CENTRO UNIVERSITÁRIO DO SUL DE MINAS
ENGENHARIA ELÉTRICA
GABRIELLE MOREIRA LUIZ

O USO DA LÓGICA FUZZY NA AUTOMATIZAÇÃO DO CORTE DE CARGAS INTELIGENTE

Varginha
2016

GABRIELLE MOREIRA LUIZ

O USO DA LÓGICA FUZZY NA AUTOMATIZAÇÃO DO CORTE DE CARGAS INTELIGENTE

Trabalho de conclusão de curso apresentado ao curso de Engenharia Elétrica do Centro Universitário do Sul de Minas- UNIS como pré-requisito para obtenção do grau de bacharel sob orientação do Prof. Msc. Eduardo Henrique Ferroni.

Varginha
2016

GABRIELLE MOREIRA LUIZ

O USO DA LÓGICA FUZZY NA AUTOMATIZAÇÃO DO CORTE DE CARGAS INTELIGENTE

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharel em Engenharia Elétrica do Centro Universitário do Sul de Minas- UNIS pela banca examinadora composta pelos membros:

Aprovado em: / /

_____

Prof. Me. Eduardo Henrique Ferroni

_____

Prof.

_____

Prof.

OBS:

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da força de vontade e a oportunidade, a minha família que estão juntos comigo nessa batalha diária pelo apoio, aos amigos que conheci no decorrer do curso e ao meu orientador Eduardo Henrique Ferroni.

“Dificuldades preparam pessoas comuns para destinos extraordinários! ”

A Cabana

RESUMO

O presente trabalho tem como objetivo automatizar totalmente um sistema de corte inteligente de cargas utilizando Lógica *Fuzzy*. Baseado nas duas variáveis fundamentais do processo produtivo em questão: pressão da tubulação de amônia e disponibilidade de energia, é possível criar meios para otimização do consumo de energia e realizar o ajuste do indivíduo gerado pelo algoritmo genético quando necessário. O *software Matlab* é utilizado, aplicando os conceitos da programação orientada a objetos para desenvolver esse sistema. Um processo industrial é dinâmico, e as escolhas feitas no corte inicial de cargas, com o tempo podem não ser a melhor opção. Por esse motivo, há necessidade que à Lógica Fuzzy ajuste as alterações efetuadas pelo operador, afim de gerenciar a energia da melhor maneira possível. O sistema se mostrou eficiente no complemento do corte inteligente de cargas, agindo de maneira rápida tanto no ajuste fino da potência, como na manutenção da automatização.

Palavras-chave: Corte inteligente de cargas. Lógica *fuzzy*. Automação. Programação orientada à objeto. Matlab.

ABSTRACT

This paper aims to fully automate an intelligent cutting system loads using Fuzzy Logic. Based on two fundamental variables of the production process in question: Pipe pressure of ammonia and energy availability, you can create ways to optimize energy consumption and make the adjustment of the individual generated by the genetic algorithm when needed. The Matlab software is used by applying the concepts of object-oriented programming to develop this system. An industrial process is dynamic, and the choices made in the initial cutting loads over time may not be the best option. For this reason, it is necessary that the Fuzzy Logic setting changes made by the operator in order to manage energy in the best possible way. The system was efficient in complementing the smart cut loads acting both quick fine adjustment of power and in the maintenance of automation.

Keywords: *Intelligent loads cut. Genetic Algorithm. Fuzzy logic. Supervisory system.*

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Produtividade, recursos naturais, competividade, lógica, otimização do tempo e racionamento de energia. Nunca o tema automação ocupara tanto espaço na mídia e nas discussões em todos os lugares – das universidades às ONGs, dos ambientes de trabalho às escolas. A meta é reduzir os impactos negativos de toda e qualquer interferência na cadeia produtiva. Através da mudança na forma de diálogo entre os integrantes desse complexo sistema e inserção em massa dos recursos presentes na automação.

Diante desse quadro, a automação surge como uma ferramenta que direcionará os esforços rumo às metas. Metas de redução de perdas, na essência da eficiência. É um meio de aflorar o que cada projeto resulte no rendimento máximo, se transformado em resultados corporativos (RIBEIRO NETO, 2011).

Inúmeras são as vantagens agregadas pela integração dos sistemas automatizados nos diversos processos produtivos, destacando-se a economia de energia, já que a grande maioria dos sistemas são de baixa potência, e a conformidade com as normas regulamentadoras, resulta na redução de acidentes do trabalho, devido a adoção das regras previstas na NR12, que aborda temas relacionados à Segurança no Trabalho em Máquinas e Equipamentos, reduzindo o afastamento do trabalho, sendo economicamente viável para a indústria (JOB, 2003).

A dependência da energia elétrica pela indústria é inquestionável, e as falhas no fornecimento pelas concessionárias causam enormes prejuízos. Diante da busca pela maximização dos lucros, muitas organizações estão partindo para a instalação de usinas de cogeração nos parques industriais, que atuam paralelamente com o sistema elétrico de potência das distribuidoras, resultando em um menor impacto nos casos de interrupções de origem acidental e/ou programadas, e como suporte no horário de pico. Dependendo do processo ou matéria prima envolvida, um simples pique de energia pode causar grandes perdas ou paradas nas linhas de produção, o que justifica a implantação de sistemas inteligentes de transferência automática de circuitos para o suprimento seletivo das cargas, (VANDERLEI, 2003).

Em alguns casos, essas usinas de cogeração não suprem a demanda de energia total da indústria. Com isso, é necessário cortar as cargas não prioritárias em caso de interrupção do fornecimento de energia. Para definir quais cargas devem permanecer ligadas durante todo o processo é utilizado um sistema supervisório. A eliminação das cargas menos prioritárias e o comportamento elétrico das mesmas mantêm a estabilidade do processo (IEEE, 2007).

Para a continuidade das linhas de produção, as empresas desprendem investimentos para evitar paradas não programadas, investimentos com retorno garantido, frente aos desgastes

oriundos de interrupções não previstas. O corte inteligente e seletivo de cargas evita a paralisação, mesmo que momentânea do processo industrial.

Ferroni (2013) utilizou algoritmo genético para criar um sistema supervisório capaz de realizar o corte inteligente de cargas de forma automática após uma interrupção no fornecimento de energia ou horário de pico de consumo de energia elétrica. Porém, após esse corte inicial de cargas ainda é necessário que um operador modifique as cargas de acordo com a dinâmica do processo produtivo.

Os valores de pressão e temperatura usados para definir a melhor combinação de cargas sofrem alterações conforme a dinâmica do processo industrial. O sistema de controle de cargas sofre algumas influências externas, como da temperatura e dos compressores de amônia. Os compressores de amônia de grande porte são acionados através de inversores de frequência. A potência de operação desses compressores pode ser menor que a nominal em dias frios, restando um saldo de energia. Essa sobra de energia ~~potência~~ pode ser utilizada para acionar mais equipamentos, de potências menores, alterando a combinação inicial de cargas proposta pelo AG (FERRONI, 2013).

O presente trabalho se propõe a automatizar totalmente o monitoramento do processo em questão para que não seja mais necessária a interferência do ser humano na tomada de decisões do sistema supervisório. Utilizando a teoria da Lógica *Fuzzy* e baseado nas duas variáveis fundamentais deste processo produtivo: pressão da tubulação de amônia e sobra de energia é proposto criar meios para gerenciar o consumo de energia e ajustar o indivíduo gerado pelo algoritmo genético conforme necessário.

Utilizando o *software* Matlab 2016 e os conceitos da Programação Orientada a Objetos, foi desenvolvido um sistema, baseado em Lógica *Fuzzy*, que faz o ajuste fino de potência após o corte inicial de cargas realizado pelo Algoritmo Genético. Esse ajuste é necessário devido à alguns compressores operarem com inversores de frequência, o que gera uma sobra de carga que precisa ser aproveitada no processo.

Após esse procedimento, com a interrupção do fornecimento de energia por um período prolongado, pode ser necessário que o operador altere os forçadores ligados ou desligados. A partir dessa modificação, podem ocorrer três situações: o extrapolamento dos 1800kW, o nível de carga abaixo do ideal, assim como o sistema pode continuar com valores abaixo de 1800kW.

A Lógica *Fuzzy*, levando em consideração os níveis ideais de pressão e carga, realiza os ajustes necessários para que o sistema mantenha o equilíbrio, mesmo após as alterações do operador. O sistema se mostra eficiente devido ao tempo necessário para o ajuste, que é irrelevante comparado ao tempo de atuação do sistema de proteção da indústria.

O Capítulo 2 apresenta as principais definições necessárias ao entendimento desse trabalho: Algoritmo Genético, sistema supervisório e Programação Orientada a Objetos. O Capítulo 3 explica o funcionamento da planta industrial em questão. O Capítulo 4 define Lógica *Fuzzy* e seus principais conceitos e aplicação. Já o Capítulo 5 apresenta os resultados desse sistema. E o Capítulo 6, apresenta uma conclusão, justificando a importância da automatização total para o sistema industrial estudado.

2 COGERAÇÃO, SISTEMAS SUPERVISÓRIOS E ALGORITMO GENÉTICO

A interrupção no fornecimento de energia, para muitas indústrias, pode causar um grande prejuízo. Para combater esse problema, muitas unidades industriais estão utilizando sistemas de geração própria em momentos de eventual suspensão no fornecimento de energia, ou no horário de pico de consumo de energia elétrica. A geração própria é viável financeiramente, pois mantém o sistema elétrico e também a produção industrial.

O corte inteligente cargas consiste em eliminar cargas não prioritárias ao processo industrial com o intuito de restabelecer o balanço de potência entre a geração e o consumo de energia, após a ocorrência de alguma anomalia ou realizar o controle de demanda de forma que a energia ativa de uma unidade consumidora fique dentro de valores limites pré- estabelecidos (SANTOS, 2009).

Os sistemas de geração próprios costumam ser limitados. Por essa razão é necessário que no horário de pico ou na interrupção do fornecimento de energia sejam adotados planos de corte de carga. As cargas menos prioritárias são eliminadas, e o comportamento elétrico das mesmas mantém a estabilidade do processo (IEEE, 2007).

Os consumidores autoprodutores possuem três classificações (CPFL, 2005):

a) Autoprodutores com venda de excedente: são os consumidores que possuem geração própria, a qual pode operar em paralelo com o sistema da concessionária de energia e vendem o excedente de sua geração para a distribuidora usando a rede de distribuição;
b) Autoprodutores sem venda de excedente: são consumidores com geração própria que pode operar em paralelo com o sistema elétrico e não possuem excedente para venda;
c) Autoprodutores com paralelismo momentâneo: são consumidores que possuem geração própria que opera em paralelo com o sistema somente por um curto período de tempo e que alimentam a carga própria. Este paralelismo dura alguns segundos e é realizado pelo relé de transferência. Muitas concessionárias de energia exigem um relé de proteção coordenado pela função 31 (função de potência reversa), que não permite o paralelismo por uma potência superior a 5% da potência máxima gerada, isto é, no momento que o sistema de geração própria alcança 5% do valor nominal de geração, o relé de proteção desconecta a rede elétrica da distribuidora (Ilhamento).

O “Ilhamento” ocorre quando uma parte da rede de distribuição torna-se eletricamente isolada da fonte de energia principal (subestação), mas continua energizada por geradores distribuídos conectados no subsistema isolado (JENKINS *et al.*, 2002). Ao se projetar uma subestação de energia que contemplará a transferência de energia entre distribuidora e geração

própria é fundamental que sejam destacados dois pontos: Um sistema de proteção contra potência reversa; e o correto dimensionamento do sistema de cogeração para que o mesmo tenha a capacidade de sustentar as cargas da indústria (VASCONCELOS, 2006).

As concessionárias de distribuição determinam normas para os autoprodutores, para que o nível de tensão se mantenha o mais constante possível. O corte inicial de cargas pode não ser adequado para determinado instante, devido a dinâmica do processo produtivo. Por isso é necessário um sistema supervisório capaz de adequar as cargas conforme necessário.

A empresa deve adotar o corte de cargas e consequente redução da produção nos horários de ponta, para não levar o sistema elétrico da indústria ao colapso. O corte de carga deve acontecer de maneira automática e rápida e isso só é possível com o uso de um sistema de controle confiável, sendo a manutenção do sistema elétrico desta indústria o motivo fundamental do uso de um sistema de controle (FERRONI, 2013).

O corte de cargas precisa ser rápido, de modo a interferir o mínimo possível no sistema industrial. O processo decisório precisa ser eficiente e rápido. O algoritmo genético proposto por Ferroni (2013), fornece a melhor combinação de cargas que deve permanecer ligadas no momento em que o sistema de cogeração for acionado, programado ou não, de maneira automática. Uma rede de sensores fornece, em tempo real, as condições do sistema produtivo. Estas informações são avaliadas por um algoritmo genético que toma uma decisão da melhor combinação de cargas. Os resultados formam um banco de dados, que um sistema supervisório acessa num momento programado ou quando o sistema de energia solicita. As mudanças na produção passam por pequenos ajustes definidos pelo sistema supervisório, sendo necessário que um operador modifique a combinação de cargas inicialmente proposta pelo AG.

Nas indústrias, a gestão de energia deve ser vista como um fator de produção tão importante quanto o trabalho, o capital e as matérias-primas. O monitoramento das cargas elétricas, além de garantir uma melhor qualidade de energia, melhora o desempenho produtivo e consequente lucro da indústria (BROLIN, 2010).

2.1 Sistemas Supervisórios

Para conseguir uma indústria energeticamente eficiente e com uma boa qualidade de energia é necessário a implementação de técnicas computacionais que auxiliem na tomada de decisão (LEITE, 2010). Para que as informações de um processo produtivo sejam monitoradas e rastreadas são utilizados sistemas supervisórios. Conhecidos como SCADA (*Supervisory Control and Data Aquisition)* esses sistemas coletam dados através de equipamentos, esses

dados são manipulados, analisados, armazenados e então apresentados ao usuário (MAIA, 2007).

Atualmente, os sistemas de automação industrial utilizam tecnologias de computação e comunicação para automatizar o monitoramento e controle dos processos industriais, efetuando coleta de dados em ambientes complexos, eventualmente dispersos geograficamente, e a respectiva apresentação de modo amigável para o operador, com recursos gráficos elaborados (interfaces homem-máquina) e conteúdo multimídia.

Os sistemas SCADA identificam todas as variáveis envolvidas no processo, podendo executar funções computacionais (operações matemáticas, lógicas, com vetores ou *strings*, etc) ou representar pontos de entrada/saída de dados do processo que está sendo controlado. Neste caso, correspondem às variáveis do processo real (ex: temperatura, nível, vazão etc), se comportando como a ligação entre o controlador e o sistema.

A partir do momento em que o monitoramento e o controle de um processo são feitos com a ajuda de um sistema supervisório, o processamento das variáveis de campo é mais rápido e eficiente. Qualquer evento imprevisto no processo é rapidamente detectado e mudanças nos set-points são imediatamente providenciadas pelo sistema supervisório, no sentido de normalizar a situação. Ao operador fica a incumbência de acompanhar o processo de controle da planta, com o mínimo de interferência, excetuando-se casos em que sejam necessárias tomadas de decisão de atribuição restrita ao operador.

2.2 Algoritmo Genético

Algoritmos genéticos são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação. Um Algoritmo Genético cria uma população de possíveis respostas para o problema a ser tratado, no processo de inicialização, para depois submetê – lo ao processo de evolução, dividido entre as etapas: avaliação, seleção, cruzamento, mutação, atualização e finalização (NAVES, 2012).

A preocupação de um programador de Algoritmo Genético não é como chegar a uma solução, mas com o que ela deve parecer. Esta é uma técnica que advém da computação evolutiva, e é uma área que estuda as técnicas de busca baseadas na evolução dos seres vivos e na teoria evolucionista de Darwin (BARCELLOS, 2000).

Os principais elementos do Algoritmo Genético são os organismos e as populações. Através da criação e manipulação de um conjunto de estruturas (organismos) o AG tenta

encontrar sua solução para o problema. Um conjunto de organismos é denominado população e cada organismo representado por um cromossomo simples. Este método é bastante eficaz em problemas de otimização com restrições diversas.

O Algoritmo Genético permite a simplificação na formulação e solução de problemas de otimização. Os problemas de otimização são baseados em três pontos principais: a codificação do problema, a função objetivo que se deseja maximizar ou minimizar e o espaço de soluções associado. Com o uso do Algoritmo Genético é absorvido uma solução potencial para um problema específico em uma estrutura semelhante a de um cromossomo e se aplica operadores de seleção e recombinação a fim de preservar informações críticas quanto à solução do problema (BRANDÃO, 2009).

Existem três tipos de representação possíveis para os cromossomos: binária, inteira ou real. A essa representação se dá o nome de *alfabeto* do AG. De acordo com a classe de problema que se deseja resolver pode-se usar qualquer um dos três tipos.

Uma implementação de um Algoritmo Genético começa com uma população aleatória de cromossomos. Essas estruturas são, então, avaliadas e associadas a uma probabilidade de reprodução de tal forma que as maiores probabilidades são associadas aos cromossomos que representam uma melhor solução para o problema de otimização do que àqueles que representam uma solução pior. A *aptidão* da solução é tipicamente definida com relação à população corrente (LINDEN, 2008).

A função objetivo de um problema de otimização é construída a partir dos parâmetros envolvidos no problema. Ela fornece uma medida da proximidade da solução em relação a um conjunto de parâmetros. Os parâmetros podem ser conflitantes, ou seja, quando um aumenta o outro diminui. O objetivo é encontrar o ponto ótimo. A função objetivo permite o cálculo da *aptidão bruta* de cada indivíduo, que fornecerá um valor a ser usado para o cálculo de sua probabilidade de ser selecionado para reprodução (MARTINS, 2012).

Algoritmos Genéticos diferem dos algoritmos tradicionais de otimização em basicamente quatro aspectos (BRANDÃO, 2009):

a) Algoritmos Genéticos trabalham com uma codificação de conjunto de parâmetros e não com os próprios parâmetros;
b) Algoritmos Genéticos trabalham com uma população e não com um único ponto;
c) Algoritmos Genéticos utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
d) Algoritmos Genéticos utilizam regras de transição probabilísticas e não determinísticas.

O algoritmo Genético funciona da seguinte maneira:

a) Uma população inicial é gerada e é composta por indivíduos criados aleatoriamente pelo algoritmo;
b) Cada um destes indivíduos da população sofre uma função de avaliação de aptidão (*fitness*) baseada em critérios para o problema desejado;
c) Em seguida, os operadores de seleção são executados e escolhem os indivíduos de melhores valores (baseado na função de aptidão), estes indivíduos escolhidos são utilizados como base para a criação de um novo conjunto de possíveis soluções, conhecido como nova geração;
d) Para a formação de uma nova geração, estes indivíduos passarão por processos que misturam suas características utilizando o cruzamento (*crossover*) e a mutação. Isto se repetirá até que seja alcançado um ponto de parada, para isso normalmente é especificado o número de gerações. Ou, até que o algoritmo não consiga encontrar uma solução melhor do que a já encontrada.

Dentre as características do AG, destacam-se a simplicidade operacional, facilidade de implementação e a eficácia na procura do ponto onde provavelmente se localiza o máximo ou o mínimo global. E é exatamente este ponto de máximo ou mínimo global, não local, que a solução é escolhida pela execução do algoritmo. A solução escolhida apresentada possui a melhor avaliação dentre todos os indivíduos da população final. Caso o algoritmo seja executado novamente, dificilmente será obtida a mesma solução anterior. Isto significa que, apesar da solução apresentada, o AG através de seus processos aleatórios pode encontrar outra solução melhor para o problema (RODRIGUES, 2007).

2.3 Programação Orientada a Objetos

A programação orientada a objetos é um paradigma que visa representar as situações do mundo real em sistemas computacionais. Rumbaugh (1994) define orientação a objetos como: "uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real. O componente fundamental é o objeto, que combina estrutura e comportamento em uma única entidade". A grande vantagem da programação orientada a objetos é a utilização do conceito de herança.

Objeto é uma entidade do mundo real que merece representação no ambiente estudado. Um objeto possui características, chamadas de atributos. Este também é capaz de executar ações, denominadas métodos.

A base da programação orientada a objetos está na utilização de classes, definida como um conjunto de objetos com características comuns. No entanto, um objeto é uma instância de

uma determinada classe. Uma classe representa a forma como objetos com características comuns estão estruturados internamente. Chamamos instanciação quando a classe produz um novo objeto.

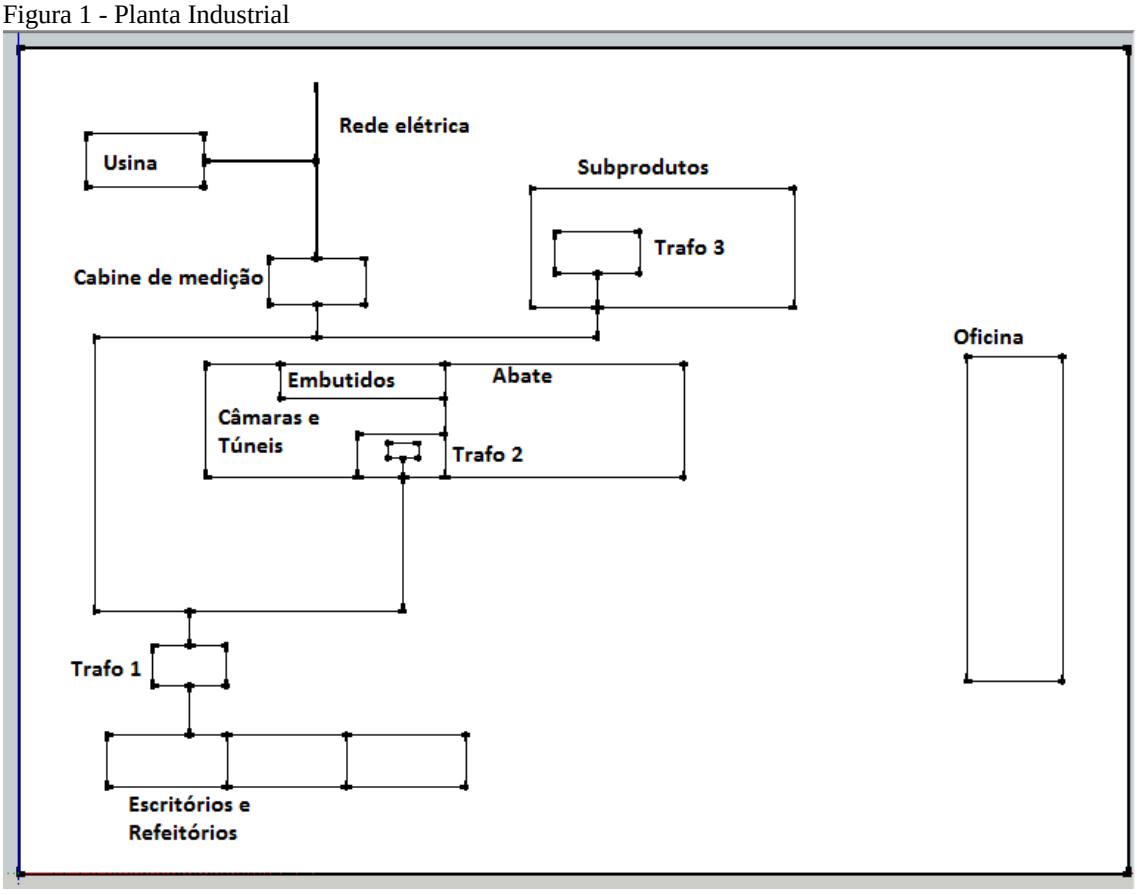
3 PLANTA INDUSTRIAL

O processo industrial utilizado nesse trabalho para o projeto do controle de demanda de energia consiste em um frigorífico de abate de aves. A indústria possui um sistema de cogeração que possui capacidade de produção de energia menor que a demanda necessária para manter o funcionamento de todo o processo produtivo.

Costa, Silva e Resende (2012) utilizou um algoritmo genético para definir a melhor combinação de cargas que devem permanecer ligadas no horário de ponta, o que foi eficiente para o processo produtivo em questão. O corte de cargas é feito de forma manual, o que pode causar um colapso em caso de falha no fornecimento de energia.

Ferroni (2013) propõe um sistema supervisório utilizando o algoritmo genético proposto por Costa, Silva e Resende (2012) que proporcione a telemetria, o telecomando e a telesupervisão para o controle da demanda de energia desta planta industrial.

Figura 1 - Planta Industrial



Fonte: (FERRONI, 2013).

O *layout* da empresa, onde o sistema de controle automático deve funcionar está apresentado na figura 1. Em caso de falta de energia os setores escritório, subprodutos e almoxarifado não são acionados, já que não comprometem a produção industrial.

A demanda média do frigorífico são 3.500 kW. A casa de máquinas é o setor de maior consumo, devido à elevada potência dos compressores de amônia. Nesse setor também estão localizados os painéis com os dispositivos de comando e proteção dos motores pertencentes ao sistema: doze compressores de potências variadas, dez forçadores (ventiladores) dos túneis de congelamento e três forçadores das câmaras de resfriamento.

Considerado primordial no processo, o setor de abate de aves, com demanda média de 800 kW não entra no corte de cargas. Motores com potências menores usados em esteiras, linhas, chillers e depenadeiras fazem parte desse setor.

O frigorífico possui três transformadores: 75kVA, 1.500kVA e 2.500kVA, que alimentavam diferentes setores da indústria. A planta conta com dois geradores de 360kVA ligados diretamente ao barramento de 380V, localizado na saída do transformador de 2.500Kva, utilizados em setores essenciais do abate.

A usina térmica de cogeração tem capacidade de 2.160kVA, sendo seis geradores de 360kVA. A usina possui um sistema de proteção e monitoramento da potência gerada pelo conjunto de geradores, assim como o sincronismo entre usina e a rede elétrica no momento da transição entre os sistemas de fornecimento de energia elétrica e um sistema eficaz e automático de controle de reativos.

A tabela 1 mostra as potências ativas medidas das cargas que podem ser desligadas do processo produtivo quando necessário, (COSTA, SILVA E RESENDE, 2012). As cargas $Co_1 a\ Co_{12}$ representam os motores dos compressores de amônia, $Ft_1\ a\ Ft_{10}$ representam os motores dos forçadores dos túneis de congelamento e $Fc_1\ a\ Fc_3$ representam os motores dos forçadores das câmaras de resfriamento.

A Tabela 1 mostra valores das potências ativas das cargas que o sistema de controle poderá atuar.

Tabela 1 – Potências elétricas do Compressores e Forçadores

Cargas	Compressores e Forçadores	Potência elétrica medida (kw)
1	Co_1	619,41
2	Co_2	474,56
3	Co_3	323,3
4	Co_4	323,3
5	Co_5	323,3
6	Co_6	323,3
7	Co_7	104,5
8	Co_8	55,2
9	Co_9	55,2
10	Co_{10}	55,2
11	Co_{11}	55,2
12	Co_{12}	33,4
13	Ft_1	42,4
14	Ft_2	54,52
15	Ft_3	54,52
16	Ft_4	54,52
17	Ft_5	36,3
18	Ft_6	36,3
19	Ft_7	36,3
20	Ft_8	36,3
21	Ft_9	36,3
22	Ft_{10}	24,22
23	Fc_1	40
24	Fc_2	40
25	Fc_3	32

Fonte: Ferroni (2013)

A demanda máxima adotada no setor de máquinas, em caso de falta de fornecimento de energia, é 1.800kVA. Isso ocorre para impedir que a usina trabalhe em sobrecarga e evita um possível desligamento da usina de cogeração.

O fornecimento de energia pela concessionária é feito em 13,8kV, por uma rede trifásica única, com 30 km de comprimento.

Costa, Silva e Resende (2012) utilizou um Algoritmo Genético para solucionar problema de definição de cargas que devem ser cortadas ou religadas quando a usina assume o fornecimento da energia. Apesar do algoritmo ser eficiente na escolha das cargas prioritárias naquele instante, o corte é manual e é necessário o religamento gradativo das cargas prioritárias.

Ferroni (2013) propôs um sistema supervisório que faz a interface entre o Algoritmo Genético proposto por Costa *et al* (2012) e uma rede de sensores sem fio. O AG utiliza os valores de temperatura das câmaras e tempo do produto dentro dos túneis como dados de entrada. Essas são informações necessárias para definir a melhor combinação de cargas. A partir desse trabalho, o corte de cargas não fica mais comprometido se houver interrupção no fornecimento de energia, preservando o processo produtivo.

Devido a dinâmica do processo produtivo os valores de pressão e temperatura usados para definir a melhor combinação de cargas sofrem alterações devido ao processo produtivo.

A temperatura ambiente também influencia o sistema de controle de cargas. Os compressores de amônia de grande porte são acionados através de inversores de frequência. A potência de operação desses compressores pode ser menor que a nominal em dias frios. Essa sobra de potência pode ser utilizada para acionar mais equipamentos, de potências menores, alterando a combinação inicial de cargas proposta pelo AG.

O objetivo do presente trabalho é utilizar a Lógica *Fuzzy* para transformar o sistema em totalmente automático, não necessitando de intervenção do ser humano. Baseado nas duas variáveis fundamentais do processo produtivo: pressão da tubulação de amônia e sobra de energia, é possível criar meios para que o sistema supervisório otimize o consumo de energia e até mesmo ajuste o indivíduo gerado pelo algoritmo genético quando necessário.

4 LÓGICA *FUZZY*

A Teoria da Lógica *Fuzzy* tem sido utilizada para lidar com o conceito de verdade parcial, ou seja, com valores de verdade entre o completamente verdadeiro e o completamente falso da lógica Booleana. Na lógica clássica existem apenas os valores verdadeiro ou falso. A maioria das situações cotidianas não podem ser resolvidas apenas com verdadeiro ou falso. Eventos como altura, temperatura, ser fumante ou não, idoso ou jovem, possuem níveis de verdade.

Na lógica *Fuzzy*, que pode ser traduzida como Lógica Nebulosa, os valores verdade são expressos linguisticamente, (exemplos: verdade, muito verdade, não verdade, falso, muito falso), onde cada termo linguístico é interpretado como um subconjunto *fuzzy* do intervalo unitário. O tempo todo utilizamos esse conceito de verdade parcial. A lógica *Fuzzy* pode ser considerada como uma das ferramentas matemáticas mais poderosas para lidar com incertezas, imprecisões e verdades parciais, permitindo a tratabilidade de problemas do mundo real muitas vezes com soluções de baixo custo. Segundo Lona (2006, p. 82):

> Pode-se definir lógica *fuzzy*, também chamada de "lógica nebulosa" ou "lógica difusa", como uma ferramenta capaz de capturar informações vagas, imprecisas e qualitativas, em geral descritas em uma linguagem natural, e convertê-las para um formato numérico de fácil manipulação por computadores (LONA, 2006, p. 82).

Em alguns casos, como em sistemas de controle de processo reais extremamente complexos, em que não se conhece detalhadamente o processo, ou cuja modelagem matemática é impraticável e muitas informações essenciais são conhecidas apenas de forma qualitativa e critérios de desempenhos só estão disponíveis em termos linguísticos, não é possível a utilização das teorias de controle clássicas. Mesmo técnicas como controle linear multivariável (DOYLE E SKIN, 1981), estimação de estado a partir de medidas ruidosas, controle ótimo (SAGE E WHITE, 1977), sistemas lineares estocásticos (BERTSEKAS, 1976), além de certas classes de problemas não-lineares determinísticos (HOLTZMAN, 1970), não são capazes de resolver problemas em que não é possível derivar o modelo matemático que descreve o problema.

A modelagem e o controle *fuzzy* (LEE, 1990) são técnicas para se manusear informações qualitativas de uma maneira rigorosa. Tais técnicas consideram o modo como a falta de exatidão e a incerteza são descritas e, fazendo isso, tornam-se suficientemente poderosas para manipular de maneira conveniente o conhecimento. A sua utilização em sistemas de controle de processos em tempo real, em computadores ou micro-controladores, é das mais convenientes, dado que,

geralmente, não envolvem nenhum problema computacional sério. A teoria de modelagem e controle *fuzzy* trata do relacionamento entre entradas e saídas, agregando vários parâmetros de processo e de controle. Isso permite a consideração de processos complexos, de modo que os sistemas de controle resultantes proporcionam um resultado mais acurado, além de um desempenho estável e robusto. A grande simplicidade de implementação de sistemas de controle *fuzzy* pode reduzir a complexidade de um projeto a um ponto em que problemas anteriormente intratáveis passam agora a ser solúveis.

4.1 Teoria dos conjuntos tradicionais

Um conjunto *A*, sobre o conjunto universo *X*, pode ser definido de três maneiras:

Um conjunto *A* cujos membros foram a_1 , a_2 e a_3 costuma ser definido de acordo com a equação 1.

$$A = \{a_1 , a_2 , a_3 \} \quad (1)$$

Válido apenas para conjuntos finitos.

Um conjunto também pode ser definido pela equação 2.

$$A = \{x \mid P(x) \} \quad (2)$$

A é o conjunto dos elementos de *X*, para os quais a proposição *P (x)* é e verdadeira.

Um conjunto é representado por uma função característica Υ, que declara quais elementos de *X* são membros do conjunto *A* e quais não são, conforme equação 3.

$$\gamma_a(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad (3)$$

As principais operações realizadas com conjuntos são União $\cup$, Intersecção $\cap$ e Complemento.

A união pode ser representada pela soma de dois conjuntos, conforme equação 4.

$$A \cup B = \{ x \mid x \in A \text{ ou } x \in B \} \quad (4)$$

A união dos conjuntos *A* e *B* contém os itens *x* tal que *x* pertence ao conjunto *A* ou x pertence ao conjunto *B.*

A intersecção de dois conjuntos é representada pela equação 5.

$$A \cap B = \{ x \mid x \in A \text{ e } x \in B \} \quad (5)$$

A intersecção dos conjuntos *A* e *B* contém os itens *x,* tal que *x* pertence ao conjunto *A* e *x* pertence ao conjunto *B.*

O complemento representa os elementos que não fazem parte de um conjunto, conforme equação 6.

$$A = \{ x \mid x \in X \text{ e } x \notin B \} \quad (6)$$

O conjunto *A* contém os elementos *x* tal que *x* pertence ao conjunto *X* e *x* não pertence ao conjunto *B.*

Segundo Zimmermann (1985), existem dois complicadores para a modelagem matemática na lógica convencional: Situações reais muito frequentemente não são determinísticas e não podem ser precisamente descritas. E a descrição completa de um sistema real frequentemente requereria mais dados detalhados do que os que um ser humano poderia sempre reconhecer simultaneamente, processar e entender. O Quadro 1 mostra as principais propriedades dos conjuntos.

Quadro 1 – Propriedades dos Conjuntos

Nº	Operação	Representação
1	Absorção	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
2	Absorção por X	$A \cup X = X$ $A \cap \emptyset = \emptyset$
3	Associatividade	$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$
4	Comutatividade	$A \cup B = B \cup A$ $A \cap B = B \cap A$
5	Distribuitividade	$A \cup (B \cup C) = (A \cup B) \cup (A \cup C)$ $A \cap (B \cap C) = (A \cap B) \cup (A \cap B)$
6	Idempotência	$A \cup A = A$ $A \cap A = A$
7	Identidade	$A \cup \emptyset = A$ $A \cap X = A$
8	Involução	$\neg\neg A = A$
9	Lei de Contradição	$A \cap \neg A = \emptyset$
10	Lei de Morgan	$\neg A \cup B = \neg A \cap \neg B$
11	Lei do Meio Excluído	$A \cup \neg A = X$

Fonte: (FERREIRA, 2001).

4.2 Teoria dos Conjuntos *Fuzzy*

Um Conjunto *Fuzzy* é definido em um universo de discurso (conjunto base) X, e caracterizado pela sua função de pertinência, onde A (x) representa o grau com que x pertence a A, de acordo com a função 7.

$$A : X \rightarrow [\ O, 1\ ] \qquad (7)$$

A teoria dos Conjuntos *Fuzzy* é uma extensão da Teoria dos Conjuntos Tradicionais. O Quadro 2 mostra as principais operações e relações de conjuntos *Fuzzy*.

Quadro 2 – Principais Operações e Relações de Conjuntos Fuzzy

Operação	**Representação**	**Natureza**
Complemento	$A^c(x) = 1 - A(x)$	Operação
Diferença	$(A \neq B)$ se $A(x) \neq B(x)$ para pelo menos um elemento de $x \in X$.	Relação
Igualdade	$(A = B)$ se $A(x) = B(x)$ para todo $x \in X$.	Relação
Inclusão	$(A \subseteq B)$ se $A(x) \leq B(x)$ para todo $x \in X$.	Relação
Intersecção	$A \cap B = A(x) \cap B(x) = \min [A(x), B(x)]$	Operação
União	$A \cup B = A(x) \cup B(x) = \max [A(x), B(x)]$	Operação

Fonte: (FERREIRA, 2001).

4.3 Proposições *Fuzzy*

As Proposições *Fuzzy* podem ser classificadas em quatro tipos:

a) Proposições *Fuzzy* Incondicionais e não Qualificadas;

p: V é F

Onde V é uma variável que assume valores x de um conjunto universo X e F é um Conjunto *Fuzzy* em que X representa um predicado *Fuzzy* tal como alto, grande, quente e outros.

b) Proposições *Fuzzy* Incondicionais e Qualificadas;

p: V é F é S

Onde V é F tem o mesmo significado, como na Equação , e S é um qualificador de verdade *Fuzzy*.

c) Proposições *Fuzzy* Condicionais e não Qualificadas;

p: Se X é A então Y é B

Onde X, Y são variáveis cujo valores estão nos conjuntos X e Y respectivamente, e A e B são Conjuntos *Fuzzy* em X e Y respectivamente.

d) Proposições *Fuzzy* Condicionais e Qualificadas;

p: Se X é A, então Y é B é S

Um elemento pode pertencer parcialmente a um dado conjunto (ZADEH, 1965). Se considerarmos uma função de pertinência que nos forneça o grau de pertinência dos diversos números ao conjunto considerado. Sendo assim, chamando de F o conjunto dos números aproximadamente iguais a 5, no universo dos números naturais $\mathbb{N}$, podemos propor por exemplo, uma função de pertinência onde o 10 ∈ F com grau 0, 0 (o que corresponde a não pertinência clássica), o 2 e o 8 ∈ F com grau de pertinência 0, 25, o 3 e o 7 ∈ F com grau 0, 5, os números 4 e 6 ∈ F com o grau 0, 75 e o 5 ∈ F com grau de pertinência 1, 0 (correspondendo a pertinência total). Esta extensão da função característica da lógica clássica para o intervalo [0, 1] originou os conjuntos *fuzzy* e possibilitou, entre outras coisas, a utilização de variáveis linguísticas, permitindo a exploração do conhecimento humano no desenvolvimento de muitos sistemas.

4.4 Número *Fuzzy*

Para definir número *fuzzy*, precisamos introduzir o conceito de α−níveis de um conjunto *fuzzy A*, que são os subconjuntos clássicos dos números reais definidos pela equação 8.

$$[A]^{\alpha} = \{x \in \mathbb{R} : A(x) \geq \alpha\} \text{ para } \alpha \in [0, 1] \quad (8)$$

Onde *A* é o conjunto *Fuzzy* que contém α níveis. O elemento *x* pertence ao conjunto dos números reais, tal que *A(x)* é maior ou igual ao número α de níveis. Um conjunto *fuzzy A* é chamado de número *Fuzzy* quando o conjunto universo, onde *A* está definido, é o conjunto dos números reais ($\mathbb{R}$) e satisfaz às condições:

a. Todos os α − níveis de A são não vazios.

b. Todos os α − níveis de A são intervalos fechados de $\mathbb{R}$.

c. O suporte de A, {x ∈ $\mathbb{R}$: A(x) > 0} = sup pA, é limitado.

A família dos números *Fuzzy* será indicada por *F(*$\mathbb{R}$*)* e, $\mathbb{R}$ é um subconjunto (clássico) de *F(*$\mathbb{R}$*)*. Segundo Lima (2003), na construção destes conjuntos *fuzzy* existe flexibilidade na escolha da forma geométrica das funções de pertinência dos elementos. As funções mais encontradas na prática são as triangulares, trapezoidais e gaussianas.

O problema da escolha das funções de pertinência não foi resolvida teoricamente e elas permanecem sendo escolhidas a depender da aplicação e do contexto do problema abordado (ORTEGA, 2001), existindo diferentes métodos para sua obtenção, inclusive alguns automatizados, mas qualquer que seja a escolha, o mais importante é retratar com máxima fidelidade o conceito apresentado (LIMA, 2003). Um número *Fuzzy* A é dito triangular se sua função de pertinência é da forma descrita pela equação 9.

$$A(x) = \begin{cases} 0, & se\ x < a \\ \dfrac{x-a}{b-a} & se\ \ a \le x \le b \\ \dfrac{x-c}{b-c} & se\ \ b \le x \le c \\ 0, & se\ x > c \end{cases} \tag{9}$$

Os números reais *a*, *b* e *c* definem o número *fuzzy* triangular *A* que será denotado pela terna ordenada (*a; b; c*). Note que o conjunto fuzzy acima não é necessariamente simétrico já que $c-b$ pode ser diferente de $b-a$.

4.5 Operações aritméticas com números *Fuzzy*

A soma dos números *Fuzzy* A e B é o número *Fuzzy*, A + B, cuja função de pertinência é definida pela equação 10.

$$(\mathrm{A} + \mathrm{B})\,(\mathrm{x}) = sup_{y+2=x} \min[A(y), B(z)] \tag{10}$$

A multiplicação de λ por A é o número *Fuzzy*, λA, cuja função de pertinência é definida pela equação 11.

$$\lambda \mathrm{A}(x) = \begin{cases} A(\lambda^{-1}x), & se\ \lambda \neq 0 \\ 0, & se\ \lambda = 0 \end{cases} \tag{11}$$

Se A e B são dois números *Fuzzy* e λ um número real, então para todo α ∈ [0, 1] tem-se as equações 12 e 13.

$$[A+B]^{\alpha} = [A]^{\alpha} + [B]^{\alpha} = \{a+b : a \in A \; e \; b \in B\} \tag{12}$$

E

$$[\lambda A]^{\alpha} = \lambda[A]^{\alpha} = \{\lambda A : a \in A\} \tag{13}$$

4.6 Relações *Fuzzy*

A teoria *Fuzzy* inclui a teoria clássica dos conjuntos, já que um conjunto clássico é um conjunto particular *Fuzzy*. Uma relação clássica indica se há ou não relação entre dois objetos e uma relação *Fuzzy* indica também o grau desta relação.

a) Definição 1: Uma relação (clássica) *R*, sobre os conjuntos *U1xU2x...xUn*, é qualquer subconjunto (clássico) do produto cartesiano *U1xU2x...xUn*. Se o produto cartesiano for formado por apenas dois conjuntos, *U1xU2*, a relação é chamada de binária sobre *U1xU2*. Se *U1 = U2 = ... = Un = U*, diz-se que *R* é uma relação sobre *U* e, se o produto cartesiano for composto por dois conjuntos iguais, *UxU*, *R* é chamada de relação binária sobre *U*. Como a relação *R* é um subconjunto do produto cartesiano, então ela pode ser representada por sua função característica C_r, conforme equação 14.

$$C_r(x_1, x_2, \dots, x_n) = \begin{cases} 1, & se \; (x_1, x_2, \dots, x_n) \in R \\ 0, & se \; (x_1, x_2, \dots, x_n) \notin R \end{cases} \tag{14}$$

A função característica C_r assume o valor 1 se o elemento *x* pertence à *R* e o valor 0 se o elemento *x* não pertence à relação *R*.

b) Definição 2: Uma relação *fuzzy R* sobre os conjuntos *U1 xU2 x...x Un* é qualquer subconjunto *fuzzy* de *U1xU2x...xUn*. Se o produto cartesiano for formado por apenas dois conjuntos, *U1xU2*, a relação é chamada de *fuzzy* binária sobre *U1xU2*. Se os conjuntos *Ui* forem todos iguais a *U*, *R* é uma relação *fuzzy* sobre *U* e, sobre *UxU*, *R* é chamada apenas de relação *fuzzy* binária. Se a função de pertinência da relação *fuzzy R* for também indicada por *R*, então o número *R (x1, x2, ..., xn)* ∈ [0, 1] indica o grau com que os elementos *xi* que compõem a n-upla (*x1, x2, ..., xn*) estão relacionados segundo a relação *R*.

c) Definição 3: O produto cartesiano *fuzzy* dos subconjuntos *fuzzy A1, A2, ... , An* de *U1, U2, ..., Un*, respectivamente, é a relação *fuzzy* descrita pela equação 15.

$$R(x_1, x_2, \dots, x_n) = A_1(x_1) \wedge A_2(x_2) \wedge, \dots, \wedge A_n(x_n) \quad (95)$$

Onde ∧ representa o mínimo.

4.7 Inferências e Regras *Fuzzy*

As regras *fuzzy* descrevem situações específicas que podem ser submetidas à análise de um painel de especialistas, e cuja inferência nos conduz a algum resultado desejado. Cada regra *fuzzy* é uma unidade capaz de capturar algum conhecimento particular e da mesma forma que uma afirmação clássica, é composta por uma parte antecedente (a parte Se) e uma parte consequente (a parte Então), resultando em uma estrutura do tipo:

Se {antecedentes} Então {consequentes}

Os antecedentes descrevem uma condição (premissas), enquanto a parte consequente descreve uma conclusão ou uma ação que pode ser esboçada quando as premissas se verificam.

Segundo Ortega (2001), os antecedentes definem uma região *fuzzy* no espaço das variáveis de entrada do sistema. Já os consequentes descrevem uma região no espaço das variáveis de saída do sistema, qual seja a sua conclusão ou ação. Sendo assim, a construção dos antecedentes muitas vezes resulta em um trabalho de classificação, enquanto a elaboração dos consequentes exige um conhecimento, ainda que empírico, sobre a dinâmica do sistema, tarefa esta que pode ser mais complexa.

A regra geral para composição de relações *fuzzy*, chamada de MAX-MIN ou MAMDANI, consiste em tomarmos o mínimo em uma 'serie de conexões', e tomarmos o máximo em 'conexões paralelas' (ORTEGA op. cit.).

4.8 Variáveis linguísticas

Variáveis linguísticas são variáveis cujos valores são nomes de conjuntos *Fuzzy*. Por exemplo, a temperatura de um determinado processo poderia ser uma variável linguística

assumindo valores como alta, baixa e média. Estes valores são descritos por intermédio de conjuntos *Fuzzy*.

Uma variável linguística é caracterizada por {*n, T, X, m(n)*} onde *n* é o nome da variável (por exemplo, temperatura, pressão, febre, etc.), *T* é o conjunto de termos linguísticos de *n* (elevado, baixo, pouco, extenso, etc.), *X* é o domínio (Universo) de valores de *n* sobre o qual o significado do termo Variáveis linguísticas é determinado (a febre pode estar, por exemplo, entre 35 e 40◦ C) e *m(t)* é uma função semântica que assinala para cada termo linguísticos $t \in T$ o seu significado, que é um conjunto *fuzzy* em *X* (ou seja, $m : T \rightarrow (X)$ onde *(X)* é o espaço dos conjuntos *fuzzy*).

As variáveis linguísticas são expressas dentro de um certo domínio de valores. As variáveis simbólicas têm conquistado cada vez maior importância devido ao desenvolvimento das áreas de inteligência artificial e processos de decisão. A capacidade de combinar variáveis simbólicas (linguísticas) e numéricas é uma das principais razões do sucesso das aplicações da Lógica *Fuzzy* em sistemas inteligentes, tanto na engenharia quanto em muitas outras áreas que lidam com domínios contínuos (ORTEGA, 2001).

4.9 Possibilidade

Os conjuntos *Fuzzy* geram vários graus de possibilidade. Uma função de pertinência gera a distribuição de possibilidades.

Sejam $f : \Omega \rightarrow [0, 1]$, com $f(x) = 1$, para algum $x \in \Omega$ e $A \subset \Omega$. A medida de possibilidade de *A* é o número real dado pela equação 16.

$$\pi(A) = \begin{cases} sup_{x \in A} f(x), & se\ A \neq \emptyset \\ 0, & se\ A = \emptyset \end{cases} \tag{16}$$

4.10 Desfuzzificação

A Desfuzzificação é o procedimento que nos permite interpretar a distribuição de possibilidades da saída de um modelo linguístico *fuzzy* de forma quantitativa, ou seja, ele nos fornece um valor numérico representativo que captura o significado essencial dessa distribuição de possibilidades. O método do Centro de Área é a técnica de desfuzzificação mais comumente usada. Ele também é citado na literatura como método do Centro de Gravidade ou do Centróide (KLIR e YUAN 1995, YEN e LANGARI 1999 apud ORTEGA, 2001).

O procedimento é similar ao usado para calcular o centro de gravidade em física, se consideramos a função de pertinência *μA(x)* como a densidade de massa de x. O método também pode ser compreendido como uma média ponderada, onde *μA (x)* funciona como o peso do valor *x*. Se *x* é discreto, então a desfuzzificação da conclusão *fuzzy A* é dada pelas equações 17 e 18.

$$y_0 = \frac{\sum_x \mu A(x).x}{\sum_x \mu A(x)} \tag{17}$$

No caso de x ser contínuo:

$$y_0 = \frac{\int \mu A(x).xdx}{\int \mu A(x)dx} \tag{18}$$

4.11 Processo de decisão *Fuzzy*

O processo de decisão é baseado em escolher qual estratégia é mais eficiente. A teoria de decisão *Fuzzy* utiliza formulações vagas e imprecisas, próximas ao pensamento humano.

No processo de decisão *Fuzzy*, os resultados devido a cada ação são apenas aproximados, esse processo é dito sobre condições de imprecisão (KLIR; YUAN 1995).

Bellmam e Zadeh (1970) sugeriram um modelo *fuzzy* de tomada de decisão no qual os objetivos e restrições relevantes são expressos em termos de conjuntos *fuzzy*, e a decisão é determinada a partir de um tipo de agregação apropriada desses conjuntos (BELLMAM; ZADEH, 1970). Um processo de decisão nesse tipo de modelo é caracterizado pelos seguintes componentes:

a) Um conjunto *A* de possíveis ações ou estratégias;

b) Um conjunto de metas *Gi* $(i \in \mathbb{N})$, cada uma das quais expressa em termos de um conjunto *fuzzy* definido sobre *A*;

c) Um conjunto de restrições *Cj* $(j \in \mathbb{N})$, cada um também sendo expresso em termos de um conjunto *Fuzzy* definido sobre *A*.

Então, dada uma situação de decisão caracterizada pelos conjuntos *fuzzy A*, *Gi* $(i \in \mathbb{N})$ e *Cj* $(j \in \mathbb{N})$, a decisão *fuzzy D*, é definida como um conjunto *fuzzy* sobre *A* que satisfaz simultaneamente as metas e as restrições, de acordo com a equação 19.

$$D\ (a) = \min_{i \in \mathbb{N}} [\inf G_i(a), \qquad \inf_{j \in \mathbb{N}} C_j(a)] \tag{19}$$

Para qualquer a $\in A$.

Ou seja, a estratégia definida como solução mais adequada será a mais pertinente do conjunto dos mínimos entre objetivos e restrições.

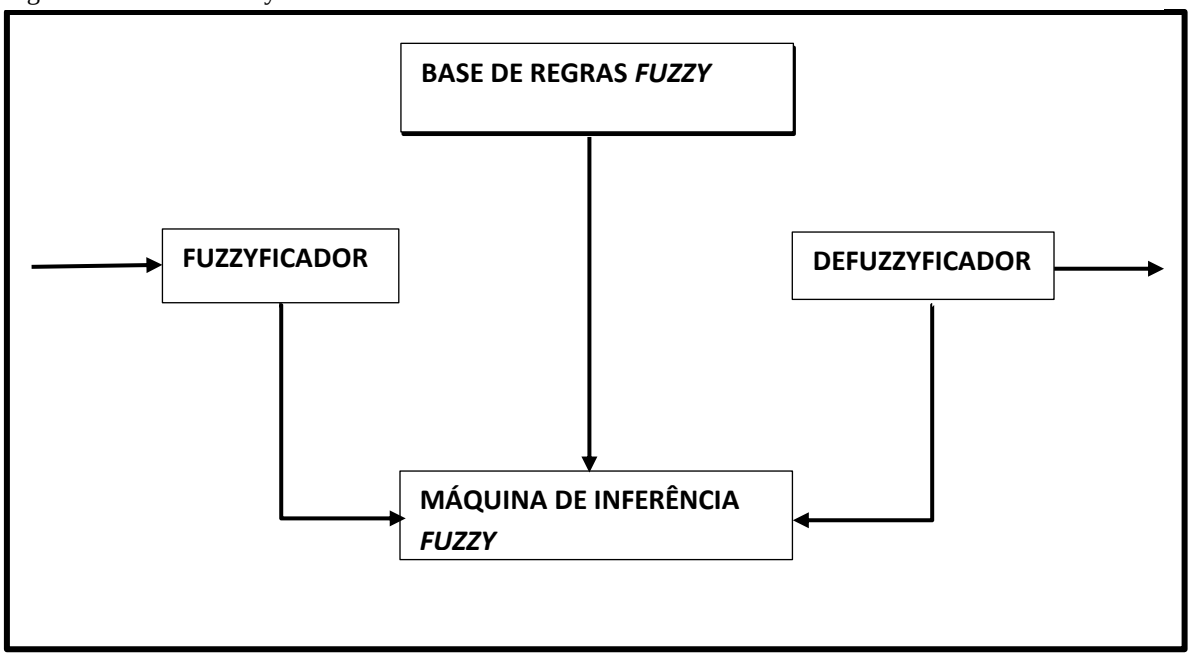
4.12 Sistema *Fuzzy*

Um sistema *fuzzy* é entendido como uma coleção de variáveis de entrada (sendo cada uma, uma coleção de conjuntos), uma ou mais variáveis de saída (sendo também cada uma, uma coleção de conjuntos), e uma coleção de regras que associam as entradas com as saídas.

As variáveis de entrada ingressam com seus valores quantitativos ou qualitativos. As regras de inferência são implementadas mediante operações lógicas do tipo união, interseção e negação, apropriadamente definidas e por fim, as variáveis de saída são desfuzzificadas por algum método para obter resultados 'nítidos' que representem adequadamente o tipo de saída desejada para o problema.

Evidentemente existem possibilidades de sistemas híbridos segundo a aplicação que se trate pelo que o nome sistema *fuzzy* é apenas indicativo. A Figura 3 ilustra um sistema típico no qual as variáveis de entrada e de saída sofrem transformações de seus valores, de nítidos para *fuzzy* e de *fuzzy* para nítidos respectivamente, mediante os chamados 'Fuzzificador' e 'Desfuzzificador'. A 'Base de Regras *fuzzy*' constitui o 'conhecimento do sistema' e a 'Máquina de Inferência *fuzzy*' contém as especificações lógicas para processar as informações.

Figura 2 – Sistema Fuzzy



Fonte: (BRANÃ, 2008)

5 DESENVOLVIMENTO E RESULTADOS

O desenvolvimento desse sistema de automatização foi feito a partir do *software* Matlab 2016, aplicando os conceitos da programação orientada a objetos.

5.1 Funcionamento

O código *Fuzzy* recebe os dois vetores que representam o estado dos compressores e forçadores (ligado ou desligado). A partir desses valores o sistema calcula a carga total ligada no momento, que varia entre 1200kW e 1780kW, de acordo com os valores gerados pelo algoritmo genético (FERRONI, 2013). A lógica *fuzzy* faz o ajuste desse valor para o mais próximo possível à 1780kW, levando em consideração os níveis de pressão, que precisam ser mantidos entre 7.5 e 8.3 Kgf/m².

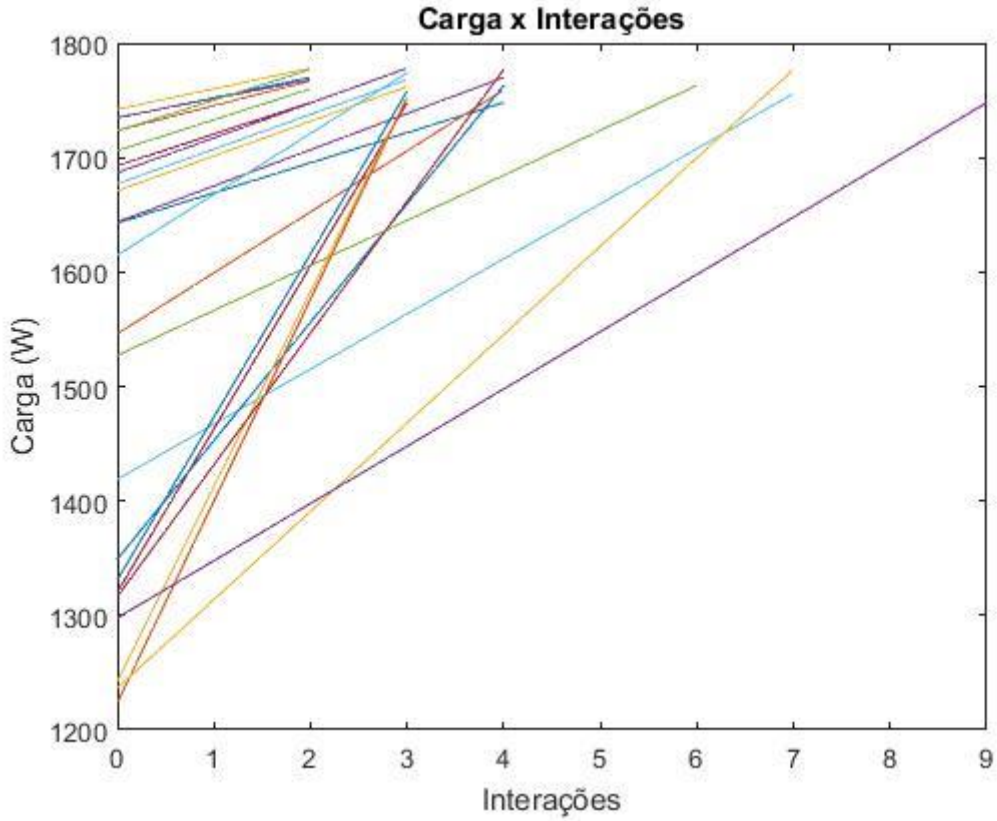
Após esse ajuste inicial de cargas é possível que o operador altere o estado dos forçadores, de acordo com a necessidade do sistema industrial em questão. O código fuzzy novamente adapta os compressores e forçadores ligados para que a carga se mantenha abaixo de 1.800kW (um mil e oitocentos) e a pressão se mantenha entre os níveis esperados para o bom funcionamento da planta industrial.

É necessário que a cada interação haja uma ponderação dos valores de pressão, devido ao impacto causado pela ligação ou retirada de compressores ou forçadores do sistema elétrico. O algoritmo genético inicialmente escolhe quais são as cargas prioritárias e em todos os testes realizados, o compressor de maior valor sempre se manteve ligado. Sendo ele, em todas as condições testadas a carga prioritária do sistema. Por esse motivo, o mesmo nunca será desligado.

5.2 Resultados

Foram registrados 50 testes mostrando o tempo e número de interações necessárias para que o sistema chegue à otimização no ajuste inicial de cargas. A Figura 3 apresenta a relação Carga *X* Interações.

Figura 3 - Carga *X* Interações

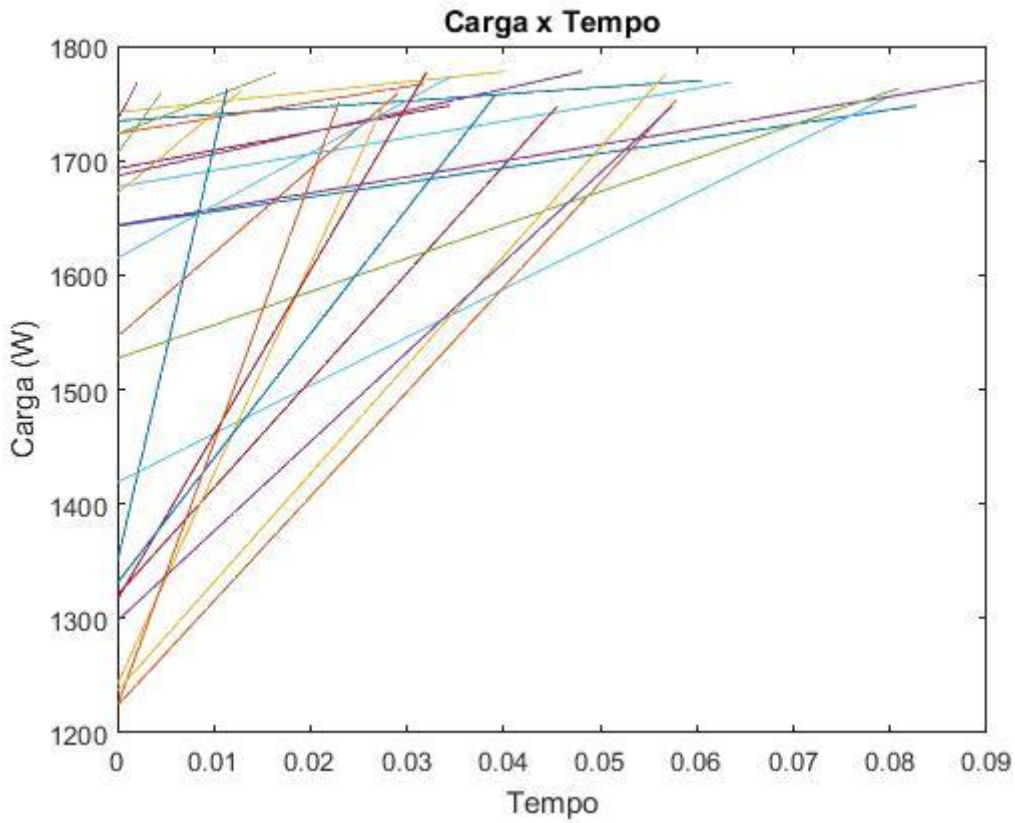


Fonte: O autor

A Figura 3 mostra que 9 foi o número máximo de interações necessárias para que o sistema atinja o nível ideal de carga e o mínimo foram apenas 2.

A Figura 4 mostra a relação Carga *X* Tempo.

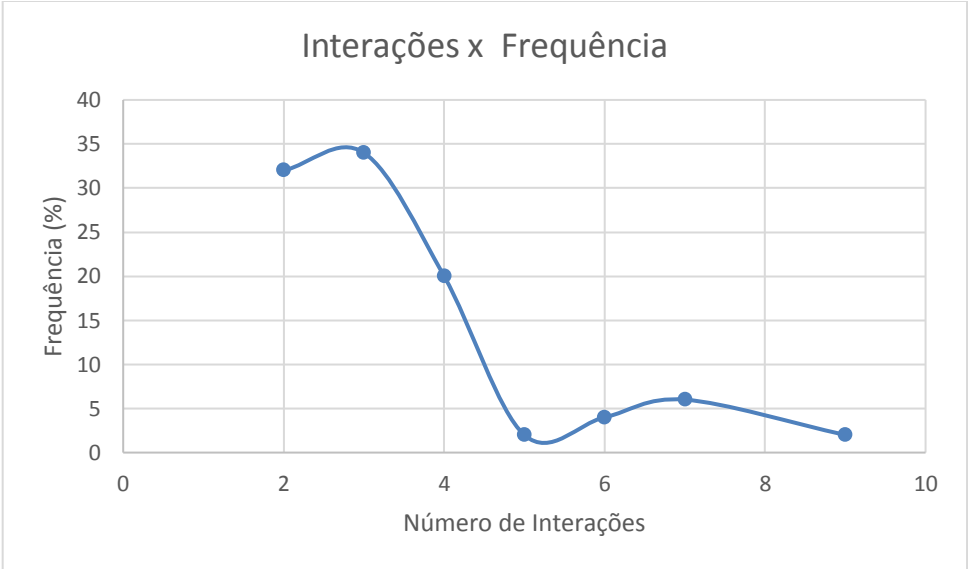
Figura 4 - Carga *X* Tempo



Fonte: O autor

O sistema se mostra eficaz devido ao seu tempo de atuação ser praticamente insignificante perto do tempo do corte de cargas (FERRONI, 2013) de 15 segundos. A Figura 5 mostra a frequência de interações, comprovando a agilidade do sistema.

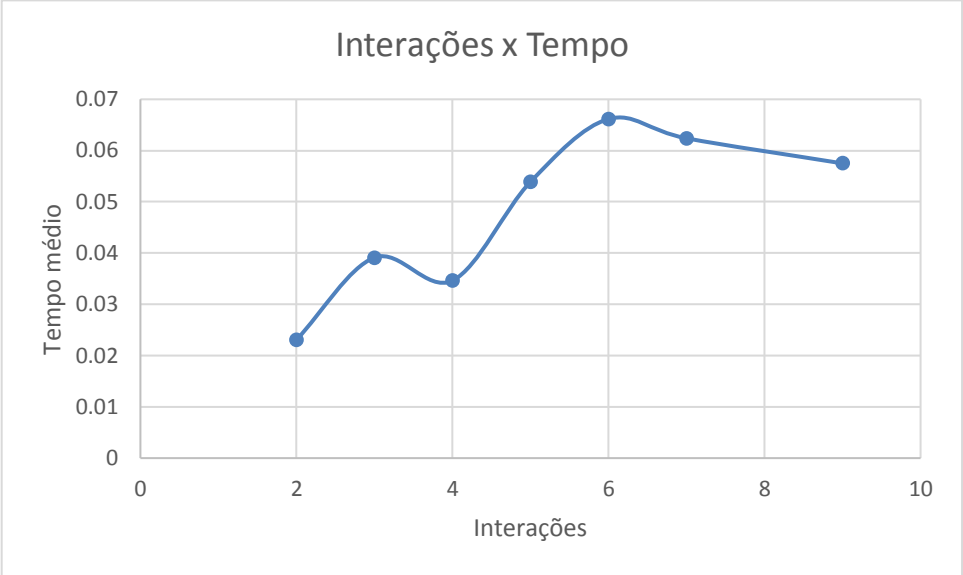
Figura 5 - Frequência x Interações



Fonte: O autor

Conforme demonstrado no Gráfico 3, a maioria dos testes do ajuste de carga, que foram realizados com 50 combinações diferentes, chegaram ao nível ótimo com apenas 2 ou 3 interações. A Figura 6 mostra que o tempo médio máximo para estabilização foi 0,07 segundos.

Figura 6 - Tempo x Interações



Fonte: O autor

Após os testes do ajuste inicial de carga, foi realizado também a automatização do sistema, no momento em que o operador precisa alterar o estado de algum forçador, fazendo com que o sistema precise ser ajustado para os valores mais próximos a 1800kW, mantendo a pressão na faixa de valores aceitáveis para o processo industrial.

A Figura 7 representa o processo após o ajuste inicial de cargas, com a necessidade de ajuste através do operador. Com a alteração efetuada, o sistema pode sair dos níveis ótimos de potência para funcionamento. A Lógica *Fuzzy* precisa readequar os níveis de pressão e carga. Foram realizados 50 testes para esta situação, e em alguns momentos, como mostra a Figura 8, o tempo de estabilização foi muito baixo, pois os níveis se mantiveram abaixo de 1800kW. Em outros testes, o desligamento de forçadores levou o sistema a um nível abaixo do ideal, fazendo com que o tempo de estabilização fosse mais alto.

Figura 7 – Manutenção da Automatização

Fonte: O autor

Conforme Figura 7, o tempo para estabilização nunca ultrapassou 0,04 segundos, tempo desprezível se comparado aos 30 segundos que o sistema de proteção da usina de cogeração demora para entrar em funcionamento em caso de falhas no fornecimento de energia elétrica.

6 CONCLUSÃO

Utilizou - se Lógica *Fuzzy* para automatizar totalmente um processo. Como a paralisação desse processo industrial pode causar grandes prejuízos ao frigorifico, há uma usina de cogeração no local. Porém, atualmente a capacidade de geração da usina é menor que a demanda da indústria. Por essa razão, em caso de falha no fornecimento de energia pela concessionária é necessário que haja um corte inteligente de cargas.

Ferroni (2013), utilizou algoritmo genético para tornar o corte de cargas automático. Contudo, após esse corte inicial, é necessário que um operador monitore o sistema, através de um sistema supervisório *SCADA (Supervisory Control and Data Acquisition)*, devido à dinâmica do processo produtivo industrial. Entretanto não foi feita a manutenção da automatização após o corte de cargas.

Para o complemento da automatização dessa planta industrial foi utilizado Lógica *Fuzzy* para manter o sistema automatizado após o corte de cargas realizado pelo Algoritmo Genético. Tornando todo o sistema totalmente automatizado com o uso de sistema supervisório, algoritmo genético, rede de sensores sem fio e Lógica *Fuzzy*.

Segundo a proposta de Ferroni (2013), criou – se um sistema automático via rádio, para corte inicial inteligente de cargas utilizando Algoritmo Genético. Entretanto, em muitas situações, a combinação ideal de cargas proposta inicialmente pelo Algoritmo Genético consumia uma quantidade de energia menor que a prevista, pois alguns compressores funcionam com inversor de frequência e todo sistema decisório utilizado nesse trabalho foi baseado no valor máximo de carga de cada equipamento.

Com o uso da Lógica *Fuzzy*, esse trabalho corrige essas possíveis diferenças que possam vir a existir entre a combinação de cargas prevista e a potência real consumida, que em muitos casos no trabalho de Ferroni (2013), ficaram abaixo desse valor.

A partir da agregação desse sistema ao proposto por Ferroni (2013), foi possível realizar o ajuste fino das cargas ligadas, afim de otimizar a utilização da potência. Chegando sempre, em todos os testes realizados, a valores próximos a 1800kW.

A Lógica *Fuzzy* apenas funciona após a atuação do Algoritmo Genético. Dado que, durante o funcionamento normal da indústria, sem falta de energia, todos os equipamentos permanecem ligados normalmente.

Em casos de períodos prolongados de falta de energia elétrica, como no horário de pico, as temperaturas das câmaras se alteravam e o operador precisava fazer alterações na

combinação de cargas gerada pelo algoritmo genético. O uso da Lógica *Fuzzy* reestabiliza automaticamente o sistema após as alterações do operador.

A rede de sensores sem fio proposta por Ferroni (2013), possui um tempo de 15 segundos para entrar em funcionamento. Já a Lógica *Fuzzy,* em todos os casos testados, apresentou um tempo menor que 1 segundo. Como o sistema de proteção da indústria possui um sistema de atuação de 30 segundos, o sistema se mostrou eficiente na automatização dessa planta industrial.

REFERÊNCIAS

BARCELLOS, J. C. H. **Algoritmos Genéticos Adaptativos:** Um estudo comparativo. 2000. 131 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo. 2000.

BELLMAN, R. E.; ZADEH, L. A. Decision making in a Fuzzy environment. **Management Science**, Bekerley, v. 17, n. 4, p. 141-154, Dezembro, 1970.

BERTSEKAS, D.P. **Dynamic Programming and Stochastic Control. Mathematics in Science and Engineering,** Illinois: Academic Press, 1976. Vol. 125.

BRANDÃO, J. S. **Aplicação de algoritmos genéticos para minimização do número de objetos processados e o setup num problema de corte unidimensional.** 2009. 99 f. Dissertação (Mestrado) – Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo. 2009.

BROLIN, L. C. **Análise de planos de corte de carga através de métodos diretos.** 2010. 73 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de São Paulo, Escola de Engenharia de São Carlos, São Carlos. 2010.

FERREIRA, J. C. **Elementos de Lógica Matemática e Teoria dos Conjuntos.** Departamento de Matemática, Instituto Superior Técnico. Lisboa, 2001.

COMPANHIA PAULISTA DE FORÇA E LUZ (São Paulo, SP). **Ligação de autoprodutores em paralelo com o sistema de distribuição da CPFL,** Versão 1.2. São Paulo, 2005.

COSTA, M. H., SILVA, V. V. R., RESENDE, L. C. **Controle de demanda por corte ideal de cargas em tempo real de um processo produtivo via algoritmo genético.** In: SIMPOSIO IBERO-AMERICANO DE APLICACIONES Y TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES, 2012, Orlando. **Anais**, Orlando: ATIC, 2012.

DOYLE, J.C. E G. SKIN. Multivariable feedback design: concept for a classical/modern synthesis. **IEEE Trans. on Automatic Control**, Bekerley, vol. AC-26, pp. 4-16, Fevereiro, 1981.

FERRONI, E. H. **Protótipo de um controle de demanda de energia baseado em sistemas supervisório e rede de sensores sem fio.** 2013. 99 f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de São João Del Rey, São João Del Rey, 2013.

HOTZMAN, J. M. **Nonlinear system theory**. New Jersey: Prentice Hall, 1970.
IEEE Power Engineering Society. **IEEE guide for application of protective relays used for abnormal frequency load shedding and restoration**, IEEE Std. C37.117, pp. c1-43, 2007.

JENKINS, N *et al*. **Embedded generation.** Londres: The Institute of Eletrical Engineers (IEE), 2002, p. 292.

JOB, F. P. P. **Os Sentidos do trabalho e a importância da resiliência nas organizações.** FGV/EAESP. São Paulo, 2003. 237f.

KLIR, G.; YUAN, B. ***Fuzzy*** **Sets and Fuzzy Logic - Theory and Applications**, New Jersey: Prentice Hall, Maio, 1995.

LEE, C.C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – parts 1 and 2. **IEEE Transactions on Systems**, Man and Cybernetics, Bekerley, v. 20, n. 2, p. 404-435, 1990.

LEITE, F. C. **Modelamento da eficiência energética para o gerenciamento sustentável no setor industrial pela medição e verificação.** 2010. 94 f. Dissertação (Doutorado) – Escola Politécnica da Universidade de São Paulo. São Paulo, 2010.

LIMA, C. J. TEIXEIRA DE. **Processo de Tomada de Decisão em Projetos de Exploração e Produção de Petróleo no Brasil:** Uma abordagem utilizando conjuntos nebulosos. 2003. 141 f. Dissertação de Mestrado, PPE/COPPE/UFRJ. Programa de Planejamento Energético, Universidade Federal do Rio de Janeiro, 2003.

LINDEN, R. **Algorítmos genéticos: Uma importante ferramenta da inteligência computacional.** 2ª Edição, Rio de Janeiro, Brasport, 2008.

LONA, N. R. **Lógica *Fuzzy* aplicada a Sistemas de Controle de Edifícios Inteligentes.** 2006. 103 f. Dissertação (Mestrado em Engenharia Mecânica) – Departamento de Mecânica, Universidade de Taubaté, Taubaté. 2006.

MAIA, L. T. **Um estudo sobre aplicação de técnicas de inteligência artificial e engenharia de software à construção de um sistema de supervisão e controle.** 2007. 87f. Dissertação (mestrado) – Departamento de Engenharia Elétrica, Universidade de Brasília. Brasília, 2007.

MARTINS, S. R. **Estudo avaliativo de um algoritmo genético auto – organizável e multiobjetivo utilizando aprendizado de máquina para aplicações de telecomunicações.** 2012. 85 f. Dissertação (Mestrado) – Centro de tecnologia, Universidade Federal do Rio Grande do Norte. Natal, 2012.

NAVES, T. F. **Uma abordagem para maximização da produção de recursos em jogos RTS.** 2012. 118 f. Dissertação (Mestrado) – Faculdade de Computação, Universidade Federal de Uberlândia. Uberlândia, 2012.

RIBEIRO NETO, E. ANÁLISE SWOT – **Planejamento Estratégico para Análise de Implantação e Formação de Equipe de Manutenção em uma Empresa de Segmento Industrial.** 2011.33f. Tese de Conclusão de Curso - MBA – ICAP/PITÁGORAS, Faculdade Pitágoras. São João Del Rey, 2011.

ORTEGA, N. R. S. **Aplicação da Teoria de Conjuntos *Fuzzy* em Problemas de Biomedicina.** 2001. 152 f. Tese (Doutorado em Ciências) – Instituto de Física, Universidade de São Paulo, São Paulo, 2001.

RODRIGUES, N.M. **Um algoritmo cultural para problemas de despacho de energia elétrica.** 2007, 99f. Dissertação de Mestrado – Universidade Estadual de Maringá, 2007.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W. "**Modelagem e Projetos Baseados em Objetos**". Campus: Rio de Janeiro, 1994.

SAGE, A. E C. WHITE . **Optimum Systems Control** - New Jersey, Prentice Hall, 1977.

SANTOS, L. F.; TOLARDO, O.; NOGUEIRA, A. S. **Esquema de Rejeição de Cargas Inteligente com Funcionalidade Distribuída Utilizando Recursos da Norma IEC61850.** VI CIERTEC – Seminário Internacional sobre Smart Grid em Sistemas de Distribuição de Energia Eléctrica. Belo Horizonte, 2009.

VANDERLEI, L. O. O. **A Cogeração como Difusão de um Processo de Inovação Tecnológica e Estratégia Empresarial no Setor Sucroalcooleiro: o Caso da Destilaria Giasa.** 2003, 193f. Dissertação – Universidade Federal de Pernambuco. Recife, 2003.

VASCONCELOS, A. **Esquema de proteção para ilhamento preventivo de indústrias cogeradoras.** Campinas: Figener S/A Engenheiros Associados, 2006.

YEN J.; LANGARI R. **Fuzzy Logic: Intelligence, Control, and Information**. EUA, Prentice Hall, 1999.

ZADEH, L. Fuzzy Sets. **Information and Control**, Bekerley, v. 8, n. 3, p. 338-353, 1965.

ZIMMERMANN, H. J. ***Fuzzy*** **Set Theory and Its Applications**, Boston, Kluwer Nijhoff Publishing, Julho, 1985.

APÊNDICE – Código *Fuzzy*

```
classdef fuzzy
   properties
      sp%a variavel sp representa o sensor de pressão que varia entre 7.5 e 8.3
      vcomp = [];%vetor com as cargas dos compressores
      vforc = [];%vetor contendo as cargas dos forçadores
      estcomp = [];%indica o estado dos compressores, se eles estao ligados ou desligados por
valores binarios 0 ou 1
      estforc = [];%indica o estado dos forçadores, se eles estão ligados ou desligados por
valores binarios 0 ou 1
      varcomp = [];
      varforc = [];
      sc %energia restante a ser usada
      ct
      pressao
      carga
      cargacomp
      cargaforc
      x
    end

   methods
      function F = fuzzy()
         F.sp = 7.5 + (8.3 - 7.5).*rand(1);%a variavel sp representa o sensor de pressão que
varia entre 7.5 e 8.3
         F.vcomp = [619.41 474.56 323.3 323.3 323.3 323.3 104.5 55.2 55.2 55.2 55.2
33.4];%vetor com as cargas dos compressores
         F.vforc = [42.4 54.52 54.52 54.52 36.3 36.3 36.3 36.3 36.3 24.22 40 40 32];%vetor
contendo as cargas dos forçadores
         F.varcomp = [1 0.304 0.208 0.208 0.208 0.208 0.064 0.032 0.032 0.032 0.032 0.032
0.02];
         F.varforc = [-0.088; -0.124; -0.124; -0.124; -0.088; -0.088; -0.088; -0.088; -0.088; -
0.064; -0.076; -0.076; -0.06];
      end

      function obj = set.estcomp(obj,estcomp)
         obj.estcomp = estcomp;
         global compest;
         compest = obj.estcomp;
      end

      function obj = set.estforc(obj,estforc)
         obj.estforc = estforc;
         global forcest;
         forcest = obj.estforc;
      end

      function cargacomp = compcarga(F)
```

```
    global compest;
    cargacomp = F.vcomp * compest; %representa a carga total ligada dos compressores
end

function cargaforc = forccarga(F)
    global forcest;
    cargaforc = F.vforc * forcest;
end

function sc = sobracarga(obj)
    sc = 1780 - (obj.compcarga() + obj.forccarga());
end

function pressao = verifpressao(F)
    global sensor;
    sensor = F.sp;
    if (sensor >= 7.5) && (sensor < 7.7)
        pressao = 1;%pressao = 1 muita pressao negativa
    elseif (sensor >= 7.7) && (sensor < 7.9)
        pressao = 2;%pressao = 2 pouca pressao negativa
    elseif (sensor >= 7.9) && (sensor < 8.1)
        pressao = 3;%pressao = 3 pouca pressao positiva
    elseif (sensor >= 8.1)
        pressao = 4;%pressao = 4 muita pressao positiva
    end
end

function carga = verifcarga(F)
    global compest;
    global forcest;
    F.sc = 1780 - ((F.vcomp * compest) + (F.vforc * forcest));
    if (F.sc >= 0) && (F.sc < 35)
        carga = 1;%carga = 1 desprezivel
    elseif (F.sc >= 35) && (F.sc < 60)
        carga = 2;%muito pouca
    elseif (F.sc >= 60) && (F.sc < 110)
        carga = 3;%carga = 3 pouca
    elseif (F.sc >= 110) && (F.sc < 330)
        carga = 4;%carga = 4 media
    elseif (F.sc >= 330) && (F.sc < 480)
        carga = 5;%carga = 5 alta
    elseif (F.sc >= 480) && (F.sc < 580)
        carga = 6;%carga = 6 maxima
    end
end

function a = acao (obj)
    tic
    global compest;
    global forcest;
```

```
global sensor;
obj.x = 1;
while (obj.verifcarga() ~= 1)&&(obj.x < 10)
      if (obj.verifcarga() == 6) && ((obj.verifpressao() == 1)||(obj.verifpressao() == 2))
         if compest(2) ~= 1
            compest(2) = 1;
            sensor = sensor + obj.varcomp(2);
            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
            obj.sc = 1780 - obj.ct
            obj.estcomp = compest
            obj.sp = sensor
         else
            for b=3:6
               if compest(b) == 0
                  compest(b) = 1;
                  sensor = sensor + obj.varcomp(b);
                  obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                  obj.sc = 1780 - obj.ct
                  obj.estcomp = compest
                  obj.sp = sensor
                  break;
               end
            end
         end

   elseif (obj.verifcarga() == 5) && ((obj.verifpressao() == 1)||(obj.verifpressao() ==
2))
         if (compest(3)==1)||(compest(4)==1)||(compest(5)~=1)||(compest(6)~=1)
            for d=3:6
               if compest(d) == 0
                  compest(d) = 1
                  sensor = sensor + obj.varcomp(d);
                  obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest)
                  obj.sc = 1780 - obj.ct
                  obj.estcomp = compest
                  obj.sp = sensor
                  break;
               end
            end
            else
               if compest(7) ~= 1
                  compest(7)= 1;
                  sensor = sensor + obj.varcomp(7);
                  obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                  obj.sc = 1780 - obj.ct
                  obj.estcomp = compest
                  obj.sp = sensor
               end
            end
```

```
        elseif (obj.verifcarga() == 4) && ((obj.verifpressao() == 1)||(obj.verifpressao() ==
2))
            if compest(7) ~= 1
                compest(7)= 1;
                sensor = sensor + obj.varcomp(7);
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct
                obj.estcomp = compest
                obj.sp = sensor
            else
                if (compest(8)==0)||(compest(9)==0)||(compest(10)==0)||(compest(11)==0)
                    for y=8:11
                        if compest(y) == 0
                            compest(y) = 1;
                            sensor = sensor + obj.varcomp(y);
                            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                            obj.sc = 1780 - obj.ct
                            obj.estcomp = compest
                            obj.sp = sensor
                            break;
                        end
                    end

                    else
                        if compest(12) ~= 1
                            compest(12)= 1;
                            sensor = sensor + obj.varcomp(7);
                            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                            obj.sc = 1780 - obj.ct
                            obj.estcomp = compest
                            obj.sp = sensor
                        end
                    end
                end


        elseif (obj.verifcarga() == 3) && ((obj.verifpressao() == 1)||(obj.verifpressao() ==
2))
            if (compest(8)==0)||(compest(9)==0)||(compest(10)==0)||(compest(11)==0)
                for y=8:11
                    if compest(y) == 0
                        compest(y) = 1;
                        sensor = sensor + obj.varcomp(y);
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct
                        obj.estcomp = compest
                        obj.sp = sensor
                        break;
                    end
```

```
            end
        else
            if compest(12) ~= 1
                compest(12)= 1;
                sensor = sensor + obj.varcomp(7);
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct
                obj.estcomp = compest
                obj.sp = sensor

            else
                for r = 1:13
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct;
                if (forcest(r)== 0)&&(obj.sc > obj.vforc(r))
                    forcest(r) = 1;
                    sensor = sensor + obj.varforc(r);
                    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                    obj.sc = 1780 - obj.ct
                    obj.estforc = forcest
                    obj.sp = sensor
                    break;
                end
            end
            end
        end

    elseif (obj.verifcarga() == 2) && ((obj.verifpressao() == 1)||(obj.verifpressao() ==
2))
        if compest(12) ~= 1
            compest(12)= 1;
            sensor = sensor + obj.varcomp(7);
            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
            obj.sc = 1780 - obj.ct
            obj.estcomp = compest
            obj.sp = sensor
        else
            for r = 1:13
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct;
                if (forcest(r)== 0)&&(obj.sc > obj.vforc(r))
                    forcest(r) = 1;
                    sensor = sensor + obj.varforc(r);
                    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                    obj.sc = 1780 - obj.ct
                    obj.estforc = forcest;
                    obj.sp = sensor
                    break;
                end
            end
```

```
            end

        elseif (obj.verifcarga() ~= 1) && ((obj.verifpressao() == 3) || (obj.verifpressao() ==
4)) %uso de forçadores
            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
            obj.sc = 1780 - obj.ct;
             if
(forcest(1)==0)||(forcest(2)==0)||(forcest(3)==0)||(forcest(4)==0)||(forcest(5)==0)||(forcest(6)=
=0)||(forcest(7)==0)||(forcest(8)==0)||(forcest(9)==0)||(forcest(10)==0)||(forcest(11)==0)||(forc
est(12)==0)||(forcest(13)==0)
                for z = 1:13
                    if (forcest(z)== 0) &&(obj.sc > obj.vforc(z))
                        forcest(z) = 1;
                        sensor = sensor + obj.varforc(z);
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct;
                        obj.estforc = forcest
                        obj.sp = sensor
                        break;
                    end
                end
            elseif compest(12)==0
                compest(12) = 1;
                sensor = sensor + obj.varcomp(12);
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct;
                obj.estforc = forcest;
                obj.sp = sensor;
            else
                for g=8:11
                    if (compest(g)==0)
                        compest(g) == 1;
                        sensor = sensor + obj.varcomp(g);
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct;
                        obj.estforc = forcest;
                        obj.sp = sensor;
                    end
                end
            end



        elseif obj.verifcarga() == 1
            break;
        end

    obj.x = obj.x + 1
    end
```

```
    obj.sp = sensor
    obj.ct = obj.ct
    compest
    forcest
    a=toc

end

function operador (obj, ind_forc)
    global compest;
    global forcest;
    global sensor;
    obj.x = 1;
    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
    while (obj.ct < 1745)
        if (obj.verifcarga() == 3) && ((obj.verifpressao() == 1)||(obj.verifpressao() == 2))
            if (compest(8)==0)||(compest(9)==0)||(compest(10)==0)||(compest(11)==0)
                for y=8:11
                    if (compest(y) == 0) && (y ~= ind_forc)
                        compest(y) = 1;
                        sensor = sensor + obj.varcomp(y);
                        obj.cargacomp = (obj.vcomp * compest);
                        obj.cargaforc = (obj.vforc * forcest);
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct;
                        obj.estcomp = compest
                        break;
                    end
                end
            else
                if compest(12) ~= 1
                    compest(12)= 1;
                    sensor = sensor + obj.varcomp(7);
                    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                    obj.sc = 1780 - obj.ct
                    obj.estcomp = compest
                    obj.sp = sensor
                else
                    for r = 1:13
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct;
                        if (forcest(r)== 0)&&(obj.sc > obj.vforc(r))
                            forcest(r) = 1;
                            sensor = sensor + obj.varforc(r);
                            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                            obj.sc = 1780 - obj.ct
                            obj.estforc = forcest
                            obj.sp = sensor
                            break;
```

```
                end
            end
        end
    end

elseif (obj.verifcarga() == 2) && ((obj.verifpressao() == 1)||(obj.verifpressao() ==
2))
    if (compest(12) == 0)
        compest(12)= 1;
        sensor = sensor + obj.varcomp(12);
        obj.cargacomp = (obj.vcomp * compest);
        obj.cargaforc = (obj.vforc * forcest);
        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
        obj.sc = 1780 - obj.ct;
        obj.estcomp = compest
    else
        for r = 1:13
            if (forcest(r)== 0)&& (r ~= ind_forc)
                forcest(r) = 1
                sensor = sensor + obj.varforc(r);
                obj.cargacomp = (obj.vcomp * compest);
                obj.cargaforc = (obj.vforc * forcest);
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct;
                obj.estforc = forcest
                break;
            else
                for g=8:11
                    if (compest(g)==0)
                        compest(g) == 1;
                        sensor = sensor + obj.varcomp(g);
                        obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                        obj.sc = 1780 - obj.ct;
                        obj.estforc = forcest;
                        obj.sp = sensor;
                        break;
                    end
                end
            end
        end
    end

elseif (obj.verifcarga() ~= 1) && ((obj.verifpressao() == 3) || (obj.verifpressao() ==
4)) %uso de forçadores
    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
    obj.sc = 1780 -obj.ct;
    p=0;
    for a = 1:12
        if (forcest(a) == 1)
```

```
            p = p+1
          end
        end

        if p >= 2
          for z = 1:13
            if (forcest(z)== 0) &&(z~=ind_forc)
              forcest(z) = 1;
              sensor = sensor + obj.varforc(z);
              obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
              obj.sc = 1780 - obj.ct;
              obj.estforc = forcest
              obj.sp = sensor
              break;
            end
          end
        else
          if compest(12)==0
            compest(12) = 1;
            sensor = sensor + obj.varcomp(12);
            obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
            obj.sc = 1780 - obj.ct;
            obj.estforc = forcest;
            obj.sp = sensor;
          else
            for g=8:11
              if (compest(g)==0)
                compest(g) == 1;
                sensor = sensor + obj.varcomp(g);
                obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                obj.sc = 1780 - obj.ct;
                obj.estforc = forcest;
                obj.sp = sensor;
                break;
              end
            end
          end
        end

      elseif obj.verifcarga() == 1
        break;
      end
      obj.x = obj.x + 1;

    end
end

function b = liga(obj, ind_forc,estado)
    tic
    obj.x = 1;
```

```
global sensor;
global compest;
global forcest;
if forcest(ind_forc) == estado
   sensor
   compest
   forcest
else
   forcest(ind_forc) = estado
   obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest)
   obj.sc = 1780 - obj.ct;
   if (estado == 0)
      sensor = sensor - (obj.varforc(ind_forc))
      obj.sp = sensor
   else
      sensor = sensor + (obj.varforc(ind_forc))
      obj.sp = sensor
   end

   if (obj.ct <= 1745)
      obj.operador(ind_forc)
      forcest
      compest
   elseif (obj.ct >= 1800)
      h = 0;
      while (obj.ct >= 1800)
         for k = 1:13
            if forcest(k)== 1
               h = h + 1;
            end
         end
         if h > 2
            for z = 1:13
               if (forcest(z)== 1)&&(z ~= ind_forc)&&((sensor - obj.varforc(z))> 7.5)
                  forcest(z) = 0;
                  sensor = sensor - obj.varforc(z);
                  obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                  obj.sc = 1780 - obj.ct;
                  obj.estforc = forcest
                  forcest
                  obj.sp = sensor
                  obj.x = obj.x + 1;
                  break;
               end
            end
         else
            for l = 12:2
               if (compest(l)== 1)&&((sensor - obj.varcomp(l))> 7.5)
                  compest(l) = 0;
                  sensor = sensor - obj.varcomp(l);
```

```
                    obj.ct = (obj.vcomp * compest) + (obj.vforc * forcest);
                    obj.sc = 1780 - obj.ct;
                    obj.estcomp = compest
                    compest
                    obj.sp = sensor
                    obj.x = obj.x + 1;
                    break;
                  end
                end

              end
            end

          elseif (obj.ct > 1745) && (obj.ct < 1800)
            compest
            forcest
            obj.x = obj.x + 1;
          end
        b = toc
      end
      end
    end
end
```