos e que não interfira no intervalo de confiança apresentado nos gráficos concernentes à Taxa de Separação Total das Fontes. Para fins de ilustração, de todos os cenários, que totalizam 375, obtivemos suas respectivas médias e, destes valores, 336 destes alcançam

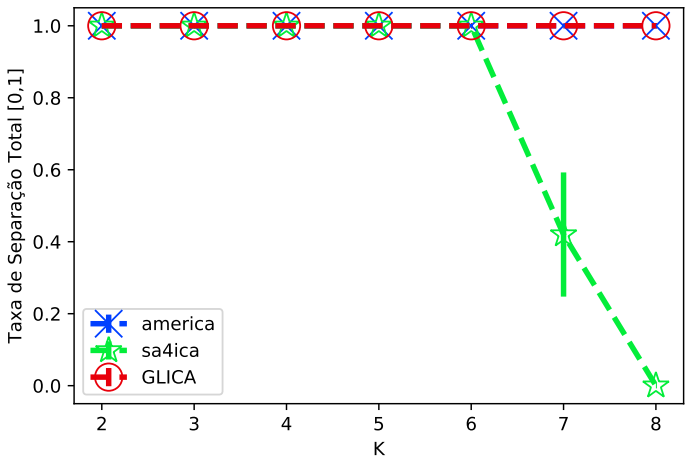

Figura 3.3: Taxa de separação total média ao final de cada técnica, para $P = 5$, $T = 4096$, em função de K. Confiança de 95%.

o valor 1 quanto à Taxa de Separação (100% de Separação), i.e. 89.6% dos cenários.

A formulação que utilizamos para a estimativa do erro padrão é definida pela equação 3.4, onde S é o desvio padrão amostral e n é a quantidade de amostras, neste experimento dada pela quantidade de realizações em cada cenário.

$$\widehat{SE}(\widehat{X}) = \frac{S}{\sqrt{n}} \tag{3.4}$$

Esta estimativa do erro padrão é utilizada para construir intervalos de confiança juntamente com a distribuição de t-student (pois não sabemos o desvio-padrão da variável aleatória) bi-caudal com $n - 1$ graus de liberdade, onde n aqui é quantidade de realizações do experimento, e 95% de confiança para o intervalo. O parâmetro α é relacionado a confiança da seguinte maneira: confiança $= 1 - \alpha$ [Casella e Berger 2019]. Ou seja, pela equação 3.5, temos que a média verdadeira, μ, está neste intervalo com 95% de probabilidade. O valor de $t_{n-1,\frac{\alpha}{2}}$, o coeficiente intervalar, é dado aproximadamente por: 2.01, para $\alpha = 0.05$ e quantidade de realizações $n = 50$.

O valor de $t_{n,\alpha}$ nos indica que, para uma variável aleatória T com distribuição de t-student e n graus de liberdade, desejamos o seguinte valor A, tal que: $Pr(T < A) = 1-\alpha$. Este valor A é $t_{n,\alpha}$. Contudo, o que desejamos aqui é encontrar A, tais que: $Pr(-A \leq \frac{\widehat{X}-\mu}{\frac{S}{\sqrt{n}}} \leq A) = 1 - \alpha$. Desta forma, A assume-se $t_{n-1,\frac{\alpha}{2}}$.

Assim, o coeficiente intervalar multiplicado pela estimativa do erro padrão é o raio do intervalo de confiança.

$$\widehat{X} - t_{n-1,\frac{\alpha}{2}}\widehat{SE}(\widehat{X}) \leq \mu \leq \widehat{X} + t_{n-1,\frac{\alpha}{2}}\widehat{SE}(\widehat{X})$$
$$\mu \in \left[\widehat{X} - t_{n-1,\frac{\alpha}{2}}\widehat{SE}(\widehat{X}), \widehat{X} + t_{n-1,\frac{\alpha}{2}}\widehat{SE}(\widehat{X})\right] \tag{3.5}$$

Pôde-se notar também que, quando o SA4ICA apresenta queda abrupta de desempenho quanto à separação, a sua Correlação Total (Informação Mútua) também aumenta e se distancia dos respectivos valores para os outros algoritmos. Isto reforça a noção desta métrica como um indicador vinculado à capacidade de separar totalmente as fontes. A Figura 3.4 apresenta esta tendência. Os algoritmos GLICA e AMERICA se encontram quase sempre empatados junto do SA4ICA até quantidade de fontes igual a 6.

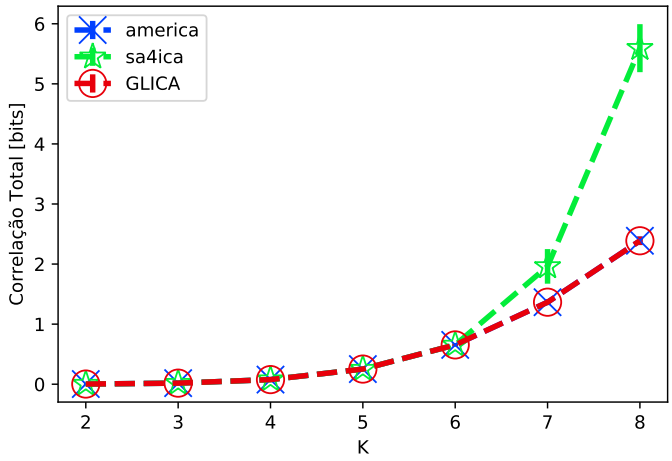

Figura 3.4: Correlação Total média ao final de cada técnica, para $P = 5$, $T = 4096$, em função de K. Confiança de 95%.

Um intervalo de confiança, em se tratando das métricas de Separação ou de tempo de execução, pode não ser visível "a olho nu" por ser de ordem de grandeza muito inferior à própria escala do gráfico. Por exemplo: na Figura 3.4 dos pontos $2, 3, 4, 5, 6$, para todos os algoritmos, as médias amostrais demonstram-se iguais e não se pode perceber o intervalo de confiança, que também se mostram iguais para cada algoritmo. Apresentamos estes valores para estes intervalos de confiança na Tabela 3.2.

Quanto ao tempo de execução dos algoritmos, o AMERICA é consistentemente a técnica mais rápida, enquanto o GLICA se mantém em segundo lugar e o SA4ICA como o mais lento destes. Podemos observar o desempenho temporal dos três algoritmos fixando duas características do problema e variando uma delas. Utilizamos escala logarítmica para melhor visualização. Primeiramente, temos a Figura 3.5 que mostra o crescimento do tempo de execução dos algoritmos para $P = 5$ e $T = 4096$.

Tabela 3.2: Raios dos intervalos de confiança para o cenário $P = 5$, $T = 4096$, para a Correlação Total.

K	Raio do intervalo de confiança
2	$3.22 \cdot 10^{-4}$
3	$8.54 \cdot 10^{-4}$
4	$2.885 \cdot 10^{-3}$
5	$8.562 \cdot 10^{-3}$
6	$2.0344 \cdot 10^{-2}$

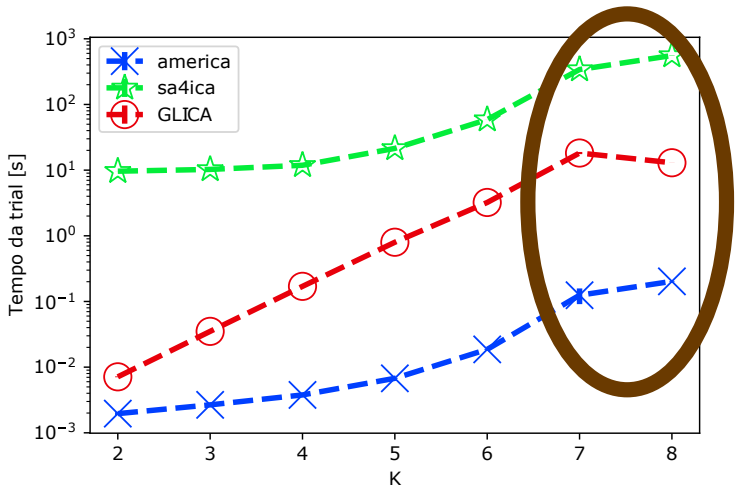

Figura 3.5: Tempo médio da realizações para cada técnica, $P = 5$, $T = 4096$, em função de K. Confiança de 95%. Neste cenário, a pequena queda se justifica pela necessidade da execução em outro computador com algumas adaptações nos algoritmos. Isto não diminui a validade quanto aos testes estatísticos, pela natureza da análise ser relativa, e não absoluta.

Note que, para o algoritmo GLICA, pela escala do gráfico ser logarítmica e o crescimento do tempo de execução nesta escala se apresentar aproximadamente linear, seu crescimento é exponencial em relação ao número de fontes. Os demais algoritmos detém comportamento exponencial mesmo na escala do gráfico, mas para o AMERICA, em ordens de grandeza inferior que a dos outros dois algoritmos. Quanto ao caso $P = 7$ e $T = 4096$, temos a Figura 3.6.

O gráfico é interrompido para K maior que 5. Como informado anteriormente, a simulação deste cenário envolveu limitações em termos de memória e tempo de execução. Conseguimos observar aqui, também, um crescimento exponencial do tempo de execução dos algoritmos. Mantendo ainda a ordem de superioridade destes.

Podemos avaliar o tempo de cada algoritmo pela quantidade de amostras, tal como a Figura 3.7. Outras visualizações são possíveis e mostram comportamentos relativos entre

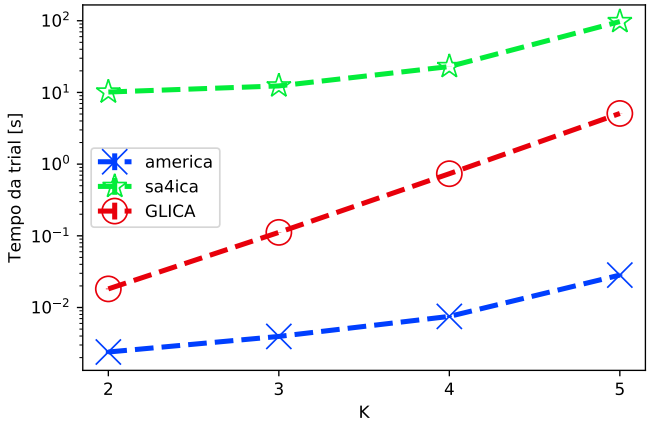

Figura 3.6: Tempo médio da realizaçõe para cada técnica, $P = 7$, $T = 4096$, em função de K. Confiança de 95%.

os algoritmos que são similares em relação ao crescimento da quantidade de amostras T.

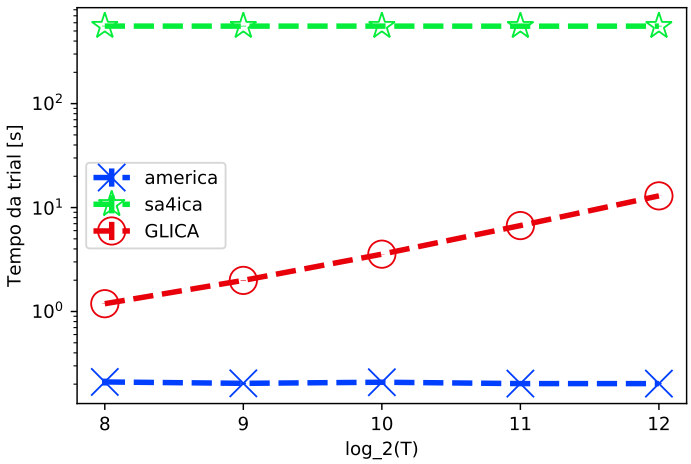

Figura 3.7: Tempo médio da realizaçõe para cada técnica, $P = 5$, $K = 8$, em função de T. Confiança de 95%.

O algoritmo GLICA apresenta, novamente, crescimento linear, e tanto o SA4ICA quanto o AMERICA são praticamente constantes em relação à quantidade de amostras. Estes trabalham com o tensor de probabilidades, como informado no Capítulo 2. Desta maneira, o ônus temporal da etapa de populacionar o tensor de probabilidade não é contabilizada, diferentemente das operações com a matriz de observações, cujo tamanho depende diretamente de T, quando se utiliza o algoritmo GLICA.

Por fim, podemos visualizar o tempo de execução dos algoritmos em relação ao crescimento da ordem do alfabeto, como ilustrado na Figura 3.8. Novamente, temos comportamentos exponenciais ao crescermos um dos parâmetros. Se crescermos P e K simultaneamente, temos um comportamento "duplamente exponencial". Isto nos indica que para valores muito grandes de P e K, todos estes algoritmos apresentariam tempos de execução demasiadamente grandes.

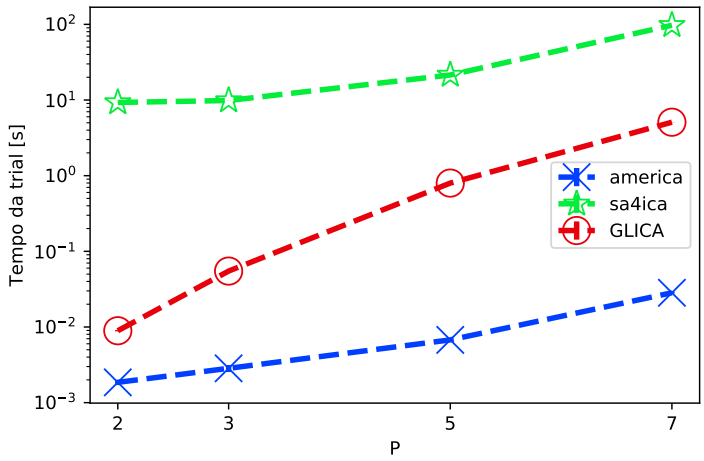

Figura 3.8: Tempo médio da realizaçõe para cada técnica, $T = 4096$, $K = 5$, em função de P. Confiança de 95%.

Um outro aspecto aspecto importante a se considerar é o custo em memória dos algoritmos. Para os tensores de probabilidades, devemos reservar P^K posições de probabilidades, e se cada probabilidade for designada por um ponto flutuante de 64 bits, temos, por exemplo, em $P = 5$, $K = 8$ e $T = 4095$, $64 \times P^K \approx 3.1$ Megabytes. A matriz de observações será $64 \times 8 \times 4096 \approx 2 \times 10^2$ Kilobytes de dados (e a das fontes também), cenário este que começa a se tornar inviável para armazenamento dos dados ou execução das simulações para algoritmos mais lentos, como o SA4ICA e o GLICA, este último com tempo de execução dependendo diretamente do tamanho de T. Se desejássemos simular um cenário com $P = 11$ e $K = 10$, teríamos, para um tipo de dado de 32 bits, 829997587232bits $\approx 1.03 \times 10^{11}$ bytes apenas para o tensor de probabilidades, aproximadamente 660 Gigabytes.

Por estas razões, somente as simulações para $P = 5$ e $K = 8$, executaram com um *script* específico em MATLAB e algumas modificações nos *scripts* invocados, em que os pontos flutuantes tomavam valores de 32 bits ao invés de 64, além de tratar alguns números inteiros, para a função do algoritmo GLICA, de forma a ocupar menor tamanho na memória. Daí, por esta mesma razão, percebe-se leve queda no ponto final da respectiva

curva de tempo de execução, na Figura 3.5, por se tratar desta simulação em ambiente MATLAB realizado em outro computador mais potente.

Para fins de comparação estatística em todos os cenários, utilizamos o teste de Wilcoxon [Wilcoxon 1992]. Os resultados dos testes quanto a Taxa de Separação Total de determinado algoritmo serem maiores que a de outro se encontram na Tabela 3.3. Nota: tanto o método AMERICA quanto o GLICA quando testados em relação ao SA4ICA, por terem maior taxa de separação que este, apresentam os mesmos resultados, bastando substituir na tabela "AMERICA", por "GLICA".

Para todos os testes realizados, mostramos apenas os p-valores em que obtivemos rejeição da hipótese nula, isto é, ao assumirmos que a capacidade de separação do SA4ICA é superior que a do AMERICA, temos um p-valor inferior a 0.05, que é a significância adotada neste trabalho. Por conseguinte, inferimos que a hipótese alternativa é a verdadeira: que o método AMERICA tem maior capacidade de separação que o método SA4ICA. Desta forma, apresentam-se somente os cenários cuja significância foi menor que 0.05 para a hipótese nula. Os testes envolvendo GLICA e AMERICA demonstraram-se inconclusivos e, portanto, não são apresentados. Afinal, estes dois algoritmos apresentam-se praticamente iguais em seus resultados. Haja visto que todos os algoritmos alcançavam, também, Separação Total na maioria dos cenários.

Tabela 3.3: Teste de Wilcoxon entre os algoritmos quanto a Separação Total das fontes.

p-valor: Hipótese nula: (AMERICA $\leq$ SA4ICA)	Médias para $P =$
$4.69 \cdot 10^{-4}$	5
$2.93 \cdot 10^{-2}$	7

Os resultados quanto ao teste de Wilcoxon comparando tempo de execução, especificamente do método AMERICA ser maior ou igual que ao tempo do método GLICA se encontram na Tabela 3.4. Não apresentamos os mesmos testes quanto à comparação do tempo de execução do algoritmo AMERICA em relação ao algoritmo SA4ICA nem o teste para o algoritmo GLICA em relação ao algoritmo SA4ICA por estes retornarem exatamente os mesmos resultados no teste. Ou seja, se substituíssemos na Tabela "AMERICA $\geq$ GLICA" por "GLICA $\geq$ SA4ICA" ou mesmo "AMERICA $\geq$ SA4ICA", obtemos os mesmos p-valores encontrados para a hipótese nula. Inferimos, portanto, que o AMERICA é o mais veloz destes e que o GLICA é mais veloz que o SA4ICA.

Os testes confirmam a intuição de que o algoritmo AMERICA, no geral, é o algoritmo mais adequado tanto em termos de Separação Total, quanto em tempo de execução em relação ao algoritmo GLICA e ao algoritmo SA4ICA; seguido do método GLICA. Como os testes se fundamentam na rejeição da hipótese nula, não podemos inferir que o GLICA ou o AMERICA sejam superiores um em relação ao outro quanto à taxa de Separação

Tabela 3.4: Teste de Wilcoxon entre os algoritmos quanto ao tempo de execução dos algoritmos.

p-valor: Hipótese nula: (AMERICA $\geq$ GLICA)	Médias para $P =$
$1.23 \cdot 10^{-7}$	2
$1.23 \cdot 10^{-7}$	3
$1.23 \cdot 10^{-7}$	5
$4.42 \cdot 10^{-5}$	7

Total. Isso já era esperado, dado que o funcionamento de ambos os algoritmos é muito similar.

3.1.2 Conclusões

Em [Silva e Attux 2018] o algoritmo SA4ICA apresenta uma quantidade praticamente constante da média avaliações da função-custo em relação ao algoritmo MEXICO. Imagina-se, portanto, que o tempo de execução seja, também, praticamente constante em relação a K. Apesar disso, em se tratando de tempo de execução, o algoritmo não apresenta tempo constante, uma das razões de ser potencialmente mais vantajoso que o MEXICO. Este comportamento foi evidenciado nas Figuras 3.5 e 3.6, quanto ao tempo de execução. Além disso, apresenta comportamento anômalo e uma queda abrupta na separação a partir de $K = 6$ quando $P = 5$, não reportada em trabalhos anteriores. Apesar de convergir para 100% de Separação Total quando a quantidade de amostras aumentam, como esperado, minora os outros algoritmos quanto a esta métrica.

A Correlação Total do SA4ICA nos cenários anômalos apresenta-se maior que a dos respectivos valores para os métodos AMERICA e GLICA. Estas técnicas, aliás, obtiveram medidas quase sempre iguais seja sob a ótica de Correlação Total ou sob a perspectiva da Taxa de Separação Total, que costumam apresentar intervalo de confiança não-nulo somente nas estimativas com os menores T observados. Era de se esperar que, como os algoritmos AMERICA e GLICA detém funcionamento muito similares, seus respectivos desempenhos fossem similares também. Apesar dos testes de hipóteses com o teste de Wilcoxon se apresentarem inconclusivos ou falharem, por terem muitos empates no processo de ranqueamento.

Por fim, infere-se que o algoritmo AMERICA, entre os três métodos, é o mais vantajoso, tanto em termos de tempo de execução, quanto em Separação Total das fontes.

3.2 Experimento de ICA pura

Para o experimento de ICA pura, tem-se os seguintes parâmetros de simulação, alguns deles similares ao casos de BSS: tamanho do alfabeto (P), 2 e 3, pois a partir de 5 ocorria um ônus temporal muito elevado para avaliação; a dimensão do problema (K) toma somente os valores de 2 a 5. Para as quantidades de amostras observadas, escolhemos duas: 2^{13} e 2^{14}. A fim de termos uma boa estimativa das probabilidades para aplicarmos ICA é necessária uma quantidade razoável de amostras observadas, tal que o foco aqui é avaliar como se dá a busca no espaço de estados feita por cada algoritmo. Por fim, adicionamos o algoritmo de ICA não-linear QICA para a lista de técnicas a serem testadas. O tempo total das simulações foi de, pelo menos, 146 horas, para 640 cenários de simulação. Aqui os cenários são dados pela tupla: (tamanho do alfabeto, dimensão, quantidade de amostras, algoritmo, distribuição). Realizaram-se 50 vezes cada cenário, a partir dos resultados armazenados, computamos médias e outros valores estatísticos de interesse, da TC e do tempo de execução dos algoritmos.

Quanto aos parâmetros, os algoritmos realizaram com as mesmas configurações do cenário de BSS. Quanto ao método QICA, usamos o método de gradiente descendente no lugar do exaustivo, pelo segundo tomar tempos de execução inviáveis, e os hiper-parâmetros os mesmos valores apresentados ao final do Capítulo 2.

A base do experimento de ICA é de que, a partir da função massa de probabilidade (ou distribuição) conjunta, geramos a matriz de observações. Optaram-se por diversas distribuições de probabilidade, falemos primeiramente da Zipf, batizada de "Zipf pura", dada pela equação 3.6. Aqui temos a probabilidade da variável aleatória X assumir o valor do i-ésimo símbolo, aqui denotado por $\mathbf{x}_i$. s é um parâmetro que pode assumir valores não-negativos. P^K é a quantidade total de símbolos, dado que nosso alfabeto tem cardinalidade P para K dimensões.

$$P(X = \mathbf{x}_i \mid s, P^K) = \frac{i^{-s}}{\sum_{l=1}^{P^K} l^{-s}} \tag{3.6}$$

Usa-se a distribuição Zipf por ser uma distribuição que corrompe o modelo gerador linear usual, além de ser uma distribuição que surge em fenômenos da linguagem, Economia, entre outros [Zipf 2016, Painsky, Rosset e Feder 2015]. O parâmetro s da distribuição Zipf toma o valor 1.05, por ser o mesmo sugerido na implementação [Painsky], para fins de comparação. O esquema que utilizamos é o seguinte: o primeiro elemento a ser sorteado da Zipf representa o elemento-vazio e de maneira crescente até o elemento cheio tal como apresentado em na Seção 2.7.1. O elemento-vazio é representado por um índice de número 1. O último pelo índice P^K. A Figura 3.9 mostra a distribuição da Zipf pura para um alfabeto de tamanho 27 ($P^K = 3^3$) e $s = 1.05$.

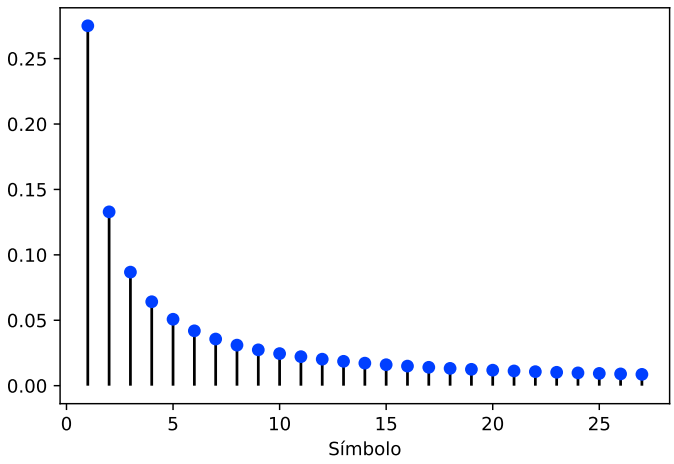

Figura 3.9: Distribuição Zipf pura para alfabeto de tamanho 27, com s=1.05.

Note que, dessa forma, o esquema gerador de símbolos *não* segue um processo de mistura linear, tal como no experimento de BSS. Adotamos também uma distribuição Zipf em que embaralhamos as probabilidades das variáveis conjuntas, da mesma maneira que se sugere na seção (III,E) em [Painsky, Rosset e Feder 2018], para que as tuplas tenham representações aleatórias. Batizamos estas distribuições de "Zipf embaralhada". A Figura 3.10 ilustra uma distribuição possível da Zipf embaralhada.

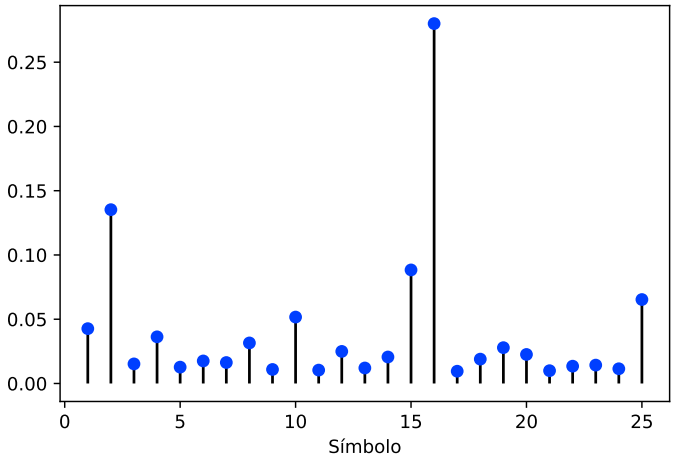

Figura 3.10: Uma possível distribuição Zipf embaralhada para alfabeto de tamanho 25, com s=1.05.

Também testamos distribuições binomiais, seguindo o mesmo esquema de ordenamento dos símbolos da Zipf pura, com o parâmetro p de "sucesso" da binomial tendo

variações. A razão de uso destas distribuições é de caráter exploratório e devido à sua grande popularidade dentro da literatura de estatística e probabilidade. O uso da binomial segue a equação 3.7 para as probabilidades dos símbolos, dada uma probabilidade de sucesso p.

$$P(X = \mathbf{x}_i \mid P^K, p) = \binom{P^K - 1}{i - 1}(p)^{i-1}(1-p)^{((P^K-1)-(i-1))} \\ i \in \{1, 2, \ldots, P^K\} \tag{3.7}$$

Para fins de exemplo, as Figuras 3.11 e 3.12 representam distribuições binomiais para $p = 0.2$ e $p = 0.8$ para um alfabeto de tamanho 16, respectivamente. Note como a escolha da probabilidade de sucesso translada o cume da distribuição .

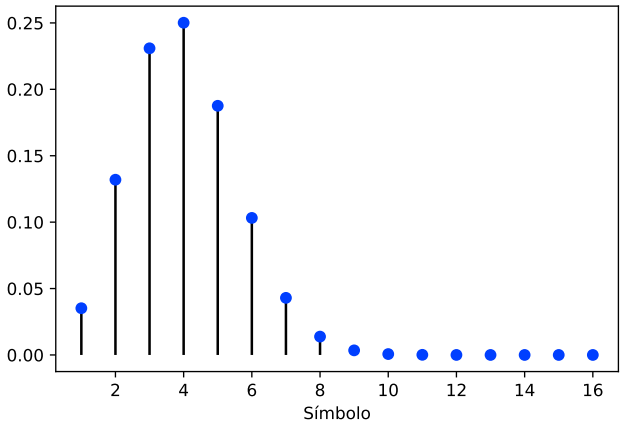

Figura 3.11: Distribuição binomial com alfabeto de tamanho 16 e probabilidade de sucesso 0.2.

Por fim, escolhemos distribuições conjuntas cujas probabilidades são definidas de maneira aleatória. Os cenários possíveis estão representados na Tabela 3.5. Os hiperparâmetros para o algoritmo QICA adotados foram: $min\_k = 2$, $max\_k = 10$, $I = 1000$. Configuração aos códigos exemplo presentes em [Painsky] e dos adotados no trabalho [Silva e Attux 2018], para o BICA.

3.2.1 Resultados

Pode-se observar que o algoritmo QICA somente foi necessariamente menor, em relação à Informação Mútua, para o caso de $P = 2$ e $K = 3$, quando a distribuição é a Zipf pura em relação aos demais algoritmos e cenários. Isso está ilustrado na Figura 3.13. Assim, mesmo com a possibilidade teórica de desempenhar melhor que os algoritmos lineares,

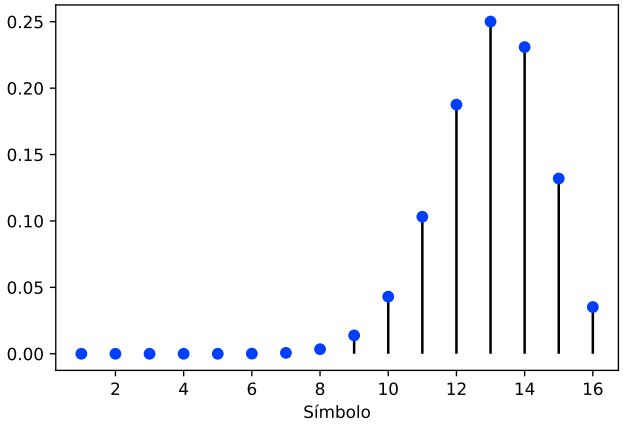

Figura 3.12: Distribuição binomial com alfabeto de tamanho 16 e probabilidade de sucesso 0.8.

Tabela 3.5: Tabela das configurações do experimento de ICA pura.

cardinalidade (P)	2, 3,
dimensão (K)	2, 3, 4, 5
quantidade de amostras (T)	$2^{13}, 2^{14}$
algoritmos	AMERICA, SA4ICA, QICA, GLICA
distribuições	zipf pura, zipf embaralhada, binomial, aleatória
prob. sucesso binomial	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
quantidade de realizações	50

para todos os outros casos o QICA se demonstra inferior, sob esta métrica, nos cenários avaliados.

Outra visualização possível, agora enfatizando as distribuições, para dados P, K, T fixos, nos revela, quando variamos as funções massa de probabilidade, como desempenham os algoritmos. Um exemplo é ilustrado na Figura 3.14.

No geral, das distribuições, uma das menos desafiadoras para todos os algoritmos foi a Zipf pura (representada na Figura 3.14 como "pure_zipf"). Contudo, se tirarmos os valores mínimos de Correlação Total para cada (P, K, T) e tomarmos aquela distribuição correspondente a esta Informação Mútua, temos a Tabela 3.6. Para fins de nomenclatura, as distribuições correspondem nos gráficos e tabelas a seguir da seguinte maneira: a distribuição Zipf pura é a "pure_zipf", a Zipf embaralhada é a "shuffled_zipf", as diversas PMFs binomiais, como dependem de um parâmetro p variante de 0.2 a 0.8, são designadas por "binomialpxx", em que "xx" é substituído pelo valor de p sem o ponto decimal. Por fim, a PMF de distribuições aleatórias é representada por "random_pmf".

Observa-se que as distribuições "mais fáceis" de minimizarem a MI para os algoritmos

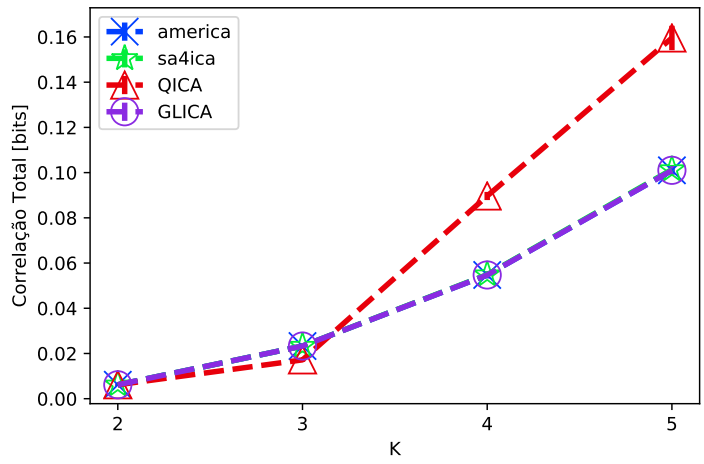

Figura 3.13: Correlação Total média para $P = 2$ e $T = 16384$, distribuição Zipf pura. Confiança de 95%.

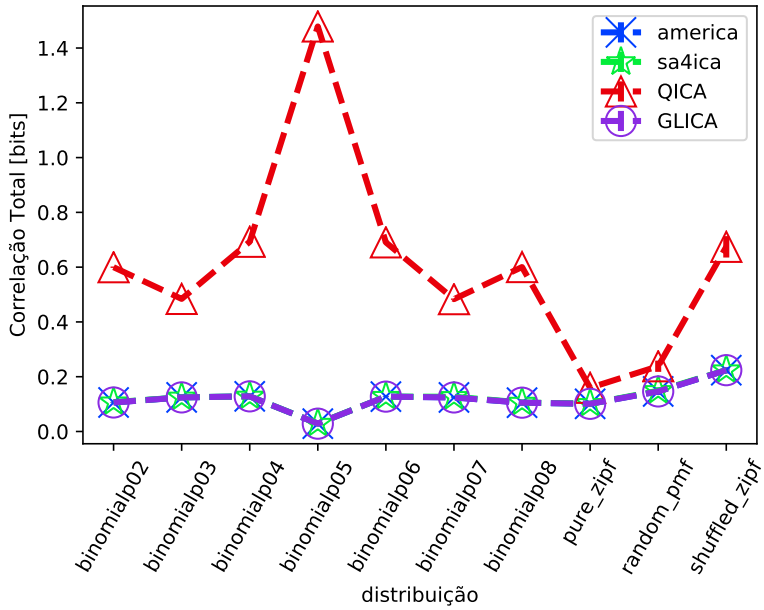

Figura 3.14: Correlação Total Média para $P = 2$, $K = 5$ e $T = 16384$, para diversas distribuições avaliadas. Confiança de 95%.

lineares coincidem. A PMF binomial com $p = 0.5$ foi a mais frequente dentre as que estes algoritmos conseguiram minimizar melhor para dados P, K e T.

Agora, se observamos o cenário oposto, as maiores Correlações Totais, dados (P, K, T) para então extrairmos a distribuição para qual isto ocorreu, temos a Tabela 3.7.

Tabela 3.6: Contagem das menores Correlações Totais em distribuição para todos os cenários (P, K, T) em ordem decrescente.

AMERICA	binomialp05: 8, pure_zipf: 6 binomialp02: 2
SA4ICA	binomialp05: 8, pure_zipf: 6 binomialp02: 2
QICA	pure_zipf: 12, random_pmf: 2
GLICA	binomialp05: 8, pure_zipf: 6 binomialp02: 2

Tabela 3.7: Contagem das maiores Correlações Totais em distribuição para todos os cenários (P, K, T) em ordem decrescente.

AMERICA	shuffled_zipf: 8, binomialp03: 3, binomialp04: 2, binomialp05: 2, binomialp07: 1
SA4ICA	shuffled_zipf: 8, binomialp03: 3, binomialp04: 2, binomialp05: 2, binomialp07: 1
QICA	binomialp05: 8, binomialp08: 3, binomialp07: 3, binomialp02: 1, binomialp04: 1
GLICA	shuffled_zipf: 8, binomialp03: 3, binomialp04: 2, binomialp05: 2, binomialp07: 1

Percebe-se que para a distribuição binomialp05, o algoritmo QICA apresenta maiores dificuldades para minimizar a MI, enquanto os algoritmos lineares tem como distribuição com a menor MI a própria binomialp05. Novamente, temos empates quanto à contagem dos valores por distribuição nos algoritmos lineares. A distribuição mais desafiadora para os algoritmos lineares, pelo critério de contagem, foi a Zipf embaralhada, que é a distribuição de representação aleatória proposta como não-linear em [Painsky, Rosset e Feder 2015]. Ainda assim, os algoritmos lineares costumam ter desempenho superior ao QICA.

Pode-se observar também, tanto na Figura 3.14 quanto em 3.15, um comportamento praticamente crescente, ao aproximarmos o parâmetro p da binomial para 0.5 tanto pela esquerda, quanto pela direita, para o método QICA. Em contrário, o ponto binomialp05 representa um comportamento praticamente oposto para os algoritmos lineares.

Para $P = 3$, quanto ao método QICA, temos uma curva acentuada para cima a partir de binomialp03, decrescendo para binomialp04. Em seguida, uma inflexão e retorno do crescimento até atingir seu máximo em binomialp07. Podemos considerar que aqui temos uma "região de pico"; diferentemente do caso $P = 2$, em que valores são menos acentuados em distribuições específicas para a binomial. Isto está ilustrado na Fig 3.16.

Considerando o tempo de execução, tanto a técnica QICA quanto a técnica SA4ICA demonstram serem praticamente constantes em relação à dimensão da entrada para $P = 2$ e $P = 3$, o que assintoticamente poderia apresentar uma vantagem. Contudo, o QICA é cerca de 5 ordens de grandeza mais lento que o AMERICA, para um desempenho inferior.

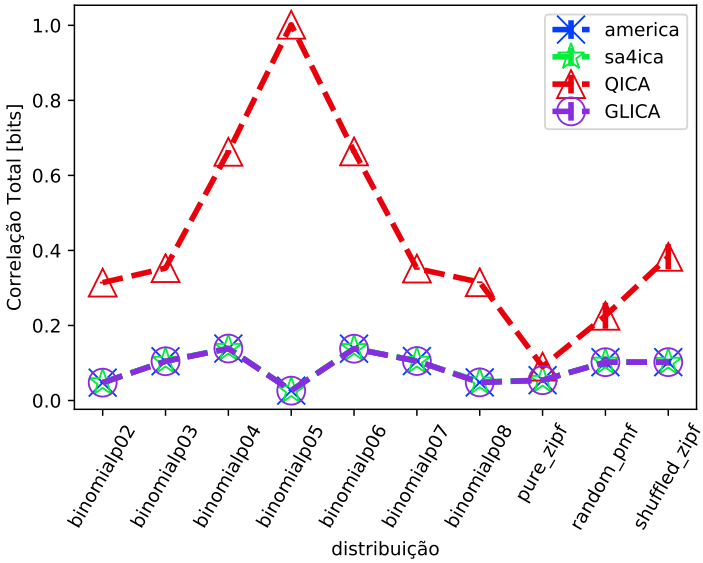

Figura 3.15: Correlação Total Média para $P = 2$, $K = 4$ e $T = 8192$, para diversas distribuições avaliadas. Confiança de 95%.

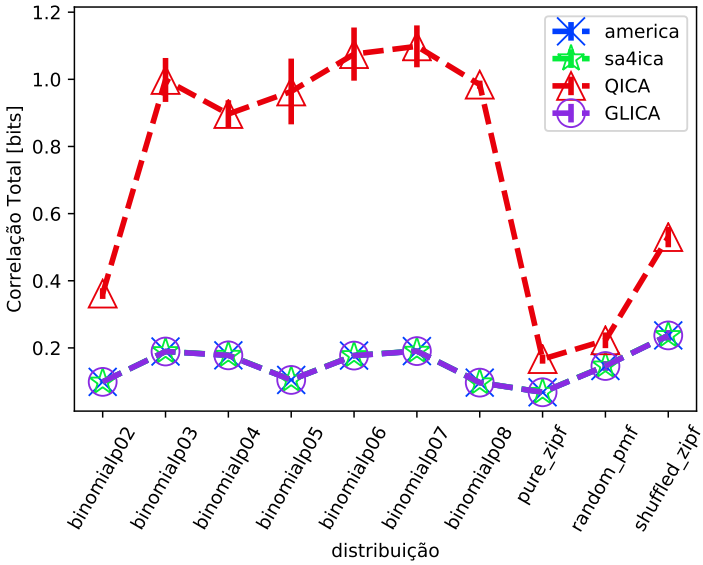

Figura 3.16: Correlação Total Média para $P = 3$, $K = 3$ e $T = 8192$, para diversas distribuições avaliadas. Confiança de 95%.

Sendo esse o mais lento dos quatro algoritmos testados. Isto pode ser observado na Figura 3.17.

Da mesma maneira que dispomos resultados do teste de Wilcoxon para aplicação de ICA em BSS, dispomos aqui os resultados quanto à Correlação Total de um dado algoritmo ser maior ou igual ao de outro e um teste quanto ao tempo do algoritmo QICA, em relação

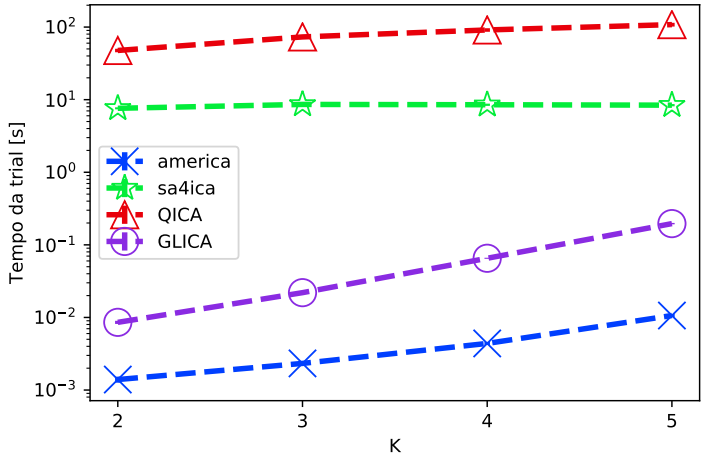

Figura 3.17: Tempo médio para cada algoritmo para $P = 3$ e $T = 16384$, distribuição Zipf pura. Confiança de 95%.

ao SA4ICA, o mais lento dos lineares. Os resultados dos p-valores é disposta na Tabela 3.8.

Tabela 3.8: Teste de Wilcoxon quanto ao QICA tomar menos tempo que o SA4ICA.

p-valor: Hipótese nula: (SA4ICA $\geq$ QICA)	Médias para $P =$
$3.92 \cdot 10^{-15}$	2
$3.92 \cdot 10^{-15}$	3

Estes resultados nos indicam que, dada a significância de 0.05, devemos rejeitar a hipótese nula e inferir que o tempo de execução do método SA4ICA é menor que a do método QICA.

Quanto à hipótese nula do QICA ter maior capacidade de minimizar a Correlação Total, quanto as médias para determinado P, não obtemos significância inferior a 0.05 em relação ao AMERICA, de tal forma que não conseguimos inferir a hipótese que o QICA é um algoritmo melhor que o AMERICA sob esta métrica.

Contudo, se restringirmos os testes para as realizações no cenário em que ele foi melhor que os demais algoritmos, i.e. foi capaz de ter menor TC; temos a Tabela 3.9. Nota: se compararmos o algoritmo QICA com o algoritmo GLICA ou SA4ICA, temos os mesmos resultados para os p-valores.

Desta forma, rejeitamos a hipótese nula nestes cenários em específico, por terem valores inferiores a 0.05 quanto à distribuição Zipf pura e $P = 2, K = 3$.

Contudo, se adotarmos a hipótese nula de que o algoritmo AMERICA obtém maior Correlação Total que o algoritmo QICA, temos a Tabela 3.10. O AMERICA e o GLICA

Tabela 3.9: Teste de Wilcoxon quanto ao QICA ter menor Correlação Total que o AMERICA.

p-valor: Hipótese nula: (QICA $\geq$ AMERICA	$P = 2, K = 3$, distribuição: Zipf pura T =
$3.77 \cdot 10^{-10}$	8192
$3.77 \cdot 10^{-10}$	16384

não são apresentados pelo teste de Wilcoxon falhar, por terem no processamento de *rankeamento* muitos empates, ou por apresentarem significância maior que 0.05.

Tabela 3.10: Teste de Wilcoxon quanto ao AMERICA ter menor Correlação Total que o QICA.

p-valor: Hipótese nula: (AMERICA $\geq$ QICA)	Médias para $P =$
$4.37 \cdot 10^{-14}$	2
$3.92 \cdot 10^{-15}$	3

Desta maneira, rejeitamos a hipótese nula e confirmamos que quanto à minimização da MI, inferimos que o algoritmo AMERICA se mostra superior ao algoritmo QICA.

3.2.2 Conclusões

Pode-se observar neste segundo experimento, com modelos geradores não necessariamente lineares, todos os algoritmos lineares avaliados, na maioria dos cenários, apresentaram minimização da Correlação Total menores que a do algoritmo não-linear, QICA. O QICA só apresentou em um cenário específico, quando a distribuição era a Zipf pura e $P = 2$, $K = 3$, uma Correlação Total necessariamente menor que a dos demais algoritmos. Contudo, analisando os cenários sobre diversas distribuições, podemos observar que, para todos os algoritmos avaliados, a distribuição Zipf pura não aparenta ser uma distribuição desafiadora para realização da tarefa de ICA, visto que estes mesmos algoritmos obtiveram desempenhos similares para esta mesma distribuição, seja ele linear ou não-linear.

Assim, para o QICA, a distribuição mais desafiadora foi a binomial com $p = 0.5$, enquanto que nesta mesma distribuição, na maioria dos cenários, foi esta que os algoritmos lineares alcançaram seus menores valores de Correlação Total para os demais parâmetros de simulação constantes. Enquanto que para os algoritmos lineares, fora a Zipf embaralhada a distribuição mais desafiadora, a distribuição sugerida em [Painsky, Rosset e Feder 2015] como o modelo gerador não-linear. O QICA também toma tempo de execução muito elevado, chegando a alcançar ordem de grandeza cerca de 5 vezes maior que a do algoritmo mais rápido avaliado.

Cabe salientar que as estimativas de probabilidades da PMF conjunta são cada vez melhores quando aumentamos a quantidade de amostras observadas. Contudo, para uma

boa estimativa das probabilidades da PMF conjunta é desejável que a quantidade de amostras observadas seja muito maior que P^K [Yeredor 2011], que aqui assumem, no máximo, o valor $3^5 = 243 \ll 8192$, que é a menor quantidade de amostras observadas para este experimento.

Desta maneira, algoritmos lineares podem ser vantajosos quando a quantidade de amostras observadas não puder ser tão grande, apesar de terem um limitante inferior quanto à TC maiores que o limitante inferior TC dos algoritmos não-lineares, como constatado em [Painsky, Rosset e Feder 2018]. Como informado no Capítulo 2, a ICA não-linear permitiria alcançar menores valores de TC por abarcar, dentro de seu espaço de estados, os estados possíveis de ICA linear.

4 Conclusão

Com base numa análise da literatura, percebe-se que alguns algoritmos não apresentam uma análise teórica de custo computacional. Para isto, realizam-se simulações para avaliar o desempenho dos algoritmos. Destas simulações, é possível encontrar cenários ou resultados anômalos não previstos anteriormente; ou ainda confirmar resultados de trabalhos anteriores, apresentando uma replicação destes. Não obstante, deseja-se que os algoritmos avaliados possam obter bom desempenho em tempos cabíveis. Assim, comparamos os algoritmos lineares a um não-linear, no caso da realização da tarefa de ICA, inclusive em cenários que não assumem modelos geradores lineares, que favoreceriam, a princípio, o algoritmo não-linear. Utilizamos de comparações estatísticas e estimativas para inferir as relações de desempenho entre estes algoritmos.

Quanto ao primeiro experimento, que visava comparar algoritmos lineares na aplicação de ICA ao problema de BSS, observa-se que os algoritmos convergem assintoticamente para 100% de Separação das Fontes. Contudo, não reportado anteriormente, um destes algoritmos, o SA4ICA, apresentou oscilações quanto à separação ao aumentarmos a cardinalidade do alfabeto para números primos maiores que 2 e 3. Também apresentou oscilações ao aumentarmos a quantidade de componentes do problema, sempre majorado pelos dois outros algoritmos lineares, AMERICA e GLICA, que são muito similares em seu funcionamento.

Também verificamos que, como informado anteriormente, o comportamento é praticamente constante em $GF(2)$ e $GF(3)$ quanto ao número das avaliações da função-custo, o que poderia tornar o algoritmo viável em relação ao MEXICO para quantidade de fontes maiores. O SA4ICA apresenta tempo de execução crescente de forma logarítmica, quando aumentamos tanto P quanto K. Afinal, o próprio MEXICO começa a valer a pena em termos de custo computacional, mesmo com acurácia inferior ao AMERICA para quantidade menor de amostras, quando se aumenta o valor de K [Yeredor 2011]. Assim, o algoritmo AMERICA continua sendo o preferível entre os algoritmos lineares, tanto em termos de separação, quanto custo computacional.

Quanto ao segundo experimento, execução de ICA por ela mesma, avaliamos a capacidade de *minimização* da Informação Mútua (ou Correlação Total) de dados gerados

aleatoriamente de uma PMF conjunta sob diversas distribuições não necessariamente lineares. Os algoritmos lineares tem capacidade limitada quanto à Correlação Total mínima alcançável [Painsky, Rosset e Feder 2018]. Contudo, para cenários que não compreendem estimação perfeita das probabilidades, devido a uma quantidade não tão grande de amostras, na casa das dezenas de milhares, os algoritmos lineares demonstraram ser capazes de alcançar valores de Correlação Total competitivos em relação ao algoritmo QICA, que é não-linear. Apesar dos algoritmos não-lineares deterem limites inferiores para Correlação Total menores que os lineares [Painsky, Rosset e Feder 2018]. Além disso, o tempo tomado para encontrar o mapeamento dos algoritmos lineares avaliados são ordens de grandeza menores que o do algoritmo QICA, chegando a 5 vezes a diferença. Desta forma, em termos de barganha entre minimização da MI e de tempo de execução, o algoritmo mais generalista não demonstrou superioridade, mesmo utilizando-se de uma heurística para acelerar a execução deste. Cabe enfatizar que o algoritmo não-linear permite trabalhar em alfabetos de cardinalidade arbitrária, diferentemente dos algoritmos lineares, que só aceitam alfabetos de cardinalidade prima ou potências de números primos, em outras palavras, trabalham no contexto de Corpos Finitos.

4.1 Perspectivas e trabalhos futuros

Quanto a trabalhos futuros, é desejável executar experimentos com outras distribuições, maior quantidade de realizações, para melhor estimativas dos intervalos de confiança, além de buscar desenvolver outros algoritmos de ICA que não necessitem de custo computacional tão elevado, ou mesmo aprimorar os já existentes. Também variar os valores para a divergência de Kullback-Leibler, pois os algoritmos podem apresentar comportamentos distintos quando as distribuições das componentes das fontes se apresentarem muito próximas da distribuição uniforme.

É inevitável não relacionar a teoria da informação quando nos referimos a ICA. Seria interessante investigar o uso do problema BSS como uma tarefa suficientemente difícil para criptografia, por exemplo. Bem como uma teoria que unifique a relação de ICA com códigos corretores de erro.

Tratamos aqui de ICA em alfabetos finitos, mas nada impediria, por exemplo, formularmos um problema cujas fontes detém alfabetos distintos, discretos e não necessariamente finitos, mas enumeráveis. Afinal, quando realizamos codificação, o alfabeto costuma ser construído só ao final do processo. Por exemplo, utilizou-se o próprio problema de BSS tendo em vista aplicações práticas em *networking coding* [Nemoianu et al. 2013, Nemoianu et al. 2014]. Desta maneira, ainda há espaço para aplicação de ICA e desenvolvimento de técnicas de separação.

Referências

[Alpaydin 2014]ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2014. 6

[Bell e Sejnowski 1995]BELL, A. J.; SEJNOWSKI, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, MIT Press, v. 7, n. 6, p. 1129–1159, 1995. 2

[Belouchrani, Cardoso et al. 1995]BELOUCHRANI, A.; CARDOSO, J.-F. et al. Maximum likelihood source separation by the expectation-maximization technique: Deterministic and stochastic implementation. In: CITESEER. *Proc. Nolta*. [S.l.], 1995. v. 95, p. 49–53. 2

[Casella e Berger 2019]CASELLA, G.; BERGER, R. L. *Inferência Estatística*. [S.l.: s.n.], 2019. ISBN 9788522108947. 10, 35

[Comon 1994]COMON, P. Independent component analysis, a new concept? *Signal processing*, Elsevier, v. 36, n. 3, p. 287–314, 1994. 2, 6, 8

[Comon 2010]COMON, P. *Handbook of blind source separation : independent component analysis and applications*. Amsterdam Boston: Elsevier, 2010. ISBN 978-0-12-374726-6. 2, 8, 13, 14

[Cover e Thomas 2012]COVER, T. M.; THOMAS, J. A. *Elements of information theory*. [S.l.]: John Wiley & Sons, 2012. 1, 8, 9, 10

[Cristea, Ottosson e Piątyszek]CRISTEA, B.; OTTOSSON, T.; PIąTYSZEK, A. *IT++ : C++ Library*. Disponível em: <http://itpp.sourceforge.net/4.3.1/index.html>. 2

[da Rosa e Silva 2019]da Rosa, M.; SILVA, D. G. Um estudo comparativo de algoritmos de ICA em alfabetos finitos sob modelos lineares e não-lineares. In: *XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2019) (SBrT 2019)*. Petrópolis, Brazil: [s.n.], 2019. 5

[Delfosse e Loubaton 1995]DELFOSSE, N.; LOUBATON, P. Adaptive blind separation of independent sources: a deflation approach. *Signal processing*, Elsevier, v. 45, n. 1, p. 59–83, 1995. 2, 16

[Eaton et al. 2019]EATON, J. W. et al. *GNU Octave version 5.1.0 manual: a high-level interactive language for numerical computations*. [S.l.], 2019. Disponível em: <https://www.gnu.org/software/octave/doc/v5.1.0/>. 32

[Ferguson, Schneier e Kohno 2010]FERGUSON, N.; SCHNEIER, B.; KOHNO, T. *Cryptography engineering : design principles and practical applications*. Indianapolis, IN: Wiley Pub., Inc, 2010. ISBN 978-0470474242. 10

[Gavrilović 1975]GAVRILOVIĆ, M. M. Optimal approximation of convex curves by functions which are piecewise linear. *Journal of Mathematical Analysis and Applications*, Elsevier, v. 52, n. 2, p. 260–282, 1975. 28

[Girolami e Fyfe 1996]GIROLAMI, M.; FYFE, C. Negentropy and kurtosis as projection pursuit indices provide generalised ica algorithms. In: *Advances in Neural Information Processing Systems Workshop*. [S.l.: s.n.], 1996. v. 9. 2

[Glover e Sörensen 2015]GLOVER, F.; SöRENSEN, K. Metaheuristics. *Scholarpedia*, v. 10, n. 4, p. 6532, 2015. Revision #149834. 19

[Gutch, Gruber e Theis 2010]GUTCH, H. W.; GRUBER, P.; THEIS, F. J. ICA over finite fields. In: SPRINGER. *International Conference on Latent Variable Analysis and Signal Separation*. [S.l.], 2010. p. 645–652. 3, 16, 17, 18, 21, 22

[Gutch et al. 2012]GUTCH, H. W. et al. ICA over finite fields—Separability and algorithms. *Signal Processing*, Elsevier, v. 92, n. 8, p. 1796–1808, 2012. 3, 14, 15, 17, 18, 20, 22

[Hérault, Jutten e Ans 1985]HÉRAULT, J.; JUTTEN, C.; ANS, B. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In: GRETSI, GROUPE D'ETUDES DU TRAITEMENT DU SIGNAL ET DES IMAGES. *10 Colloque sur le traitement du signal et des images, FRA, 1985*. [S.l.], 1985. 2

[Hyvärinen 2013]HYVÄRINEN, A. Independent component analysis: recent advances. *Phil. Trans. R. Soc. A*, The Royal Society, v. 371, n. 1984, p. 20110534, 2013. 8

[Hyvärinen e Oja 1997]HYVÄRINEN, A.; OJA, E. A fast fixed-point algorithm for independent component analysis. *Neural computation*, MIT Press, v. 9, n. 7, p. 1483–1492, 1997. 2

[Jutten 1987]JUTTEN, C. *Calcul neuromimétique et traitement du signal: analyse en composantes indépendantes*. Tese (Doutorado) — Grenoble INPG, 1987. 2

[Jutten e Herault 1991]JUTTEN, C.; HERAULT, J. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, Elsevier, v. 24, n. 1, p. 1–10, 1991. 2

[Kirkpatrick, Gelatt e Vecchi 1983]KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. 18, 19

[Lang 2002]LANG, S. *Algebra, volume 211 of Graduate texts in mathematics*. [S.l.]: Springer-Verlag, New York,, 2002. 12, 26

[Lathi 2012]LATHI, D. Z. B. P. *Sistemas de Comunicações Analógicos e Digitais Modernos*. 4. ed. [S.l.: s.n.], 2012. ISBN 9788521620273. 1, 2

[Lozano 2019]LOZANO, R. W. H. J. A. *Foundations of MIMO Communication*. [S.l.]: Cambridge University Press, 2019. ISBN 0521762286,9780521762281. 2

[Marchini, Heaton e Ripley]MARCHINI, J.; HEATON, C.; RIPLEY, B. *FastICA Algorithms to Perform ICA and Projection Pursuit*. Disponível em: <https://cran.r-project.org/web/packages/fastICA/fastICA.pdf>. 2

[Marcuzzo 2019]MARCUZZO, M. *Comparação de algoritmos de ICA para o caso de BSS e ICA pura, 2019*. 2019. Disponível em: <https://github.com/MateusMarcuzzo/sbrt2019_ICA_comparative>. 12, 22, 23, 29, 32

[Moon 2005]MOON, T. K. *Error Correction Coding : mathematical methods and algorithms*. Hoboken, N.J: Wiley-Interscience, 2005. ISBN 978-0471648000. 10

[Morettin e Bussab 2010]MORETTIN, P.; BUSSAB, W. *Estatistica básica*. São Paulo: Saraiva, 2010. ISBN 978-85-02-08177-2. 34

[Moura 2009]MOURA, J. What is signal processing?[president's message]. *IEEE Signal Processing Magazine*, IEEE, v. 26, n. 6, p. 6–6, 2009. 1

[Nemoianu et al. 2013]NEMOIANU, I. et al. On a practical approach to source separation over finite fields for network coding applications. In: IEEE. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. [S.l.], 2013. p. 1335–1339. 3, 53

[Nemoianu et al. 2014]NEMOIANU, I.-D. et al. On a hashing-based enhancement of source separation algorithms over finite fields with network coding perspectives. *IEEE Transactions on Multimedia*, IEEE, v. 16, n. 7, p. 2011–2024, 2014. 3, 53

[Painsky]PAINSKY, A. *Matlab® implementation of A. Painsky, S. Rosset and M. Feder, "Generalized Independent Component Analysis Over Finite Alphabets", Transactions on Information Theory, 2015*. Disponível em: <https://sites.google.com/site/amichaipainsky/software>. 29, 42, 44

[Painsky, Rosset e Feder 2015]PAINSKY, A.; ROSSET, S.; FEDER, M. Generalized independent component analysis over finite alphabets. *IEEE Transactions on Information Theory*, IEEE, v. 62, n. 2, p. 1038–1053, 2015. 3, 18, 27, 28, 31, 42, 47, 50

[Painsky, Rosset e Feder 2016]PAINSKY, A.; ROSSET, S.; FEDER, M. Binary independent component analysis: Theory, bounds and algorithms. In: IEEE. *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. [S.l.], 2016. p. 1–6. 27

[Painsky, Rosset e Feder 2018]PAINSKY, A.; ROSSET, S.; FEDER, M. Linear independent component analysis over finite fields: Algorithms and bounds. *IEEE Transactions on Signal Processing*, IEEE, v. 66, n. 22, p. 5875–5886, 2018. 4, 21, 22, 43, 51, 53

[Pedregosa et al. 2011]PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011. 2

[Pfleeger e Pfleeger 2007]PFLEEGER, C. P.; PFLEEGER, S. L. *Security in computing*. Upper Saddle River, NJ: Prentice Hall, 2007. ISBN 978-0132390774. 10

[Romano et al. 2011]ROMANO, J. M. T. et al. *Unsupervised signal processing : channel equalization and source separation*. Boca Raton: CRC Press, 2011. ISBN 9780849337512. 2

[Russell e Norvig 2009]RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach (3rd Edition)*. [S.l.]: Pearson, 2009. ISBN 0136042597. 4, 19

[Sayood 2012]SAYOOD, K. *Introduction to data compression*. Waltham, MA: Morgan Kaufmann, 2012. ISBN 978-0124157965. 10

[Shannon 1948]SHANNON, C. E. A mathematical theory of communication. *Bell system technical journal*, Wiley Online Library, v. 27, n. 3, p. 379–423, 1948. 9, 10

[Silva e Attux 2018]SILVA, D. G.; ATTUX, R. Simulated Annealing for Independent Component Analysis Over Galois Fields. *IEEE Signal Processing Letters*, IEEE, v. 25, n. 4, p. 516–520, 2018. 18, 32, 41, 44

[Silva et al. 2011]Silva, D. G. et al. An immune-inspired information-theoretic approach to the problem of ica over a galois field. In: *2011 IEEE Information Theory Workshop*. [S.l.: s.n.], 2011. p. 618–622. 3

[Silva et al. 2014]SILVA, D. G. et al. A michigan-like immune-inspired framework for performing independent component analysis over galois fields of prime order. *Signal Processing*, v. 96, p. 153 – 163, 2014. ISSN 0165-1684. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0165168413003472>. 3

[Strang 2006]STRANG, G. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006. ISBN 978-0030105678. 13

[Tůma]TůMA, M. *libICA - ICA library: FastICA C Implementation*. Disponível em: <http://tumic.wz.cz/fel/online/libICA/>. 2

[Watanabe 1960]WATANABE, S. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, IBM, v. 4, n. 1, p. 66–82, 1960. 8, 10

[Waterhouse 1987]WATERHOUSE, W. C. How often do determinants over finite fields vanish? *Discrete mathematics*, Elsevier, v. 65, n. 1, p. 103–104, 1987. 26

[Wilcoxon 1992]WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics*. [S.l.]: Springer, 1992. p. 196–202. 40

[Yeredor 2007]YEREDOR, A. ICA in boolean XOR mixtures. In: SPRINGER. *International Conference on Independent Component Analysis and Signal Separation*. [S.l.], 2007. p. 827–835. 3, 16, 18, 20, 22

[Yeredor 2011]YEREDOR, A. Independent component analysis over galois fields of prime order. *IEEE Transactions on Information Theory*, IEEE, v. 57, n. 8, p. 5342–5359, 2011. 3, 9, 14, 15, 16, 17, 18, 20, 22, 51, 52

[Yeredor 2011]YEREDOR, A. *Matlab® Code for* $\mathbb{GF}(P)$*, AMERICA and MEXICO*. 2011. Disponível em: <http://www.eng.tau.ac.il/ arie/ICA4GFP.rar>. 16, 17, 18

[Zipf 2016]ZIPF, G. K. *Human behavior and the principle of least effort: An introduction to human ecology*. [S.l.]: Ravenio Books, 2016. 42

[Zito et al. 2009]ZITO, T. et al. Modular toolkit for data processing (mdp): a python data processing framework. *Frontiers in Neuroinformatics*, v. 2, p. 8, 2009. ISSN 1662-5196. Disponível em: <https://www.frontiersin.org/article/10.3389/neuro.11.008.2008>. 2