Relatório final de iniciação científica

Computação imuno-inspirada na análise de componentes independentes em campos finitos

Submetido ao
Programa Institucional de Bolsa de Iniciação Científica da UNICAMP
Submetido por
Faculdade de Ciências Aplicadas (FCA)
Rua Pedro Zaccaria, 1300, Jardim Santa Luiza
CEP 13484-350 – Limeira - SP

Bolsista: Daniel de Araújo Pereira
Orientador: Prof. Dr. Leonardo Tomazeli Duarte
Duração Prevista: 12 meses

1. INTRODUÇÃO

O presente relatório tem como objetivo descrever as atividades desenvolvidas pelo bolsista durante sua iniciação científica. Inicialmente, faremos uma breve recapitulação dos problemas tratados e dos objetivos deste projeto, além de descrever as atividades desenvolvidas. Em seguida, os materiais e métodos são descritos na seção 2. A seção 3 abordará a descrição e análise dos resultados encontrados, e, por fim, na seção 4 tecemos nossas considerações finais.

1.2 Recapitulação do problema

O problema de separação cega de sinais (BSS, do inglês *Blind Source Separation*) e a sua solução através da análise de componentes independentes (ICA, do inglês *Independent Component Analysis*) é comumente considerado no campo de sinais reais ou complexos, entretanto, esse projeto tem como foco o estudo do caso em que os sinais pertencem a um conjunto discreto e finito, denominado campo (ou corpo) finito (ou de Galois) [GUTCH, 2012]. Neste cenário, é possível tratar o problema como uma busca combinatória, sendo então adequado ao contexto da utilização de técnicas de computação bio-inspirada para se determinar a solução [SILVA, 2013].

A computação bio-inspirada é uma metodologia que se inspira em fenômenos biológicos para o desenvolvimento de ferramentas (algoritmos) aplicáveis a diversos problemas, nas mais diversas áreas [DE CASTRO, 2006]. Vale ressaltar que essa abordagem pode ser vista como uma estratégia alternativa para solucionar problemas de otimização, principalmente empregada quando o problema é muito complexo (muitas variáveis, muitas possíveis soluções, etc.), o que demandaria um tempo possivelmente inviável, se abordado pelos métodos clássicos, ou quando há necessidade de obter-se múltiplas soluções / ótimos (o que implica na exploração de soluções candidatas diversas).

Há diversos paradigmas de algoritmos bio-inspirados, em especial, os Sistemas Imunológicos Artificiais (SIA) têm sido empregados com sucesso em problemas de separação de sinais [SILVA, 2013]. O algoritmo escolhido para o problema em foco neste projeto é a cob-aiNet[C], uma técnica imuno-inspirada estado-da-arte, elaborada para tarefas de otimização combinatória e que combina duas teorias que embasam diversos algoritmos imunológicos modernos: o principio da Seleção Clonal e a Teoria da Rede Imunológica [COELHO, 2011].

1.3 Recapitulação dos objetivos

Dado o contexto do estudo de métodos bio-inspirados no problema de separação de sinais em campos finitos, foram propostos, no projeto inicial, os seguintes objetivos:

- Realização de um estudo sobre computação bio-inspirada, iniciando com os principais conceitos de computação evolutiva e encerrando com os algoritmos imunológicos estado-da-arte.
- Realização de um estudo sobre processamento de sinais em campos finitos, com ênfase nas abordagens baseadas em algoritmos populacionais.

- Aplicação dos métodos estudados ao problema de ICA em campos finitos, e sua subsequente análise.
- Redação de um relatório científico final descrevendo todas as etapas realizadas durante o trabalho.
- Participação na redação de pelo menos um artigo acerca das análises realizadas, e que exponha tais resultados em um fórum científico.

1.4 Atividades desenvolvidas

Inicialmente, foi necessária a familiarização com os principais conceitos da computação bio-inspirada, estudando suas abordagens baseadas na teoria da evolução (computação evolutiva) e no sistema imunológico dos vertebrados (computação imuno-inspirada), com ênfase final no algoritmo cob-aiNet[C]. Foi também necessário o estudo de princípios da teoria da informação, e em seguida ICA, para a compreensão dos métodos de resolução do problema de BSS, em especial no domínio de campos finitos. Por fim, foram realizadas simulações para testar a eficiência da cob-aiNet[C] na separação de sinais em corpos binários. Devido ao prolongamento, além do previsto, das atividades anteriores, a participação na redação de um artigo científico foi a única atividade do planejamento inicial que não pôde ser realizada.

1.4.1 Computação Evolutiva

A computação evolutiva, como o nome sugere, busca sua inspiração no processo evolutivo para construir ferramentas para solução de problemas, em geral aqueles de otimização. O algoritmo estudado pelo bolsista foi o algoritmo genético [BACK, 2000; DE CASTRO, 2006], que toma emprestado conceitos e princípios dos campos da teoria da evolução e da genética.

Um algoritmo genético trabalha com uma população inicial de indivíduos, um conjunto $P(t) = \{\boldsymbol{x}_1^t, \dots, \boldsymbol{x}_n^t\}$, x representando o indivíduo e t representando a iteração. Cada indivíduo representa um candidato à solução e corresponderia, em termos biológicos, a um cromossomo, que usualmente se implementa através de uma lista de atributos (os genes) ou vetor. Para a determinação da melhor solução, é utilizada uma função objetivo que indica o grau de adaptabilidade (*fitness*) da solução, simulando a pressão do ambiente. Então, na iteração seguinte, $(t + 1)$, uma nova população é formada por meio de uma seleção dos indivíduos mais adaptados (melhor *fitness*). Estes, então, são submetidos a processos de alteração no seu vetor de atributos para a formação de novos indivíduos / soluções.

Um algoritmo genético usualmente possui a seguinte estrutura:

- Inicializar o conjunto $P(t)$
- Avaliar $P(t)$
- Enquanto (não atingida condição de parada) faça
    - Selecione $P(t + 1)$ a partir de $P(t)$
    - Altere $P(t + 1)$
    - Avalie $P(t + 1)$

Os operadores genéticos utilizados para alteração da população são o *crossover* (ou recombinação) e a mutação [BACK, 2000]. O primeiro é uma operação que se inspira no processo de *crossover* entre cromossomos, criando novos indivíduos a partir da recombinação de dois candidatos distintos. Uma estratégia comumente utilizada, neste caso, é o *crossover* de um ponto: dois indivíduos (pais) geram dois novos indivíduos (filhos), que são formados por segmentos de genes de ambos os pais. O primeiro filho recebe os genes do primeiro pai até o ponto de corte e, depois do corte, os genes são do segundo pai. O segundo filho é formado no mesmo processo, porém os genes antes do ponto de corte são do segundo pai e os genes após pertencem ao primeiro.

O operador de mutação altera aleatoriamente um ou mais genes do indivíduo. A probabilidade de ocorrência do operador é definida como taxa de mutação e normalmente emprega-se um valor pequeno [DE CASTRO, 2006]. No caso binário, um operador bastante empregado realiza a troca do valor de um (ou mais) genes; por exemplo, um valor originalmente 0 passa a ser 1 após a mutação, e vice-versa. Para o caso de valores reais, é frequentemente utilizada a mutação gaussiana: todos os atributos do cromossomo são somados a uma pequena perturbação de modo que $\boldsymbol{x} = \boldsymbol{x} + N(0, \sigma)$ onde N é um vetor de variáveis aleatórias, com distribuição normal de média 0 e desvio padrão σ.

Para sedimentar o conhecimento de computação evolutiva, foram realizados alguns estudos de caso, com a implementação em linguagem C de um algoritmo genético para solução dos problemas do Caixeiro Viajante e de otimização de funções com variáveis independentes reais.

1.4.2 Computação imuno-inspirada

Algoritmos imunológicos são técnicas inspiradas no sistema imunológico de vertebrados. Embora possua diversos pontos de contato com os algoritmos evolutivos, eles se distinguem das outras técnicas bio-inspiradas pela maior capacidade de manutenção de diversidade na população, em outras palavras, conseguem manter soluções diversas durante a sua execução, o que potencializa a localização de múltiplos ótimos [DE CASTRO, 2006].

Duas importantes teorias por trás desses algoritmos são o Princípio da Seleção Clonal [BURNET, 1959] e a Teoria da Rede Imunológica [JERNE, 1974]. A Seleção Clonal afirma que quando um indivíduo é exposto a um agente patogênico (antígeno), o linfócito – célula do sistema imune responsável pelo reconhecimento do intruso – será submetido a um processo de clonagem, gerando (*i*) células especializadas para secretar anticorpos específicos contra aquele antígeno ou (*ii*) gerando células de memória para uma possível reincidência no futuro. Além disso, a replicação é acompanhada de um processo de hipermutação, que introduz mutações aleatórias nos clones, variáveis na intensidade de acordo com a afinidade de cada um no reconhecimento do antígeno. Quanto menor a afinidade, mais intensa é a mutação.

A Teoria da Rede Imunológica atesta que as células do sistema imune também são capazes de se reconhecer, portanto cada componente é capaz de interferir e ser interferido pelo outro, de modo que o sistema como um todo se mantenha em um estado de equilíbrio dinâmico.

O primeiro algoritmo imuno-inspirado estudado neste projeto foi o CLONALG. O algoritmo tem embasamento no princípio da Seleção Clonal e seus principais passos, para ser aplicado a um problema de otimização, são:

- Inicializar uma população Ab de células (indivíduos), definidas de maneira aleatória
- Repita enquanto não atingir a condição de parada:
  - Avaliação da afinidade de cada célula de Ab (ou seja, determinar o valor da função objetivo para cada Ab_i)
  - Geração de nCl clones para cada célula i em Ab
  - Clones sofrem um processo de mutação inversamente proporcional à afinidade de seu pai
  - Avaliar a afinidade dos clones gerados
  - Selecionar a melhor célula entre os clones e o pai
  - Substituir um porcentual b das piores células por novas células geradas aleatoriamente

Após o CLONALG, que lida somente com o primeiro princípio mencionado no começo da seção, surgiram novos algoritmos incorporando também a teoria da Rede Imunológica, o que agregaria o controle dinâmico do número de indivíduos da população. O pioneiro foi a ai-Net (*Artificial Immune Network*) e aquele que foi estudado durante o segundo semestre do projeto foi a cob-aiNet[C], a qual lida com problemas de otimização combinatória [COELHO, 2011].

As novas operações inseridas, em relação ao CLONALG, consistem na interação entre as próprias células da população: caso duas soluções candidatas sejam similares, ou seja, em termos de uma métrica de distância estejam muito próximas (menor que um limiar previamente estabelecido) ocorre uma comparação e a célula com o menor *fitness* tem a sua concentração reduzida. Caso haja a contínua redução da concentração de uma célula, ela pode ser suprimida quando esse valor alcançar zero. A concentração também pode aumentar de acordo com o *fitness*. Vale ressaltar que, diferentemente do CLONALG, na cob-aiNet[C] a afinidade não corresponde ao valor da função objetivo, é na verdade uma medida própria baseada nos parâmetros de concentração e distância de soluções vizinhas [COELHO, 2011].

A estrutura geral da cob-aiNet[C] é:

Parâmetros:

- nAb tamanho inicial da população
- $maxAb$: tamanho máximo da população
- nCl^{max}: número máximo de clones por célula
- nCl^{min} número mínimo de clones por célula
- C_0: concentração inicial de cada célula;
- σ_s: limiar de supressão
- β^i: parâmetro inicial do operador de mutação
- β^f: parâmetro final do operador de mutação
- LS_{it}: número de iterações de busca local
- LS_{freq}: frequência da busca local

Passos:

- Inicializar uma população Ab de células com nAB elementos definidos de maneira aleatória
- Avaliar o *fitness* das células de Ab
- Avaliar a afinidade entre as células de Ab
- Enquanto (iteração ≤ número máximo de iterações) faça
    - Definir o número nCl_i de clones que serão gerados para cada célula i
    - Gerar nCl_i clones para cada célula i da população
    - Aplicar operador de mutação nos clones
    - Avaliar *fitness* das células geradas
    - Selecionar as células que serão mantidas para a próxima geração (com inserção)
    - Se iteração for divisível por LS_{freq} então
        - Aplicar o operador de busca local a todos os indivíduos
        - Atualizar o *fitness* de cada indivíduo
    - Avaliar a afinidade entre as células da população
    - Atualizar a concentração das células e eliminar aquelas de concentração nula
- Aplicar o operador de busca local a todos os indivíduos
- Atualizar *fitness* de cada indivíduo
- Avaliar a afinidade entre as células da população
- Atualizar a concentração das células e eliminar as de concentração nula

Para consolidar o estudo em computação imuno-inspirada, primeiramente foi implementado na linguagem C o algoritmo da cob-aiNet, para otimização real, que em seguida foi validado em problemas-exemplo de otimização de funções contínuas.

1.4.3 Teoria da informação e entropia

A teoria da informação, desenvolvida por Claude E. Shannon em seu famoso artigo científico intitulado "*A Mathematical Theory of Communication"* [SHANNON, 1948] tem como problema primordial a transmissão de informações através de um canal com ruídos. A medida fundamental de informação, proposta por Shannon, é a entropia, geralmente expressa pela média de número de *bits* necessária para guardar ou comunicar um símbolo em uma mensagem. A entropia quantifica a incerteza média associada aos valores de uma variável aleatória.

Seja a função f definida por:

$$f(p) = -p\log(p), \text{ para } 0 \leq p \leq 1 \quad (1.1)$$

O logaritmo geralmente é na base 2, entretanto, ele só altera a escala de medida da entropia. f é uma função não negativa que tem valor zero para $p = 0$ e $p = 1$, e valor positivo para os valores entre o intervalo]0,1[, como ilustra a Figura 1.1. Usando essa função, entropia é definida como [HYVARINEN, 2001]:

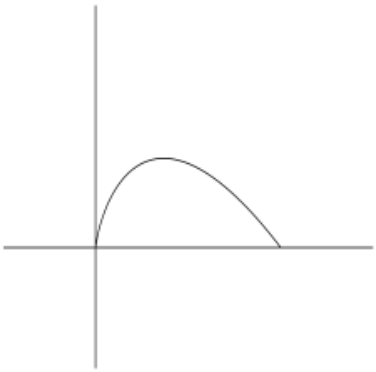

Figura 1.1: função f plotada no intervalo [0,1], retirada de [HYVARINEN, 2001].

$$H(X) = \sum_i f(P(X = a_i)) = \sum_i -P(X = a_i)\log\left[P(X = a_i)\right] \quad (1.2)$$

Considerando a forma da função f, é possível perceber que seu valor é pequeno para probabilidades $P(X = a_i)$ próximas de 0 e 1, mas torna-se alto nos valores intermediários. Isto corrobora com a noção de que entropia define o grau médio de incerteza para os valores de uma variável aleatória, já que para os casos em que o evento é certo ($P(X = a_i) = 1$) ou é certo que não ocorreu ($P(X = a_i) = 0$), a incerteza associada é nula, afinal, deparamo-nos com um evento completamente previsível.

Já para um número n de eventos, se $P(X = a_i) = {}^{1}/_{n}$, ou seja, a distribuição é uniforme, o grau médio de incerteza acerca da variável aleatória X é máximo, já que não é possível prever, com base nos valores das probabilidades, qual evento tenha maiores possibilidades de ocorrer [HYVARINEN, 2001].

Então, quanto mais imprevisível, na média, é essa variável, maior é sua entropia. Como veremos mais adiante, a entropia é utilizada para a determinação do *fitness* de cada indivíduo na cob-aiNet[C], no qual os indivíduos com menor entropia representam os mais adaptados.

1.4.4 Análise de componentes independentes

Um problema clássico no qual a Análise de Componentes Independentes (ICA – *Independent Component Analysis*) é utilizado como método para sua resolução é a Separação Cega de Fontes (BSS – *Blind Source Separation*) [HYVARINEN, 2001]. Uma maneira de exemplificar o BSS é se considerarmos que, em um determinado ambiente, esteja ocorrendo a emissão de um conjunto de sinais estatisticamente independentes, sejam sinais de rádio emitidos pelo celular, pessoas conversando ou até mesmo áreas diferentes do cérebro emitindo sinais elétricos. Há também nesse ambiente receptores em diferentes localizações de modo que os sinais captados são distintos. Entretanto, é possível que os receptores captem a mistura de dois ou mais sinais. O objetivo, portanto, é separar as misturas captadas por cada receptor de modo que tenhamos os sinais isolados sem interferência, como foram originalmente emitidos pela fonte.

Seja $\boldsymbol{x}(n)$ o vetor que compõem todos os sinais observados, $\boldsymbol{s}(n)$ os sinais originais emitidos pela fonte e $\boldsymbol{A}$ a matriz de mistura de dimensões k por k inversível, temos que:

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{s}(n) \quad (1.3)$$

O objetivo é encontrar uma matriz de separação $\boldsymbol{W}$ que restaure os sinais originais $\boldsymbol{s}(n)$. Uma estratégia de solução ocorre por meio de ICA, pois, considerando que os sinais originais são estatisticamente independentes, que há no máximo uma fonte gaussiana e que os coeficientes da mistura são tais que a matriz $\boldsymbol{A}$ seja inversível, é possível chegar à solução:

$$\boldsymbol{y}(n) = \boldsymbol{W}\boldsymbol{x}(n) = \boldsymbol{P}\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{x}(n) = \boldsymbol{P}\boldsymbol{D}\boldsymbol{s}(n) \quad (1.4)$$

desde que $\boldsymbol{y}(n)$ tenha seus componentes também mutuamente independentes [HYVARINEN, 2001]. Neste caso, $\boldsymbol{y}(n)$ equivale à $\boldsymbol{s}(n)$, com a ressalva de que seus elementos podem estar multiplicados por algum escalar (representado pela matriz diagonal $\boldsymbol{D}$) e com ambiguidades de permutação, representadas pela matriz de permutação $\boldsymbol{P}$.

Embora seja comumente empregada com sinais de valores reais, neste projeto a técnica de ICA é estudada no contexto em que o modelo, aqui apresentado, possui os valores pertencentes a um campo finito [SILVA, 2013].

1.4.5 Campos Finitos

Um campo (ou corpo) *F* é um estrutura matemática que generaliza as noções de soma e multiplicação, ele envolve um conjunto de elementos que obedecem aos axiomas de associatividade, comutatividade, distributividade, identidade e elemento inverso. Um corpo finito ou corpo de Galois, assim chamado em homenagem a Évariste Galois, matemático francês que desenvolveu os primeiros estudos sobre o tema, é um campo que contém um número finito de elementos, chamado de ordem (o tamanho do conjunto). O corpo de Galois é denominado *GF(q),* onde *q* é a sua ordem. Como em todos os corpos, as operações de multiplicação, adição, subtração e divisão cumulativas foram definidas. Vamos definir as operações de um campo binário $GF(2) = F = \{0,1\}$, o qual será utilizado nesse trabalho. As operações básicas de soma e multiplicação, nesse caso, correspondem ao ou-exclusivo (XOR) e AND, respectivamente: 0+1 = 1; 0+0 = 1+1 = 0; 0·0 = 0·1 = 0; 1·1 = 1.

O modelo do ICA em corpos finitos permanece o mesmo descrito na seção 1.4.4, com as diferenças que (*i*) as operações e todos os valores estão agora definidos à luz dos corpos de Galois e (*ii*) enquanto no caso com valores contínuos havia a restrição de haver no máximo uma fonte gaussiana, no modelo em corpos finitos há uma restrição mais rigorosa que só permite haver fontes com distribuição não-uniforme [GUTCH, 2012; SILVA, 2013].

Algumas considerações devem ser feitas ao modelar um algoritmo baseado na cob-aiNet[C] para esse tipo de problema. Em ICA, procura-se uma matriz de separação

que restaure as fontes originais. Portanto, a população do algoritmo terá indivíduos que representam, cada um, uma matriz candidata à solução, com determinante não nulo (devido à restrição da matriz ser inversível) e pertencente ao corpo finito, de modo que as operações envolvendo essa matriz deverão seguir as normas de adição e multiplicação apresentadas previamente.

A função *fitness* desse problema é a soma das entropias marginais dos m sinais extraídos pela matriz de separação candidata , a qual [SILVA, 2013] mostra que sua minimização equivale à maximização da independência entre os sinais observados:

$$\begin{aligned} g(\boldsymbol{y}) &= \frac{1}{k}\sum_{i=1}^{k} \widehat{H}(y_i), \\ \boldsymbol{y} &= \boldsymbol{W}\boldsymbol{x}, \end{aligned} \quad (1.5)$$

onde $\widehat{H}(y_i)$ é a entropia estimada de cada componente, medida a partir da equação 1.2 e com os valores de probabilidade (p) estimados pela sua frequência relativa em y_i.

Como os indivíduos são matrizes de elementos discretos e finitos, não podemos usar como métrica de distância da cob-aiNet[C], por exemplo, a norma euclidiana entre pontos de um espaço vetorial de valores reais. A distância é utilizada no cálculo da afinidade – o que influencia na concentração do indivíduo – e para a inserção de novos indivíduos caso eles estejam fora do limiar de supressão. Para este problema, então, escolhemos a distância de Hamming como métrica, que retorna o número de elementos que diferem entre duas matrizes, dividido pela dimensão total (k^2).

A operação de mutação sobre os indivíduos é feita substituindo uma das linhas da matriz representada pela soma desta linha com uma distinta. As duas linhas são escolhidas de forma aleatória. Uma outra possibilidade de mutação seria a troca de um elemento em determinada posição arbitrária da matriz. Entretanto, essa maneira foi descartada porque ela poderia gerar indivíduos fora do escopo de soluções factíveis, em outras palavras, matrizes não-inversíveis (com determinante nulo). Ao somar duas linhas e substituir uma destas por essa soma, o determinante se mantém não nulo e não caímos neste cenário indesejado.

A operação de busca local utilizada foi definida de maneira semelhante ao operador de mutação, entretanto, neste caso a substituição de uma linha só é aceita se ela traz uma melhoria no *fitness* do indivíduo, caso contrário, tenta-se uma outra possibilidade de troca.

Ao final da execução do algoritmo, a solução de ICA para o problema corresponderá ao indivíduo de melhor *fitness* da população.

2. SIMULAÇÕES

Uma vez que os conceitos foram assimilados, o passo final do projeto foi desenvolver um programa de computador, na linguagem C, que implementasse ICA em corpos finitos através do algoritmo cob-aiNet[C], conforme a especificação dada na seção 1.4, e, assim, analisar sua eficiência no problema de BSS e quais parâmetros influenciam seu desempenho. Para tanto foi desenvolvida uma simulação, com sinais em GF(2) – binários –, que funciona da seguinte maneira:

1. Geração do vetor de sinais originalmente emitidos pelas fontes hipotéticas, cujas probabilidades dos símbolos são: 0,2 e 0,8; 0,7 e 0,3; 0,9 e 0,1; 0,4 e 0,6; para os símbolos 0 e 1 e as fontes 1, 2, 3 e 4 respectivamente. As amostras de cada fonte são geradas de forma aleatória, obedecendo as distribuições especificadas. É gerada então a matriz de mistura $\boldsymbol{A}$, definida arbitrariamente, que, ao multiplicar o vetor de sinais $\boldsymbol{s}(n)$ – que contém as fontes geradas para o experimento – forma o vetor dos sinais observados $\boldsymbol{x}(n)$, o qual é dado como entrada para o programa.
2. Executar a cob-aiNet[C] para encontrar a matriz de separação, que estima o vetor de fontes original.
3. Repetir os passos 1 até 3 por 30 vezes.

Cabe aqui enfatizar que as amostras das fontes e a matriz de mistura para essas simulações eram sempre distintas a cada rodada/iteração, geradas de forma aleatória. A população inicial da cob-aiNet[C] é composta de matrizes com determinantes não nulos que também foram geradas de forma aleatória. Como já mencionado, o campo finito adotado nos testes foi o binário e testes foram realizados para separação de duas, três e quatro fontes.

Um subconjunto de parâmetros da cob-aiNet[C] foi definido *a priori* e permaneceu o mesmo em todas as simulações, conforme exibe a Tabela 1.

Tabela 1 – Parâmetros da cob-aiNet[C] constantes em todas as simulações

nAb	2
$maxAb$	100
nCl^{max}	10
nCl^{min}	3
β^{i}	2
β^{f}	0.2
LS_{it}	1
LS_{freq}	1

Dois parâmetros do algoritmo foram propositalmente alterados, a cada simulação, a fim de analisar a influência no desempenho da técnica:

- Número de Amostras dos sinais.
- Número de Iterações da cob-aiNet[C].

O número de amostras influencia a estimação da probabilidade de cada símbolo processado pela matriz de separação, o que por sua vez impacta na estimação das entropias destes sinais, consequentemente, no *fitness*. Quanto maior o número de amostras, mais preciso tende a ser o valor estimado de *fitness*. Os números de amostras variam de 2^5 a 2^{10}. O número de iterações que a cob-aiNet[C] foi executada, que impacta na convergência da busca, variou entre 300, 500 e 700 iterações máximas. O intuito da

análise do número de iterações é avaliar seu potencial efeito sobre a convergência do método para a solução.

Por fim, os parâmetros de concentração inicial – C_0 – e limiar de supressão – σ_s – foram definidos após a realização de testes preliminares, que levaram a C_0 = 1 e 0.5 e σ_s = h/k^2 com k sendo o número de fontes utilizadas no experimento e h sendo um fator multiplicador (o número de elementos diferentes entre duas matrizes).

3. RESULTADOS E DISCUSSÃO

Nesta seção estão contidos os resultados obtidos nas simulações desenvolvidas. Os gráficos apresentam no eixo das ordenadas a porcentagem de vezes em que o algoritmo conseguiu recuperar todas as fontes e no eixo das abscissas o número de amostras. As figuras 3.1 à 3.6 mostram os resultados obtidos nas simulações para duas fontes, com C_0 nos valores de 1 e 0.5 e σ_s com seu fator h variando entre 0.5, 1 e 2.

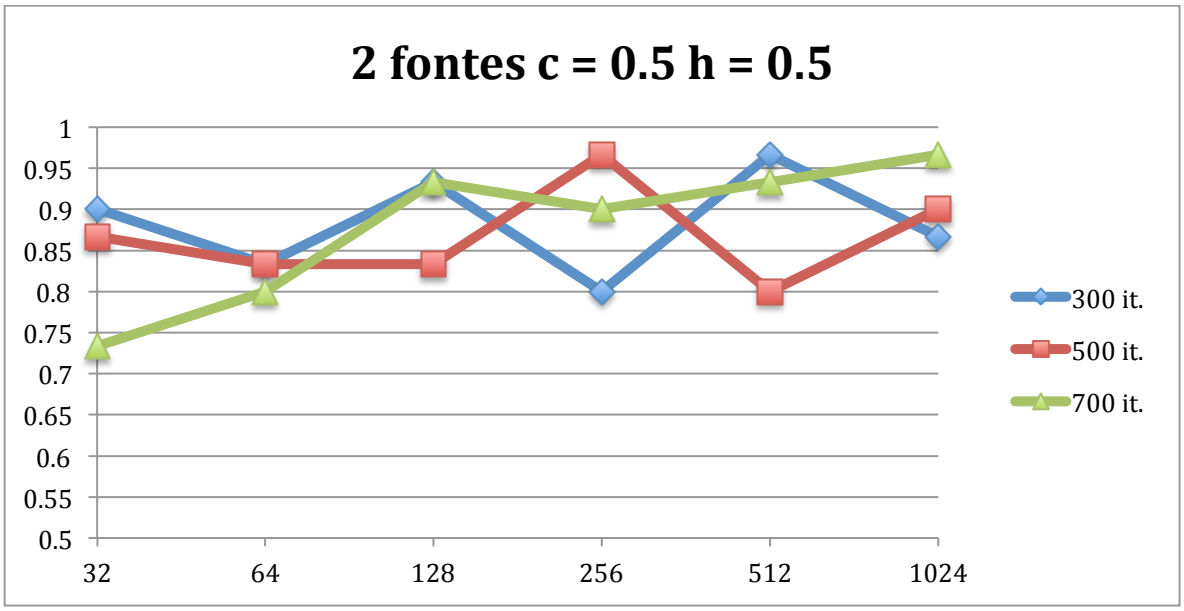

Figura 3.1 – Desempenho do algoritmo com C_0 = 0.5 e σ_s = h/k^2 com h = 0.5 para 2 fontes

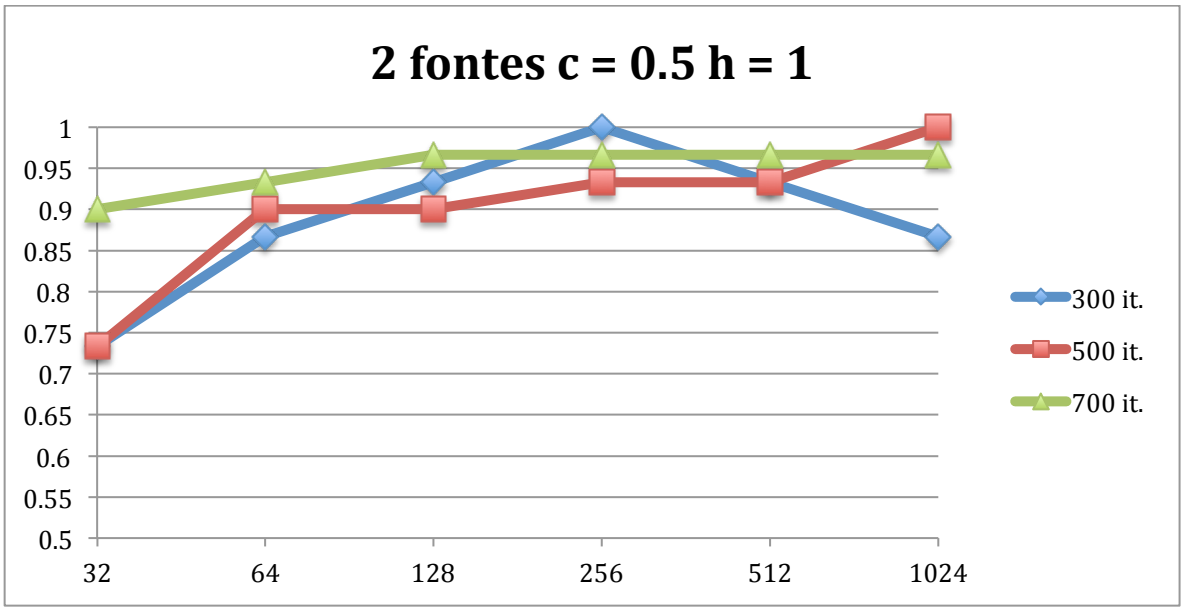

Figura 3.2 – Desempenho do algoritmo com $C_0 = 0.5$ e $\sigma_s = h/k^2$ com $h = 1$ para 2 fontes

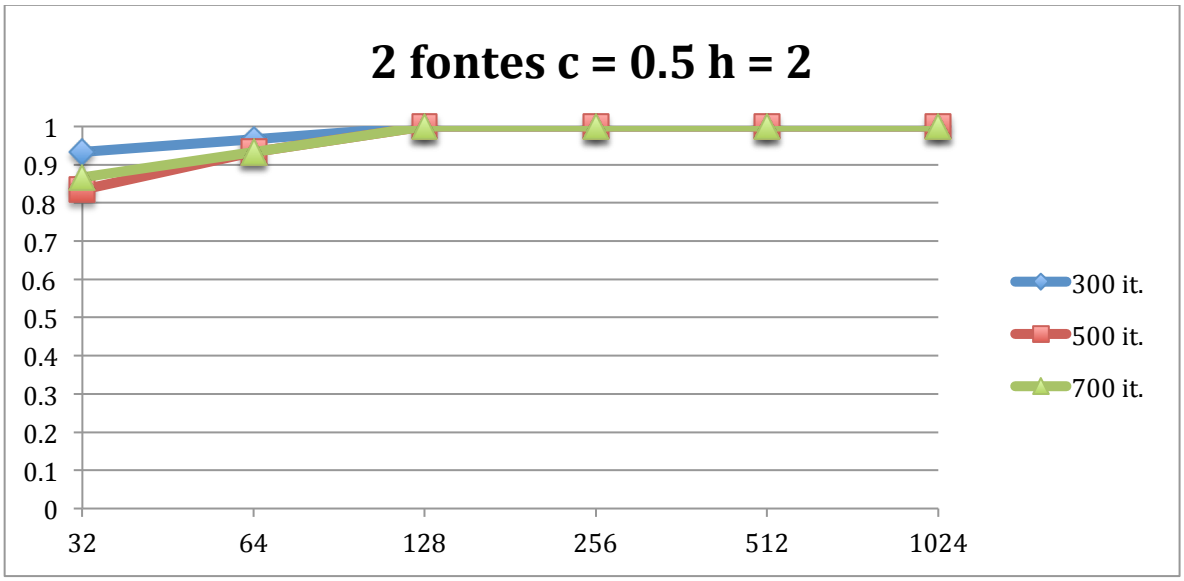

Figura 3.3 – Desempenho do algoritmo com $C_0 = 0.5$ e $\sigma_s = h/k^2$ com $h = 2$ para 2 fontes

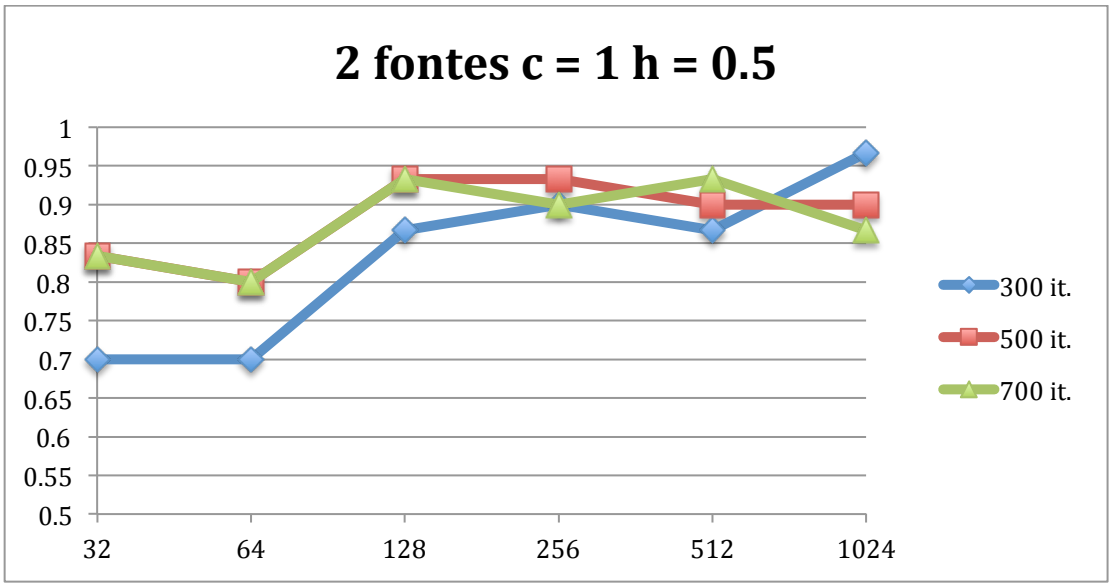

Figura 3.4 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 0.5$ para 2 fontes

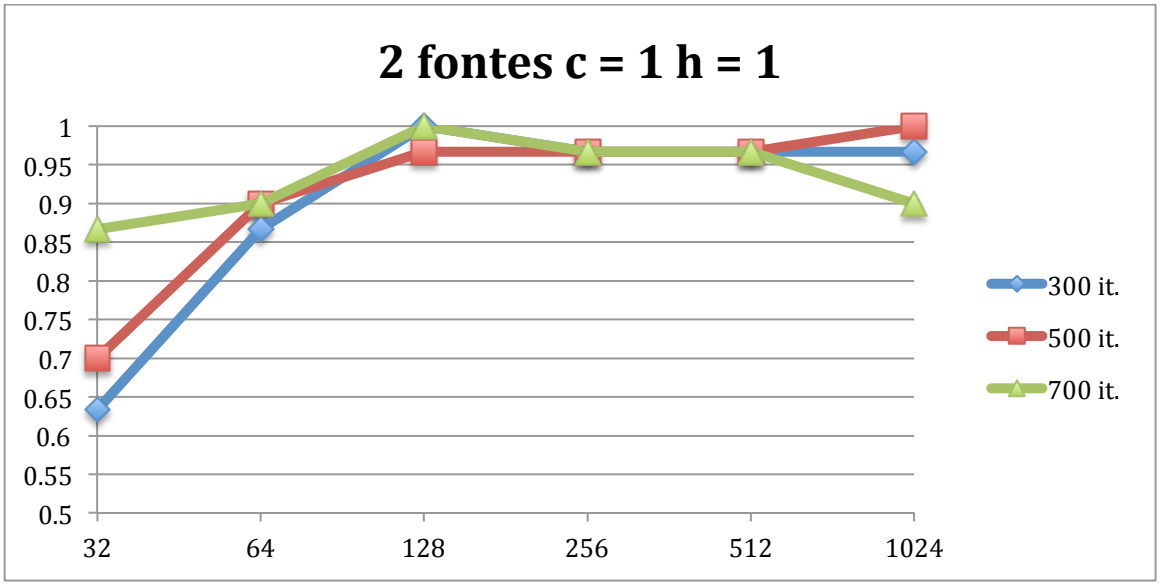

Figura 3.5 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 1$ para 2 fontes

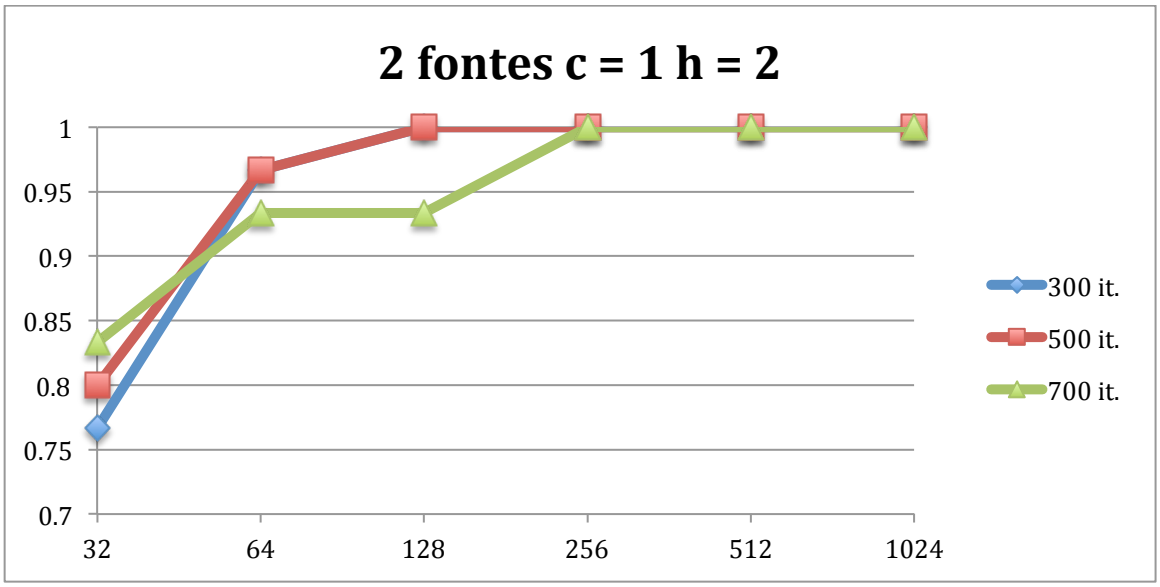

Figura 3.6 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 2$ para 2 fontes

Analisando os gráficos, percebemos que para os casos onde h assumiu o valor de 2, o crescimento da taxa de separação mostra-se constante à medida que as amostras aumentam, característica esta ainda mais acentuada no caso de $C_0 = 1$. Essa combinação apresentou o maior número de separações, atingindo 100% de separação em muitos dos casos e com o menor valor de 88,83% para o caso de 32 amostras, 500 iterações.

Para os casos com $h = 0.5$ a melhoria do algoritmo pelo aumento do número de amostras não seguiu o padrão esperado, como o caso da Figura 3.3, e teve seu comportamento ainda mais inesperado na Figura 3.1, caso onde $C_0 = 0.5$.

Deste modo, os experimentos indicam que para o caso binário e de 2 fontes, a melhor combinação do par, C_0 e h de σ_s são os valores 0.5 e 2 respectivamente, já que este cenário apresentou o maior número de separações totais.

Os resultados para 3 fontes seguiram um padrão semelhante aos das simulações para duas fontes: os experimentos apontaram que o melhor par (C_0, h) também foi o de valores 0.5 e 2, respectivamente. A partir dos gráficos 3.7 à 3.12, pode-se notar que a taxa de separação tem valores reduzidos em comparação aos valores para o caso de 2 fontes, isso por que a dimensão do espaço de busca (combinações possíveis de matrizes inversas) é agora maior, o que aumenta a complexidade do problema e dificulta a busca do algoritmo. Os gráficos para as simulações com 3 fontes são apresentados a seguir, nas figuras de 3.6 à 3.12.

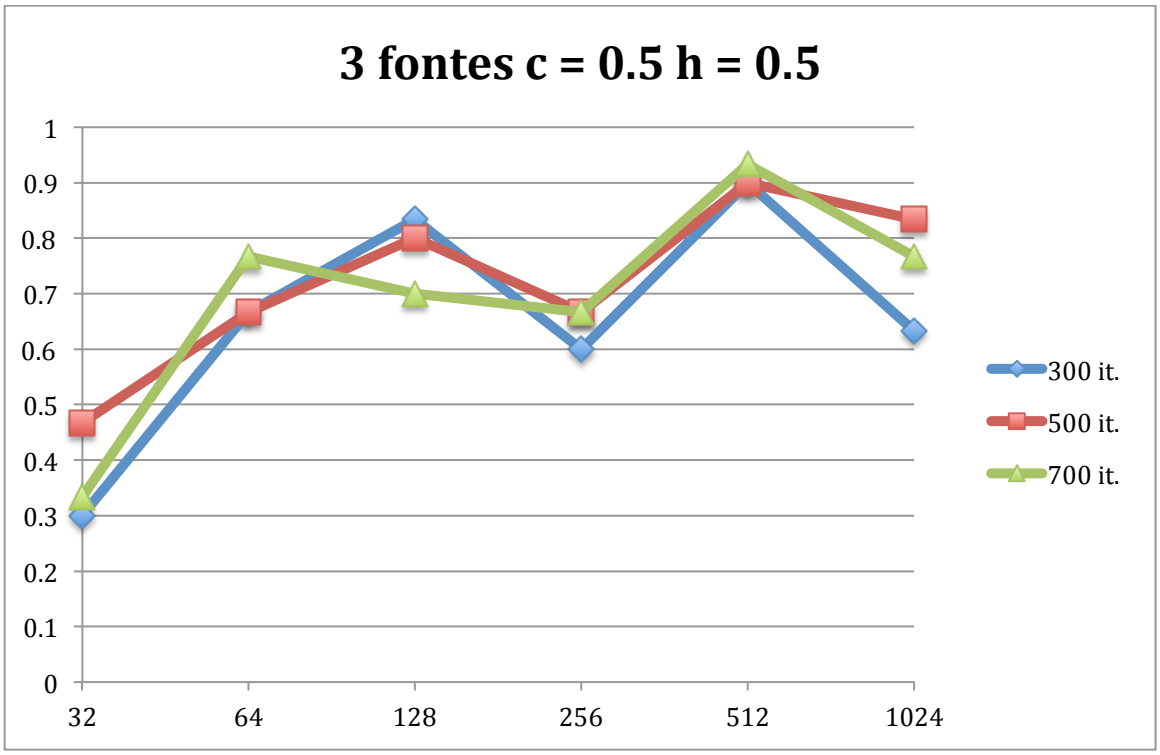

Figura 3.7 – Desempenho do algoritmo com $C_0 = 0.5$ e $\sigma_s = h/k^2$ com $h = 0.5$ para 3 fontes

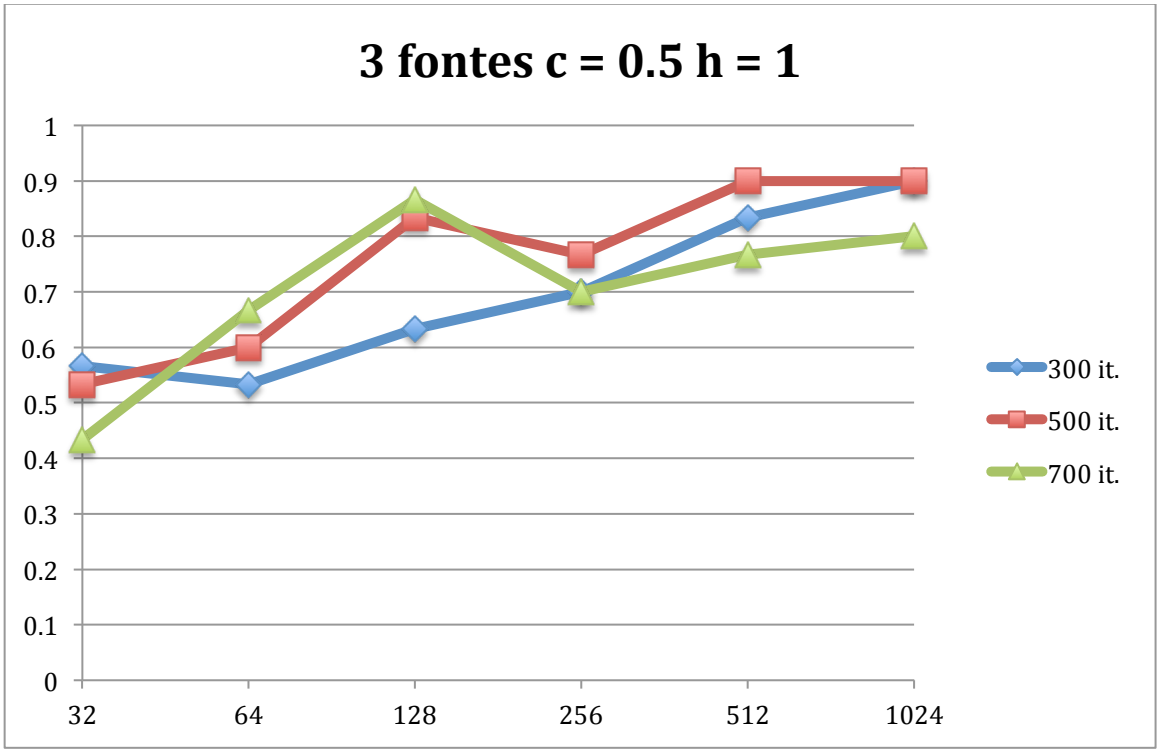

Figura 3.8 – Desempenho do algoritmo com $C_0 = 0.5$ e $\sigma_s = h/k^2$ com $h = 1$ para 3 fontes

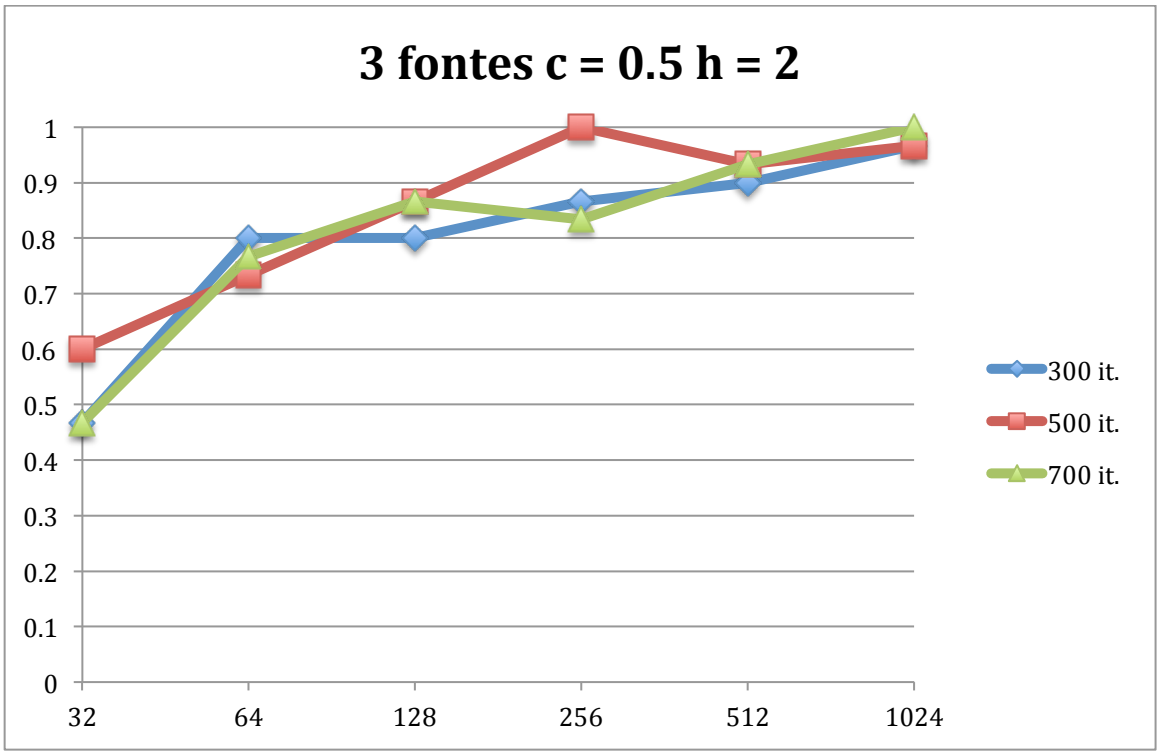

Figura 3.9 – Desempenho do algoritmo com $C_0 = 0.5$ e $\sigma_s = h/k^2$ com $h = 2$ para 3 fontes

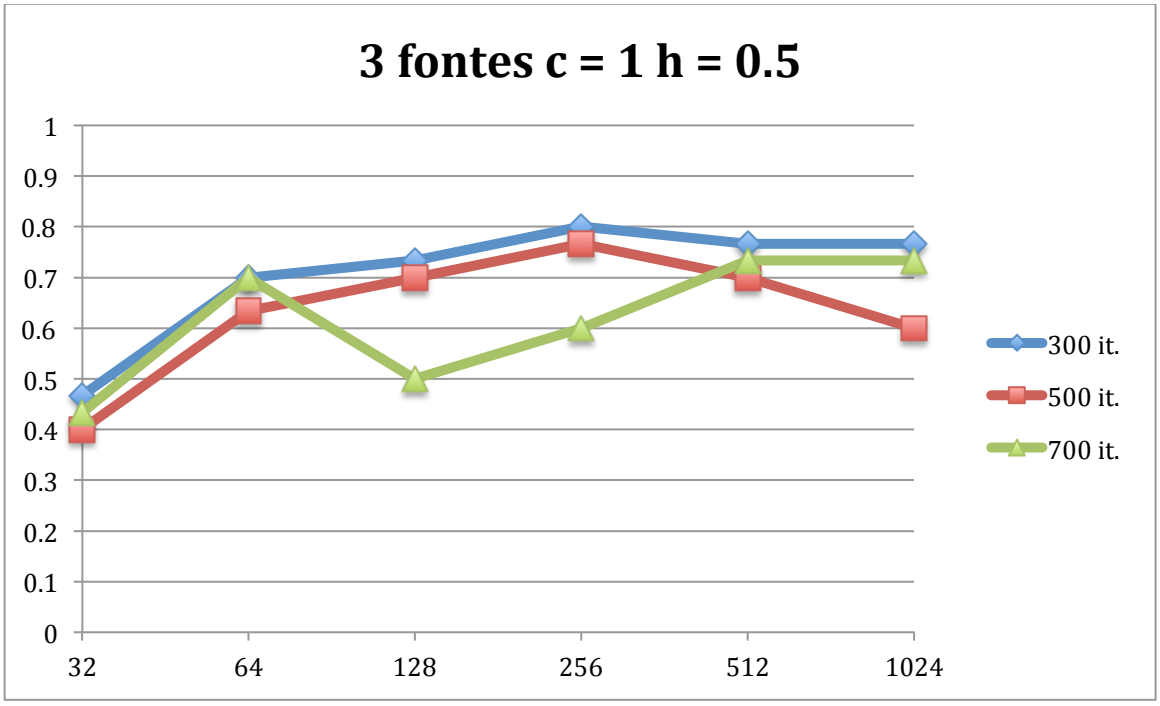

Figura 3.10 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 0.5$ para 3 fontes

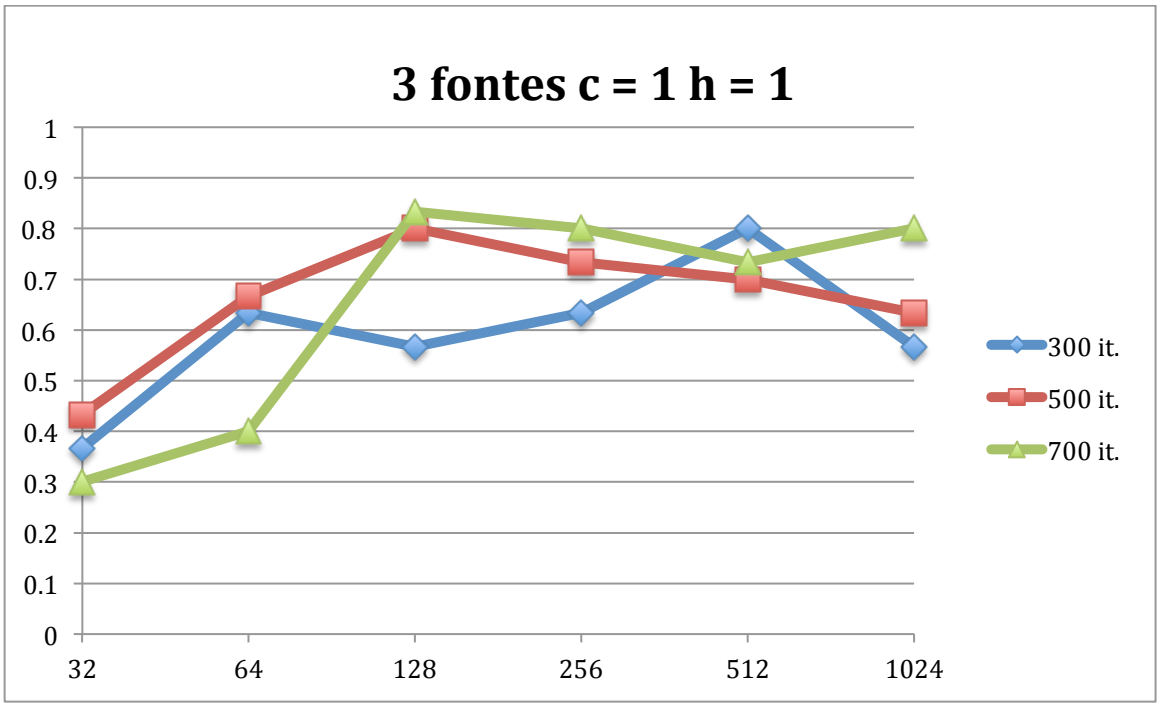

Figura 3.11 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 1$ para 3 fontes

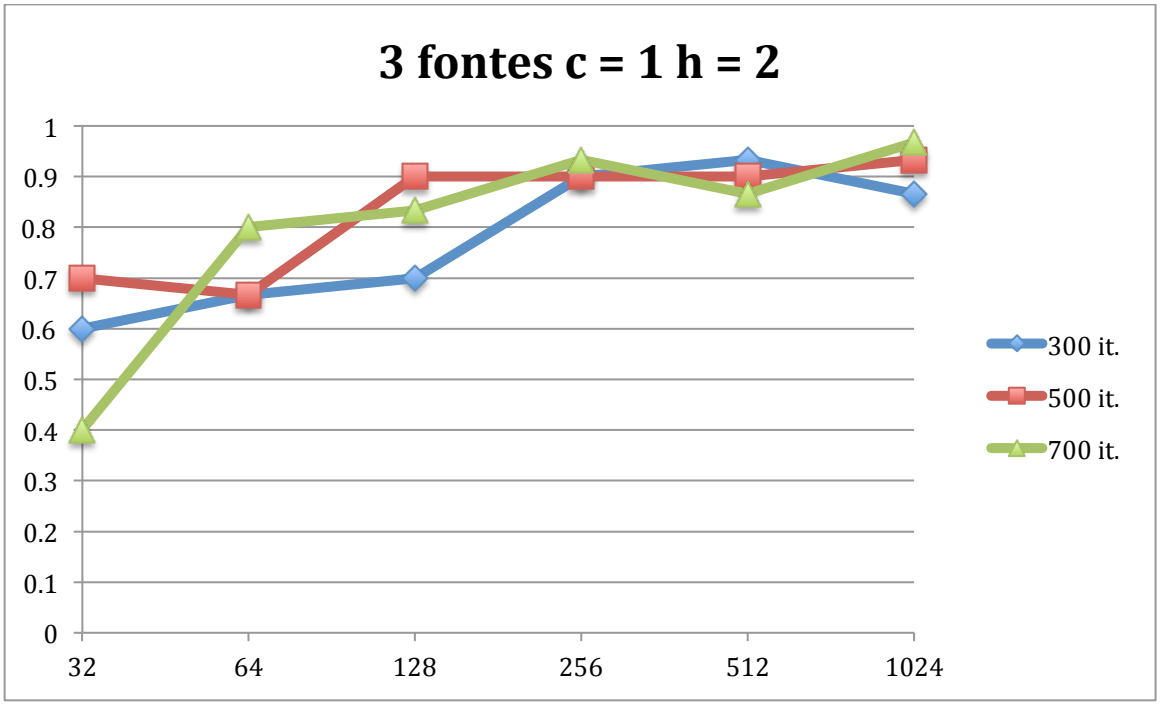

Figura 3.12 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 2$ para 3 fontes

Para o caso de 4 fontes, fica ainda mais nítida a necessidade de haver um maior número de amostras para refinar a estimativa da probabilidade e, consequentemente, o cálculo do *fitness* da solução candidata, e que haja uma determinação cuidadosa dos

parâmetros, já que os casos onde n era menor que 2, os resultados foram inferiores aos cenários de duas e três fontes, com um baixo aproveitamento no caso de $C_0 = 1$ e $h = 0.5$, como ilustram as figuras 3.13 à 3.15.

Analisando o comportamento geral do método, nas figuras 3.1 à 3.15, percebe-se que não há uma relação direta de melhora de desempenho com o número de iterações do algoritmo. É então possível levantar a hipótese de que 300 iterações são suficientes para a busca convergir e aumentar esse valor não deve impactar de forma positiva na eficiência do algoritmo.

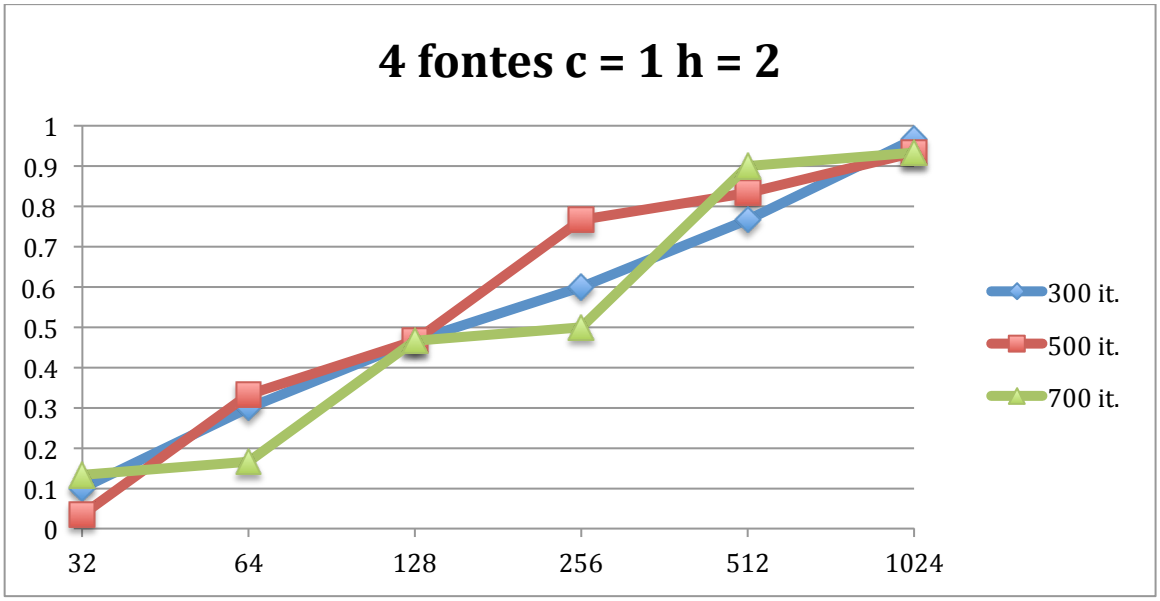

Figura 3.13 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 2$ para 4 fontes

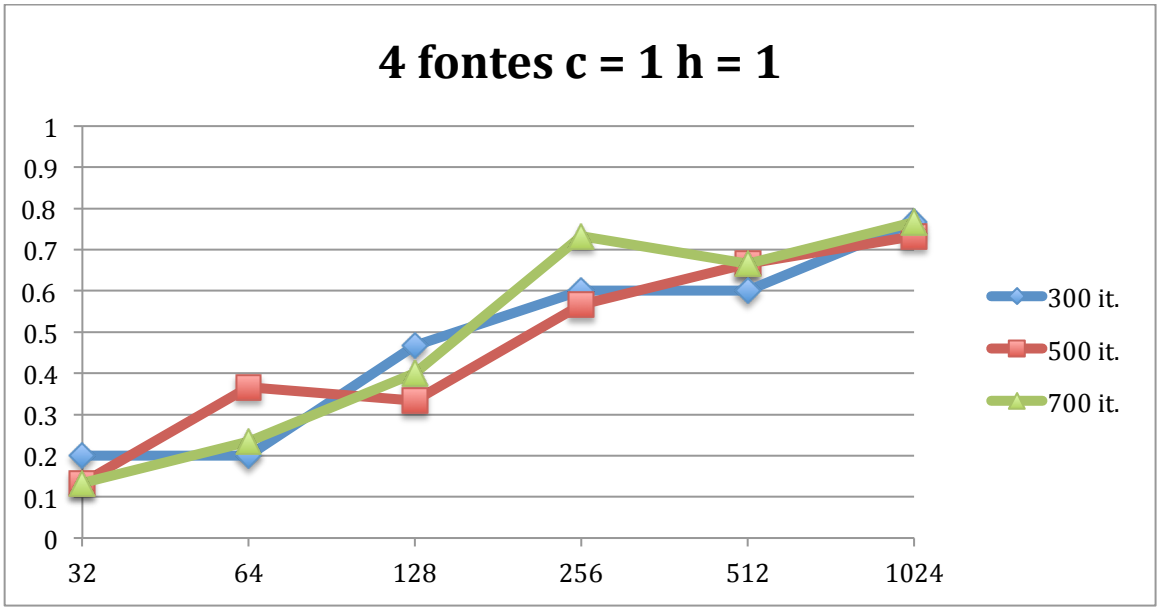

Figura 3.14 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 1$ para 4 fontes

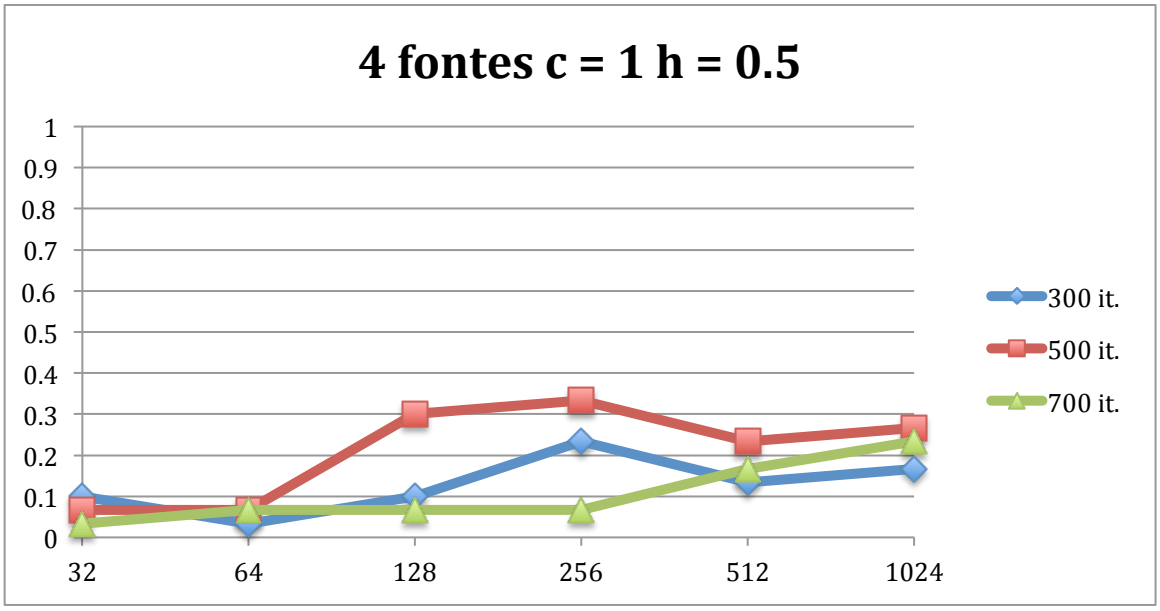

Figura 3.15 – Desempenho do algoritmo com $C_0 = 1$ e $\sigma_s = h/k^2$ com $h = 0.5$ para 4 fontes

4. Conclusão

Analisando o projeto como um todo, o aluno acredita que o trabalho teve um desempenho bastante positivo, tanto no âmbito dos resultados quanto no que assiste ao desenvolvimento do aluno. Ao âmbito dos resultados, analisados na seção anterior, foi atribuído esse valor uma vez que o algoritmo analisado, a cob-aiNet[C], apresentou um eficiente desempenho como ferramenta de ICA para a separação sega de sinais em campo binário, com resultados experimentais de até 100% de separação para os cenários com duas e três fontes, e 90% para o cenário com quatro fontes. Não obstante, a quase totalidade dos objetivos estabelecidos no projeto foram cumpridos de maneira bem sucedida.

No que compete ao desenvolvimento do aluno, o resultado foi satisfatório e de grande importância, uma vez que foi o seu primeiro contato com o ambiente acadêmico de pesquisas, e ele adquiriu grande experiência por meio de leituras de livros e artigos científicos nos temas de computação bio-inspirada, probabilidade, teoria da informação e processamento de sinais, contando com o auxílio do professor por meio de reuniões semanais para orientação. Adicionalmente, o horizonte de aplicação dos conceitos aprendidos em matérias como LE303 – Algoritmos e Programação de Computadores e GL301 – Estatística foi expandido, de modo que o aluno agora possui novas ferramentas e conhecimentos para abordar novos problemas que venham a aparecer em sua carreira acadêmica.

Agradecimentos

Gostaria de formalmente agradecer o Prof. Daniel Guerreiro e Silva e o Prof. Leonardo Tomazelli Duarte pela orientação na confecção desse trabalho, assim como no ensino das técnicas e teorias necessárias para desenvolvimento desse trabalho que foram importantes para motivar ainda mais o aluno a continuar o estudo de algoritmos e programação de computadores. Sou grato também aos membros do Laboratório de Processamento de Sinais para Comunicações da Faculdade de Engenharia Elétrica da Unicamp, à Faculdade de Ciências Aplicadas da Unicamp e ao CNPq e seu Programa Institucional de Bolsas de Iniciação Científica.

Bibliografia

BACK, T., FOGEL, D. B., & MICHALEWICZ, Z. (Eds.). *Evolutionary Computation 1: Basic Algorithms and Operators*. Bristol, UK: Taylor & Francis. 2000.

BURNET, F. M. . *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press. 1959.

COELHO, G. P. *Redes imunológicas artificiais para otimização em espaços contínuos : uma proposta baseada em concentração de anticorpos*. 2011. 233 f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas. 2011.

DE CASTRO, L. N. Fundamentals of Natural Computing: Basic Concepts, Algorithms and Applications. Boca Raton, USA: Chapman & Hall/CRC, 2006.

GUTCH, H. W., GRUBER, P., YEREDOR, A., & THEIS, F. J. ICA Over Finite Fields - Separability and Algorithms. *Signal Processing*, *92*(8), 1796–1808. 2012.

HYVARINEN, A.; KARHUNEN, J.; OJA, E. Independent Component Analysis. [S.l.]: John Wiley & Sons, 2001.

JERNE, N. K. Towards a Network Theory of the Immune System. *Annales d'Immunologie*, *125*(1-2), 373–389. 1974.

SHANNON, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal*, v. 27, p. 379–423, 623–656, 1948.

SILVA, D. G. *Aprendizado de máquina baseado na teoria da informação: contribuições à separação de sinais em corpos finitos e inversão de sistemas de Wiener*. 2013. 135 f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas. 2013.