ESTUDO DE TÉCNICAS DE PROGRAMAÇÃO INTEIRA NO PROBLEMA DE MONITORAÇÃO NÃO INTRUSIVA DE CARGAS RESIDENCIAIS

Pedro Paulo Marques do Nascimento

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Maurício Aredes
Heraldo Luís Silveira de Almeida

Rio de Janeiro
Março de 2013

ESTUDO DE TÉCNICAS DE PROGRAMAÇÃO INTEIRA NO PROBLEMA DE MONITORAÇÃO NÃO INTRUSIVA DE CARGAS RESIDENCIAIS

Pedro Paulo Marques do Nascimento

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO.

Examinado por:

Prof. Maurício Aredes, Dr.-Ing.

Prof. Heraldo Luís Silveira de Almeida, D.Sc.

Prof. Jorge Lopes de Souza Leão, Dr.Ing.

Prof. Juliano Freitas Caldeira, M.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2013

Marques do Nascimento, Pedro Paulo

Estudo de técnicas de programação inteira no problema de monitoração não intrusiva de cargas residenciais/Pedro Paulo Marques do Nascimento. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2013.

XII, 43 p.: il.; 29, 7cm.

Orientadores: Maurício Aredes

Heraldo Luís Silveira de Almeida

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Eletrônica e de Computação, 2013.

Referências Bibliográficas: p. 42 – 43.

1. Monitoração não intrusiva. 2. Programação inteira. 3. Clusterização. I. Aredes, Maurício *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Eletrônica e de Computação. III. Título.

A quem se considera digno desta dedicatória.

Agradecimentos

Em primeiro lugar, aos fatores aleatórios da vida que me fizeram chegar nesse momento.

Aos meus pais, Paulo e Edna, por me incentivarem nos estudos e todo apoio dado ao longo dos anos.

A minha irmã Isabela, pelas risadas.

Aos meus orientadores, pelos ensinamentos e apoio durante o projeto.

Aos amigos da faculdade e fora dela, pelos momentos de diversão e estudo, em especial ao Juliano Caldeira e ao Rodrigo Paim, pelas ajudas de caráter mais técnico durante a execução deste projeto.

A todos os outros que não foram citados, mas contribuíram de alguma forma para que isso acontecesse.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

Estudo de técnicas de programação inteira no problema de monitoração não intrusiva de cargas residenciais

Pedro Paulo Marques do Nascimento

Março/2013

Orientadores: Maurício Aredes
Heraldo Luís Silveira de Almeida

Curso: Engenharia Eletrônica e de Computação

O monitoramento não intrusivo de cargas é de grande importância pela sua praticidade e baixo custo na instalação e manutenção dos equipamentos de monitoração. No entanto, possui maior imprecisão e dificuldade de implementação em relação ao monitoramento intrusivo. Visando isso, grandes esforços vem sendo feitos para melhorarem os algoritmos e técnicas via software para que consigamos a maior fidelidade e velocidade possível na monitoração. Este trabalho propõe uma técnica para treinamento usando clusterização e técnicas de programação inteira para monitoramento em tempo real de cargas residências, a qual em grande parte podem ser modeladas como máquinas de estado finitos.

Palavras-chave: Monitoração não intrusiva, Programação inteira, Clusterização.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

RESEARCH OF INTEGER PROGRAMMING TECHNIQUES ON NONINTRUSIVE RESIDENTIAL LOAD MONITORING

Pedro Paulo Marques do Nascimento

March/2013

Advisors: Maurício Aredes
Heraldo Luís Silveira de Almeida

Course: Electronic Engineering

The nonintrusive load monitoring has great importance for its convenience and low cost of installation and maintenance of monitoring equipment. However possesses less accuracy and greater difficulty of implementation in relation to intrusive load monitoring. Aiming at this, great efforts are being made to improve the algorithms and techniques via software so that we can achieve the highest possible speed and fidelity in monitoring. This paper proposes a technique for training using clustering and integer programming techniques for real-time residential load monitoring, which at most can be modeled as a finite state machines.

Keywords: Nonintrusive load monitoring, Integer Programming, Clustering.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Tema

O tema do trabalho é o estudo de técnicas para a resolução do problema de monitoração não intrusiva de cargas residenciais. Neste sentido, pretende-se responder ao problema fazendo uma modelagem baseada em programação inteira[1] e vendo como diferentes técnicas de resolução do mesmo respondem. É esperado que as diferentes técnicas apresentem diferentes velocidades na resolução do problema. Além disso, variações nos parâmetros de coleta dos dados, tais como a taxa de amostragem, devem influenciar na taxa de acerto na identificação de cargas. Portanto, o foco deste trabalho é analisar variações nos parâmetros da modelagem, assim como os diferentes algoritmos de resolução de programação inteira[2] e com isso verificar a taxa de acerto e velocidade de resolução dos mesmos.

1.2 Delimitação

Os objetivos principais do trabalho são avaliar parâmetros de um sistema de monitoração não intrusiva de cargas residências e algoritmos de clusterização para o treinamento. A partir da potência total consumida em uma residência e de modelos preestabelecidos de cargas é possível em cada instante de tempo identificar quais cargas estão operando e em qual modo de operação as mesmas se encontram. Serão

utilizados para isso algoritmos, tais como: *backtracking* e *simulated annealing*.

1.3 Justificativa

Originalmente o monitoramento de cargas era feito de forma intrusiva, isto é, colocando-se sensores individuais em cada carga. Essa abordagem possui problemas, pois requer a instalação de um sensor em cada equipamento, o qual pode ser bastante custoso e inconveniente. Esse tipo de abordagem traz um foco no hardware, enquanto que a implementação de software é extremamente simples. Portanto a abordagem não intrusiva se torna muito atraente, em função de ser necessário apenas um medidor no ramo principal ou em cada fase, o qual mede a energia agregada, que no caso é de uma residência, e assim o trabalho de inspeção do funcionamento das cargas é feito utilizando software.

Existem diversas utilidades para um sistema de monitoramento não intrusivo[3] como, por exemplo: identificação de perfil detalhado de uso de cargas, gerenciamento de equipamentos, detecção de falhas e furtos, identificação de demandas de uso, entre outros. Portanto, conseguimos um sistema inteligente o qual é parte extremamente importante num contexto de *smart grid* e que pode aumentar em muito a eficiência energética de residências e outras instalações nas quais esse sistema venha a ser utilizado. Neste sentido, o presente projeto é uma complementação de estudos anteriores, buscando avançar na compreensão dos métodos de programação inteira (e o quão uteis eles podem ser) e como variações no processo de aquisição dos dados afetam a precisão e velocidade dos métodos. Sua originalidade reside no fato de não existir literatura ainda tão ampla e documentada, principalmente tratando-se de uma abordagem de programa inteira para a monitoração não intrusiva de cargas. Além disso ainda não há uma abordagem bem definida como melhor, sendo assim este um problema ainda em aberto.

1.4 Objetivo

O objetivo geral deste trabalho é, então, usando uma modelagem de programação inteira para o problema de monitoração não intrusiva de cargas, estudar como variações nos parâmetros de aquisição dos dados e diferentes algoritmos de resolução de programação inteira se comportam, avaliando o *trade off* entre velocidade e precisão dos mesmos. Desta forma, tem-se como objetivos específicos:

1. Estudar os aspectos gerais do problema de monitoração não intrusiva de cargas;
2. Estudar algoritmos de clusterização para o treinamento;
3. Estudar e pesquisar algoritmos de programação inteira, tais como o *backtracking* e *simulated annealing*;
4. Fazer a implementação dos algoritmos estudados;
5. Analisar variações dos algoritmos, variações dos parâmetros do problema e heurísticas nas buscas, verificando assim a sua eficiência e velocidade.

1.5 Metodologia

Este trabalho utilizará dados agregados de um medidor único em cada fase e dados de circuitos previamente medidos, para assim modelar os estados dos equipamentos e identificar a utilização das cargas no tempo.

Em um primeiro momento será realizado um extenso estudo sobre o problema de monitoração não intrusiva de cargas, o qual continuará até o fim do trabalho, usando para isso livros e artigos, além de entender melhor o funcionamento das técnicas de otimização combinatória e dos algoritmos de clusterização.

Entre os muitos métodos que vem sendo utilizados na tentativa de solucionar o problema de monitoração não intrusiva de cargas, escolhemos uma modelagem de programação inteira, procurando obter um melhor entendimento, aperfeiçoamento no modelo e resolução do problema, o qual será validado por meio de simulações.

O modelo proposto suporta aplicações que tem um funcionamento modelável por uma máquina de estados, ou seja, possuem um número discreto de estados distintos com relação ao seu consumo. Será medido para cada estado (de cada equipamento) o comportamento da potência ao longo do tempo.

A partir dos dados agregados e dos dados dos equipamentos, será simulado o algoritmo de monitoração dos mesmos. Assim tendo um gabarito dos estados de funcionamento dos equipamentos em cada momento, compararemos os resultados gerados pelo algoritmo com este gabarito verificando assim o percentual de acerto do mesmo. Além disso verificaremos a velocidade dos algoritmos, algoritmos com tempo de identificação muito rápido são essências para o monitoramento de cargas em tempo real.

1.6 Descrição

No Capítulo 2 serão discutido os conceitos básicos de monitoração de cargas em um contexto geral e do projeto apresentado especificamente.

No Capítulo 3 são discutidos os fundamentos teóricos inerentes as simulações que serão posteriormente realizadas.

No Capítulo 4 são mostrados os resultados das simulações, através de gráficos e tabelas, os quais foram obtidos através da teoria desenvolvida.

No Capítulo 5 encontram-se as conclusões obtidas, assim como os trabalhos a serem realizados no futuro.

Capítulo 2

Monitoração de cargas

2.1 Conceitos básicos

Os primeiros estudos na tentativa de a partir dos dados agregados de uma residência descobrir-se o estado de cada carga em um dado momento, foram propostos por HART [4] nos anos 80 e 90, focando em cargas residenciais, as quais são em grande parte apenas resistivas. O método utilizado focava nas variações da potência total, para a partir dessas mudanças identificar quais cargas estão sendo utilizadas. Ele foi pensado principalmente para identificação de cargas com apenas 2 estados ($On/\ Off$) e com consumo constante (em contraste com um consumo variável).

A partir de então, muitas abordagens vem sendo desenvolvidas. Do ponto de vista do hardware (aquisição de dados), pode ser utilizado um medidor de baixa amostragem (1 Hz), sendo o medidor assim mais barato ou um medidor com alta amostragem (1kHz), o qual possui um custo mais elevado. Do ponto de vista do software, muitas técnicas distintas vêm sendo utilizadas e desenvolvidas, tais como: programação inteira, modelo oculto de *markov*, redes neurais, algoritmos genéticos, clusterização, entre muitos outros. Ainda não há uma técnica que se sobressai claramente sobre as demais.

No contexto de monitoração de cargas alguns conceitos básicos e terminologias básicas são comumente utilizados, alguns exemplos são:

- **Monitoração intrusiva**: A monitoração intrusiva é quando cada equipamento possui um monitor e assim tem seus dados extraídos diretamente. A vantagem desse método é a precisão e confiabilidade da medição, a qual fica sujeito apenas a erros de medida. As desvantagens são a dificuldade de instalação em cada equipamento e o custo de instalação dos mesmos.

- **Monitoração não intrusiva**: Quando a monitoração é feita com poucos medidores, o qual medem dados agregados de um conjunto de equipamentos, em contraste com apenas um monitor por equipamento, dizemos que o monitoramento foi feito de forma não intrusiva, em geral é utilizado um monitor para cada fase ou apenas um para a casa toda. Este método perde confiabilidade, dependendo agora em grande parte dos softwares utilizados, porém é muito mais simples e barata a instalação de poucos medidores, sendo assim objeto de estudo atualmente.

- **Treinamento supervisionado**: Este termo é utilizado quando é utilizado para o treinamento dos algoritmos de monitoração, dados individuais dos equipamentos de uma residência, ou seja, é feito o monitoramento intrusivo do equipamento da residência em um primeiro momento, para esses dados serem assim utilizados na monitoração não intrusiva.

- **Treinamento não-supervisionado**: No treinamento não supervisionado não há dados previamente calculados, a partir dos dados agregados da residência o algoritmo automaticamente consegue identificar os diferentes equipamentos existentes em uma residência, sendo assim esse método muito desejado, porém de difícil implementação.

- **Amostragem em baixa frequência**: A amostragem em baixa frequência em geral possui frequência de amostragem menor ou da ordem de uma amostra por segundo, muitas vezes são coletadas amostras uma vez por minuto ou hora. Com taxas de amostragem dessa ordem, frequência ativa e reativa não são distinguíveis, assim como harmônicos e outros recursos, isso se dá em função

do teorema de *Nyquist-Shannon*.

- **Amostragem em alta frequência**: São amostragens da ordem de kHz e MHz, em geral é feita a amostragem da corrente e da voltagem em cada fase. Com dados amostrados em alta frequência temos muitos recursos que podem ser utilizados nos algoritmos de monitoração, tais como uma série temporal com um certo grau de fidelidade, potência ativa e reativa, harmônicos de alta ordem, entre outros. A desvantagem se dá pelo preço de medidores com taxa de amostragem dessa ordem.

2.2 *Reference Energy Disaggregation Data Set*(REDD)

O REDD[5] é um *dataset* livre, proposto pelo MIT, o qual foi designado para ser um *dataset* de referência em monitoração não intrusiva de cargas. Entre os seus principais objetivos esta ser um *dataset* de *benchmarking* padrão para algoritmos de monitoração. Antes do mesmo, muitos *datasets* eram utilizados, muitos coletados pelo próprio autor do algoritmo, assim era difícil a comparação entre algoritmos, em função de cada autor utilizar seus dados e suas métricas de avaliação da precisão dos mesmos.

O REDD foi pensado para ser um *dataset* amplo, já que o seu objetivo é comparar diferentes algoritmos e técnicas. Foram coletados dados da forma de onda AC de corrente e tensão, a uma frequência de 15kHz. Com isso é possível extrair muitos recursos, assim como avaliar tanto abordagens de alta e baixa (fazendo subamostragem) frequência de amostragem.

Os dados foram coletados de 10 residências da região de *Massachussetts*, nos EUA, em residências onde o sistema era bifásico. Um dos problemas do *dataset* é o fato de não ter sido gravado dados dos equipamentos individualmente, apenas de circuitos. Sendo assim, só é possível identificar equipamentos que utilizem um circuito sozinho, no entanto a maior parte dos circuitos são utilizados por apenas

um equipamento ou equipamentos com características parecidas.

O *dataset* é separado em 3 conjuntos de dados:

- **low_freq.tar.bz2** - Contém dados de 6 residências, cada uma com cada fase amostrada a uma frequência de 1 Hz e de circuitos individuais amostrados a cada 3 ou 4 segundos. No arquivo contem a potência ativa gasta pelo equipamento, assim como o UTC *timestamp* do momento. Cada circuito tem um *label* que busca caracterizar da melhor forma possível o mesmo.

- **high_freq.tar.bz2** - Esse arquivo contém os dados das formas de onda de corrente das duas fases e tensão de uma das fases amostrados em alta frequência. No entanto, usando o fato de que as formas de onda permanecem praticamente constante por grandes períodos, os dados são comprimidos e a forma de onda é descrita apenas nos pontos onde há "mudança no sinal".

- **high_freq_raw.tar** - Esse arquivo contém a forma de onda de corrente e tensão na sua forma "pura", sem métodos de compressão.

No referente projeto utilizaremos uma abordagem de baixa frequência, para isso usaremos o arquivo **low_freq.tar.bz2**. Foram utilizados mais especificamente os dados da casa 3, porém poderíamos utilizar a mesma abordagem para qualquer uma das outras casas. O arquivo da casa 3 contém os dados amostrados de cada circuito, a qual possui um total de 22 circuito sendo os circuitos 1 e 2 referentes a cada uma das fases (fases 1 e 2), convêm lembrar que as fases são amostradas a cada 1 segundo e os circuitos são amostrados a cada 3 ou 4 segundos. No total foram coletados 17 dias de medição. A tabela 2.1 mostram os *labels* que melhor descrevem cada circuito da casa, em relação as cargas que estao associadas com aquele circuito.

Tabela 2.1: *Labels* que descrevem os tipos de cargas associadas a cada circuito

circuito	*label*
1	*mains*
2	*mains*
3	*outlets_ unknown*
4	*outlets_ unknown*
5	*lighting*
6	*electronics*
7	*refrigerator*
8	*disposal*
9	*dishwasher*
10	*furance*
11	*lighting*
12	*outlets_ unknown*
13	*washer_ dryer*
14	*washer_ dryer*
15	*lighting*
16	*microwave*
17	*lighting*
18	*smoke_ alarms*
19	*lighting*
20	*bathroom_ gfi*
21	*kitchen_ outlets*
22	*kitchen_ outlets*

Capítulo 3

Fundamentos Teóricos

3.1 Programação dinâmica

Programação dinâmica consiste em um método de resolver um problema subdividindo o mesmo em subproblemas menores. As duas propriedades básicas de um problema que permitem a aplicação de um algoritmo de programação dinâmica são a sobreposição de subproblemas e a existência de uma subestrutura ótima [6]. A técnica de programação dinâmica é comumente empregada em problemas de otimização.

Na solução de um problema usando programação dinâmica os passos necessários são:

1. Caracterizar a estrutura de uma solução ótima.

2. Recorrentemente definir a solução ótima.

3. Definir o caso base da recorrência.

Um exemplo clássico de problema que pode ser resolvido usando o método de programação dinâmica é o problema da mochila inteiro, o qual é enunciado a seguir:

Dado um conjunto de N itens, cada um com valor e peso dados, e uma mochila com uma capacidade máxima de peso fixa, quais itens devem ser colocados na mochila de forma a maximizar o valor total carregado?

Problema:

$$\text{maximizar} \sum_{i=1}^{N} v_i * x_i$$
$$\text{sujeito a} \sum_{i=1}^{N} p_i * x_i \leq P \tag{3.1}$$

- v_i é o valor de cada item
- p_i é o peso de cada item
- $x_i = 0, 1$
- P é a capacidade máxima da mochila

A modelagem por programação dinâmica é dada pela seguinte função:

1. Estrutura ótima: $f[i, capacidade]$, essa função indica qual o máximo valor que pode ser alcançado usando itens na mochila com índices de 1 até i e com mochila de tamanho *capacidade*.

2. Recorrência:

Se $p_i > capacidade$, então:

$$f[i, capacidade] = f[i-1, capacidade], \tag{3.2}$$

pois o peso atual do índice a ser colocado excede a capacidade restante da mochila, logo a única opção é descartar o mesmo.

Se $p_i \leq capacidade$, então:

$$f[i, capacidade] = max(f[i-1, capacidade], f[i-1, capacidade - p_i] + v_i), \tag{3.3}$$

a recorrência analisa o máximo entre duas possibilidades: descartar o item atual ou colocar o mesmo na mochila, reduzindo assim o seu peso da capacidade total e adicionando o seu valor ao resultado.

3. Caso base:

$$f[0, capacidade] = 0 \tag{3.4}$$

A solução do problema é encontrada em $f[N, capacidadeTotalMochila]$.

3.2 Algoritmo de programação dinâmica para clusterização 1D

O algoritmo usando programação dinâmica para a clusterização 1D é de grande importância, pela sua velocidade e garantia de eficiência, pois ele consegue obter a clusterização ótima no sentido de minimizar a seguinte função:

$$\begin{aligned} min \sum_{j=1}^{k} \sum_{i=1}^{n} (x_{i,j} * (P_i - u_j)^2) \\ u_j = \frac{\sum_{i=1}^{n} x_{i,j} * P_i}{\sum_{i=1}^{n} x_{i,j}} \\ P_i \leq P_{i+1} \\ \sum_{j=1}^{k} x_{i,j} = 1, \end{aligned} \tag{3.5}$$

1. n é a quantidade total de pontos
2. k é a quantidade de *clusters*
3. u_j é o centro do *cluster*
4. P_i é o i-ésimo ponto
5. $x_{i,j}$ vale 1 se o P_i pertence ao *cluster* j e 0 caso contrário

a qual é conhecida como WCSS(*within-cluster sum of squares*).

Para resolver esse problema é utilizada a seguinte programação dinâmica:

1. Estrutura ótima:

$F[i,k]$ indica o mínimo que se pode obter na Eq. (3.5), usando pontos com índices maiores ou iguais a i e sendo necessário formar k *clusters*.

2. Recorrência:

$$F[i,k] = min\{F[j+1,k-1] + C[i,j]\}, i \leq j \leq n-k$$
$$C[i,j] = \sum_{l=i}^{j}(P_l - u_k)^2$$
$$u_i = \frac{x_i + (i-1) * u_{i-1}}{i}$$
$$P_i \leq P_{i+1}, \tag{3.6}$$

3. Caso base:

$$F[n,0] = 0$$
$$F[i,0] = \infty, \forall i < n, \tag{3.7}$$

A figura 3.1 ilustra o método de programação dinâmica. É preciso encontrar o valor de j que minimiza a soma das duas parcelas vistas.

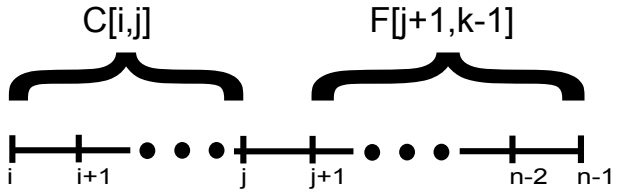


Figura 3.1: Recorrência do algoritmo de programação dinâmica para clusterização 1D.

Temos que $C[i,j]$ é o *cluster* usando pontos de i ate j inclusive. Calculando $C[i,j]$ da forma trivialmente enunciada a complexidade fica $O(kn^3)$, entretanto como indicado em WANG e SONG [7] usando a recorrência 3.8 para o cálculo de $C[i,j]$,

$$C[i,j] = C[i,j-1] + \frac{(j-i)\,(x_{j-i+1} - u_{j-i})^2}{j-i+1}$$

$$C[i,i] = 0, \tag{3.8}$$

a complexidade total do algoritmo de programação dinâmica fica em $O(kn^2)$.

3.3 Uma otimização baseada em divisão e conquista e programação dinâmica para clusterização 1D

Dependendo da quantidade de pontos e do número de *clusters*, o algoritmo de programação dinâmica básico, pode ser lento. Este é o nosso caso, onde temos um n da ordem de 10^5 e um k da ordem de 10, um algoritmo com complexidade $O(kn^2)$ se torna inviável pois o número total iterações do algoritmo estaria na ordem de 10^{11}. Tendo em vista essa limitação, analisaremos propriedades da recorrência e da estrutura do problema, para assim melhorarmos a complexidade do algoritmo. Utilizaremos uma otimização baseada na técnica de divisão e conquista [8, 9].

Recorrências da forma:

$$F[i,k] = min_{i \leq j \leq n-k}\{F[j+1,k-1] + C[i,j]\}, \tag{3.9}$$

onde $C[i,j]$ é alguma função de custo dada e quando a função de custo satisfaz a desigualdade quadrangular [10]:

$$C[a,c] + C[b,d] \leq C[a,d] + C[b,c], a \leq b \leq c \leq d, \tag{3.10}$$

podemos otimizar esse tipo de recorrência, o qual originalmente possui complexidade $O(kn^2)$ para $O(knlog\ n)$, o qual em muitas aplicações, como no caso do nosso problema, é uma otimização imprescindível.

Vamos definir $opt[i,k]$ como o menor índice j, tal que: $F[i,k] = F[j+1,k-1] +$

$C[i,j]$. Nós temos que: $opt[0,k] \leq opt[1,k] \leq opt[2,k] \leq ... \leq opt[n-1,k]$.

Vamos calcular $F[i,j]$ iterativamente e crescentemente para valores de k.

O pseudocódigo da otimização é dado pelo algoritmo 1.

Algoritmo 1 Algoritmo de clusterização 1D

Calcular(k, L, R, optL, optR) =

Caso especial: L>R: fim.

Defina M = (L+R) / 2.

Resolva F[M,k] e opt[M,k], limitando o *loop* para no máximo usar (optR-optL+1) operações.

Calcular(k, L, M-1, optL, opt[M,k])

Calcular(k, M+1, R, opt[M,k], optR)

O código acima possui complexidade $O(nlog\ n)$, usando o mesmo para cada valor de *cluster*, obtemos uma complexidade de $O(knlog\ n)$. Ele basicamente calcula todos os valores de $F[i,k]$ para $L \leq i \leq R$. Para isso subdivide em 2 casos, o calculo de $F[i,k]$ para $L \leq i \leq M-1$ e $F[i,k]$ para $M+1 \leq i \leq R$, onde $M = \frac{(L+R)}{2}$ e usando o fato de que $opt[i,k]$ é monotonicamente crescente para valores também crescentes de j, podemos provar que a cada passo o loop principal usa $O(n)$ passos e árvore tem altura no máximo $O(log\ n)$, levando a complexidade $O(nlog\ n)$ para cada *cluster*.

Podemos também calcular em tempo $O(1)$ com pré-computação de $O(n)$ o valor de $C[i,j]$, o qual é fundamental para que o algoritmo fique com tempo $O(knlog\ n)$.

3.4 Critério de identificação automática do número de *clusters*

Um critério de identificação automática do número de *clusters* é de grande importância para a correta classificação de quais cargas estão em quais estados em cada momento. Portanto, este é um ponto o qual deve ser dada uma grande importância. Usaremos esse método para identificar o número de modos de operação de cada circuito, pois cada *cluster* esta associado com um modo de operação.

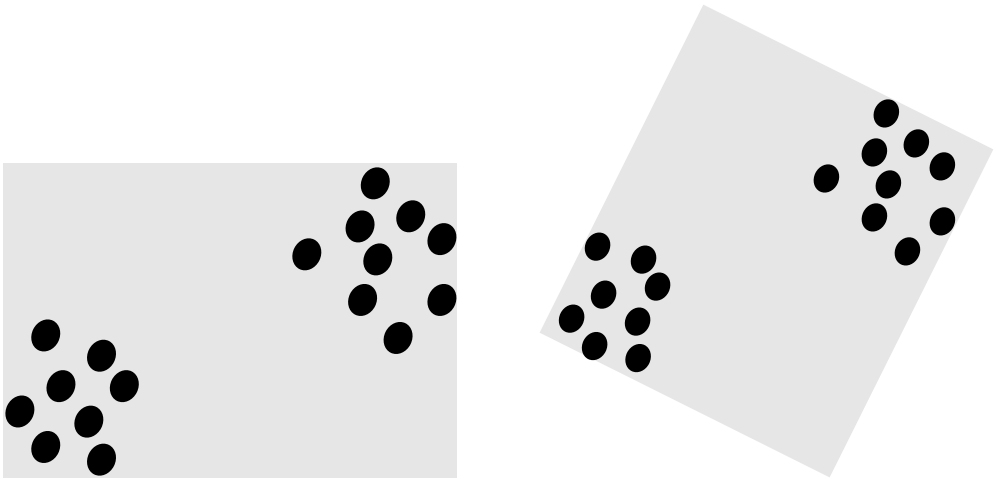
Por muito tempo o método mais utilizado foi o *elbow method*, no entanto alguns problemas eram encontrados, como por exemplo, não ter uma clusterização de referência e as diferenças não serem normalizadas para comparação. Um método que vem sendo utilizado é o método de *gap statistic*[11], o qual tem se demonstrado melhor na prática.

Problemas de usar o *gap statistic* surgem quando, por exemplo, o algoritmo *k-means* fica com desempenho baixo, no sentido de minimizar o WCSS, o que pode acabar por prejudicar o *gap statistic*. Usando a abordagem com programação dinâmica (a qual sempre acha o minimo WCSS), esse problema não existe.

Definiremos o conceito de *gap* pela seguinte equação:

$$Gap_n(k) = E_n^*[log(W_k)] - log(W_k), \tag{3.11}$$

onde E_n^* é o valor esperado amostrando uma distribuição de referência de tamanho n. Para calcular $E_n^*[log(W_k)]$ o artigo propõe amostragem em uma distribuição uniforme. A distribuicao uniforme escolhida é determinada pelo *bounding box* dos pontos originais. Há 2 modos de determinar o *bounding box*, como mostrados na figura 3.2. No modo da figura 3.2a, calculamos o *bounding box* com base nos eixos dos dados. No modo da figura 3.2b, calculamos com base na análise de componentes principais dos dados.



(a) Alinhado com eixo dos *features* (b) Alinhado com os eixos principais

Figura 3.2: Tipos de *bounding box* dos dados

Para o caso do nosso problema, onde apenas usaremos a potência ativa dos equipamentos e por isso os dados estão em apenas uma dimensão, o *bounding box* nada mais é do que um segmento de reta limitado pelos elementos máximo e mínimo do conjunto. Portanto as duas abordagens são equivalentes.

O algoritmo para cálculo do número de *clusters* é descrito a seguir:

1. Variar o número total de *clusters* K e calcular o WCSS para cada K.

2. Gerar B *datasets* de referência, amostrados de uma distribuição uniforme e assim computar o *gap statistic* estimado:

$$Gap(k) = (\frac{1}{B}) \sum_{b} log(W^*_{kb}) - log(W_k). \tag{3.12}$$

3. Vamos definir $\bar{l} = (\frac{1}{B}) \sum_b log(W^*_{kb})$, para assim calcularmos o desvio padrão:

$$sd_k = \sqrt{[\frac{1}{B} \sum_{b} log(W^*_{kb} - \bar{l})^2]} \tag{3.13}$$

Definindo $s_k = sd_k \sqrt{1 + \frac{1}{B}}$, temos que o número de *clusters* estimado é dado por:

$$\hat{k} = menor\ k\ tal\ que\ Gap(k) \geq Gap(k+1) - s_{k+1} \tag{3.14}$$

3.5 Algoritmos para monitoração não intrusiva de cargas

Após descobrimos o número de *clusters* pelo *gap statistic* e assim usando o algoritmo de programação dinâmica descobrirmos a clusterização ótima para cada *cluster*, temos os modos de operação de cada circuito (o qual é o centro de cada *cluster*). A partir do momento que temos os modos de operação dos circuitos modelados, precisamos encontrar, usando para isso os dados agregados de potência ativa em cada fase, qual o modo de operação de cada equipamento. Para atingir esse objetivo usaremos 2 abordagens, *backtracking* e *simulated annealing*, ambas buscando minimizar o erro quadrático entre os dados medidos e os estados estimados.

3.5.1 *Backtracking*

A descrição geral de um algoritmo de *backtracking* é feita em KLEINBERG e TARDOS [12].

O algoritmo de *backtracking* designado para resolver o problema é descrito a seguir:

1. Para cada circuito, selecionamos um dos possíveis modos para ser o modo atual de operação do circuito.

2. Para cada modo escolhido associamos a ele a uma das duas fases do sistema, fase1 ou fase2.

3. Fazemos isso ate todos os circuitos estiverem com o seu modo de operação definido, como visto na figura 3.3 onde cade elemento C_{ij} indica o j-ésimo modo do i-ésimo circuito. Em azul estão os modos associados a fase1 e em vermelho os modos associados a fase2.

4. Definiremos assim:

   - $soma\_fase1$ = soma de todos os modos associados a $fase1$.
   - $soma\_fase2$ = soma de todos os modos associados a $fase2$.

5. Entre todas as possibilidades de modo de operação para os circuitos e a fase associada a esse modo, buscamos a configuração que minimiza a seguinte função:

$$F = min \sum_{t=ti}^{tf} (medida\_fase1(t) - soma\_fase1)^2 + (medida\_fase2(t) - soma\_fase2)^2, \tag{3.15}$$

onde $medida\_fase1(t)$ e $medida\_fase2(t)$ são os valores medidos em cada fase no instante de tempo t, temos também que tf e ti são os instantes final e inicial, respectivamente. Visto que a taxa de amostragem nas fases é maior que nos circuitos(3 ou 4 vezes maior), consideramos que os equipamentos não mudam de modo de operação entre amostras consecutivas dos circuitos.

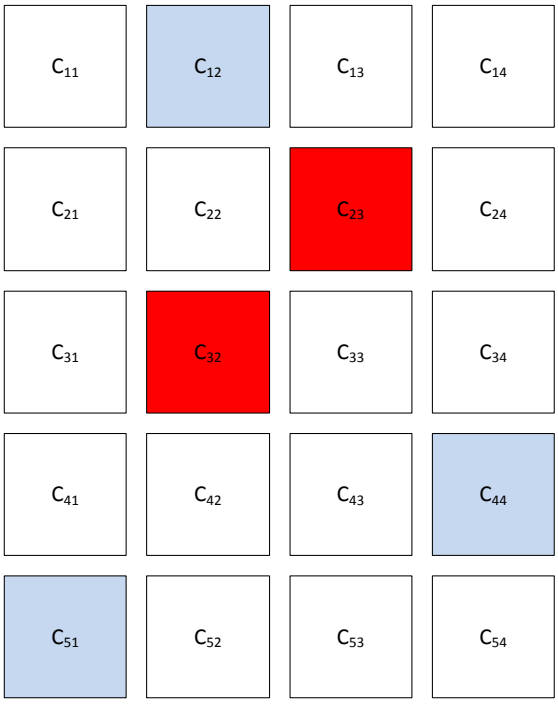


Figura 3.3: Exemplo de escolhas do algoritmo de *backtracking*.

3.5.2 *Simulated Annealing*

Algoritmos como o de *backtracking* podem ser muito custosos (do ponto de vista computacional) quando o espaço de busca é muito grande, como no caso do problema de monitoração não intrusiva de cargas, com isso o algoritmo de *simulated annealing* torna-se uma ótima opção pelo fato de ele conseguir ótimos resultados, com um baixo custo computacional. Uma descrição geral do algoritmo de *simulated annealing* pode ser vista em KELLY [13].

O algoritmo de *simulated annealing* designado para resolver o problema é descrito no algoritmo 2.

Algoritmo 2 *Simulated annealing* para monitoração não intrusiva de cargas

Definir $t0$, $tmin$, $coeficiente\_resfriamento$

S = solução inicial aleatória

$T = t0$ (temperatura inicial)

$melhor\_S = S$

enquanto $T > tmin$ **faça**

escolher um vizinho aleatório S' de S

se $custo(S) < custo(S')$ **ou** $e^{\frac{custo(S)-custo(S')}{T}} > 2 *$ (número aleatório entre 0 e 1) **então**

$S = S'$

se $custo(S') < melhor\_S$ **então**

$melhor\_S = S'$

fim se

fim se

$T = T * coeficiente\_resfriamento$

fim enquanto

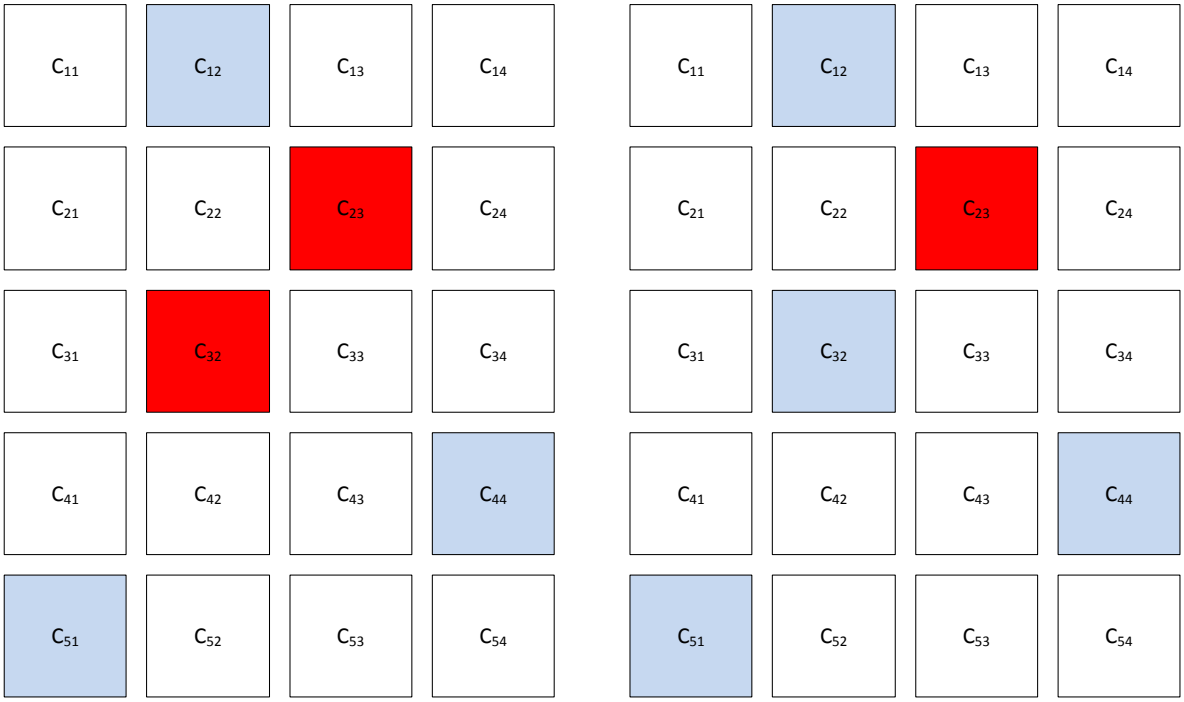
Para o nosso problema específico a solução aleatória inicial deve ser escolhida entre as possíveis soluções válidas, assim como já visto na figura 3.3. A função custo é a dada pela equação 3.15. As variáveis $t0$, $tmin$, $coeficiente\_resfriamento$ são escolhidas empiricamente, buscando o melhor resultado possível(em relação a função

custo), no menor tempo.

Outra parte importante do algoritmo não especificada claramente no algoritmo 2 é como é feita a escolha de um vizinho de S. Para criarmos uma solução vizinha a S, fazemos aleatoriamente a escolha entre uma das duas possibilidades de vizinhança a seguir:

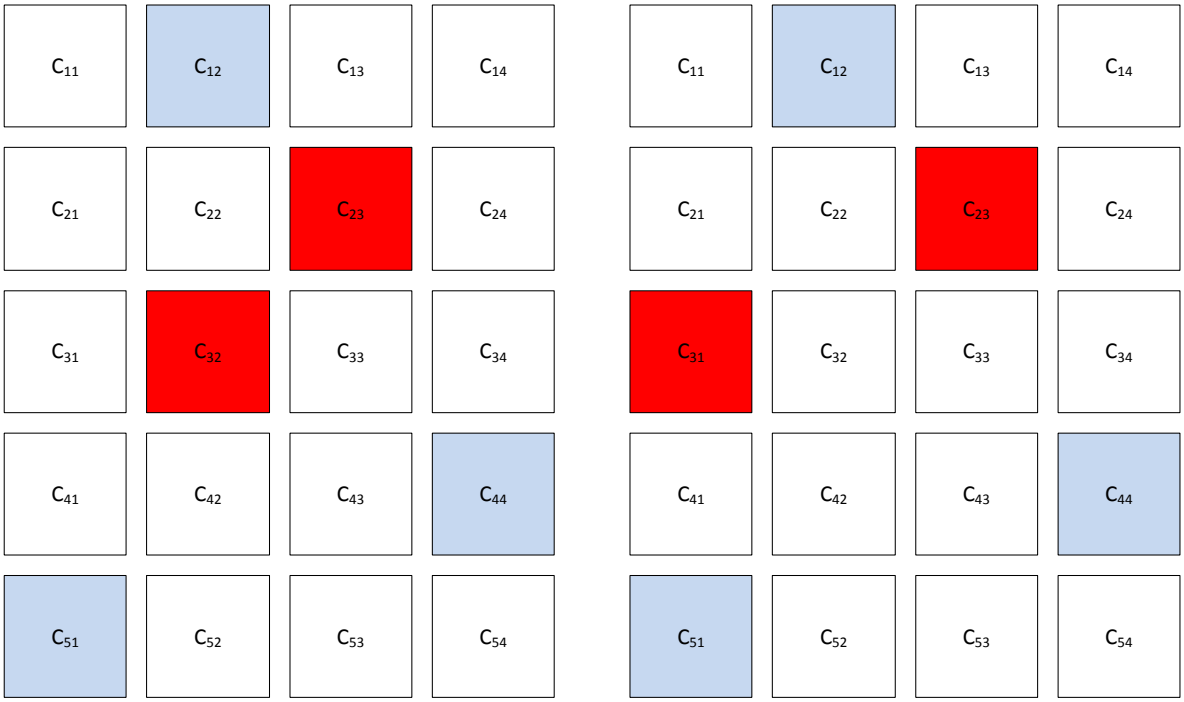
1. Escolhemos um equipamento aleatoriamente e mantendo o mesmo modo de operação, caso este equipamento esteja associado a fase1 ele agora é associado a fase2, caso ele esteja associado a fase2 ele passa a ser associado a fase1. Como pode ser visto na figura 3.4, foi escolhida de forma aleatória a linha 3 (equipamento 3) e mudou-se fase associada com aquele equipamento.

2. Escolhemos um equipamento aleatoriamente e agora de forma também aleatória selecionamos um novo modo de operação para o mesmo, no entanto mantemos a fase originalmente associada ao equipamento. Uma ilustração pode ser vista na figura 3.5, foi escolhida de forma aleatória a linha 3 (equipamento 3) e mudou-se o modo de operação daquele equipamento (do modo 2 para o modo 1), mantendo a fase associada ao equipamento.



(a) Solução atual S (b) Solução vizinha S'

Figura 3.4: Vizinhança do tipo 1 no *simulated annealing*



(a) Solução atual S (b) Solução vizinha S'

Figura 3.5: Vizinhança do tipo 2 no *simulated annealing*

3.6 Métrica de avaliação do algoritmo de monitoração

Muitas métricas distintas são usadas para avaliar os métodos de monitoração de cargas, isso acaba vindo a dificultar a comparação entre os diferentes métodos e algoritmos de monitoração de cargas. Num momento inicial onde o foco era apenas cargas com 2 estados (*on/off*), a métrica utilizada era saber se em um ponto de mudança de consumo a carga correta era associada com aquela mudança.

Uma métrica bastante utilizada é a que olha para o percentual de energia corretamente classificado. Essa métrica é muito mais geral em função de poder ser utilizada por vários tipos de métodos de monitoração. Outra questão é o fato de ela dar grande peso para cargas com grande consumo, o que em geral é o desejado, visto que nos estamos interessados no consumo total. Entretanto, algumas vezes as cargas a qual estão no controle do usuário são cargas de baixo consumo e assim mesmo com uma classificação errônea das mesmas, a métrica indicara uma boa performance total.

Temos então que a métrica de energia total classificada corretamente é descrita matematicamente por:

$$Acc = 1 - \frac{\sum_{t=1}^{T}\sum_{i=1}^{n}|\hat{y}_t^{(i)} - y_t^{(i)}|}{2\sum_{t=1}^{T}\overline{y_t}}, \tag{3.16}$$

onde $y_t^{(i)}$ e $\hat{y}_t^{(i)}$ indicam respectivamente o consumo e o consumo estimado do i-ésimo equipamento no t-ésimo instante de tempo e $\overline{y}_t = \sum_{i=1}^{n} y_t^{(i)}$ é o consumo total no t-ésimo instante de tempo.

Capítulo 4

Simulações

4.1 Introdução

Os resultados das simulações apresentados foram divididos em três partes:

1. Clusterização dos circuitos
2. *Backtracking*
3. *Simulated Annealing*

A seção 4.2 mostra os resultados usando os algoritmos de clusterização de programação dinâmica e o *gap statistic* para determinar o número de *clusters* , para assim conseguirmos identificar os modos de operação de cada circuito.

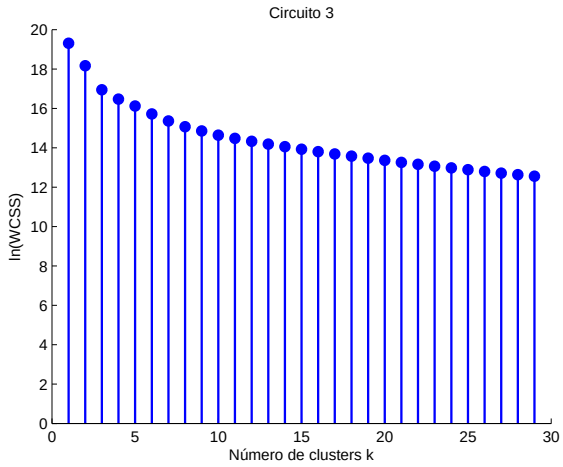
A seção 4.3.1 mostra a tentativa de predição do consumo de cada circuito e seus respectivos resultados, usando para isso um método de forca bruta.

A seção 4.3.2 mostra a tentativa de predição do consumo de cada circuito e seus respectivos resultados, usando para isso o método de *simulated annealing* , para assim diminuir o custo computacional.

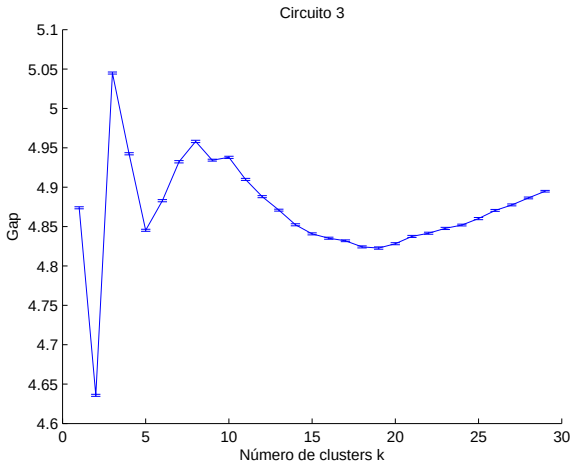
Todos os programas utilizados para simulação foram implementados usando a linguagem de programação C++, utilizando para isso um computador Intel Core i5-3337U CPU, com 8 GB de RAM DDR3 e usando o sistema operacional Windows 8 de 64 bits.

4.2 Clusterização de circuitos

Para descobrirmos o número de *clusters* de cada circuito é importante observar a variação do WCSS com k (o número de *clusters*) , a seguir mostramos os gráficos do WCSS e do *gap* com a variação de k, para cada circuito:

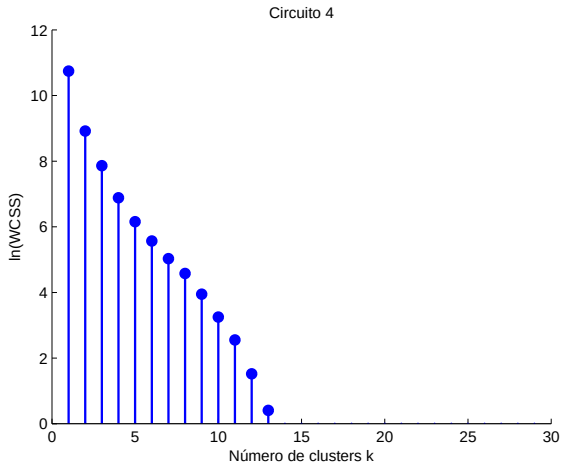


(a) Logaritmo do WCSS

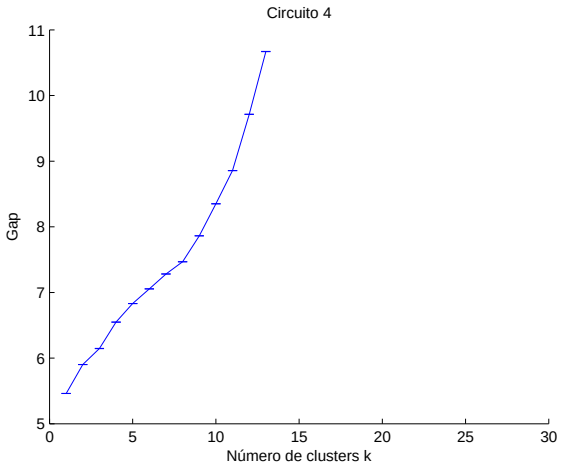


(b) Gap

Figura 4.1: Variação de k - circuito 3

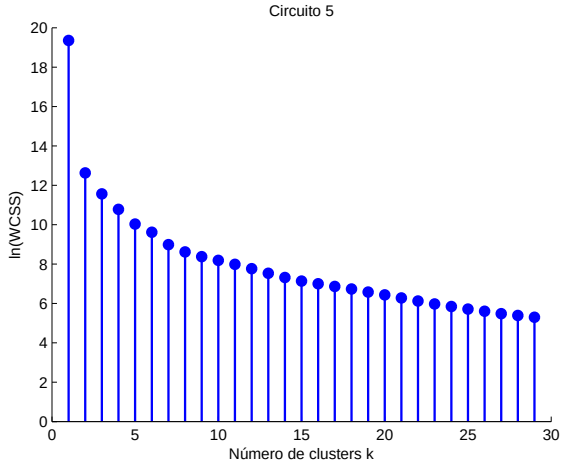


(a) Logaritmo do WCSS

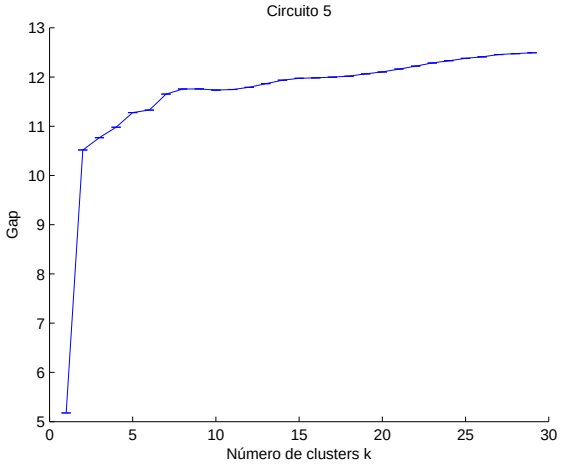


(b) Gap

Figura 4.2: Variação de k - circuito 4

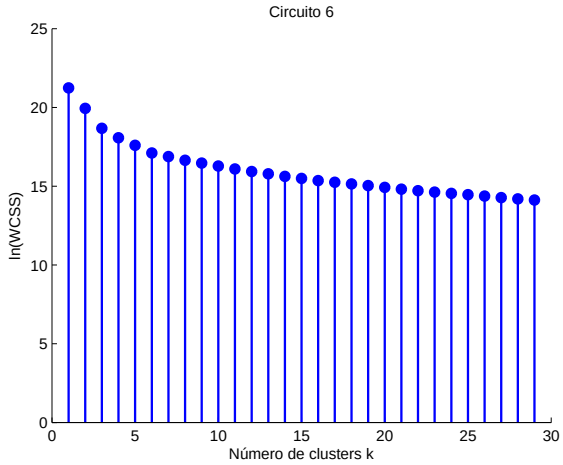


(a) Logaritmo do WCSS

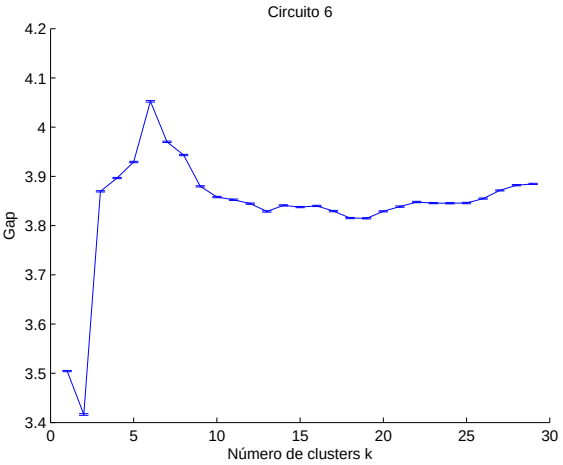


(b) Gap

Figura 4.3: Variação de k - circuito 5

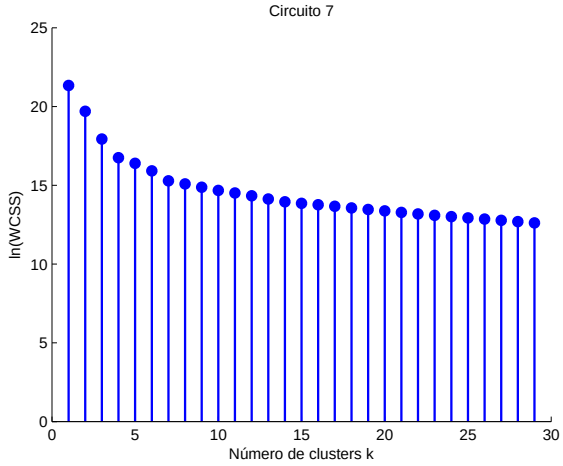


(a) Logaritmo do WCSS

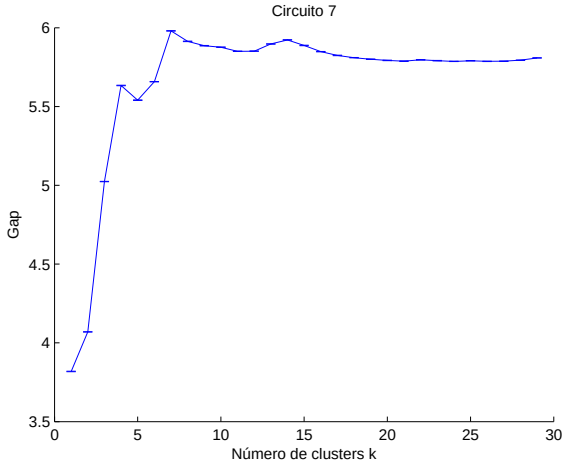


(b) Gap

Figura 4.4: Variação de k - circuito 6

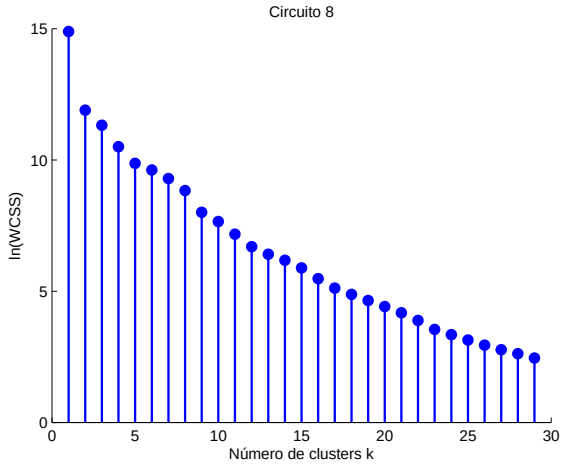


(a) Logaritmo do WCSS

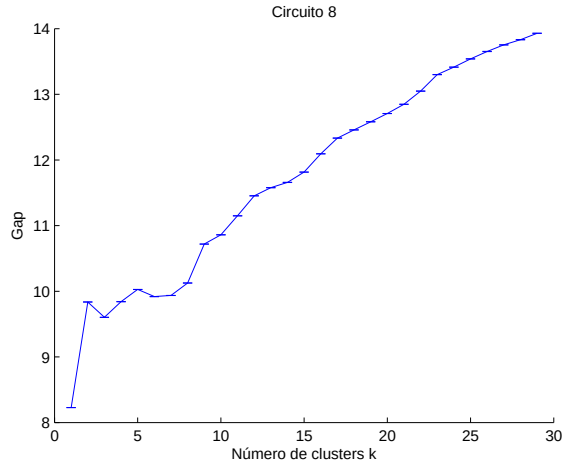


(b) Gap

Figura 4.5: Variação de k - circuito 7

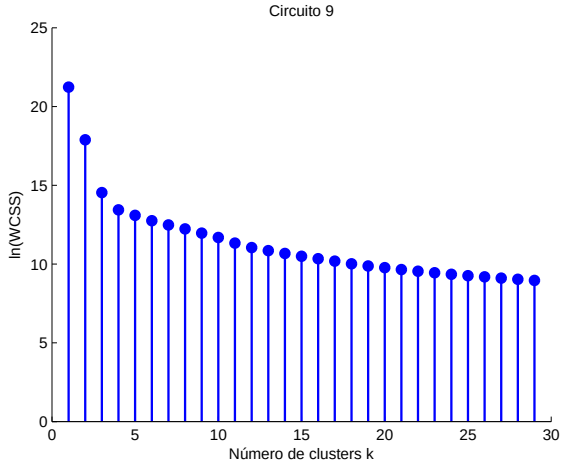


(a) Logaritmo do WCSS

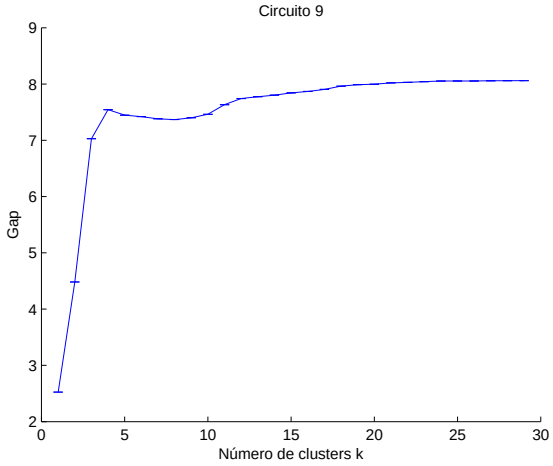


(b) Gap

Figura 4.6: Variação de k - circuito 8

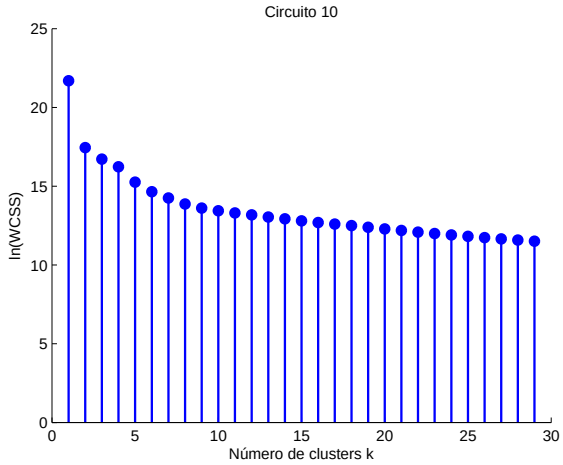


(a) Logaritmo do WCSS

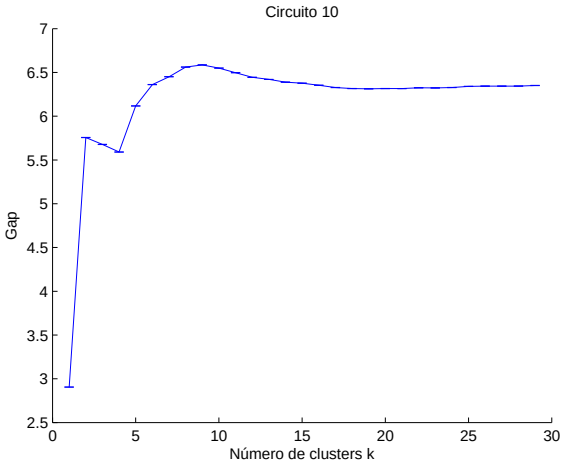


(b) Gap

Figura 4.7: Variação de k - circuito 9

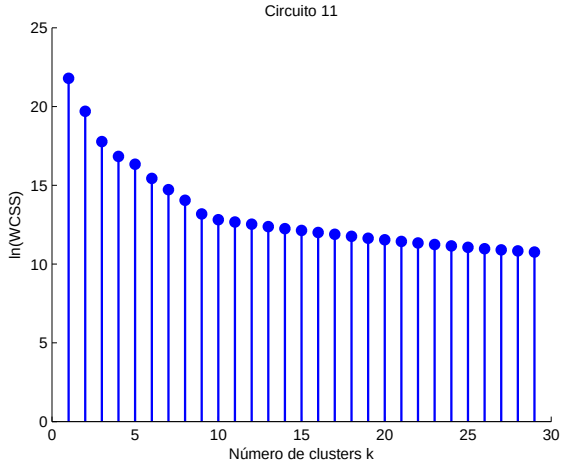


(a) Logaritmo do WCSS

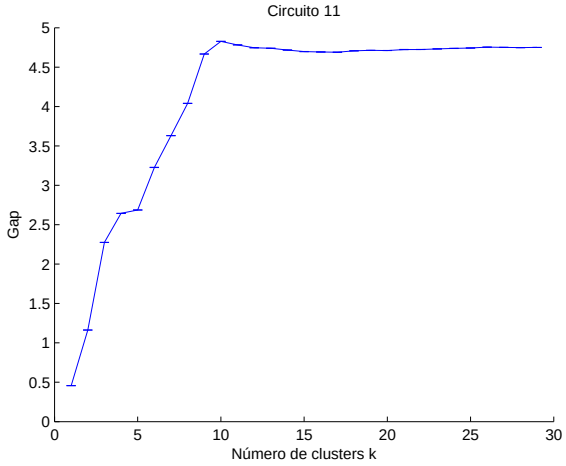


(b) Gap

Figura 4.8: Variação de k - circuito 10

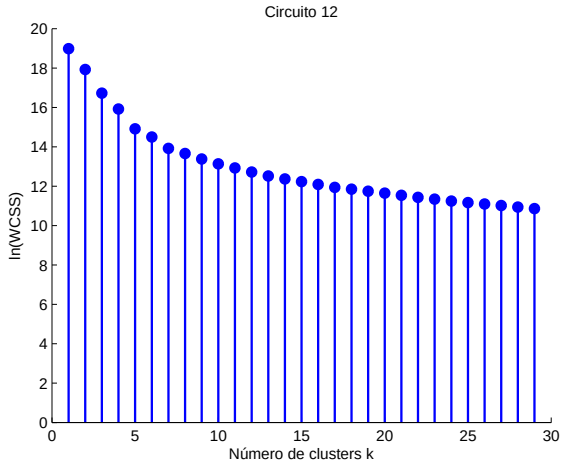


(a) Logaritmo do WCSS

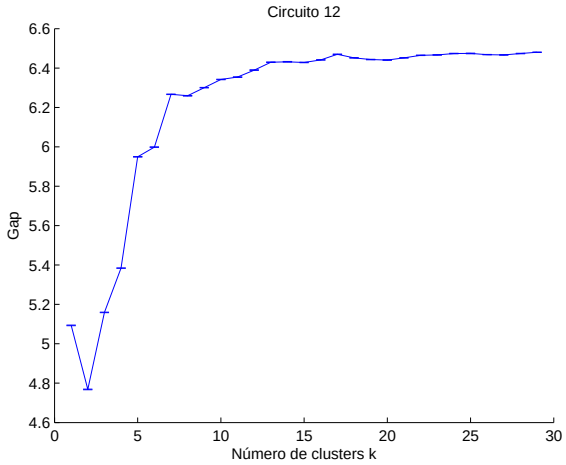


(b) Gap

Figura 4.9: Variação de k - circuito 11

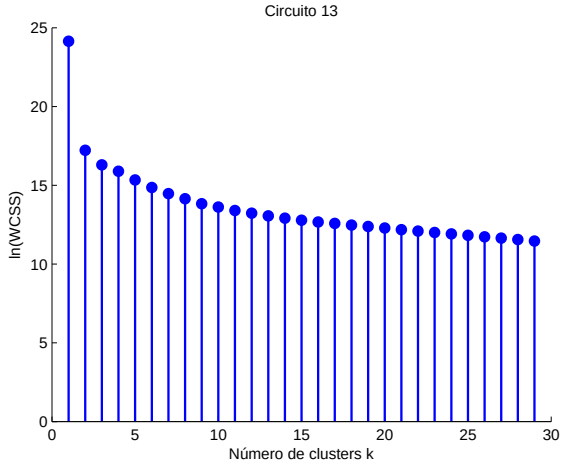


(a) Logaritmo do WCSS

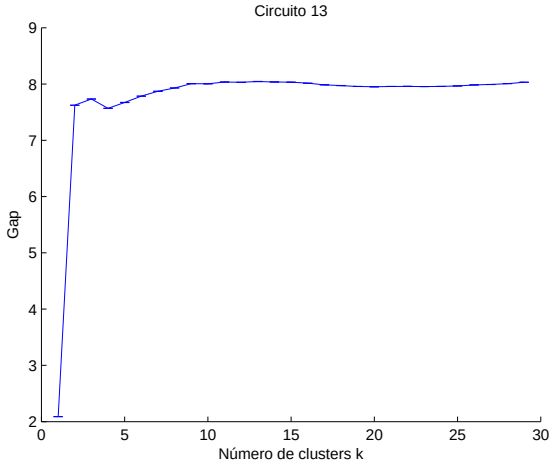


(b) Gap

Figura 4.10: Variação de k - circuito 12

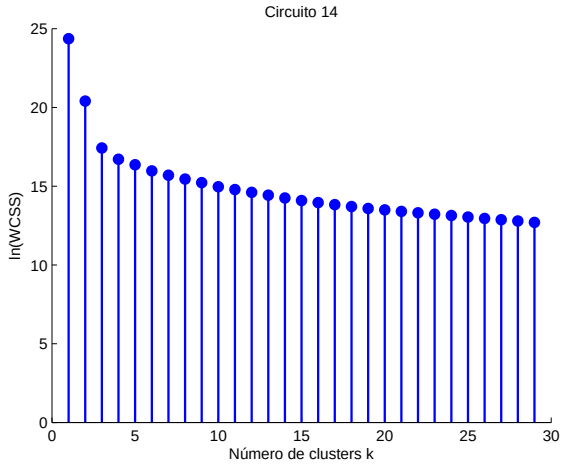


(a) Logaritmo do WCSS

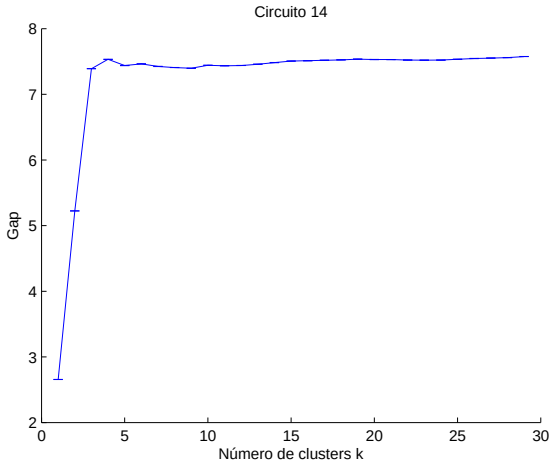


(b) Gap

Figura 4.11: Variação de k - circuito 13

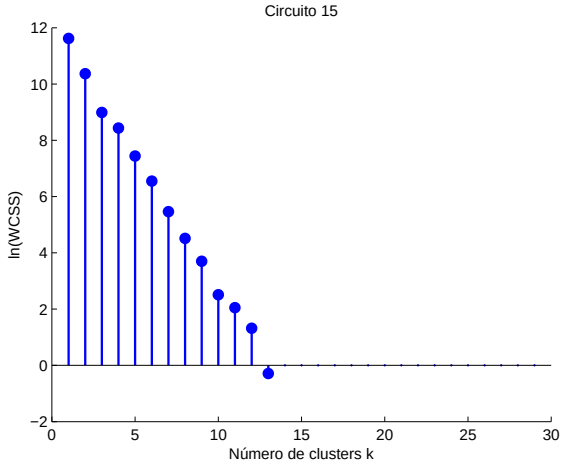


(a) Logaritmo do WCSS

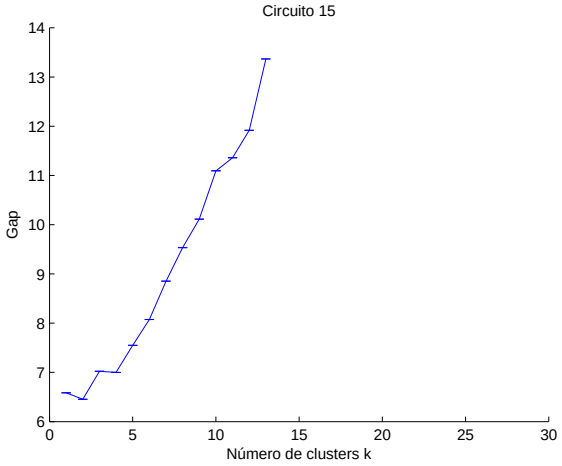


(b) Gap

Figura 4.12: Variação de k - circuito 14

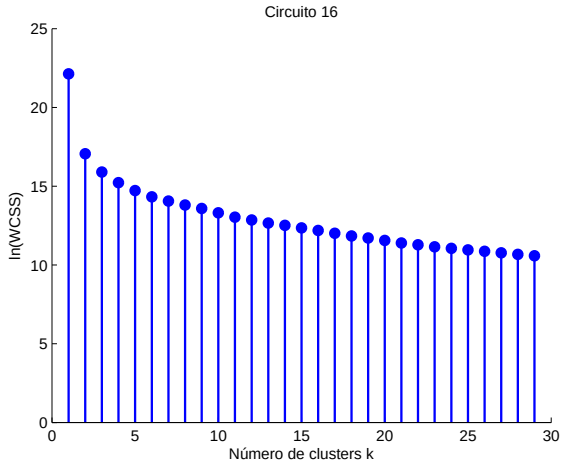


(a) Logaritmo do WCSS

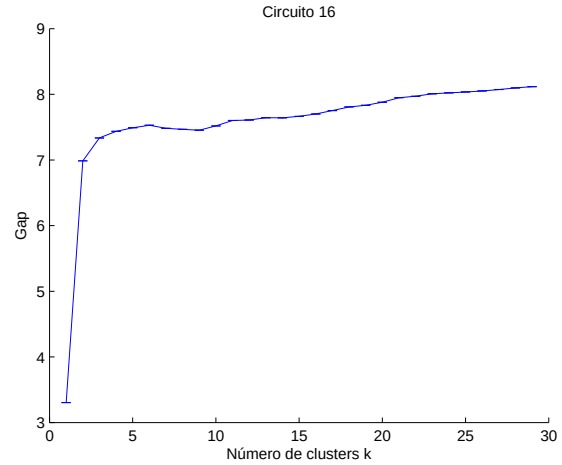


(b) Gap

Figura 4.13: Variação de k - circuito 15

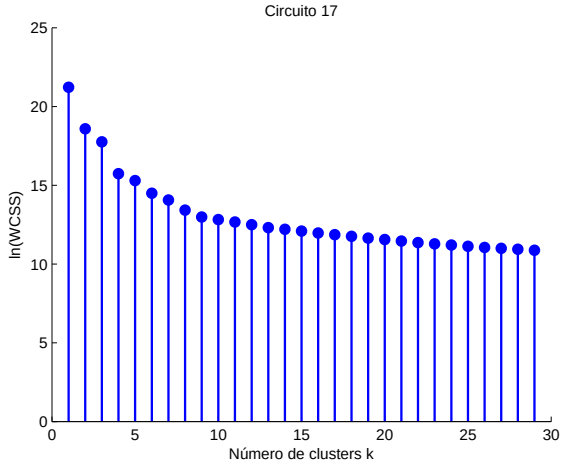


(a) Logaritmo do WCSS

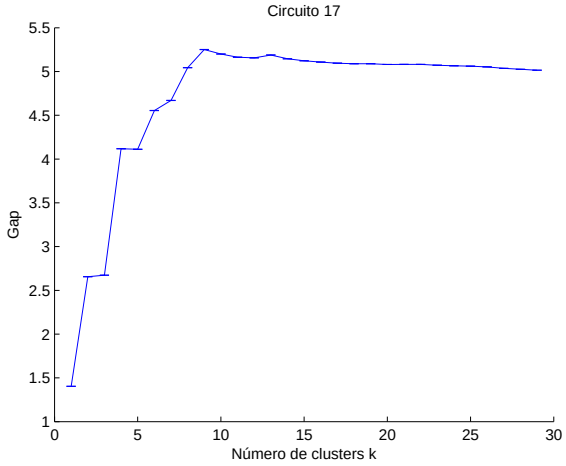


(b) Gap

Figura 4.14: Variação de k - circuito 16

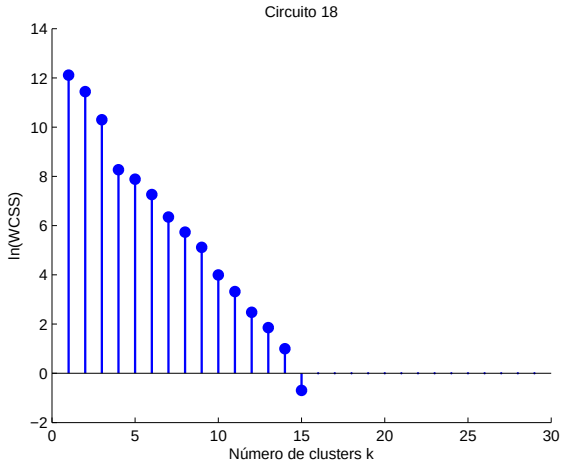


(a) Logaritmo do WCSS

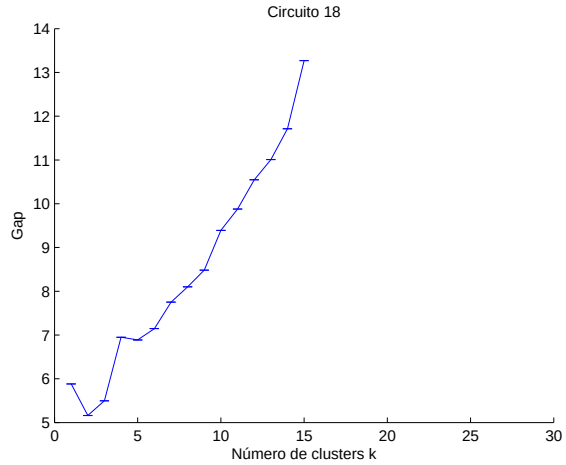


(b) Gap

Figura 4.15: Variação de k - circuito 17

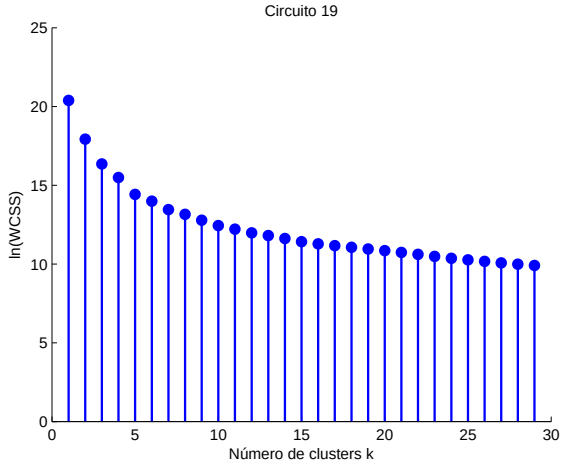


(a) Logaritmo do WCSS

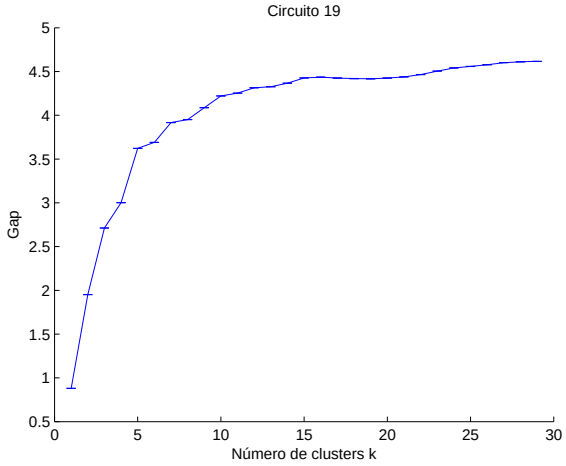


(b) Gap

Figura 4.16: Variação de k - circuito 18

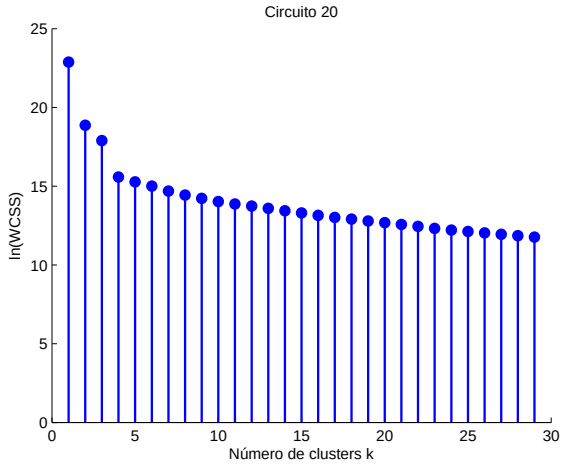


(a) Logaritmo do WCSS

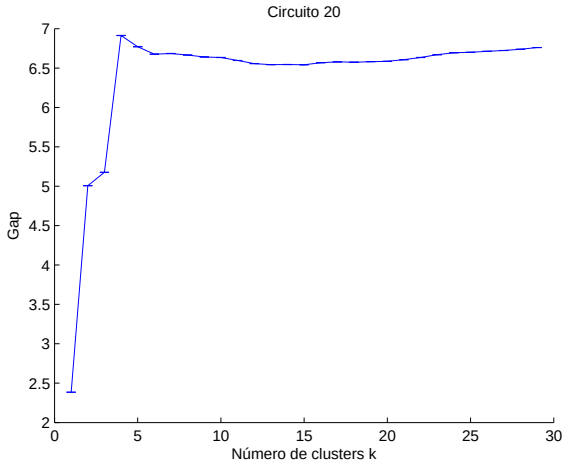


(b) Gap

Figura 4.17: Variação de k - circuito 19

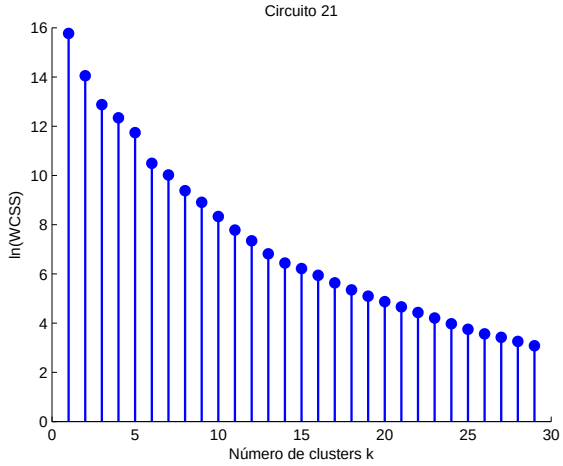


(a) Logaritmo do WCSS

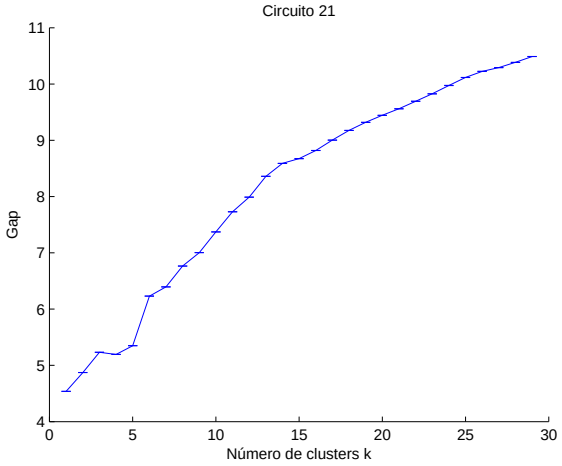


(b) Gap

Figura 4.18: Variação de k - circuito 20

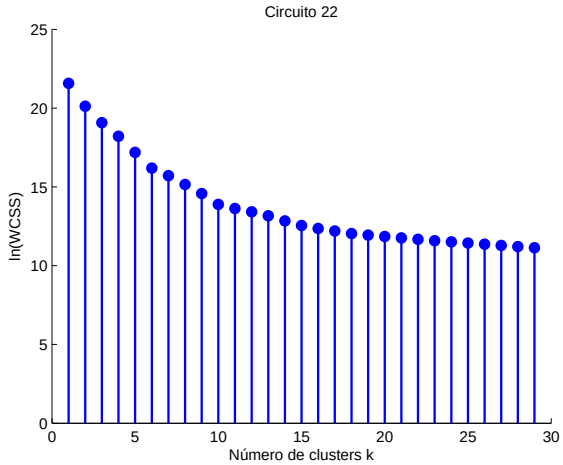


(a) Logaritmo do WCSS

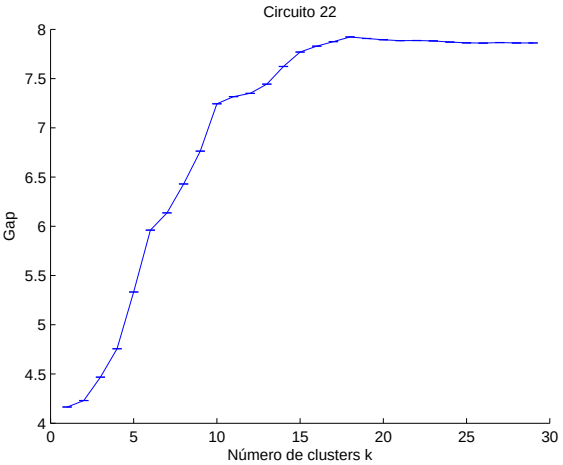


(b) Gap

Figura 4.19: Variação de k - circuito 21



(a) Logaritmo do WCSS



(b) Gap

Figura 4.20: Variação de k - circuito 22

Com os dados obtidos, determinamos assim o número de *clusters* para cada circuito, usando para isso a equação 3.14. Além disso foi utilizado que cada circuito trabalha em pelo menos 2 estados. O número de *clusters* estimado($\hat{k}$) e os modos de operação obtidos usando o algoritmo de programação dinâmica são mostrados na tabela 4.1.

Tabela 4.1: Número de *clusters* estimado e modos de operação dos circuitos

Circuito	$\hat{k}$	Estados de consumo de energia(W)
3	3	0.6440 147.1588 497.9250
4	14	0.0000 1.0000 2.0000 3.0000 4.0000 5.0000 6.0000 7.0000 8.0000 9.0000 10.0000 11.0000 12.0000 18.0000
5	9	0.0000 1.0000 2.0393 941.2727 950.1569 958.5287 969.1129 981.7500 1142.8333
6	6	76.7238 96.1048 132.3276 170.6402 228.1307 1215.7925
7	4	0.4239 119.3959 403.0436 1228.6719
8	2	0.0665 355.9091
9	4	0.8609 134.0514 210.1351 737.1860
10	2	5.1764 679.1843
11	10	0.0368 21.7347 50.3832 74.4987 154.2941 177.2514 181.4674 204.6241 335.8733 357.6382
12	7	0.1943 11.7631 22.7937 44.3655 67.1584 115.9878 910.4000
13	3	0.0261 2261.2760 2470.1389
14	4	0.0302 263.3409 2504.6582 2717.9425
15	3	0.0000 1.0000 2.7454
16	6	2.0917 145.4458 392.7368 919.2500 1676.9625 1746.0921
17	4	0.8808 55.6990 133.3647 191.3864
18	4	0.0000 1.0000 2.0515 8.8914
19	16	0.0000 1.0001 22.7983 24.3244 26.3683 34.2143 46.2844 48.8372 56.8851 149.6452 172.7594 180.5279 184.4134 197.7439 203.5276 207.5593
20	4	0.2487 891.4648 1283.1821 1609.5855
21	3	0.0002 4.2157 11.5977
22	18	0.0197 2.5000 6.7850 41.9976 127.2290 210.9360 322.3864 366.2122 406.5255 444.7358 895.7075 912.6859 948.7736 1205.7677 1226.8784 1246.6321 1523.2500 2102.1154

4.3 Monitoração não intrusiva de cargas

4.3.1 *Backtracking*

Nessa seção serão apresentados os resultados do algoritmo de monitoração não intrusiva de cargas usando a técnica de *backtracking*. Em função de um *backtracking* completo ser extremamente custoso, devido ao tamanho de espaço de busca do referido problema, limitaremos o tempo de busca e compararemos assim os diferentes resultados para diferentes tempos de busca. Avaliaremos assim como a escolha da nossa função de minimização da equação 3.15 influencia na acurácia do algoritmo.

Tabela 4.2: Algoritmo de *backtracking* para dados agregados

tempo(s)	erro(W^2)	erro médio(W^2/amostra)	acurácia(%)
9.496	187023005616.473360	462805.656958	27.3025
16.832	69118903256.411285	171041.093711	29.0316
98.034	45846552202.053513	113451.517054	29.9665
747.134	33070661681.637444	81836.3989776	30.5112
6921.652	30528226470.904293	75544.9088259	32.2956

Vemos na tabela 4.2 os resultados para diferentes tempos de execução, os quais se mostra muito superior ao *Simple Mean*(18.8% de acurácia) e equivalente a um FHMM [14] (33.3% de acurácia), como mostrados em [5]. Temos também que o tempo total de execução e erro total, são calculados acumuladamente em torno das 404107 amostras, assim temos na tabela o erro médio por execução do algoritmo (a cada 3 ou 4 segundos). Vemos também que pelo tempo de execução para 404107 amostras, o algoritmo é mais do que suficiente para monitoramento em tempo real.

Os dados nas fases do *REDD* possuem consumo de fontes desconhecidas, as quais não são provenientes dos circuitos o qual o *dataset* fornece. Visto que o treinamento e o algoritmo de monitoração são feitos para se ajustar com o consumo dos circuitos, desconsiderando assim outros consumos de fontes desconhecidas, o algoritmo pode ser bastante afetado tentando minimizar a equação 3.15, quando na verdade esse

não é o ideal. Como nós temos os gabaritos de cada um dos circuitos, podemos agregar esses dados para que assim criemos dados agregados em uma única fase, sem consumo de fontes desconhecidas. A tabela 4.3, mostra os resultados de se aplicar o algoritmo de *backtracking* nesses novos dados.

Tabela 4.3: Algoritmo de *backtracking* para dados agregados desconsiderando consumo de fontes desconhecidas

tempo(s)	erro(W^2)	erro médio(W^2/amostra)	acurácia(%)
6.771	32393676236.238632	80161.1361254	27.5692
12.550	15161681635.925171	37518.9779834	29.4439
58.151	11560279630.700285	28606.976941	31.1328
425.603	6965824519.802353	17237.5745033	35.0546
3846.629	152517366.746968	377.418274732	33.7595
17399.582	1285766.376961	3.18174735147	42.3579

Como já era esperado, obtivemos um resultado superior de acerto quando não há consumo de fontes desconhecidas.

4.3.2 *Simulated Annealing*

Nessa seção serão apresentados os resultados do algoritmo de monitoração não intrusiva de cargas usando a técnica de *simulated annealing*. É esperado que o *simulated annealing* obtenha melhores resultados do que o algoritmo de *backtracking*. Vemos os resultados das simulações na tabela 4.4.

Tabela 4.4: Algoritmo de *simulated annealing* para dados agregados

tempo(s)	erro(W^2)	erro médio(W^2/amostra)	acurácia(%)
220.260	691029609262.160890	1710016.42947	17.0973
837.878	17018756690.040476	42114.4812885	22.9594
1099.885	7378500787.029205	18258.7799445	27.9807
1767.965	9182050563.696262	22721.8300195	26.6925
2055.885	5589053135.043295	13830.6268762	28.0281
9593.486	4675704325.832830	11570.4611052	28.7298
10922.792	6391327933.730214	15815.9297754	28.4981

O resultado foi pior do que o algoritmo de *backtracking* em termos de acurácia, mesmo o erro tendo sido menor. Isso se deve ao fato de tentar ajustar os modos de operação dos equipamentos de forma a minimizar o erro, quando esse não é o caso, em função de fontes de consumo desconhecidas, o que acaba por prejudicar a acurácia.

Assim com feito na seção 4.3.1, faremos simulações quando não há consumo de fontes desconhecidas. Os resultados podem ser vistos na tabela 4.5.

Tabela 4.5: Algoritmo de *simulated annealing* para dados agregados desconsiderando consumo de fontes desconhecidas

tempo(s)	erro(W^2)	erro médio(W^2/amostra)	acurácia(%)
129.807	29319903600.550446	72554.802566	32.9543
136.067	24084248362.233288	59598.6913422	35.4070
343.535	1137417611.431066	2814.64466449	38.1107
384.159	275298567.770010	681.251667925	39.1537
503.613	13285657.631648	32.8765837554	40.9295
1441.234	1792923.034657	4.43675322293	41.9969
2389.980	1344316.861800	3.32663592019	41.7491
4235.876	1288075.194419	3.18746073297	41.8264
9953.806	1284809.935924	3.17938055001	42.4751

Dessa vez o algoritmo de *simulated annealing* obteve resultados extremamente superiores. Tanto na minimização da função custo(obtendo assim um baixo erro) , quanto uma maior acurácia, já que agora não há consumo de fontes desconhecidas.

Um ponto importante a ser observado é como todos os algoritmos a partir de um certo tempo de execução tendem a se estabilizar em termos de acurácia, mesmo com aumento significativo do tempo, a melhoria passa a não ser significativa e muitas vezes inexistente.

Capítulo 5

Conclusões

5.1 Conclusões Gerais

O objetivo deste trabalho foi realizar o projeto e estudo algoritmos de programação inteira no problema de monitoração não intrusiva de cargas, assim como a realização de uma técnica de treinamento não supervisionada.

O objetivo foi cumprido com êxito. Em um momento inicial foi feito o treinamento e aprendizado dos modos de operação dos circuitos de forma não-supervisionada, porém ainda intrusiva (no sentido de que cada circuito possui um medidor nesse momento inicial). Não foi possível a obtenção de uma métrica para avaliação da etapa de treinamento, em função de não termos o real modo de operação dos mesmos. Na parte final através dos circuitos já modelados, buscamos validar os algoritmos de programação inteira para monitoração. Vemos que o *simulated annealing* se mostrou superior ao *backtracking*, no sentido de minimizar a função custo, porém no caso onde há consumo de fontes desconhecidas isso acabou por prejudicar o acurácia do algoritmo. No entanto, no caso onde foi retirado esse consumo desconhecido, o algoritmo de *simulated annealing* foi extremamente superior. Sendo assim, vemos que as técnicas aqui empregadas se comparam e até superam algumas técnicas da literatura, ainda mais se considerando a granularidade da amostragem e termos apenas um recurso disponível(potência ativa), relação a outros trabalhos onde se usam uma quantidade muito maior de recursos.

5.2 Trabalhos Futuros

Como trabalho futuro, necessita-se o estudo de novas técnicas de estatística e aprendizado de máquina, para que assim possamos melhorar e estender as técnicas apresentadas neste trabalho. Conseguindo um nível de acurácia alto e usando equipamentos de medição o mais simples possíveis, nós poderemos implementar o sistema como uma solução real.

Referências Bibliográficas

[1] SUZUKI, K., INAGAKI, S., SUZUKI, T., et al. "Nonintrusive appliance load monitoring based on integer programming". In: *SICE Annual Conference, 2008*, pp. 2742–2747, Aug 2008. doi: 10.1109/SICE.2008.4655131.

[2] NEMHAUSER, G. L., WOLSEY, L. A. *Integer and combinatorial optimization*. New York, NY, USA, Wiley-Interscience, 1988. ISBN: 0-471-82819-X.

[3] CARRIE ARMEL, K., GUPTA, A., SHRIMALI, G., et al. "Is disaggregation the holy grail of energy efficiency? The case of electricity", *Energy Policy*, v. 52, n. C, pp. 213–234, 2013. Disponível em: <http://EconPapers.repec.org/RePEc:eee:enepol:v:52:y:2013:i:c:p:213-234>.

[4] HART, G. "Nonintrusive appliance load monitoring", *Proceedings of the IEEE*, v. 80, n. 12, pp. 1870–1891, Dec 1992. ISSN: 0018-9219. doi: 10.1109/5.192069.

[5] KOLTER, J. Z., JOHNSON, M. J. "REDD: A Public Data Set for Energy Disaggregation Research". In: *SustKDD Workshop on Data Mining Applications in Sustainability*, 2011.

[6] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al. *Introduction to Algorithms, Third Edition*. 3rd ed. , The MIT Press, 2009. ISBN: 0262033844, 9780262033848.

[7] WANG, H., SONG, M. "Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming", *The R Journal*, v. 3, n. 2, pp. 29–33, 2011. Disponível em: <http://journal.r-project.org/archive/2011-2/RJournal_2011-2_Wang+Song.pdf>.

[8] HIRSCHBERG, D. S., LARMORE, L. L. "The least weight subsequence problem", *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, v. 0, pp. 137–143, 1985. ISSN: 0272-5428. doi: http://doi.ieeecomputersociety.org/10.1109/SFCS.1985.60.

[9] EPPSTEIN, D., GALIL, Z., GIANCARLO, R. "Speeding up Dynamic Programming". In: *In Proc. 29th Symp. Foundations of Computer Science*, pp. 488–496, 1988.

[10] YAO, F. F. "Efficient Dynamic Programming Using Quadrangle Inequalities". In: *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pp. 429–435, New York, NY, USA, 1980. ACM. ISBN: 0-89791-017-6. doi: 10.1145/800141.804691. Disponível em: <http://doi.acm.org/10.1145/800141.804691>.

[11] TIBSHIRANI, R., WALTHER, G., HASTIE, T. "Estimating the number of clusters in a dataset via the Gap statistic", v. 63, pp. 411–423, 2000.

[12] KLEINBERG, J., TARDOS, E. *Algorithm Design*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321295358.

[13] KELLY, J. P. *Meta-Heuristics: Theory and Applications*. Norwell, MA, USA, Kluwer Academic Publishers, 1996. ISBN: 0792397002.

[14] GHAHRAMANI, Z., JORDAN, M. I. "Factorial Hidden Markov Models", *Mach. Learn.*, v. 29, n. 2-3, pp. 245–273, nov. 1997. ISSN: 0885-6125. doi: 10.1023/A:1007425814087. Disponível em: <http://dx.doi.org/10.1023/A:1007425814087>.