UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO ACADÊMICO EM ENGENHARIA ELÉTRICA

STÉPHANIE ALENCAR BRAGA DOS SANTOS

UTILIZAÇÃO DA META-HEURÍSTICA PSO PARA OTIMIZAÇÃO MULTIOBJETIVO DE UM *SMART HOME CONTROLLER*

FORTALEZA
2019

STÉPHANIE ALENCAR BRAGA DOS SANTOS

UTILIZAÇÃO DA META-HEURÍSTICA PSO PARA OTIMIZAÇÃO MULTIOBJETIVO DE UM *SMART HOME CONTROLLER*

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências e Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Engenharia Elétrica. Área de Concentração: Sistemas Elétricos

Orientador: Prof. Dr. Giovanni Cordeiro Barroso

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S238u Santos, Stephanie Alencar Braga dos.
Utilização da meta-heurística PSO para otimização multiobjetivo de um SMART HOME CONTROLLER / Stephanie Alencar Braga dos Santos. – 2019.
111 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2019.
Orientação: Prof. Dr. Giovanni Cordeiro Barroso.

1. Smart Home. 2. Eficiência Energética. 3. Otimização. 4. Meta-Heurística. 5. Resposta a Demanda. I. Título.

CDD 621.3

STÉPHANIE ALENCAR BRAGA DOS SANTOS

UTILIZAÇÃO DA META-HEURÍSTICA PSO PARA OTIMIZAÇÃO MULTIOBJETIVO DE UM *SMART HOME CONTROLLER*

Dissertação apresentada ao Curso de Mestrado Acadêmico em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências e Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Engenharia Engenharia Elétrica. Área de Concentração: Sistemas Elétricos

Aprovada em: 11 de fevereiro de 2019

BANCA EXAMINADORA

Prof. Dr. Giovanni Cordeiro Barroso (Orientador)
Universidade Federal do Ceará (UFC)

Prof ª.Ph.D. Ruth Pastora Saraiva Leão
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Luiz Marcelo Antunes
Universidade Federal do Ceará (UFC)

Prof. Dr. Bruno de Athayde Prata
Universidade Federal do Ceará (UFC)

Prof. Dr. José Roberto Bezerra
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

À minha família, por todo auxílio afetivo. Meu esposo, o meu maior exemplo de superação e determinação. Meus pais, por sempre me incentivarem, apoiarem e acreditarem que eu era capaz. Meus irmãos e sobrinhos, a quem tenho tanto afeto.

AGRADECIMENTOS

Ao meu Deus, por ter me dado a capacidade intelectual de aprender e por me sustentar nos momentos em que pensei que não iria conseguir.

Ao Prof. Dr. Giovanni Cordeiro Barroso por sua orientação, compreensão e objetividade.

À minha família: meu esposo, por sempre me apoiar e compreender nas fases de dedicação. Aos meus pais, que sempre me apoiam e vibram a cada conquista acadêmica. Minha tia Doninha, quem me serviu de grande apoio na minha trajetória acadêmica.

Ao Prof. Dr Bruno Prata pelas sábias contribuições.

Aos alunos colegas do grupo de Otimização pelos conhecimentos compartilhados através de encontros e discussões.

Agradeço também a todos os professores, não só da pós-graduação, mas também da graduação em Engenharia de Telecomunicações, por terem feito muito mais que repassar seus conhecimentos.

E à Fundação Cearense de Apoio ao Desenvolvimento (Funcap) pelo financiamento da pesquisa de mestrado via bolsa de estudos.

"E tudo quanto fizerdes, fazei-o de todo o coração, como para o Senhor e não para homens."

(Cl 3.23)

RESUMO

A eficiência energética é uma das principais preocupações atuais de empresas, governos, investidores e outros participantes do mercado de energia nas últimas décadas. Dessa forma, buscando atingir os objetivos de fornecimento eficiente e sustentável de energia, redução de emissão de gases, economia de energia e alcançar níveis satisfatórios de segurança e qualidade no fornecimento de energia, as tecnologias de *Smart Grid* (SG) estão atraindo uma atenção crescente devido a sua capacidade inerente de alcançar um sistema de gestão de energia sustentável. Partindo desse contexto, esta dissertação está no cenário de resposta à demanda baseada em preço, em que, de acordo com o preço anunciado, os consumidores residenciais podem regular as operações de seus aparelhos. A satisfação do usuário é um dos aspectos importantes que devem ser considerados juntamente com a redução de conta de energia elétrica. Esta dissertação apresenta um *Smart Home Controller* (SHC), que utiliza uma técnica de otimização meta-heurística baseada em população. A proposta apresenta a formulação matemática do problema através de uma função multiobjetivo que busca alcançar um *tradeoff* entre o custo total do consumo de energia elétrica e a otimização do nível de conforto do usuário, além de garantir que o consumo total não ultrapasse a demanda máxima considerada. A solução para o problema de otimização descrito é realizada através da implementação do algoritmo *Particle Swarm Optimization* (PSO), fazendo uso de duas modalidades tarifárias diferentes, a tarifa com o valor fixo e a Tarifa Branca, a tarifa do tipo *Time Of Use* utilizada no Brasil para consumidores residenciais. A implementação e os resultados das simulações mostram que o SHC proposto pode prover uma redução nos custos de energia elétrica enquanto considera o nível de conforto do usuário em níveis satisfatórios.

Palavras-chave: *Smart Home*. Eficiência Energética. Otimização. Meta-Heurística. Conforto. Resposta à Demanda

ABSTRACT

Energy efficiency is a top concern for companies, governments, investors and other energy market participants in recent decades. Thus, in pursuit of the objectives of efficient and sustainable energy supply, reduction of gas emissions, energy savings and achieving satisfactory levels of safety and quality in energy supply, Smart Grid technologies are attracting increasing attention due to its inherent ability to achieve a sustainable energy management system. From this context, this dissertation is in price-based demand response scenario, in which, by to advertised price, residential consumers can regulate the operations of their handsets. User satisfaction is one of the most relevant aspects that should be considered along with energy bill reduction power. This dissertation presents a Smart Home Controller (SHC), which uses a population-based metaheuristic optimization technique. The proposal presents the mathematical formulation of the problem through a multiobjective function that seeks to achieve a tradeoff between the total cost of electricity consumption and the optimization of level of comfort of the user, besides ensuring that the total consumption does not exceed the considered maximum demand. The solution to the described optimization problem is performed through the implementation of the Particle Swarm Optimization algorithm (PSO), making use of two tariffs, flat tariff and the White Tariff, the Time Of Use tariff used in Brazil to residential users. The implementation and simulations results show that SHC can provide a reduction in electricity costs while considering satisfactory levels to user comfort offered.

Keywords: Smart Home. Energy Efficiency. Optimization. Metaheuristic. Comfort. Demand Response

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ACIW	*Adaptive Chaotic Inertia Weight*
AEMO	Algoritmos Evolucionários Multiobjetivo
AIW	*Adaptive Inertia Weight*
BPSO	*Binary Particle Swarm Optimization*
CA	Cargas Agendáveis
CNA	Cargas Não Agendáveis
CPP	*Critical Peak Pricing*
DAP	*Day Ahead Price*
DIW	*Dynamic Inertia Weight*
DR	*Demand Response*
DSM	*Demand Side Management*
EIW	*Exponential Inertia Weight*
GA	*Genetic Algorithm*
GUI	*Guide User Interface*
IDE	*Integrated Development Environment*
LDIW	*linear decreasing inertia weight*
MKP	*Multiple Knapsack Problems*
NSPSO	Nondominated Sorting Particle Swarm Optimizer
PAR	*Peak to Average Ratio*
POM	problemas de otimização multiobjetivo
POO	Programação Orientada a Objetos
PSO	*Particle Swarm Optimization*
RTP	*Real Time Pricing*
SG	*Smart Grid*
SH	*Smart Homes*
SHC	*Smart Home Controller*
SSM	*Supply Side Management*
TLBO	*Teacher Learning Based Optimization*
TLGO	*Teacher Learning Genetic Optimization*
ToU	*Time Of Use*

LISTA DE SÍMBOLOS

M	Número de cargas planejáveis
N	Número total de amostras
$\bar{P_m}$	Vetor da potência média da m-ésima carga
$\hat{P_m}$	Vetor da potência de pico da m-ésima carga
N_m	Duração, em número de amostras, da m-ésima carga
I_{Sm}	Amostra no horário mínimo de início da m-ésima carga
I_{Em}	Amostra no horário máximo de término da m-ésima carga
I_{Bm}	Amostra associada com o melhor horário de início da m-ésima carga
I_{Cm}	Hora de início agendada para m-ésima carga
C_{Lm}	Nível de conforto da m-ésima carga
P_k	Limite de pico no k-ésimo instante de tempo
C	Vetor do custo do consumo de energia elétrica no período
T_s	Taxa de Amostragem

SUMÁRIO

1 INTRODUÇÃO

A eficiência energética é uma das principais preocupações atuais de empresas, governos, investidores e outros participantes do mercado de energia nas últimas décadas. Na busca de atingir os objetivos de fornecimento eficiente e sustentável de energia, redução de emissão de gases, economia de energia e alcançar níveis satisfatórios de segurança e qualidade no fornecimento de energia, as tecnologias de *Smart Grid* (SG) estão atraindo atenção crescente devido a sua capacidade inerente de alcançar um sistema de gestão de energia sustentável. A principal contribuição de SG à rede convencional é prover fluxo bidirecional de energia e sinais de comunicação. A estrutura de comunicação e controle permite a SG reagir às mudanças que ocorrem em qualquer parte da geração, transmissão, distribuição e consumo.

Estas questões energéticas motivam os pesquisadores e, ao longo dos anos, a investigação centrou-se em diferentes estratégias tecnológicas para responder à crescente procura de fonte de alimentação de qualidade e confiável. A conservação e gerenciamento de energia elétrica é proeminente destas pesquisas. O gerenciamento de energia pode ser classificado em duas categorias: Gerenciamento pelo lado da oferta, *Supply Side Management* (SSM) e gerenciamento pelo lado da demanda, *Demand Side Management* (DSM) (OGUNJUYIGBE *et al.*, 2017).

A primeira, SSM, garante a eficiência na geração, transmissão e distribuição de eletricidade, além de fornecer energia a um custo econômico mínimo. Já DSM visa o planejamento e atividades de monitoramento. Normalmente, o objetivo do DSM é incentivar o consumidor a usar menos energia durante os horários de pico. Devido aos benefícios relacionados ao custo, ela foi adotada em grande escala pelos consumidores para a gestão da energia residencial.

Existem muitas técnicas de DSM, como controle de equipamentos de uso final, técnicas de preenchimento de vale e de recorte de pico, entre outras (GELLINGS; SAMOTYJ, 2013). A resposta à demanda, *Demand Response* (DR), é um componente do DSM que incentiva aos consumidores modificarem seu padrão de consumo de energia e mudarem suas cargas do horário de pico para horas fora de pico.

Os programas de DR podem ser classificados em dois grupos: baseados em incentivos e baseados em preços. Nesse segundo grupo, alguns tipos de tarifas são oferecidas diretamente aos clientes de varejo com o objetivo de promover a resposta à demanda do

cliente baseada em sinais de preço. Incluem as tarifas preço de tempo de uso, *Time Of Use* (ToU); preço de pico crítico, *Critical Peak Pricing* (CPP), e preço em tempo real, *Real Time Pricing* (RTP).

No Brasil, a tarifa do tipo ToU é a Tarifa Branca, que determina três postos tarifários distintos durante dias úteis para clientes residenciais: horário de fora-ponta, horário intermediário e horário de ponta; e começou a vigorar desde janeiro de 2018. Porém, segundo o Jornal do Comércio, o número de unidades consumidoras que optaram pela modalidade correspondeu a menos de 1% do potencial. Um dos fatores apontados é a falta de conhecimento suficiente para decidir, pois para a maioria a conta de luz ainda é um grande mistério. Neste caso, existe o receio de que a adesão tenha o efeito de aumentar a conta de energia, já que esse é um risco real.

Ainda segundo o jornal, um levantamento feito pela ANEEL, a redução máxima chega a 15% no caso da Enel São Paulo; 12% na Light, do Rio; 13% na CEB, de Brasília; e 17% na Celpa, do Pará. A autarquia alerta que essa seria a economia máxima, se nada for consumido na ponta. Mas a diminuição real depende de quanto o consumo pode ser deslocado do horário de ponta e do horário intermediário.

Esta dissertação está inserida no cenário de DR baseado em preço, em que, de acordo com o preço anunciado, os consumidores residenciais podem regular as operações dos aparelhos. Dessa forma, além da economia financeira, outros aspectos devem ser considerados, como problemas operacionais, balanceamento de carga e satisfação do usuário. Nesse contexto, esta dissertação se propõe a fazer o agendamento das cargas em uma residência ao longo do dia, de forma que considere a economia e a satisfação do usuário, solucionando o problema que o usuário encontra em não saber alocar as cargas, correndo o risco de um aumento da conta de energia, ao invés de diminuição, ao aderir a Tarifa Branca.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver uma solução computacional para o problema de otimização multiobjetivo de um *Smart Home Controller* - SHC, utilizando a técnica *Particle Swarm Optimization* - PSO. O SHC faz o agendamento de um conjunto de cargas em uma residência, buscando minimizar o custo financeiro resultante, levando em consideração o nível de conforto, definido pelo usuário, de cada uma das cargas.

Para atingir o objetivo geral proposto, tem-se os seguintes objetivos específicos:

- Realizar um levantamento do estado da arte sobre os modelos de *Smart Home Controller* (SHC) e suas respectivas técnicas de solução; algoritmo *Particle Swarm Optimization* (PSO), principalmente as estratégias multiobjetivo.
- Propor um modelo matemático de SHC baseado nos modelos apresentados na revisão de estado da arte.
- Validar o modelo de SHC: solucionar o problema de otimização, usando a técnica PSO, e realizar simulações em diferentes cenários.
- Realizar análises comparativas dos resultados alcançados.
- Embarcar a solução: pretende-se implementar a solução em uma raspberry py, que acionará as cargas de acordo com o agendamento realizado. O usuário também poderá configurar suas preferências, além de acompanhar o agendamento das cargas.

1.2 Publicação

O artigo descrito nesta dissertação foi publicado e apresentado no Congresso Brasileiro de Automática (CBA 2018), João Pessoa - Paraíba, em setembro de 2018 com o título "*Smart Home Controller*: Otimização Multi-Objetivo utilizando a meta-heurística PSO ". O artigo pode ser encontrado nos apêndices desta dissertação.

1.3 Organização

Os demais capítulos desta dissertação estão estruturados como segue: no Capítulo 2 é apresentada a fundamentação teórica necessária para a melhor compreensão deste trabalho.

No Capítulo 3 é apresentada a revisão de estado da arte tanto dos sistemas de SHC existentes como das técnicas utilizadas para solucionar o problema de otimização do sistema.

No Capítulo 4 é descrito como o trabalho foi organizado, o modelo do SHC, qual o método utilizado para elaborar e solucioná-lo.

No Capítulo 5 são apresentados os resultados obtidos através das simulações computacionais realizadas nos diferentes cenários e casos. São apresentadas também as análises e discussões dos resultados apresentados.

Finalmente, no Capítulo 6 são apresentadas as conclusões sobre as implicações dos resultados obtidos e das contribuições deste trabalho e, no final, é apresentado um cronograma dos trabalhos futuros programados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os conceitos fundamentais de gerenciamento pelo lado da demanda e modelos de tarifação, que compõem o contexto em que esse trabalho está inserido. Também são abordados os conceitos e fundamentos do algoritmo PSO, método utilizado para solução do problema de otimização.

2.1 Gerenciamento pelo lado da Demanda

O conceito de Gerenciamento pelo Lado da Demanda, do inglês, *Demand Side Management*, foi proposto pela primeira vez por Gellings (1985): é o planejamento, implementação e monitoramento das atividades do fornecedor de energia que são projetadas para influenciar o uso da eletricidade pelo lado do cliente, de maneira a produzir as mudanças desejadas no perfil de carga do fornecedor.

DSM é uma função importante no gerenciamento de energia em SG que fornece apoio a funcionalidades em diversas áreas, como controle do mercado de eletricidade, gestão, construção de infraestrutura e gerenciamento de recursos energéticos descentralizados. Controlar e influenciar a demanda de energia pode reduzir a demanda de carga de pico, reformular o perfil de demanda e aumentar o sustentabilidade da rede, reduzindo os custos globais e os níveis de emissão de carbono (ESTHER; KUMAR, 2016).

Os autores Esther e Kumar (2016) apresentam os componentes necessários para um *framework DSM*, que são projetados para gerenciar de forma ótima os recursos de eletricidade.

- Geração local: plantas energéticas locais geram eletricidade que pode ser usada tanto localmente como injetada na rede.
- *Smart Devices*: dispositivos eletrônicos (*appliances*) que podem ser monitorados e disponibilizam dados do consumo de energia, podendo ser controlados remotamente.
- Sensores: usados para medir e monitorar os dados dentro da residência como: movimento, temperatura e iluminação.
- Sistemas de armazenamento de energia: dispositivos de armazenamento que permitem a flexibilidade do sistema DSM.
- Unidade de gerenciamento de energia: permite a troca de informações entre os outros elementos do sistema e gerencia os recursos do usuário baseado em mecanismos

inteligentes.

Na literatura, os principais programas e atividades DSM que vêm sendo implementados são: 1-Controle de Carga, 2-Eficiência Energética e 3-Resposta à Demanda.

2.1.1 Controle de Carga

Consiste em técnicas de controle que visam influenciar o modo como o consumidor usa a energia elétrica, de forma a produzir alterações nos diagramas de carga. No conceito de DSM, existem seis tipos de técnicas: deslocamento de carga (*Load Shifting*), recorte de pico (*Peak Clipping*), preenchimento de vales (*Valey Fillings*), eficiência energética (*Energy Efficiency*), eletrificação (*Eletrification*) e carga flexível (*Flexible Load Shape*) (GELLINGS; SAMOTYJ, 2013).

2.1.2 Eficiência Energética

A eficiência energética é um conceito amplo que inclui técnicas, equipamentos e construções para alcançar um ganho econômico sem aumentar o consumo de energia, mantendo ao mesmo tempo, o conforto e níveis produtivos. Está relacionado com a quantidade de energia utilizada para realizar determinada atividade e a quantidade de energia disponibilizada para sua realização. Na literatura há muitas ações e técnicas empregadas para melhorar a eficiência energética, como substituição de equipamentos, melhor dimensionamento das instalações elétricas, conscientização do consumo, entre outros.

2.1.3 Resposta à Demanda

Refere-se a mecanismos para gerenciar a demanda de clientes em resposta às condições de fornecimento. Por exemplo, reduzir o consumo de eletricidade do cliente em momentos críticos ou em resposta aos preços de mercado. A resposta à demanda pode ser, em geral, de dois tipos:

- Baseada em incentivos: nesse tipo de programa, é oferecido algum tipo de pagamento/incentivo aos clientes para reduzir o uso de eletricidade durante os períodos de necessidade do sistema ou estresse. Esse tipo de programa é acionado por razões de confiabilidade ou econômicas. Inclui controle direto de carga, programas de

oferta/recompra de demanda, programas de resposta à demanda de emergências, programas de capacidade de mercado e programas de serviços de auxiliares.

- Tarifas baseadas no tempo: nesse tipo de programa, alguns tipos de tarifas são oferecidas diretamente aos clientes de varejo com o objetivo de promover a resposta à demanda do cliente baseada em sinais de preço. Incluem as tarifas preço de tempo de uso, do inglês ToU, preço de pico crítico, do inglês CPP e preço em tempo real, do inglês RTP.

2.2 Tarifação de Energia Elétrica

O serviço de energia elétrica é essencial no dia a dia da sociedade, seja nas residências ou nos diversos segmentos da economia. Para o uso desse bem é necessária a aplicação de tarifas que remunerem o serviço de forma adequada, que viabilize a estrutura para manter o serviço com qualidade e que crie incentivos para eficiência.

Mesmo com os avanços em SG, a maioria dos clientes, principalmente residenciais e comerciais de pequeno porte, possuem medidores de uso de massa com um único registro de dados, que simplesmente acumula o uso ao longo do tempo. Dessa forma, esses clientes só podem ser cobrados pela eletricidade que usam de acordo com os seguintes tipos de preços (SMARTGRID.GOV, 2013):

- Taxas fixas: é cobrada a mesma taxa durante todo o período de tempo (por exemplo, ciclo de faturamento de 30 dias).
- Taxas escalonadas: normalmente se cobra um preço diferente com base em blocos de uso (por exemplo, primeiros 500 kWh *versus* próximos 500 kWh) durante um determinado período de tempo.

Essas concepções de tarifação não transmitem a variabilidade ao longo do tempo (por exemplo, hora a hora, dia a dia, estação a estação) no custo para produzir eletricidade. Usando medidores inteligentes, as concessionárias agora são capazes de registrar o consumo de eletricidade com uma frequência maior (por exemplo, a cada 15 minutos), permitindo que os clientes, que anteriormente tinham medidores de uso de massa, sejam introduzidos a novos tipos de programas de preços que reflitam melhor essas diferenças ao longo do tempo no custo para produzir eletricidade (SMARTGRID.GOV, 2013).

As formas de programas de tarifas baseadas em tempo incluem:

- Preço de tempo de uso (ToU) - normalmente se aplica ao uso em grandes blocos

de horas (por exemplo, pico = 6 horas para a tarde da semana de verão; fora de pico = todas as outras horas nos meses de verão) em que o preço de cada período é predeterminado e constante.

- Preços em tempo real (RTP) - as taxas de preços geralmente se aplicam ao uso por hora.
- Preços de Pico Variável (VPP) - um híbrido de tempo de uso e preço em tempo real em que os diferentes períodos de precificação são definidos com antecedência (por exemplo, pico = 6 horas para o dia da semana de verão), mas o preço estabelecido para o período de pico varia de acordo com as condições da concessionária e do mercado.
- Preço máximo crítico (CPP) - quando as concessionárias observam ou antecipam altos preços de mercado no atacado ou condições de emergência do sistema de energia, eles podem chamar eventos críticos durante um período de tempo especificado (por exemplo, 15h - 18h em um dia quente de verão), o preço para eletricidade durante esses períodos de tempo é substancialmente aumentada.
- Descontos de pico críticos (CPR) - quando as concessionárias observam ou antecipam altos preços no mercado atacadista ou condições de emergência do sistema de energia, eles podem chamar eventos críticos durante períodos de tempo pré-especificados (por exemplo, 15h - 18hs tardes de semana de verão), o preço da eletricidade durante esses períodos de tempo permanece o mesmo, mas o cliente é reembolsado em um valor único e predeterminado para qualquer redução no consumo em relação ao que a concessionária considerou que o cliente deveria consumir.

Todas essas formas de programas de taxa baseados no tempo são ativadas pelo investimento e instalação de medidores inteligentes (SMARTGRID.GOV, 2013).

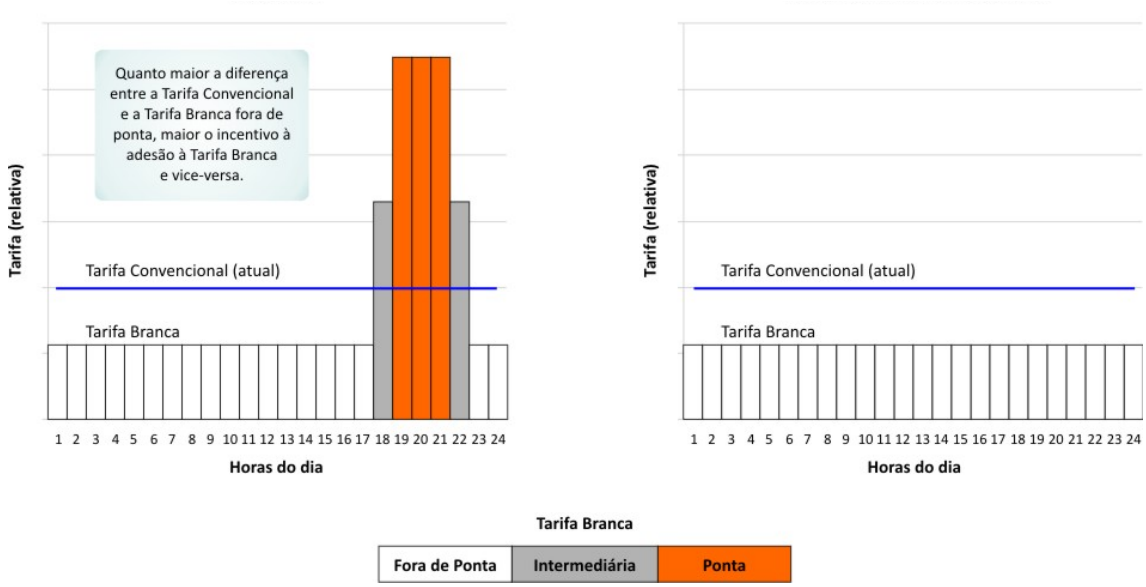
2.2.1 Tarifa Branca

No Brasil, o programa de tarifação do tipo ToU é a Tarifa Branca (ANEEL, 2017), uma nova opção que sinaliza aos consumidores a variação do valor da energia conforme o dia e o horário do consumo. Ela é oferecida para as unidades consumidoras que são atendidas em baixa tensão (127, 220, 380 ou 440 Volts), denominadas de grupo B.

Na Figura 1 é apresentada uma comparação entre as tarifas convencional e branca. A Tarifa Branca determina três postos tarifários distintos durante dias úteis

para clientes residenciais, a saber: horário de fora-ponta, horário intermediário e horário de ponta. Os postos tarifários ponta e intermediário apresentam tarifação superior à tarifa convencional, enquanto o horário de fora-ponta apresenta tarifação inferior à tarifa convencional. Nos finais de semana e feriados a tarifa branca apresenta tarifação inferior à tarifa convencional (lado direito da figura).

Figura 1 – Comparação entre Tarifa Branca e Tarifa Convencional



Fonte: (ANEEL, 2017)

A Tarifa Branca cria condições que incentivam alguns consumidores a deslocarem o consumo dos períodos de ponta para aqueles em que a rede de distribuição de energia elétrica tem capacidade ociosa. Quanto mais o consumidor deslocar seu consumo para o período fora de ponta e quanto maior for a diferença entre essas duas tarifas, maiores serão os benefícios da Tarifa Branca. Porém, se o consumo for maior nos períodos de ponta e intermediário e não houver possibilidade de deslocamento do uso dessa energia elétrica para o período fora de ponta, a Tarifa Branca não é recomendada (ANEEL, 2017).

2.3 *Particle Swarm Optimization*-PSO

O algoritmo Otimização por Enxame de Partículas, do inglês *Particle Swarm Optimization* (PSO) é um método de otimização baseado em população que encontra a solução ideal usando uma população de partículas, proposto por Kennedy e Eberhart (1995).

Está no conjunto de algoritmos que se inspiram no comportamento de colônias

de insetos sociais e outras sociedades. Assim como os enxames naturais, PSO baseia-se no conceito de cooperação, possui as características de individualidade e sociabilidade e possui habilidades para: trocar informações entre vizinhos, memorizar uma posição anterior e utilizar informações para a tomada de decisões.

Os elementos do algoritmo são:

A : população de partículas

$\vec{x_i}$: vetor posição da partícula i no espaço de soluções

f : função de avaliação (*fitness*)

$\vec{v_i}$: vetor velocidade da partícula i

$\vec{pbest_i}$: vetor da melhor posição pessoal da partícula i que corresponde à posição no espaço de busca onde a partícula apresenta o melhor valor da função avaliação f

$\vec{gbest}$: vetor da melhor posição global, representa a posição que produz o nelhor valor entre todos os $\vec{pbest_i}$.

As Equações 2.1 e o 2.2 definem como o $\vec{pbest_i}$ e $\vec{gbest}$ são atualizados no tempo t, respectivamente. É assumido que o enxame possui n partículas, em um problema de minimização da função f.

$$\vec{pbest_i}(t+1) = \begin{cases} \vec{pbest_i}(t) \ se \ f(\vec{pbest_i}) \leq f(\vec{x_i}(t+1))) \\ \vec{x_i}(t+1) \ se \ f(\vec{pbest_i}) > f(\vec{x_i}(t+1))) \end{cases} \tag{2.1}$$

$$\vec{gbest}(t+1) = min\{f(\vec{pbest}), f(\vec{gbest})\} \tag{2.2}$$

$$\vec{pbest} \in \{\vec{pbest_0}, \vec{pbest_1}, ..., \vec{pbest_n}\}$$

2.3.1 Atualização da Velocidade e Posição

A partícula i é definida por três vetores d-dimensionais: posição $\vec{x_i}$, velocidade $\vec{v_i}$ e melhor posição $\vec{pbest}$, todos já definidos acima. Três termos definem a velocidade $\vec{v}$ para uma partícula i.

1. Termo de Inércia: é a velocidade anterior do enxame, influencia a partícula a mover-se na mesma direção em que está.

2. Termo Cognitivo: expressa a experiência individual da partícula de onde a solução está, influencia a partícula a mover-se a uma posição melhor que a atual.
3. Termo Social: representa a experiência do enxame, influencia a partícula a seguir a direção de seus melhores vizinhos.

A velocidade e posição atual de cada partícula i são atualizadas como apresentado nas Equações 2.3 e 2.4, respectivamente:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1(\vec{pbest}_i - \vec{x}_i) + c_2 r_2(\vec{gbest} - \vec{x}_i) \tag{2.3}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{2.4}$$

em que:
c_1 e c_2 são duas constantes positivas, que representam os parâmetros cognitivo e social, respectivamente;
r_1 e r_2 são dois números aleatórios dentro do intervalo [0, 1];
w é o peso de inércia.

As três partes da Equação 2.3, em conjunto, determinam a capacidade do espaço de pesquisa (ESMIN, 2007).

2.3.2 Fluxograma

No fluxograma da Figura 2 estão as etapas do algoritmo básico PSO.

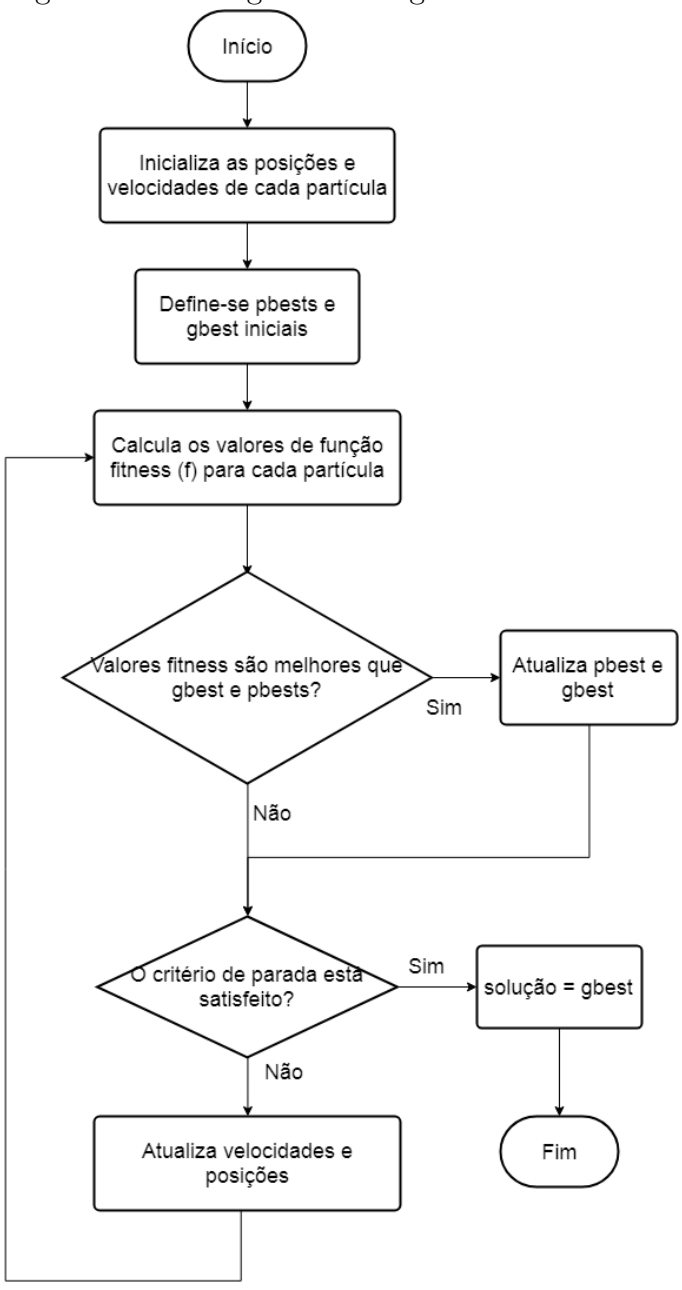
2.4 Conclusões

Neste capítulo primeiramente foram abordados os conceitos fundamentais de gerenciamento pelo lado da demanda: definições e os principais programas e atividades que vêm sendo implantados.

Também foi feita uma abordagem da tarifação de energia elétrica, apresentando os principais tipos utilizados no mundo, principalmente as tarifas baseadas em tempo. Em seguida, foi apresentada a Tarifa Branca, que é a tarifa baseada no tempo, utilizada no Brasil e adotada nessa dissertação.

Por fim, foi feita uma explanação da técnica PSO, sendo abordados os conceitos, fundamentos e características, bem como um fluxograma do algoritmo.

Figura 2 – Fluxograma do algoritmo básico PSO



Fonte: A autora

3 ESTADO DA ARTE

Neste capítulo é apresentada a revisão do estado da arte relevante para este trabalho: modelos de controladores de cargas inteligentes, do inglês *Smart Home Controller* (SHC) encontrados na literatura, as técnicas utilizadas para solucionar os problemas de otimização e quais são os avanços na técnica de meta-heurística PSO, já que foi a técnica escolhida para ser utiliza neste trabalho. A revisão do estado da arte neste trabalho é dividida em duas vertentes:

1. SHC: Modelos e Técnicas
2. PSO: Variações e Melhorias

Ao fim do capítulo está uma Seção que contém uma explanação das referências que contribuíram diretamente como base para esse trabalho.

3.1 *Smart Home Controllers*

Junto com o conceito de DSM e resposta à demanda, surgiram várias técnicas e algoritmos para solucionar problemas de redução de custos, de como usar a energia de forma eficiente, utilizando a infra-estrutura proposta em SG. Nesse contexto, há muitas pesquisas para agendamento ótimo de cargas/*appliances*.

Os primeiros estudos sobre modelos de sistemas para gerenciamento de energia no domínio residencial datam da década de 90. Wacks (1991) apresenta um dos primeiros estudos no ramo da automação residencial, conceituando gerenciamento de energia e controle de carga em uma residência. Wacks (1991) conceituou um controlador inteligente:

> Um controlador inteligente é parte de um sistema de gerenciamento de energia que tem a responsabilidade de balancear os custos com as preferências do usuário, otimizando os parâmetros de consumo e conveniência.

Os autores Rahim *et al.* (2016) trazem um bom relato dos trabalhos desenvolvidos na área de SHC, principalmente aqueles baseados em meta-heurística e *Multiple Knapsack Problems* (MKP). Javaid *et al.* (2016) também fazem um bom resumo das pesquisas feitas, dividindo em duas vertentes: a primeira relata os desafios na minimização dos custos, pico e desconforto. A segunda vertente discute as técnicas de agendamento utilizadas no gerenciamento de energia em *Smart Homes* (SH).

Nesta seção são apresentados os modelos de SHC e suas técnicas de solução. Ao fim da Seção é apresentado um quadro comparativo, resumindo as referências citadas e

suas respectivas características.

3.1.1 Modelos e Técnicas

Os autores Gudi *et al.* (2010) apresentam uma ferramenta que simula um ambiente residencial e provê ao usuário uma análise em tempo real do agendamento dos *appliances*. O modelo matemático do SHC proposto não apresenta uma função para avaliar o nível de conforto, apenas o custo financeiro da execução das cargas. A ferramenta foi desenvolvida em linguagem *Java* e banco de dados *SQL*, utilizando a técnica *Binary Particle Swarm Optimization* (BPSO) e considera que há um sistema de SG para troca de dados (informações sobre tarifas) entre a concessionária e o usuário. Na *Guide User Interface* (GUI) desenvolvida, o usuário cadastra os *appliances* e também pode selecionar fontes de energia renováveis. Apesar de ter sido desenvolvida como pesquisa para futuros produtos comerciais, a ferramenta teve a função apenas de simulação, não foi embarcada e testada em um ambiente real.

Giorgio e Pimpinella (2012) propõem um SHC, modelado como um problema de otimização inteira que minimiza o consumo de energia elétrica residencial para um determinado conjunto de cargas. O SHC leva em consideração as tarifas correntes e o consumo médio das cargas que se deseja planejar. O modelo geral do trabalho apresentado leva em consideração cargas planejáveis, que são aquelas que o SHC pode mudar seu horário de execução livremente, cargas controláveis, que possuem controle do tipo *on/off* e cargas monitoráveis, que são estimadas estatisticamente com a ajuda de um medidor inteligente de energia elétrica. O modelo SHC proposto não leva em consideração o conforto das cargas e não é capaz de minimizar o consumo de pico do conjunto de cargas, apesar de utilizar uma restrição do consumo de pico para evitar sobrecargas na rede.

Em seu trabalho, BEZERRA FILHO *et al.* (2015) propõe um modelo baseado no modelo de Giorgio e Pimpinella (2012), acrescentando uma função capaz de medir o conforto do usuário, fazendo um *trade-off* entre os dois objetivos. O autor levanta o estado da arte sobre estratégias de otimização de consumo energético e conforto, propõe um modelo de otimização baseado em programação linear inteira que traz melhorias em relação aos modelos estudados, valida o modelo proposto através de simulações, realiza análises comparativas do modelo proposto com os estudados. Apesar de não implementar na prática, o autor propõe uma arquitetura para que tal modelo possa ser implementado

na prática.

O trabalho proposto por Ma *et al.* (2016), também tem o objetivo de alcançar um *tradeoff* entre o desconforto do usuário e o custo com o consumo de energia elétrica, agendando dois tipos de *appliances*: flexíveis na potência, do inglês *power flexible* e flexíveis no tempo, do inglês *time flexible*. O diferencial do trabalho é a modelagem do desconforto utilizando a função de perda de Taguchi (*Taguchi Loss function*), usando um parâmetro de *tradeoff*. O modelo é simulado utilizando a tarifa *Day Ahead Price* (DAP). Apesar dos autores apresentarem e detalharem o modelo de SHC, não citam como fizeram as simulações e não explicam os parâmetros escolhidos para cada carga.

Manzoor *et al.* (2017) apresentam um SHC que segue o mesmo modelo matemático de (MA *et al.*, 2016), também buscando um *tradeoff* entre o desconforto do usuário e o custo. Os autores solucionaram o problema proposto usando três técnicas de otimização já existentes: *Genetic Algorithm* (GA), *Teacher Learning Based Optimization* (TLBO) e programação linear e proposeram uma nova técnica de otimização, baseada em GA e TLBO, o *Teacher Learning Genetic Optimization* (TLGO). As principais contribuições dos autores foram: analisaram o efeito de diferentes esquemas de preços DAP e CPP; analisaram o impacto de usar diferentes tempos de amostragem em relação ao custo, conforto e complexidade do modelo; minimizam *Peak to Average Ratio* (PAR) e potência de pico. Os autores simulam o SHC usando o mesmo cenário utilizado por Ma *et al.* (2016), considerando as mesmas cargas e parâmetros. Após os resultados, os autores concluem que as cargas flexíveis na potência têm um efeito importante no custo da fatura.

O trabalho proposto por Ogunjuyigbe *et al.* (2017) é o modelo que mais se diferencia dos que já foram citados acima, é uma outra abordagem (Orçamento Fixo). O objetivo do trabalho é encontrar um perfil de uso das cargas que garanta o maior conforto (satisfação) ao usuário, dado um orçamento predefinido. Os autores desenvolveram um conceito de satisfação que: é quantificável, ou seja, pode ser avaliada numericamente; é uma variável *fuzzy*, dessa forma não existe apenas 0 (insatisfeito) e 1 (satisfeito), mas para todos os níveis de satisfação existe um valor; é comparável e relacionável. São definidos 2 níveis de relatividade: Baseada no tempo (*Time-based*) e baseada nos dispositivos (*Devices-based*). Com os resultados, os autores mostram que o algoritmo é eficiente, no que se propõe. Os autores não mencionam se implantaram o modelo, apenas simularam.

Os autores Albuquerque *et al.* (2018) apresentaram uma proposta de SHC

formalizada como um problema de programação linear inteira multiobjetivo, buscando minimizar o consumo de energia e maximizar o conforto. Uma função objetiva de conforto é testada em vários cenários tarifários incluindo um com fontes renováveis e micro-geração local fora da rede. Dada a restrição de carga e preferências do usuário, o modelo proposto especifica os melhores horários para ativar eletrodomésticos reais baseados em dados de consumo de energia, utilizando uma função de agregação ponderada. Os cenários propostos mostraram excelentes resultados para economia de energia sem redução no conforto.

Lin e Hu (2018) apresentam um modelo de SHC que considera custo e conforto, além de se preocupar com o pico de demanda. O modelo também permite a inserção de uma geração local de energia, como por exemplo fontes fotovoltaicas ou eólicas. O SHC é simulado usando a técnica PSO. Esse trabalho também se diferencia dos demais pelo fato de considerar a satisfação do usuário baseado em comportamentos passados do usuário, ou seja a intervenção do usuário é mínima. Em uma primeira fase, é estudado o padrão de comportamento dos usuários. Na segunda fase, é aplicado o algoritmo de agendamento das cargas. Os autores implantaram o modelo de SHC usando linguagem R e embarcaram a solução usando uma placa *beagleboard* com o sistema operacional *Linux SO*. O modelo foi testado em uma residência com cinco *applicances*. Durante 30 dias foi executada a fase de treinamento, depois foi feito o agendamento das cargas.

No Quadro 1 estão resumidas as principais informações do estado da arte: referência; técnica utilizada para solucionar o problema de otimização; se o usuário considera ou não o conforto no modelo; se o usuário implementou ou não o modelo e como implementou; vantagens e desvantagens.

Quadro 1 – Resumo do estado da arte

Referência	Técnica	Conforto	Implementou	Vantagens	Desvantagens
(GUDI *et al.*, 2010)	BPSO	Não	Ferramenta WEB em linguagem *Java*, apenas para simulação.	É capaz de receber os dados em tempo real, selecionar *appliances* e fontes de energia. Disponibiliza as informações em tempo real.	Não considera o conforto do usuário. Não foi implantado em ambiente real, é apenas um modelo teórico.
(GIORGIO; PIMPINELLA, 2012)	Prog. Linear	Não	Simulação em Matlab.	É um controlador orientado a eventos. Testam o modelo em diferentes cenários.	Não considera o conforto. Não permite multi-acionamento das cargas.
(BEZERRA FILHO *et al.*, 2015)	Prog. Linear	Sim	Simulação em Matlab.	Considera o conforto, usa parâmetros de *tradeoff* entre custo e conforto	Não permite multi-acionamento das cargas. Utiliza apenas Programação Linear.
(MA *et al.*, 2016)	Prog. Convexa	Sim	Os autores não citam como simularam.	Consideram custo e conforto, tratam o conforto nos doi tipos de *appliances*: flexíveis na potência e no tempo.	Cenário de teste com poucas cargas, não especificam os parâmetros escolhidos. Não mostram como testaram o modelo.
(MANZOOR *et al.*, 2017)	GA, Prog.Linear, TLBO e TLGO	Sim	Apenas Simulação.	Consideram Conforto. Solução do problema com 3 técnicas diferentes, fazem comparação e propõem uma nova técnica. Mostram uma análise dos resultados.	Cenário de teste com poucas cargas, não especificam os parâmetros escolhidos.
(OGUNJUYIGBE *et al.*, 2017)	GA	Sim	Apenas Simulação.	Analisam o conceito de satisfação tanto baseado nas preferências de tempo como de dispositivos(*appliances*).	Consideram apenas um orçamento já prefixado.
(ALBUQUERQUE *et al.*, 2018)	Prog. Linear	Sim	Sim, validaram com um sistema em laboratório com SCADA e PLC	Consideram conforto e fontes de geração local. Testaram em ambiente real.	Não embarcaram o sistema. Utilizam apenas programação linear, tornando a solução inviável para alguns casos.
(LIN; HU, 2018)	PSO	Sim	Implementaram o modelo em linguagem R, em uma *beagleboard* e *Linux S.O.*	Consideram conforto e fontes de geração local. Embarcaram o sistema.	As preferências do usuário são geradas apenas pelo treinamento prévio do sistema. Pelos resultados não conseguem uma boa economia (0.85%).

Fonte: elaborado pela autora.

3.2 PSO: técnicas de melhoramento e Adaptações

Uma explicação sobre o funcionamento do algoritmo PSO já foi apresentada na Seção 2.3. Então, nesta Seção é apresentada uma revisão de estado da arte das técnicas variantes para adaptar e/ou melhorar a eficiência do algoritmo nos problemas de otimização.

Desde que foi proposto por Kennedy e Eberhart (1995), o algoritmo PSO tem sido aplicado em muitas áreas de pesquisas, como problemas de otimização, treinamento de redes neurais artificiais, sistemas de controle *fuzzy*, entre outros. Consequentemente, muitas pesquisas também vêm sendo feitas para aumentar sua eficiência no tratamento de problemas de otimização. Apesar de suas vantagens, como rápida convergência e fácil codificação, alguns autores perceberam uma tendência de convergência prematura em ótimos locais (CLERC; KENNEDY, 2002). Vários trabalhos já foram publicados com novas técnicas e adaptações envolvendo peso de inércia w e/ou parâmetros cognitivo e social c_1 e c_2, bem como as técnicas que buscam explorar as pesquisas locais e globais no espaço de soluções.

Esta Seção é dividida em duas partes: na primeira são apresentadas algumas técnicas de melhoramento baseadas em peso de inércia w e, na segunda parte, são apresentados variantes PSO utilizados para solucionar problemas de otimização multiobjetivo (POM).

3.2.1 Técnicas de Peso de Inércia

Bansal *et al.* (2011) apresentam 15 diferentes estratégias de peso de inércia e as comparam entre si, usando cinco funções objetivo diferentes. Já Arasomwan e Adewumi (2013b) apresentam uma revisão de estado da arte das estratégias de peso de inércia (w). Os autores apresentam um resumo de sete técnicas diferentes, dentre elas *linear decreasing inertia weight* (LDIW), e comparam o desempenho das mesmas, usando seis tipos de funções com diferentes características.

A seguir estão descritas algumas das técnicas mais usadas, encontradas na literatura:

- LDIW: estudos mostram que um decréscimo linear no peso de inércia é capaz de melhorar o desempenho do PSO (SHI; EBERHART, 1998). Sabendo que um valor de

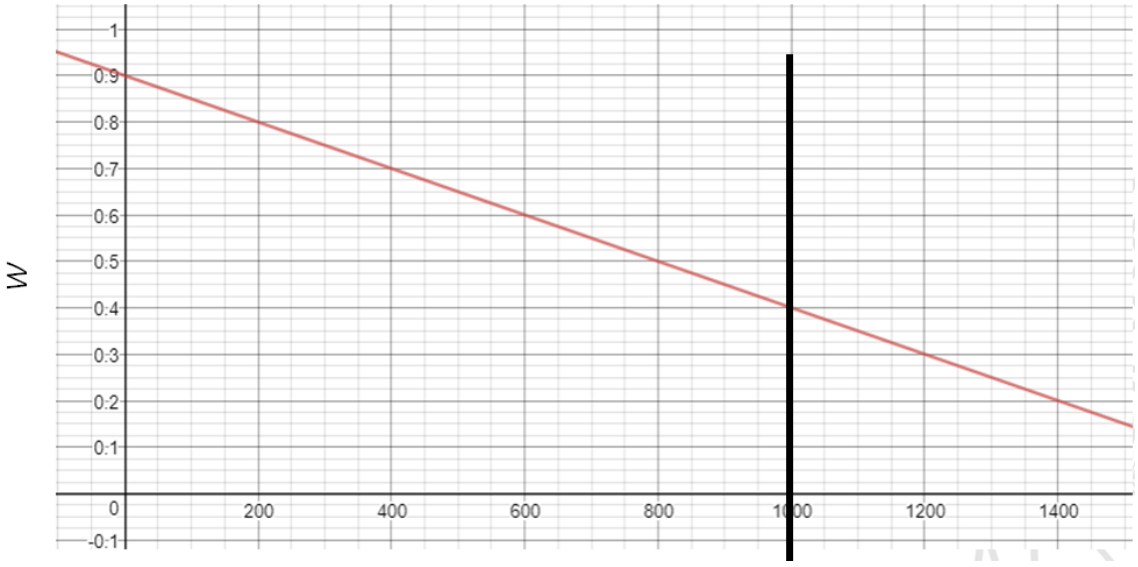
w alto facilita a busca global e que um valor pequeno facilita a busca local, utiliza-se então o valor inicial de $0,9$ e final de $0,4$. A Equação 3.1 representa o peso de inercia na iteração t.

$$w_t = (w_{start} - w_{end})(\frac{T_{max} - t}{T_{max}}) + w_{end} \tag{3.1}$$

em que w_{start} e w_{end} são os pesos de inércia inicial e final, respectivamente,
t é a geração atual,
T_{max} é o número máximo de iterações,
$w_t \in [0,1]$.

Na Figura 3 está ilustrada a variação de w ao longo das iterações, utilizando a técnica LDIW.

Figura 3 – Variação do peso de inércia - LDIW



Fonte: A autora

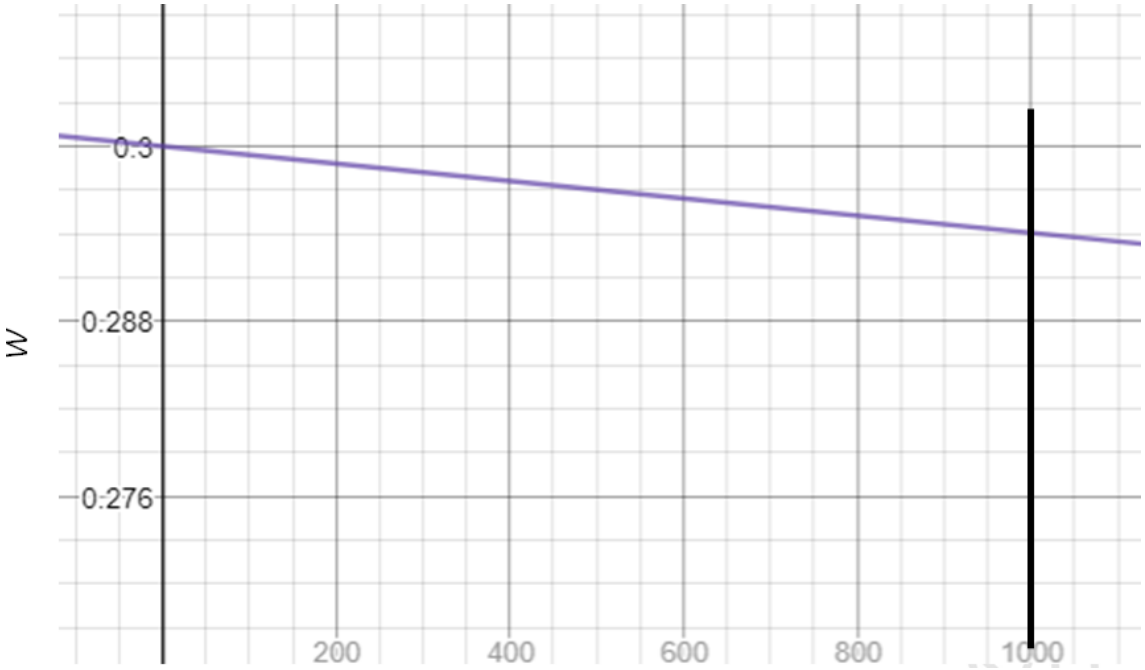
- *Dynamic Inertia Weight* (DIW): nessa técnica, o peso de inércia foi modificado multiplicando-o por uma constante elevada a uma potência dada pela contagem de iteração (JIAO *et al.*, 2008). Um peso de inércia inicial é especificado pelo usuário, então é reduzido de acordo com a Equação 3.2

$$w_t = w_{start}\ u^{-t} \tag{3.2}$$

em que u é uma constante ligeiramente maior que 1 e t é a geração atual. Os parâmetros geralmente empregados são $w_{start} = 0,3$ e $u = 1,00002$.

Na Figura 4 está ilustrada a variação de w ao longo das iterações, utilizando a técnica DIW.

Figura 4 – Variação do peso de inércia - DIW



Fonte: A autora

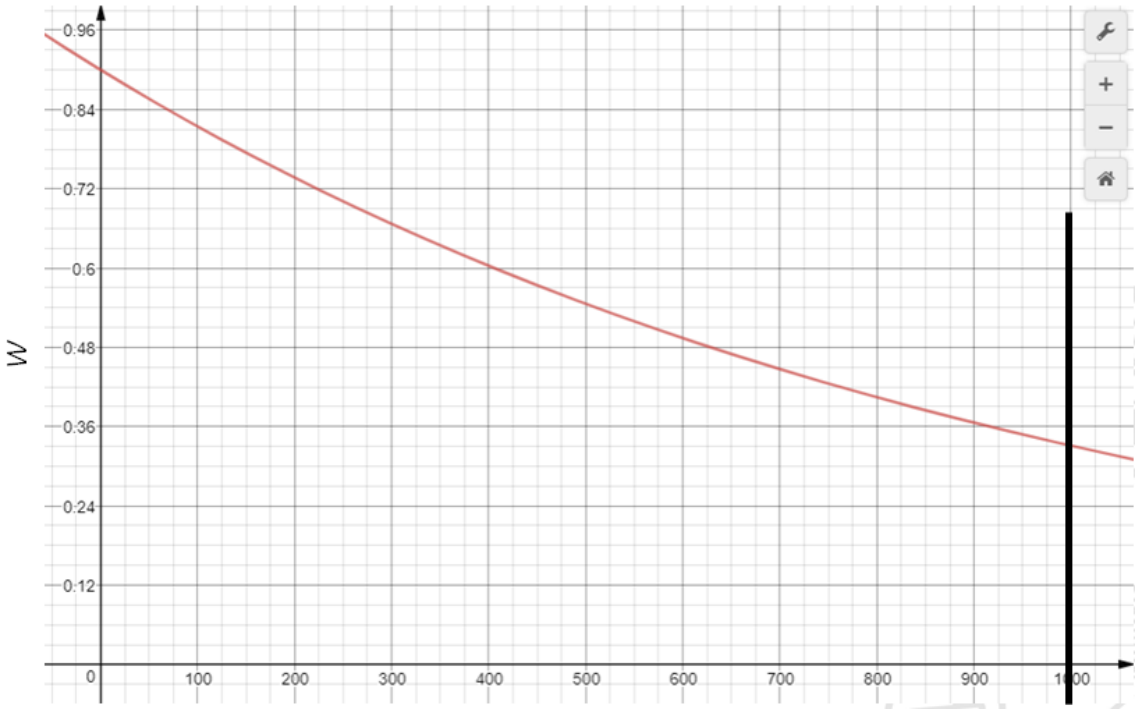
- *Exponential Inertia Weight* (EIW): foi proposto por Ting *et al.* (2012) e calcula o peso de inércia pela Equação 3.3

$$w_t = w_{start} \ \exp[-t/t_{max}] \tag{3.3}$$

em que w_{start} é inicializado com $0,9$.

Na Figura 5 está ilustrada a variação de w ao longo das iterações, utilizando a técnica EIW.

Figura 5 – Variação do peso de inércia - EIW



Fonte: A autora

- *Adaptive Inertia Weight* (AIW): nessa técnica, o peso de inércia é adaptado de acordo com o valor de *fitness* em iterações consecutivas (NICKABADI *et al.*, 2011). O peso de inércia é calculado a partir da Equação 3.4.

$$w_t = (w_{start} - w_{end})\ S + w_{end} \tag{3.4}$$

em que a taxa de sucesso S é dada por:

$$S = \frac{\sum_i \hat{S}_i}{N}\ ,\ \hat{S}_i = \begin{cases} 1, & pbest_t^i < pbest_{t-1}^i \\ 0, & outros\ casos \end{cases} \tag{3.5}$$

N é o número de partículas,

$pbest_t^i$ é o melhor valor de *fitness* na partícula i obtida na iteração t,

w varia de $w_{start} = 1$ a $w_{end} = 0,1$

- *Adaptive Chaotic Inertia Weight* (ACIW): essa técnica também usa função exponencial. O peso de inércia é modificado de acordo com uma sequência caótica e a taxa de sucesso (ARASOMWAN; ADEWUMI, 2013a).

$$w_t = [(w_{start} - w_{end})(\frac{T_{max} - t}{T_{max}}) + w_{end}]\ z \tag{3.6}$$

em que $z = 4S(S-1)$,
a sequência caótica, S, é $\sum[f_t^i - f_{t-1}^i]/N$,
w varia de $w_{start} = 0,9$ a $w_{end} = 0,4$.

3.2.2 Estratégias de PSO multi-objetivo

Na otimização de enxame de partículas multiobjetivo, a avaliação e, consequentemente, a orientação das partículas vêm sendo realizada com base em sua adequação no espaço de soluções. Dessa forma, um processo evolutivo se origina do espaço de soluções. No entanto, o espaço do problema também contém informações úteis que podem melhorar o processo de evolução do algoritmo em problemas de seleção de recursos. O conjunto de arquivos que armazena as melhores soluções em termos de objetivos (a precisão da classificação e o número de recursos) também pode determinar as características importantes (AMOOZEGAR; MINAEI-BIDGOLI, 2018).

O primeiro algoritmo multiobjetivo baseado em PSO foi sugerido por Coello e Lecsuga (2002). No algoritmo, o conceito de dominância de Pareto foi sugerido para determinar as melhores partículas globais e pessoais, e um arquivo foi mantido para salvar as partículas não dominantes e as melhores partículas globais. Embora o algoritmo proposto tenha demonstrado desempenho competitivo na resolução de problemas de otimização multiobjetivo, se comparado com outras técnicas tradicionais de Algoritmos Evolucionários Multiobjetivo (AEMO) como NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) (DEB *et al.*, 2002) e PAES (*Pareto Archived Evolution Strategy*) (KNOWLES; CORNE, 2000), ainda assim tem dificuldades em resolver POM com cenários complicados, por exemplo, aqueles com múltiplos objetivos locais. Para resolver esse problema, os autores Sierra e Coello (2005) propuseram um algoritmo multiobjetivo baseado em PSO melhorado, onde diferentes operadores de mutação foram sugeridos para diferentes subgrupos divididos pelos usuários antecipadamente. Resultados experimentais mostraram que o algoritmo melhorado tem um desempenho melhor do que o primeiro algoritmo PSO multiobjetivo em POM.

Li (2003) apresenta um algoritmo PSO modificado, o Nondominated Sorting Particle Swarm Optimizer (NSPSO). O algoritmo estende a forma básica de PSO, fazendo um melhor uso de melhores partículas pessoais *pbests* e suas descendentes para uma comparação de não-dominância mais efetiva. Em vez de uma única comparação entre a

pbest e sua descendente, o NSPSO compara todas as *pbest* e suas descendentes em toda a população. Isso se mostra eficaz em fornecer uma pressão de seleção adequada para impulsionar a população de enxames em direção à frente ótima de Pareto. Os resultados e comparação com o NSGA II (DEB *et al.*, 2002) mostram que o NSPSO é altamente competitivo com os AEMO existentes e algoritmos PSO multiobjetivo.

Zhang *et al.* (2017) apresentam o primeiro estudo de otimização de enxame de partículas de múltiplos objetivos (PSO) para problemas de seleção de recursos baseados em custo. O objetivo do trabalho é gerar uma frente de Pareto das soluções não dominadas para atender a diferentes requisitos de tomadores de decisão em aplicações do mundo real. A fim de melhorar a capacidade de pesquisa do algoritmo proposto, uma tecnologia de codificação baseada em probabilidade e um operador híbrido eficaz, juntamente com as ideias da distância de aglomeração, arquivo externo e a relação de dominação de Pareto, são aplicadas ao PSO. O algoritmo proposto é comparado com vários algoritmos de seleção de recursos multiobjetivos em cinco conjuntos de dados. Resultados experimentais mostram que o algoritmo proposto pode evoluir automaticamente um conjunto de soluções não dominadas, e é um método de seleção de recursos altamente competitivo para resolver problemas de seleção de recursos baseados em custo.

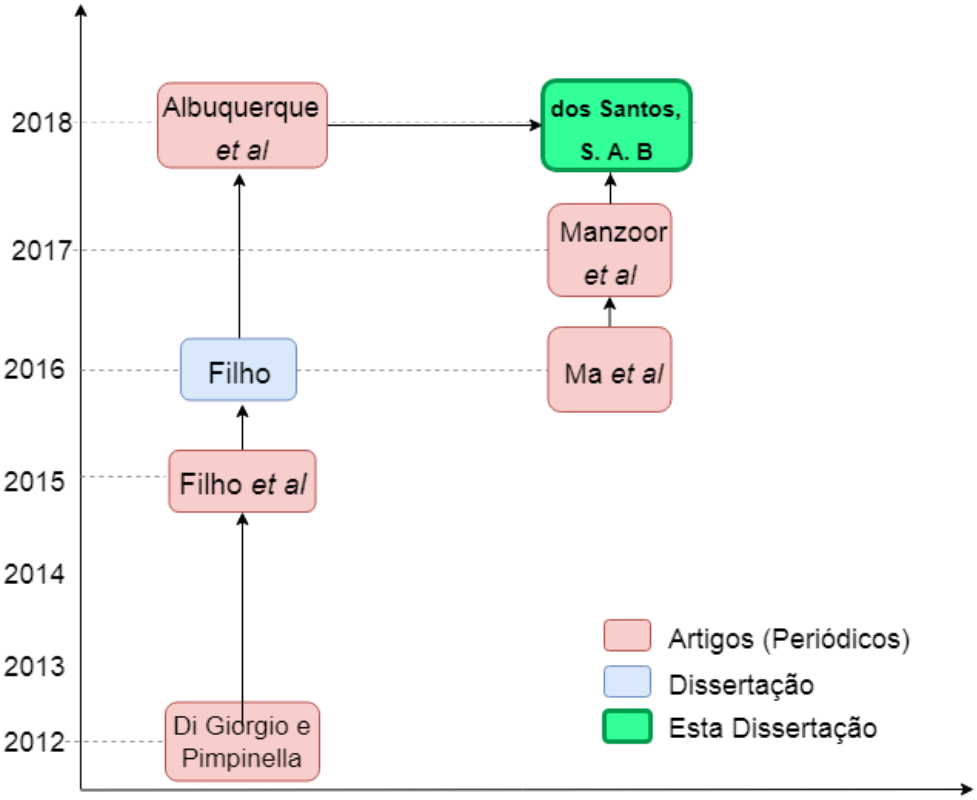
Zhang *et al.* (2018) propõem um mecanismo multiobjetivo otimizador de enxame de partículas baseado em mecanismo competitivo, onde as partículas são atualizadas com base nas competições em pares realizadas no enxame atual em cada geração. O desempenho do otimizador de enxame de partículas multiobjetivo competitivo proposto é verificado por comparações de *benchmark* com vários otimizadores de múltiplos objetivos de última geração, incluindo três algoritmos de otimização de enxame de partículas de múltiplos objetivos e três multiobjetivos algoritmos evolutivos. Resultados experimentais demonstram o desempenho promissor do algoritmo proposto em termos de qualidade de otimização e velocidade de convergência.

3.3 Esquemático Estado da Arte

Na Figura 6 é apresentado um esquemático das referências que serviram como base para o presente trabalho. As referências estão relacionadas de acordo com o ano de publicação e o tipo (Artigo ou Dissertação).

A referência em (GIORGIO; PIMPINELLA, 2012) serviu como base para o

Figura 6 – Esquemático - Estado da Arte



Fonte: A autora

modelo matemático do SHC. Apesar do modelo apresentado não considerar o conforto, a modelagem dos custos e a classificação das cargas serviram como base para o modelo apresentado nessa dissertação.

BEZERRA FILHO *et al.* (2015) e BEZERRA FILHO (2016) faz uma evolução do modelo de Giorgio e Pimpinella (2012), acrescentando o conforto ao modelo matemático, que é apresentado como um problema de programação inteira. Essa referência serve como base para a modelagem da função conforto.

Albuquerque *et al.* (2018) também se baseia no modelo de Giorgio e Pimpinella (2012), acrescentando o conforto ao modelo matemático. Utiliza as técnicas GA e programação linear, comparando-as e implanta o SHC em um cenário com cargas reais. O modelo e o cenário de cargas reais serviram como base para esta dissertação.

Em (MA *et al.*, 2016) os autores apresentam um modelo matemático que considera um *tradeoff* entre custo e conforto. A modelagem da função conforto é feita utilizando a função de Taguchi, dando uma maior precisão aos parâmetros do usuário. Essa referência também serve como base para a modelagem matemática do SHC.

Os autores em (MANZOOR *et al.*, 2017) apresentam o mesmo modelo apre-

sentado em (MA *et al.*, 2016), resolvendo o problema de otimização com quatro tipos de técnicas meta-heurísticas diferentes. Essa referência serve a nível de comparação do modelo apresentado.

3.4 Conclusões

Neste capítulo foi apresentada a revisão do estado da arte. Primeiramente, foi abordado o conceito de SHC, posteriormente foram apresentados os principais modelos e técnicas de soluções presentes na literatura. No final, foi apresentado um quadro resumindo as referências utilizadas e suas respectivas características.

Também foi apresentado a revisão do estado da arte referente a técnica PSO, apresentando as variantes e técnicas para melhorar o desempenho da técnica.

Ao final do capítulo foi apresentada uma figura, contendo um esquemático das referências que contribuíram diretamente para esta dissertação.

4 METODOLOGIA

As atividades realizadas nesta dissertação foram organizadas e executadas nas seguintes etapas:

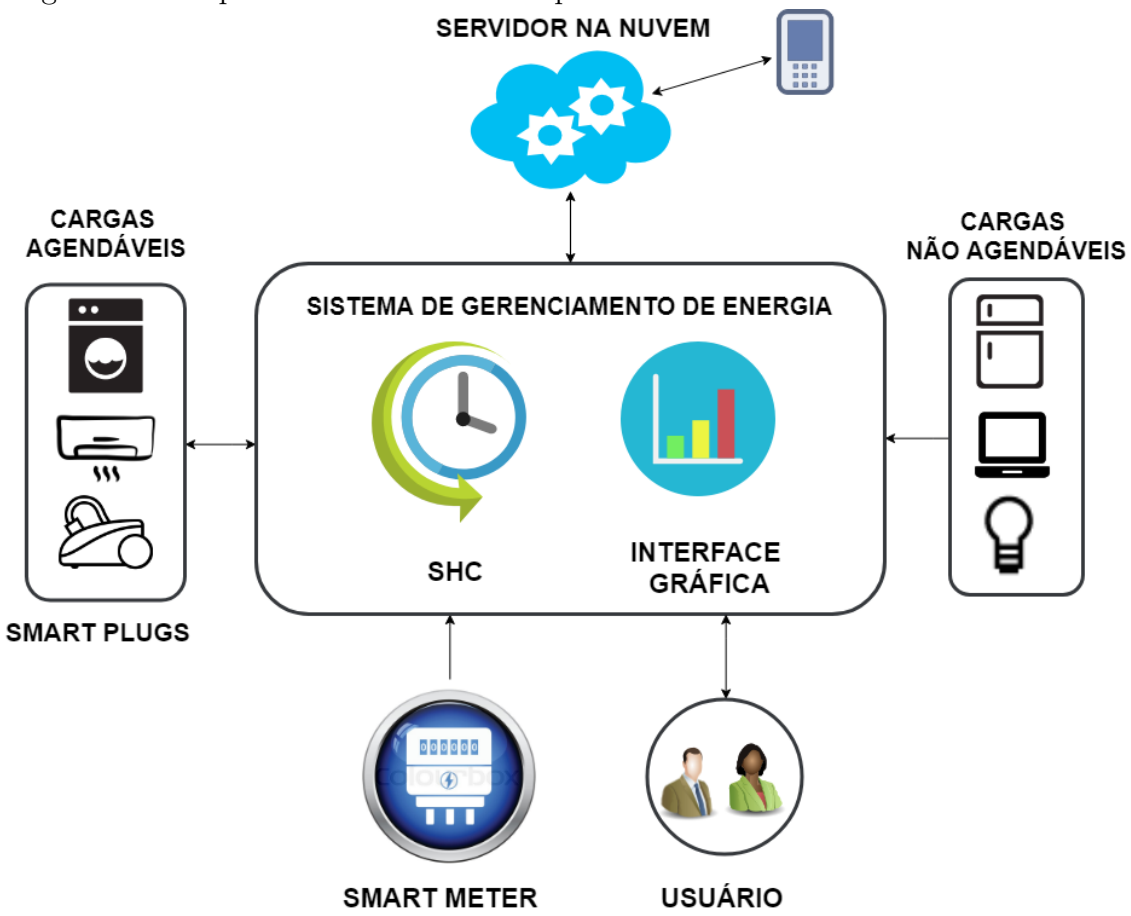
- Levantamento do estado da arte dos modelos e técnicas de SHC.
- Levantamento das técnicas para melhorar o desempenho e variantes PSO.
- Proposição de uma arquitetura de um sistema residencial de gerenciamento de energia.
- Formulação do modelo matemático do SHC, baseado nas referências estudadas, que considere a economia financeira e o conforto do usuário.
- Modelagem do software SHC, utilizando Programação Orientada a Objetos (POO).
- Construção do algoritmo PSO, para a solução do problema de otimização multiobjetivo, usando a linguagem Python.
- Validação da solução, por meio de testes utilizando diferentes cenários, a fim de comparação, inclusive um cenário com cargas reais.
- Implementação do modelo de SHC: construção do sistema embarcado.

4.1 Arquitetura Geral do Sistema Proposto

Na Figura 7 está ilustrada a arquitetura geral do modelo proposto nesta pesquisa. As cargas são classificadas em duas categorias, conforme a sua aplicação e possibilidade de controle de seu funcionamento:

- Cargas Agendáveis (CA): são aquelas cargas que podem ser ligadas/desligadas por um determinado período de tempo com ou sem a degradação da qualidade do serviço. Estas cargas são conectadas a tomadas inteligentes, *Smart Plugs*, ou diretamente ao SHC. Exemplos de CA são: condicionadores de ar, bomba do filtro de piscina, máquina de lavar roupas não programável, máquina de lavar louças, iluminação externa.
- Cargas Não Agendáveis (CNA): são as cargas que não são controláveis, que podem ter o consumo estimado pela diferença na medição de energia do medidor inteligente, *Smart Meter*, e de todas as CA e *Smart Plugs*. Exemplos de CNA são: equipamentos audiovisuais, equipamentos de informática, iluminação, torradeiras, batedeiras, geladeira, freezer.

Figura 7 – Arquitetura do Sistema Proposto



Fonte: A autora

O sistema é conectado a um *Smart Meter* para obter as informações referentes à tarifação. Através do módulo de interface gráfica (GUI), o usuário pode entrar com as informações referentes às cargas e suas respectivas preferências, além de obter os resultados do agendamento das mesmas. O módulo SHC é responsável pelo agendamento ótimo das cargas agendáveis CA. Os resultados do agendamento e informações sobre as cargas também estarão disponíveis na nuvem, onde o usuário pode obter as informações via WEB.

4.2 Modelo Matemático do SHC

O SHC tem o objetivo de escolher o momento de início de execução de cada carga em uma residência de forma que o custo financeiro resultante da operação do conjunto de cargas seja minimizado. As decisões do SHC são tomadas baseadas nos seguintes parâmetros: custo das tarifas ao longo do dia, limite máximo de demanda considerado, consumo de energia de cada carga, horário mínimo de início de cada carga (escolha do usuário), horário máximo de término de cada carga (escolha do usuário) e

melhor horário de início de cada carga (escolha do usuário).

O modelo matemático do SHC é um sistema de tempo discreto que opera a uma certa taxa de amostragem T_s, o dia é dividido em N amostras e uma residência possui M cargas planejáveis/controláveis. Na abordagem proposta, foram modeladas duas funções custo, f_1 e f_2: a primeira define a economia financeira alcançada e a segunda define o nível de conforto do usuário, respectivamente. A notação dos símbolos utilizados está definida na Tabela 1.

Tabela 1 – Lista de Símbolos

Símbolo	Descrição
M	Número de cargas planejáveis
N	Número total de amostras
$\bar{P_m}$	Vetor da potência média da m-ésima carga
$\hat{P_m}$	Vetor da potência de pico da m-ésima carga
N_m	Duração, em número de amostras, da m-ésima carga
I_{Sm}	Amostra no horário mínimo de início da m-ésima carga
I_{Em}	Amostra no horário máximo de término da m-ésima carga
I_{Bm}	Amostra associada com o melhor horário de início da m-ésima carga
I_{Cm}	Hora de início agendada para m-ésima carga
C_{Lm}	Nível de conforto da m-ésima carga
P_k	Limite de pico no k-ésimo instante de tempo
C	Vetor do custo do consumo de energia elétrica no período
T_s	Taxa de Amostragem

Fonte: a autora.

4.2.1 Maximização da Economia Financeira

O primeiro objetivo na otimização do agendamento de cargas é a minimização do custo financeiro total de eletricidade durante o período de 24 horas, maximizando assim a economia financeira. Para a análise de economia financeira das cargas elétricas do SHC, esta dissertação tem por base a modelagem de Giorgio e Pimpinella (2012).

O custo financeiro total resultante do agendamento do SHC, durante o dia, é expresso pela Equação 4.1. Como a taxa de amostragem T_s é expressa em minutos, utiliza-se $\frac{T_s}{60}$.

$$f_{Fcost} = \sum_{m=1}^{M} \sum_{k=I_{Cm}}^{I_{Cm}+N_m} (\bar{P_m}[k] \frac{T_s}{60} C[k]) \tag{4.1}$$

sujeito às restrições:

$$I_{Sm} \leq I_{Cm} \leq I_{Em} \tag{4.2}$$

$$\sum_{k=1}^{N}(\sum_{m=1}^{M}\hat{P_m}[k]) \leq P_k \tag{4.3}$$

Na restrição dada pela Equação 4.2 está definido que o instante agendado para a execução da carga m esteja entre os limites mínimo e máximo definidos pelo usuário (esses limites estão relacionados às preferências de horário do usuário). A restrição definida pela Equação 4.3 garante que o acionamento das cargas não irá exceder o valor de limite de demanda considerado em nenhum instante k, ao longo do dia.

A economia financeira do agendamento está relacionada a quanto o custo financeiro do perfil agendado pelo SHC foi menor em relação ao custo financeiro do perfil preferencial escolhido pelo usuário. O custo financeiro do perfil preferencial escolhido pelo usuário está definido na Equação 4.4. Logo, a função que representa a economia financeira (f_1), referenciada pela Equação 4.5, é a diferença entre o custo resultante do perfil preferencial do usuário e o custo resultante do agendamento do SHC.

$$f_{UFcost} = \sum_{m=1}^{M}\sum_{i=I_{Bm}}^{I_{Bm}+N_m}(\bar{P_m}[i]\frac{T_s}{60}C[i]) \tag{4.4}$$

$$\begin{aligned} f_1 &= \sum_{m=1}^{M}[\sum_{i=I_{Bm}}^{I_{Bm}+N_m}(\bar{P_m}[i]\frac{T_s}{60}C[i]) - \sum_{i=I_{Cm}}^{I_{Cm}+N_m}(\bar{P_m}[i]\frac{T_s}{60}C[i])\ ] \\ &= f_{UFcost} - f_{Fcost} \end{aligned} \tag{4.5}$$

4.2.2 Maximização do Conforto

O outro objetivo que se deseja alcançar com a otimização é o conforto, que se refere ao agendamento das cargas elétricas obedecendo à configuração de preferências do usuário. O usuário seleciona os horários mínimo, máximo e ideal para o acionamento de cada uma das cargas e quão mais próxima a otimização resultante do SHC para o acionamento das cargas for desta configuração do usuário, maior será o conforto. Desta forma, o conforto é calculado a partir da diferença entre o horário preferível de acionamento da carga selecionado pelo usuário e o horário de acionamento definido pelo SHC.

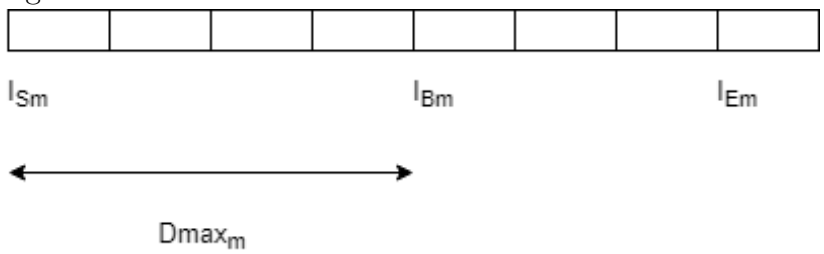
Na Equação 4.6 está definido o desconforto gerado pela execução de uma carga m. Esta função é a diferença entre o instante agendado pelo SHC e o instante escolhido

pelo usuário, multiplicado por C_{Lm}, que é o nível de conforto atribuído à carga m. O valor de C_{Lm} varia de 0 a 1 e está relacionado à importância da carga m: se esse valor é igual a 1, é muito importante que a carga inicie no melhor instante escolhido pelo usuário (I_{Bm}), se o valor é igual a zero, não há importância que a carga inicie no melhor horário, apenas que respeite os limites de instantes inicial (I_{Sm}) e final (I_{Em}).

$$fDISC_m = C_{Lm}|I_{Cm} - I_{Bm}| \tag{4.6}$$

Para levar em consideração o tamanho da janela de acionamento (Figura 8) de uma dada carga m, criou-se a função de distância máxima $Dmax_m$.

Figura 8 – Distância Máxima



Fonte: A autora

A função $Dmax_m$ (Equação 4.7) é a maior distância entre o melhor horário selecionado pelo usuário I_{Bm} e os limites inicial e final do acionamento I_{Sm}, I_{Em}, para uma dada carga m.

$$Dmax_m = \max(|I_{Sm} - I_{Bm}|, |I_{Em} - I_{Bm}|) \tag{4.7}$$

Dessa forma, a função f_2 representa o conforto total do usuário, definido pela Equação 4.8.

$$f_2 = \frac{\sum_{m=1}^{M}(Dmax_m - fDISC_m)}{\sum_{m=1}^{M} Dmax_m} \tag{4.8}$$

4.2.3 Função Custo Total Multiobjetivo

Uma única função custo é apresentada como uma combinação linear das funções custo relacionadas à economia financeira e ao conforto do usuário.

$$f = \alpha\ f_1 + (1-\alpha) f_2 \tag{4.9}$$

em que $\alpha \in [0,1]$

Ao variar α, o usuário pode obter uma economia de energia agregada a um baixo impacto no seu conforto. Ao fazer $\alpha = 1$, o usuário determina ao controlador que o nível de conforto não é importante para a execução das cargas e o controlador minimizará apenas os custos do consumo de eletricidade. Ao fazer $\alpha = 0$, o controlador obterá a solução que melhor mantém os níveis de conforto do usuário, obedecendo às restrições impostas.

4.3 Desenvolvimento da Solução

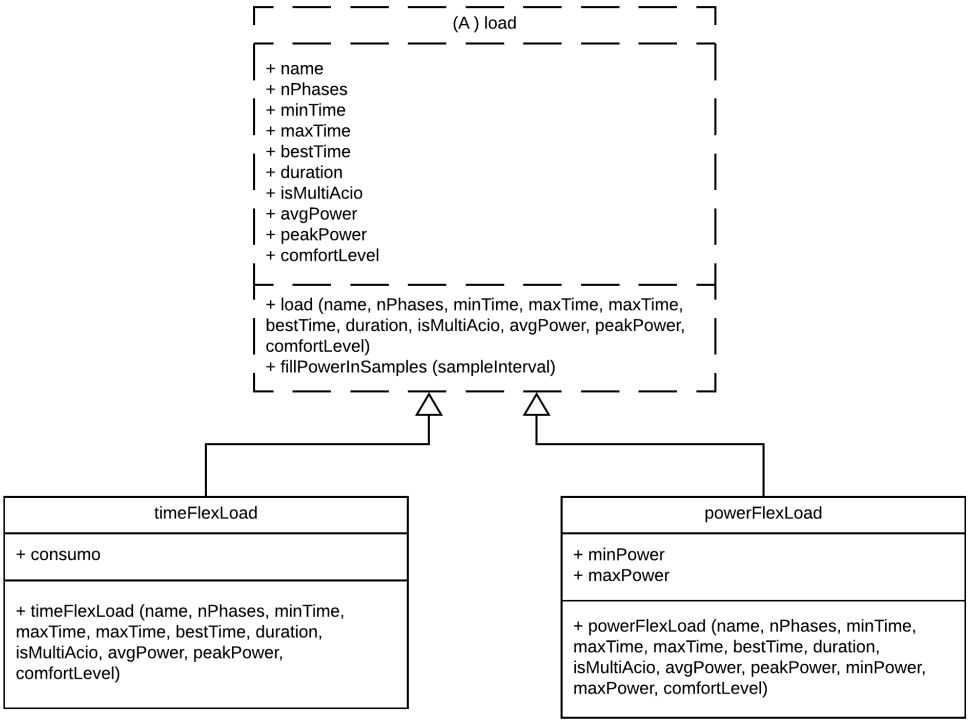
4.3.1 Modelagem

A solução foi executada na linguagem Python, utilizando o paradigma de Programação Orientada a Objetos (POO). As Figuras 9 e 10 contém os diagramas de classes da solução executada.

A Carga foi modelada segundo o diagrama da Figura 9. Foi criada a classe abstrata ***load.py*** e outras duas classes que herdam da classe dela: ***timeFlexLoad.py*** e ***powerFlexLoad.py***. No modelo proposto nessa dissertação está sendo utilizada apenas a classe ***timeFlexLoad.py***, pois apenas estão sendo utilizadas as cargas flexíveis em relação ao tempo de início de execução. A segunda classe foi criada para posteriores trabalhos em que possa se utilizar cargas flexíveis em relação à potência também.

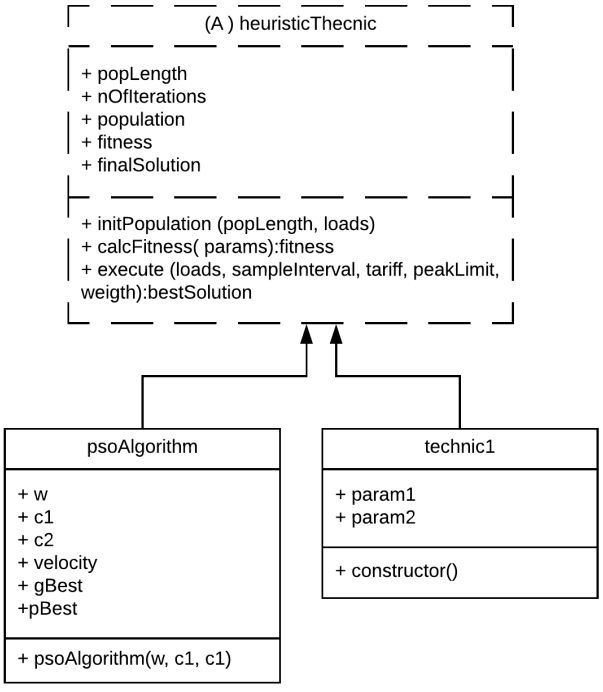
A técnica de solução da otimização foi modelada da forma ilustrada na Figura 10. Foi criada a classe abstrata ***heuristicThecnic.py*** que representa o algoritmo de meta-heurística utilizado para solucionar o problema de otimização. Dessa forma, podem ser criadas outras subclasses contendo técnicas baseadas em população, como por exemplo GA, Evolução Diferencial e outras; sem que haja uma grande mudança no código. Nesta dissertação foi utilizada apenas a técnica PSO.

Figura 9 – Modelagem da Carga



Fonte: A autora

Figura 10 – Modelagem da técnica de otimização



Fonte: A autora

4.3.2 Otimização Multiobjetivo com PSO

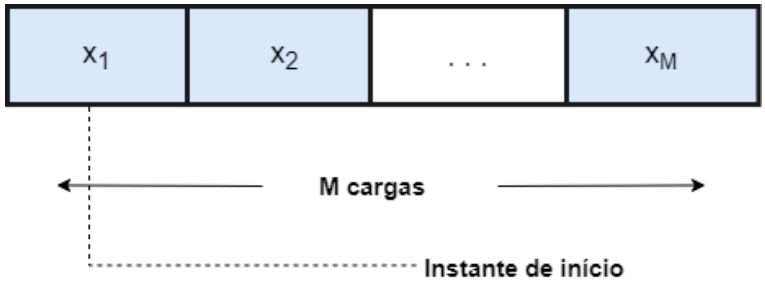
Nesta Subseção é descrita a forma como o problema de otimização multiobjetivo foi solucionado, utilizando a técnica de meta-heurística PSO.

4.3.2.1 Codificação

Para o problema proposto, a solução ótima é um conjunto com os melhores horários de início de execução das M cargas, ou seja um perfil que obtenha o máximo de economia financeira, gerando o mínimo desconforto.

Foi utilizada a codificação inteira. A posição de cada partícula é composta por um vetor de números inteiros de tamanho M (Figura 11). A variável x, que representa o instante de início, pode assumir qualquer valor no intervalo $[0,N]$, em que N é o número de amostras dentro de 24 horas. Por exemplo, se uma taxa de amostragem $T_S = 5$ minutos, o valor de N será 288, considerando que $N = \frac{24 \times 60}{5}$.

Figura 11 – Vetor que representa a posição da Partícula



Fonte: A autora

A inicialização da população inicial é feita de forma aleatória, dentro do intervalo tolerado pelo usuário, $[I_{Sm}, I_{Em}]$. O vetor velocidade é inicializado com valor zero.

4.3.2.2 Função Fitness

Tendo em vista que o problema de otimização é um problema de maximização com restrições, foi modelada uma função *Fitness* que leve em consideração os objetivos a serem maximizados e as restrições impostas. Dessa forma a função *fitness* acrescenta a parcela f_{rest} à função custo total (apresentada anteriormente na Equação 4.9). Essa parcela representa a restrição apresentada na Equação 4.3. A função *fitness* está definida

na Equação 4.10.

$$f_{fitness} = \alpha\ f_1 + (1-\alpha) f_2 + f_{rest} \tag{4.10}$$

$$f_{rest} = \frac{1}{(1+\sum_{k=1}^{N} r_k)},\ r_k = \begin{cases} 1, & \hat{P}_k > P_k \\ 0, & \text{se não} \end{cases} \tag{4.11}$$

4.3.2.3 Atualização da Velocidade e Posição

Os vetores velocidade e posição são atualizados, respectivamente, pelas equações já apresentadas (Equação 2.3, Equação 2.4), sendo que o peso de inércia w é atualizado a cada geração, usando quatro técnicas diferentes, também já apresentadas, LDIW, DIW, EIW, AIW.

4.3.2.4 Critérios de Parada

Foram utilizados dois critérios de parada:

1. Número máximo de iterações (K) ou
2. Critério de convergência: se não houver melhora na solução em $(0,2 \times K)$ iterações, ou seja 20% do número máximo de iterações.

4.3.2.5 Operador de Restart

Foi implementado um operador que reinicia a população preservando a melhor partícula (*gbest*), se não houver melhora na solução em $(0,1 \times K)$ iterações, ou seja 10% do número máximo de iterações.

4.3.3 Pseudo-Código PSO

No algoritmo 1 é apresentado um pseudo-código da solução PSO que foi implementada nessa dissertação.

Algoritmo 1: Pseudo-código da solução PSO

Data: $Loads, S, c1, c2, K$ <Loads são as cargas com suas características, S é o tamanho da população, c1 e c2 são os parâmetros cognitivo e social, K é o número máximo de iterações>
Result: *gBest* <Retorna a partícula com a melhor solução>

```
1  pop = iniPop(Loads); <Inicializa a posição de forma aleatória e a
     velocidade com valores 0>
2  fitness = calcFitness(); <Calcula a função fitness (Equação 4.10)>
3  Determina pbest e gbest; <Equação 2.1 e Equação 2.2>
4  k=0
5  while k <= K & saida = 0 do
6      w = inerciaTecnica(); <Calcula o peso de inercia de acordo com 4
         técnicas diferentes (Equações 3.1, 3.2, 3.3, 3.4)>
7      for i = 1 to S do
8          if fitness_i > gbestFit then
9              gbest = pop_i
10             last = k
11         if fitness_i > pbestFit_i then
12             pbest_i = pop_i
13         r1 = rand(); r2 = rand() <números aleatórios para atualização da
             velocidade>
14         Atualiza pop_i.V <Atualiza a velocidade, usando Equação 2.3>
15         Atualiza pop_i.X <Atualiza posição, usando Equação 2.4>
16         fitness = calcFitness()
17         if (k - last) > (0.1 × K) then
18             if k <= (0.2 × K) then
19                 <Reinicia a população, porém mantém o gbest.O gbest é
                     inserido na pior solução da nova população>
20                 pop = iniPop(Loads)
21                 fitness = calcFitness()
22                 p = argmin(fitness)
23                 pop_p = gbest
24             else
25                 <Convergência>
26                 saida = 1
27     k = k+1 <incrementa k>
```

4.4 Conclusões

Neste capítulo foi apresentada a metodologia usada nesta Dissertação. Primeiramente foi apresentada uma arquitetura geral do sistema de gerenciamento de energia proposto. Posteriormente, foi apresentado o modelo matemático do SHC e, por fim, foi apresentado como foi solucionado o problema: modelagem do *software* da solução,

bem como as características específicas, operadores e um pseudo-código da solução PSO implementada nessa dissertação.

5 RESULTADOS DOS EXPERIMENTOS

A avaliação do modelo do SHC multiobjetivo é realizada neste Capítulo por meio de simulações em dois diferentes cenários de tarifas: Tarifa com valor fixo e Tarifa Branca. A solução é avaliada em termos de economia financeira, nível de conforto e tempo de execução.

Todas as simulações são realizadas utilizando linguagem *Python*, através da *Integrated Development Environment* (IDE) Spyder (*The Scientific Python Development Environment*). Spyder é um poderoso ambiente científico escrito em *Python* para *Python*, e projetado por e para cientistas, engenheiros e analistas de dados. A IDE oferece uma combinação de edição avançada, análises, *debugging* e perfil de funcionalidades de uma ferramenta com exploração de dados, execução interativa, inspeção e visualização (SPYDER, 2018).

O código-fonte dos arquivos em *Python* estão disponíveis no apêndice B desta dissertação. O computador utilizado para execução dos ambientes simulados possui as seguintes características:

- Processador: Intel Core i3-4005U @ 1,70 GHz;
- Memória RAM: 3,80 GB;
- Disco Rígido: HD 309,8 GB;
- Sistema Operacional: Linux - Ubuntu 14.04 64bits;

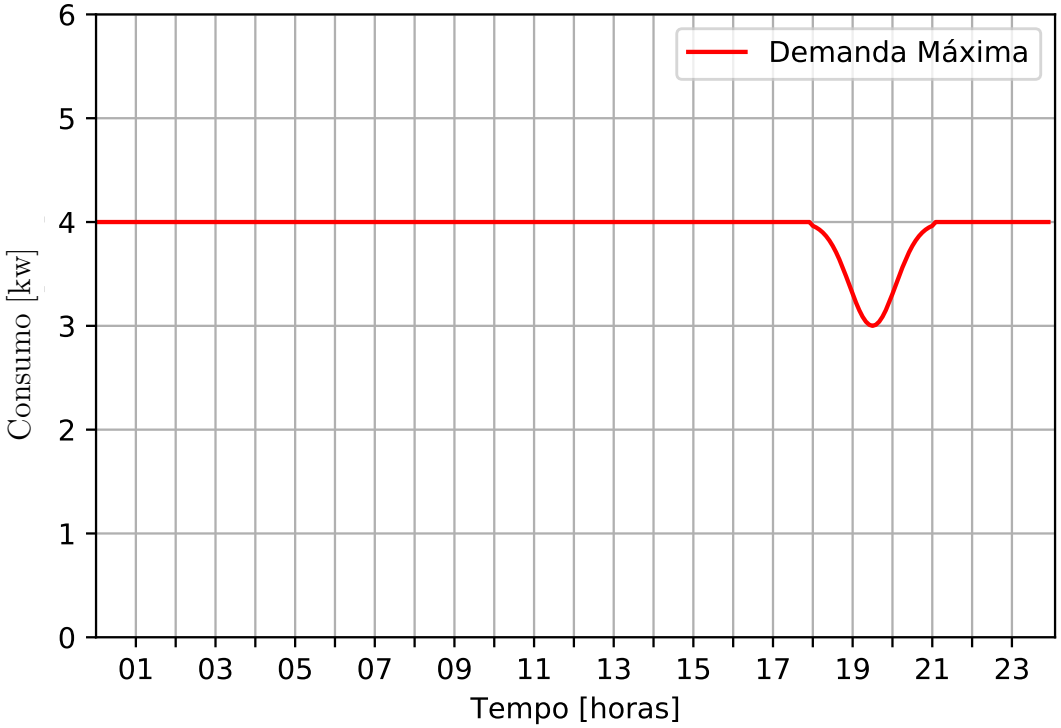
5.1 Parâmetros de Simulação

Nesta Seção são descritas as características comuns a todos os cenários de simulação.

- Taxa de amostragem T_s: 5,0 minutos.
- Limite de demanda considerada: 4,0 kW (Figura 12).
- Representação das cargas não agendáveis (CNA), apresentadas na seção 4.1: após medições reais do comportamento das cargas (Tabela 2), no período de um dia, foi constatado o padrão de maior consumo dessas cargas. Dessa forma, é usada uma gaussiana invertida centrada às 19h30 com amplitude de 1,0 kW para simular uma diminuição no limite de demanda considerado, (Figura 12).

Em relação à tarifação, foram utilizados dois tipos de tarifa: tarifa com o valor

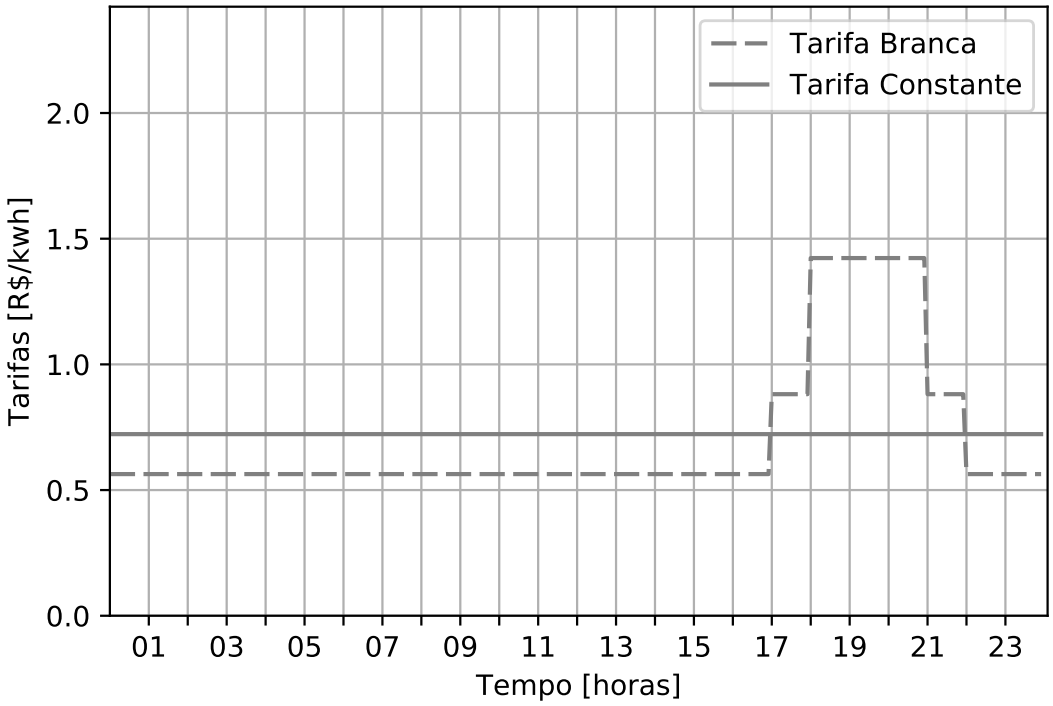
Figura 12 – Limite de Demanda Considerado



Fonte: A autora

constante e a tarifa branca, ambas mostradas na Figura 13.

Figura 13 – Modelos de Tarifas



Fonte: A autora

As cargas e suas respectivas características consideradas nas simulações estão mostradas na Tabela 2. Esse é um cenário real de cargas instaladas em uma residência de teste.

Tabela 2 – Características das cargas

ID	Carga	Ciclos	$\Delta t(min)$	$\overline{P}[kw]$	$\widehat{P}[kw]$	Melhor Hora	Hora Mínima	Hora Máxima	C_{Lm}
1	Bomba de Recalque	1	[20]	[0.02, 1.96, 0.02, 0.02, 0.02, 0.05]	[0.15, 2.10, 0.15, 0.15, 0.20, 0.55]	08h00 e 16h00	07h00	17h00	1,0
2	Bomba Filtro	1	[120]	[2.36]	[2.70]	08h00	07h00	17h00	0,9
3	Máq.(Roupas)	8	[10, 10, 5, 10, 5, 5, 5, 10]	[0.07, 1.40, 0.10, 0.07, 2.02, 0.01]	[0.10, 2.10, 1.20, 0.10, 2.15, 0.02]	08h00	07h00	17h00	0,7
4	Ilum.Externa	1	[270]	[0.27, 0.05, 2.10, 0.11, 0.11, 0.10, 0.10, 0.26]	[2.10, 0.30, 2.20, 0.20, 0.60, 0.80, 0.80, 1.10]	18h00	17h00	24h00	0,6
5	Ilum. Interna	1	[270]	[0.07, 2.00, 0.07, 0.07, 1.80, 0.01]	[0.10, 2.10, 0.10, 0.25, 2.30, 0.02]	18h00	17h00	23h00	0,5
6	Ar Cond. 1	14	[10, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[0.80, 0.50, 1.00]	[1.00, 0.80, 1.20]	16h00 e 20h00	15h00	24h00	0,4
7	Ar Cond. 2	7	[30, 20, 5, 5, 5, 5, 5]	[1.40, 0.50, 0.60, 1.00]	[1.60, 0.80, 0.60, 1.00]	20h00	17h00	24h00	0,3
8	Ar Cond. 3	1	[240]	[0.60, 0.70, 1.00]	[0.60, 0.70, 1.00]	20h00	17h00	24h00	0,2
9	Ar Cond. 4	7	[10, 10, 5, 5, 5, 5, 5]	[0.60, 0.70, 1.00]	[0.60, 0.70, 1.00]	20h00	17h00	24h00	0,2
10	Máq.(Louças)	5	[5, 10, 15, 5, 10]	[0.60, 0.70, 1.00]	[0.60, 0.70, 1.00]	21h00	18h00	22h00	0,2

As cargas podem ser executadas em diferentes ciclos sequenciais, com duração em minutos (Δt), operando com $\overline{P}[kw]$ de potência média e $\widehat{P}[kw]$ de potência de pico, cada ciclo. Cada carga possui um melhor instante de início e os instantes limite mínimo e máximo, que formam o intervalo de início de cada carga. Também é atribuído a cada carga o nível de conforto C_{Lm}, que está relacionada ao quão é importante que a carga inicie no melhor horário escolhido. Há, também, a possibilidade da carga ter mais de um momento de execução, que é o caso das cargas 1 (Bomba de Recalque) e 6 (Ar Condicionado 1), que têm dois momentos de execução.

Os parâmetros de simulação para o algoritmo PSO foram:

- Tamanho do Enxame: 10 partículas. Cada partícula representa uma possível solução.
- Técnicas para mudança de peso de inércia w: quatro técnicas diferentes (LDIW, DIW, EIW, AIW).
- Parâmetro cognitivo e social: $c_1 = 2$ e $c_2 = 2$ (valores recomendados na literatura).
- Número máximo de iterações: 1000

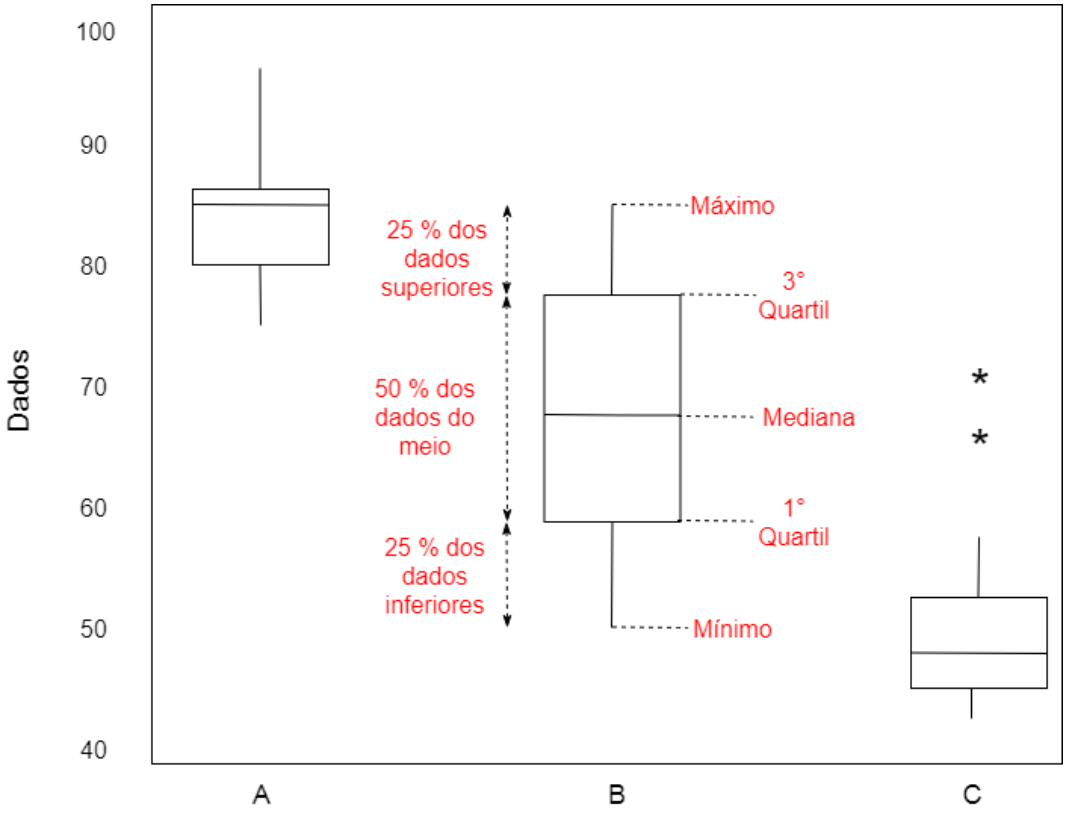
Cada solução, em qualquer um dos cenários, foi executada 20 vezes. Todos os outros operadores e características do algoritmo PSO já foram descritos na subseção 4.3.2.

5.2 Método de Análise dos Resultados

O desempenho da solução é verificado por meio da análise dos resultados das simulações em relação aos valores de *Fitness*, economia financeira, conforto relativo, tempo de execução e seus respectivos valores de desvio padrão σ.

Também é utilizada a ferramenta gráfica Diagrama de Caixa, que avalia a distribuição dos resultados, ao longo das 20 execuções. Essa ferramenta é formada pela mediana, limites inferior e superior e os 1º e 3º quartis (Q1 e Q3), fornecendo as características de localização, dispersão e simetria dos resultados. Na Figura 14 está ilustrado um exemplo de Diagrama de Caixa para demonstrar as informações contidas.

Figura 14 – Diagrama de Caixa - Exemplo



Fonte: A autora

O diagrama de caixa começa sempre no valor mínimo da base de dados e termina no valor máximo, assim como o histograma, ou seja, todo o diagrama de caixa representa 100% da base de dados. A vantagem dessa ferramenta é que cada região do gráfico representa uma parte dos dados.

Informações Fornecidas:

- Dispersão: a dispersão é representada pela amplitude do gráfico, que pode ser

calculada como máximo valor – mínimo valor. Quanto maior for a amplitude, maior a variação nos dados.

- Simetria: um conjunto de dados que tem uma distribuição simétrica, terá a linha da mediana no centro do retângulo (caso B da Figura 14).
- Assimetria Negativa: quando a posição da linha da mediana é próxima ao terceiro quartil, os dados são assimétricos negativos (caso A da Figura 14). Significa que a maior parte dos 50% dos dados centrais está abaixo da mediana.
- Assimetria Positiva: quando a linha da mediana está próxima ao primeiro quartil, os dados são assimétricos positivos (caso C da Figura 14). Significa que a maior parte dos 50% dos dados centrais está acima da mediana.
- *Outliers*: os *outliers* em um diagrama de caixa aparecem como pontos ou asteriscos fora das "linhas" desenhadas (asteriscos no caso C da Figura 14).

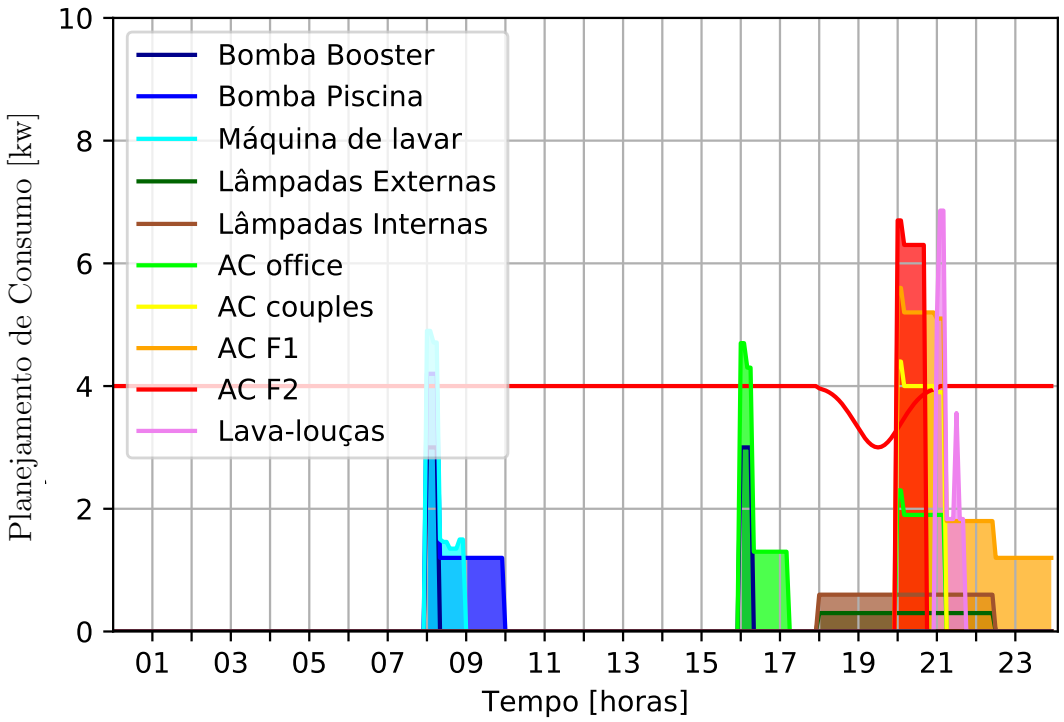
5.3 Cenário 1: Tarifa Constante

Este cenário utiliza a tarifa constante durante todo o dia no valor de $0,722495$ R$/kwh, situação similar a como é realizada a tarifação residencial no Brasil, se não houver aderência à Tarifa branca. Neste tipo de situação o controlador trabalha apenas para manter as cargas dentro do intervalo desejado pelo usuário e para garantir que o limite de pico não seja ultrapassado. Não há ganhos econômicos em acionar as cargas em momentos distintos do dia. Logo, para este tipo de cenário, qualquer solução que mantenha o conjunto de cargas abaixo do limite de pico e atenda aos requisitos do usuário de tempo de início e de término do acionamento de cada carga, é uma solução ótima do ponto de vista de economia energética.

Na Figura 15 são mostradas as cargas ao longo do dia, nos horários preferenciais selecionados pelo usuário. Como se pode observar, a configuração solicitada pelo usuário ultrapassa a demanda máxima contratada.

Na Tabela 3, estão os resultados para o cenário com a tarifa fixa, com o valor de $\alpha = 0$, visto que não há como ter economia financeira, já que a tarifa tem um valor fixo. A simulação é executada utilizando as diferentes técnicas de mudança de peso de inércia do algoritmo PSO. Na Tabela estão os resultados do custo com o perfil preferencial do usuário (Custo US), custo após o agendamento (Custo SHC) e conforto relativo percentual. Todos os resultados mostrados são os valores médios das 20 execuções.

Figura 15 – Perfil de Consumo Preferencial selecionado pelo Usuário



Fonte: A autora

Tabela 3 – Resultados Cenário 1

Técnica	Custo Us (R$)	Custo SHC (R$)	Conforto Relativo
LDIW	11,84	11,84	86,48%
DIW	11,84	11,84	88,51%
AIW	11,84	11,84	88,30%
EIW	11,84	11,84	88,02%

Fonte: a autora.

Na Tabelas 4 e 5 está uma comparação do desempenho das quatro técnicas, em termos de valores de função *Fitness* e do tempo de execução Δt, respectivamente. São mostrados os valores piores, melhores e médios, com seus respectivos valores de desvio padrão.

Tabela 4 – Comparação *Fitness* - Cenário 1

	Valor de *Fitness*			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	1,2152	1,2471	1,2701	0,015
DIW	1,2110	1,2528	1,2689	0,014
AIW	1,2095	1,2523	1,2690	0,016
EIW	1,2253	1,2514	1,2691	0,009

Fonte: a autora.

Tabela 5 – Comparação Tempo - Cenário 1

	Δt (segundos) (Pior)	(Média)	(Melhor)	Desvio Padrão σ
LDIW	99	63	46	13,16
DIW	76	65	52	6,47
AIW	84	66	53	8,70
EIW	101	57	46	14,25

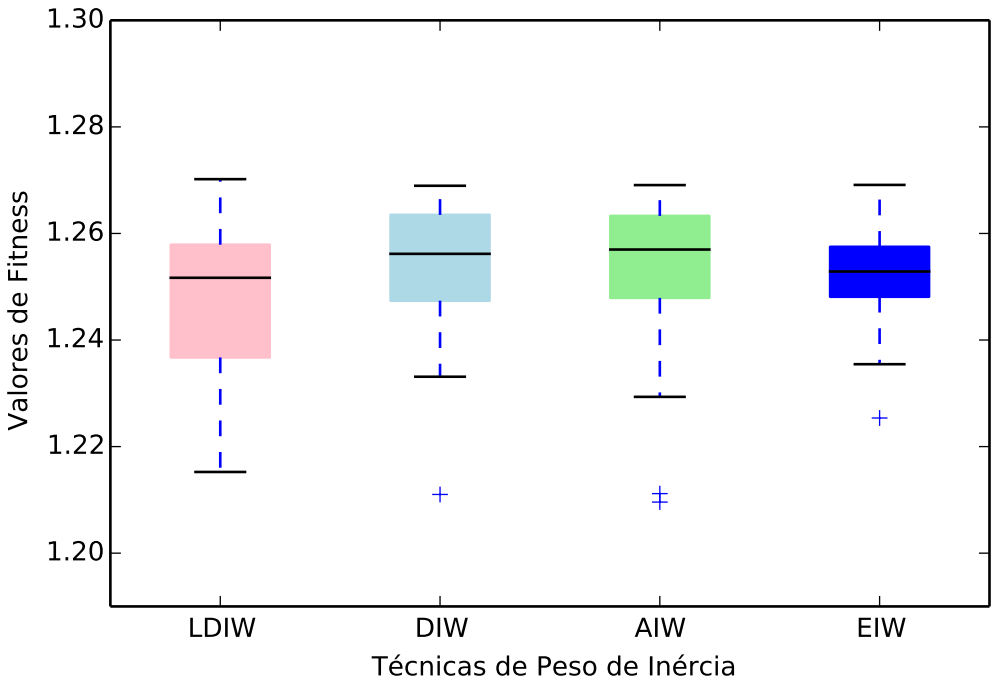
Fonte: a autora.

5.3.1 Análise dos Resultados

Com base nos resultados das tabelas, a técnica com o melhor desempenho é a DIW, pois apresenta os melhores valores médios de economia, conforto e *Fitness*. A técnica LDIW tem os piores resultados de economia e conforto, de *Fitness*. As outras duas técnicas (AIW e EIW) apresentam resultados semelhantes. Em relação ao tempo de execução, as técnicas apresentam valores bem semelhantes, porém a técnica DIW também se sobressai, por ter o menor desvio padrão, mostrando uma menor variação em torno da média.

Na Figura 16 está o diagrama de caixa dos valores de *Fitness* para as diferentes técnicas de peso de inércia.

Figura 16 – Diagrama de Caixa



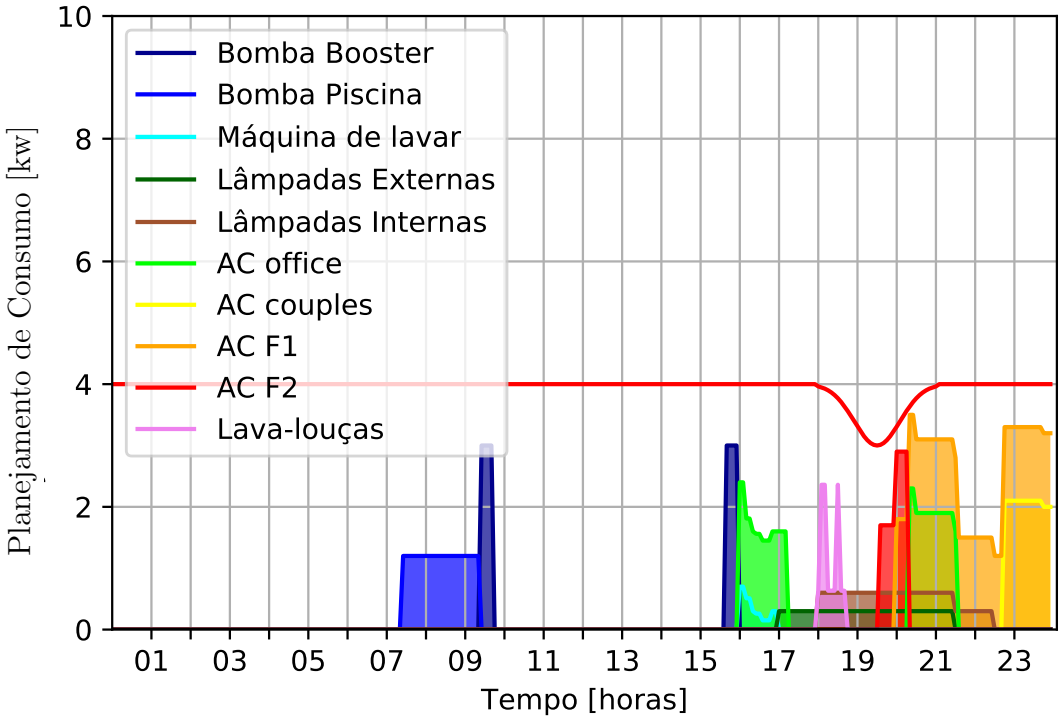
Fonte: A autora

Pelo gráfico pode-se observar que as técnica LDIW possui a maior amplitude e assimetria negativa, mostrando uma maior dispersão dos resultados de valor *Fitness*. As outras técnicas possuem resultados semelhantes, no que se refere a amplitude, porém todas apresentam pontos discrepantes (*outliers*). Apesar disso, em relação à dispersão dos resultados e assimetria, a técnica EIW apresenta melhores resultados. Dessa forma, mesmo EIW se comportando de forma mais estável, a técnica DIW possui os melhores resultados da média e mediana do valor *Fitness*.

5.3.2 Gráficos de distribuição das Cargas ao longo do dia

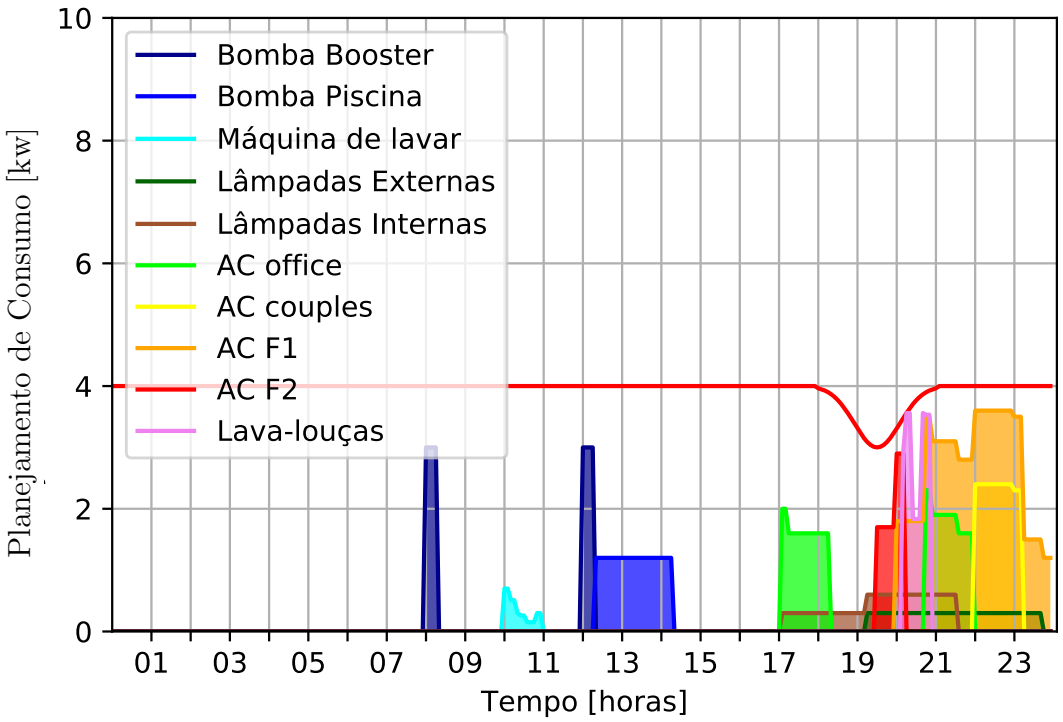
Nas Figuras 17, 18, 19, 20 é mostrado o agendamento das cargas para cada caso.

Figura 17 – Agendamento Cenário 1 - LDIW



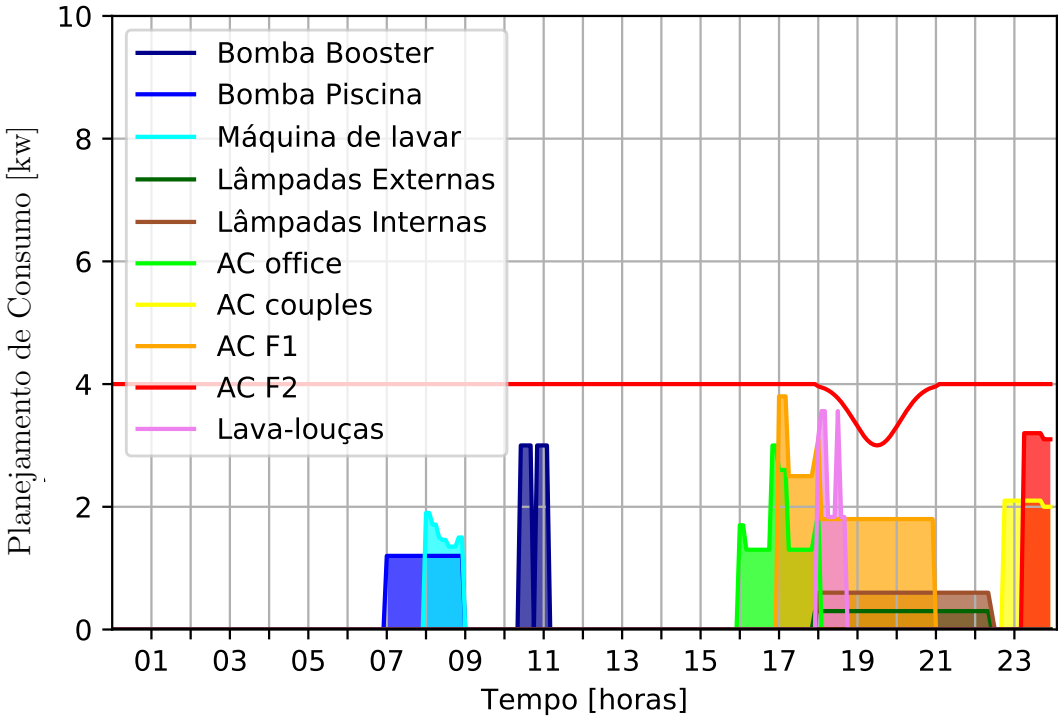
Fonte: A autora

Figura 18 – Agendamento Cenário 1 - DIW



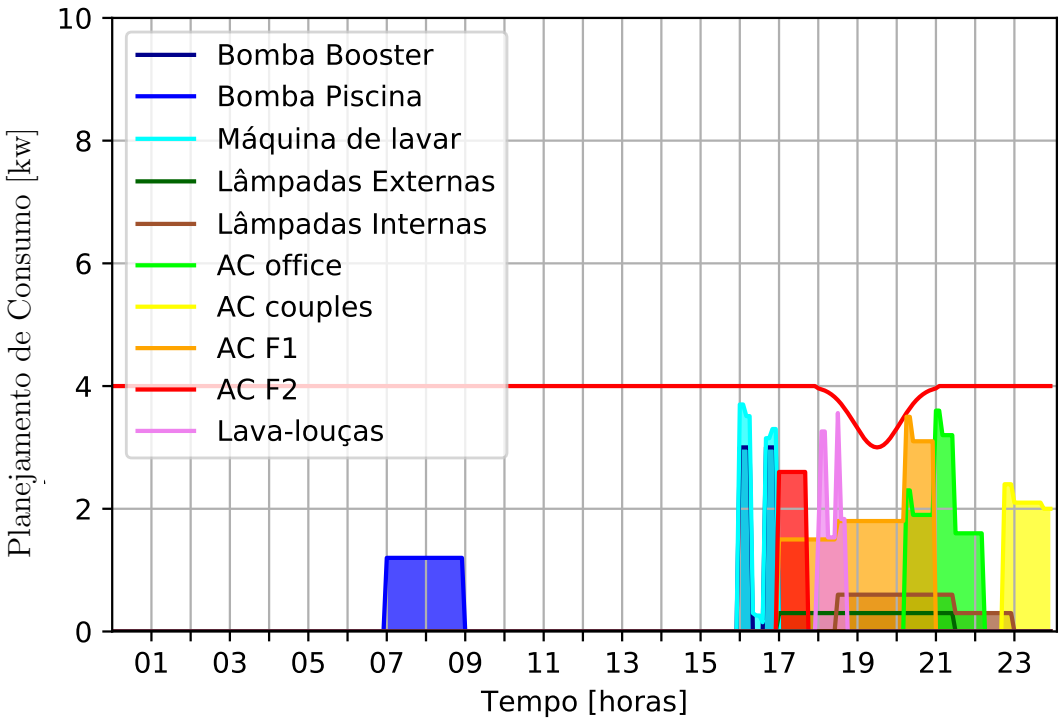
Fonte: A autora

Figura 19 – Agendamento Cenário 1 - AIW



Fonte: A autora

Figura 20 – Agendamento Cenário 1 - EIW



Fonte: A autora

5.4 Cenário 2 - Tarifa Branca

Nesse cenário é utilizada a tarifa branca, seguindo a legislação brasileira, possuindo três postos horários distintos, com os respectivos preços mostrados na Tabela 6:

Tabela 6 – Preços da Tarifa Branca

Tarifa	Período	Preço [R$/kwh]
Ponta	18h00 - 21h00	1,42294
Intermediário	17h00 - 18h00 e 21h00 - 22h00	0,88144
Fora de Ponta	restante do tempo	0,56355

São simulados três casos diferentes, variando os valores de α, de acordo com as preferências: $\alpha = 0,3$, maior preferência para o conforto; $\alpha = 0,5$, preferência igual por conforto e economia; e $\alpha = 0,7$, preferência maior pela economia. Para cada um desses casos, a solução é simulada usando quatro diferentes técnicas de mudança de peso de inércia do algoritmo PSO, cada um sendo executado 20 vezes.

5.4.1 Caso 1 - $\alpha = 0,3$

Na Tabela 7 estão os resultados para o cenário com a tarifa branca, com o valor de $\alpha = 0,3$. Os resultados mostrados são: custo com o perfil preferencial do usuário (Custo Us), custo após o agendamento (Custo SHC), economia financeira (percentual) e conforto relativo percentual. Os resultados mostrados são os valores médios das 20 execuções.

Tabela 7 – Resultados Cenário 2 ($\alpha = 0,3$)

Técnica	Custo Us (R$)	Custo SHC (R$)	Economia Financeira	Conforto Relativo
LDIW	15,76	12,90	18,09%	81,56%
DIW	15,76	12,80	18,72%	83,42%
AIW	15,76	12,88	18,24%	83,32%
EIW	15,76	12,97	17,65%	81,16%

Fonte: a autora.

Nas Tabelas 8 e 9 está uma comparação do desempenho das quatro técnicas, em termos de valores de função *Fitness* e do tempo de execução Δt, respectivamente. São mostrados os valores piores, melhores e médios, com seus respectivos valores de desvio padrão.

Tabela 8 – Comparação *Fitness* - Cenário 2 ($\alpha = 0,3$)

	Valor de *Fitness*			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	1,1712	1,2174	1,2407	0,020
DIW	1,1889	1,2230	1,2509	0,016
AIW	1,1792	1,2213	1,2480	0,020
EIW	1,1828	1,2152	1,2409	0,019

Fonte: a autora.

Tabela 9 – Comparação Tempo - Cenário 2 ($\alpha = 0,3$)

	Δt (segundos)			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	115	54	51	14,82
DIW	74	57	49	6,72
AIW	73	56	51	5,22
EIW	84	54	46	9,81

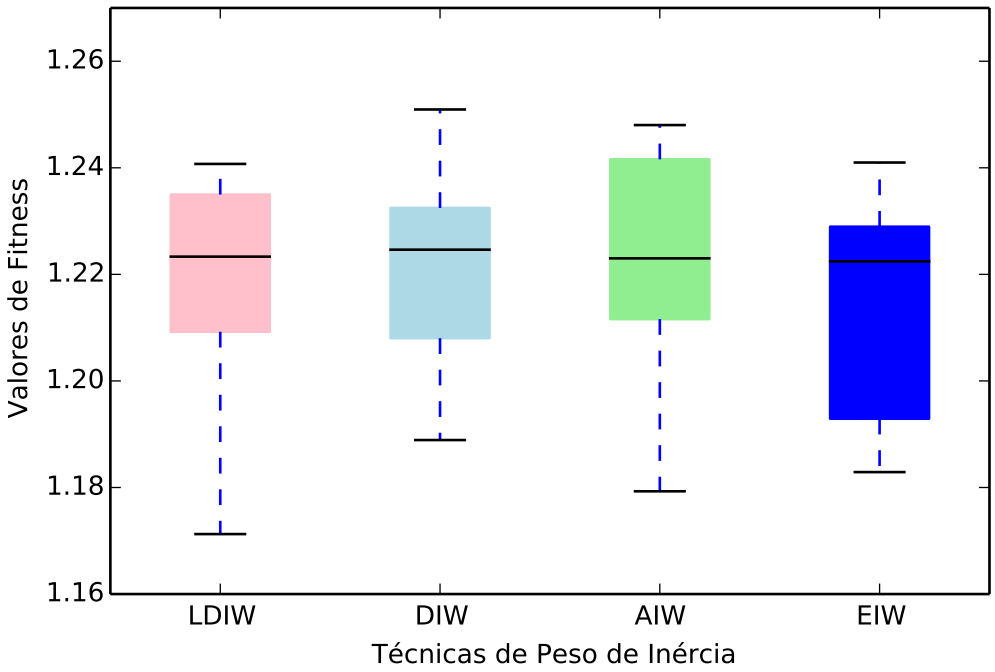
Fonte: a autora.

5.4.1.1 Análise dos Resultados

Com base nos resultados das tabelas, a técnica com o melhor desempenho é a DIW, pois apresenta os melhores valores médios de economia, conforto e *Fitness*, apresentando também o menor desvio padrão. A técnica EIW tem os piores resultados de economia e conforto, de *Fitness*. Em relação ao tempo de execução, as técnicas apresentam valores bem semelhantes, porém a técnica LDIW possui o maior desvio padrão, mostrando-se mais instável que as outras técnicas.

Na Figura 21 está o diagrama de caixa dos valores de *Fitness* para as diferentes técnicas de peso de inércia e $\alpha = 0,3$.

Figura 21 – Diagrama de Caixa - $\alpha = 0,3$



Fonte: A autora

Pelo gráfico pode-se observar que as técnicas LDIW e AIW apresentam as maiores amplitudes, mostrando uma maior dispersão dos resultados. A técnica LDIW é simétrica, porém apresenta valores muito dispersos. A técnica com menor amplitude é a EIW, o que significa que os resultados possuem uma menor dispersão, mostrando-se mais estável. Porém o Q1 é o menor, mostrando que apesar da pouca dispersão, a maioria dos resultados está abaixo da mediana. A técnica DIW também apresenta uma amplitude pequena (perdendo apenas para EIW), além disso apresenta simetria melhor e o Q1 mais

alto. Dessa forma, pelos resultados das tabelas e do gráfico, a técnica DIW apresenta o melhor desempenho, pois possui pouca dispersão e melhores resultados médios.

5.4.2 Caso 2 - $\alpha = 0,5$

Na Tabela 10 estão os resultados para o cenário com a tarifa branca, com o valor de $\alpha = 0,5$. Os resultados mostrados são: custo com o perfil preferencial do usuário (Custo Us), custo após o agendamento (Custo SHC), economia financeira (percentual) e conforto relativo percentual. Os resultados mostrados são os valores médios das 20 execuções.

Tabela 10 – Resultados Cenário 2 ($\alpha = 0,5$)

Técnica	Custo Us (R$)	Custo SHC (R$)	Economia Financeira	Conforto Relativo
LDIW	15,76	12,88	18,21%	82,13%
DIW	15,76	12,93	17,93%	82,01%
AIW	15,76	12,95	17,79%	82,87%
EIW	15,76	13,04	17,20%	80,97%

Fonte: a autora.

Nas Tabelas 11 e 12 está uma comparação do desempenho das quatro técnicas, em termos de valores de função *Fitness* e do tempo de execução Δt, respectivamente. São mostrados os valores piores, melhores e médios, com seus respectivos valores de desvio padrão.

Tabela 11 – Comparação *Fitness* - Cenário 2 ($\alpha = 0,5$)

	Valor de *Fitness*			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	1,1695	1,2084	1,2461	0,020
DIW	1,1435	1,2068	1,2483	0,030
AIW	1,1677	1,2073	1,2467	0,024
EIW	1,1469	1,2017	1,2405	0,014

Fonte: a autora.

5.4.2.1 Análise dos Resultados

Com base nos resultados das tabelas, a técnica LDIW apresenta os melhores valores médios de economia, conforto e *Fitness*. As técnicas DIW e AIW apresentam resultados bastante parecidos. Já a técnica EIW apresenta os piores desempenhos. Em

Tabela 12 – Comparação Tempo - Cenário 2 ($\alpha = 0{,}5$)

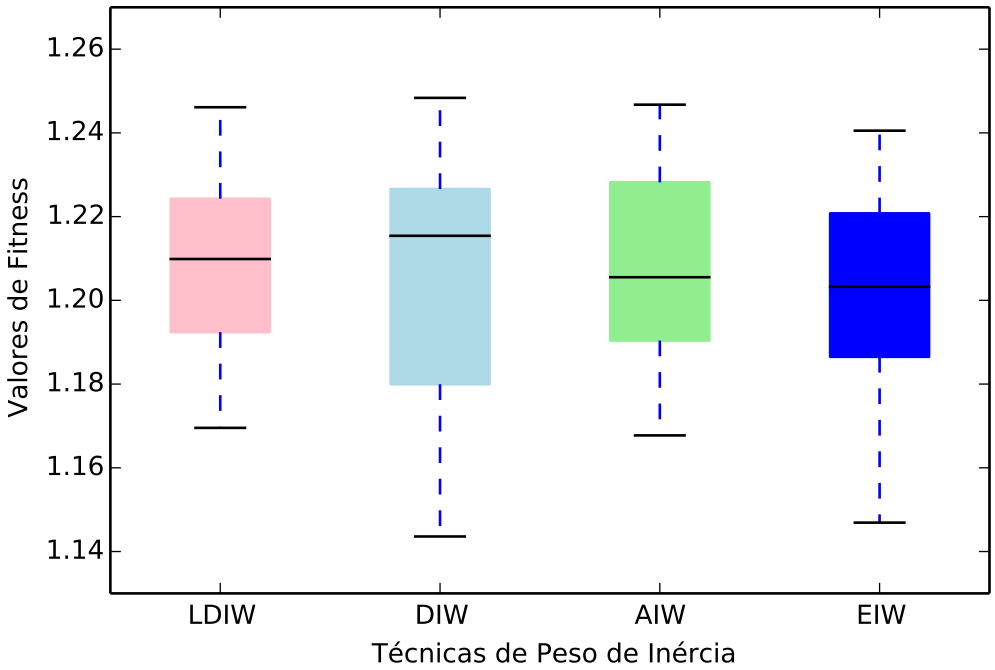
	Δt (segundos)			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	88	62	47	12,59
DIW	84	67	52	8,20
AIW	91	68	51	12,46
EIW	85	59	49	10,25

Fonte: a autora.

relação ao tempo, as quatro técnicas possuem resultados bem semelhantes, porém a técnica DIW apresenta um desvio padrão menor, em relação às outras, mostrando uma maior estabilidade no que se refere ao tempo de execução.

Na Figura 22 está o diagrama de caixa dos valores de *Fitness* para as diferentes técnicas de peso de inércia e $\alpha = 0{,}5$.

Figura 22 – Diagrama de Caixa - $\alpha = 0{,}5$



Fonte: A autora

Pelo gráfico pode-se observar que as técnicas DIW e EIW apresentam as maiores amplitudes, sendo que DIW é ainda pior, pois , além de ter uma maior amplitude, o limite inferior é mais baixo e possui uma assimetria negativa. As técnicas com menor amplitude são LDIW e AIW, mostrando resultados menos dispersos, sendo que a LDIW ainda se sobressai, pois apresenta, além de uma baixa dispersão, uma simetria nos resultados.

Nesse caso, pelos resultados das tabelas e do gráfico, a técnica LDIW apresenta o melhor desempenho.

5.4.3 Caso 3 - $\alpha = 0,7$

Na Tabela 13 estão os resultados para o cenário com a tarifa branca, com o valor de $\alpha = 0,7$. Os resultados mostrados são: custo com o perfil preferencial do usuário (Custo Us), custo após o agendamento (Custo SHC), economia financeira (percentual) e conforto relativo percentual. Os resultados mostrados são os valores médios das 20 execuções.

Tabela 13 – Resultados Cenário 2 ($\alpha = 0,7$)

Técnica	Custo Us (R$)	Custo SHC (R$)	Economia Financeira	Conforto Relativo
LDIW	15,76	12,83	18,58%	82,20%
DIW	15,76	12,75	19,09%	82,31%
AIW	15,76	12,63	19,83%	82,63%
EIW	15,76	12,89	18,16%	80,99%

Fonte: a autora.

Nas Tabelas 14 e 15 está uma comparação do desempenho das quatro técnicas, em termos de valores de função *Fitness* e do tempo de execução Δt, respectivamente. São mostrados os valores piores, melhores e médios, com seus respectivos valores de desvio padrão.

Tabela 14 – Comparação *Fitness* - Cenário 2 ($\alpha = 0,7$)

	Valor de *Fitness*			Desvio Padrão
	(Pior)	(Média)	(Melhor)	σ
LDIW	1,1191	1,1988	1,2495	0,030
DIW	1,1371	1,2042	1,2448	0,035
AIW	1,1422	1,2096	1,2550	0,026
EIW	1,1347	1,1965	1,2489	0,032

Fonte: a autora.

5.4.3.1 Análise dos Resultados

Com base nos resultados das tabelas, a técnica AIW apresenta os melhores valores médios de economia, conforto e *Fitness*. Já a técnica EIW apresenta os piores resultados. As técnicas LDIW e DIW apresentam resultados semelhantes. Em relação ao

Tabela 15 – Comparação Tempo - Cenário 2 ($\alpha = 0{,}7$)

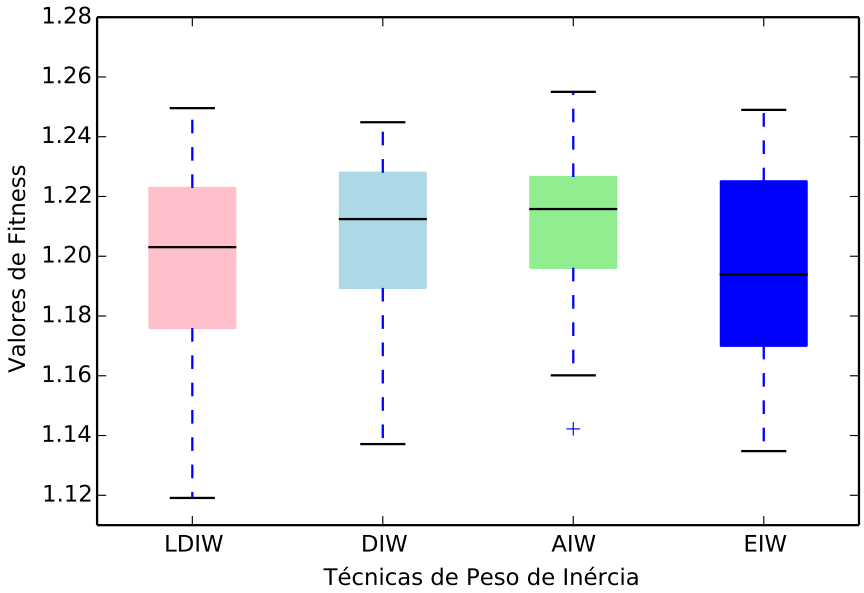
	Δt (segundos) (Pior)	(Média)	(Melhor)	Desvio Padrão σ
LDIW	79	58	48	8,04
DIW	74	63	52	6,94
AIW	65	56	47	4,86
EIW	82	62	50	9,99

Fonte: a autora.

tempo de execução, as quatro técnicas possuem resultados bem semelhantes, porém as técnicas AIW e DIW apresentam um desvio padrão menor, em relação às outras, mostrando uma maior estabilidade no que se refere ao tempo de execução.

Na Figura 23 está o diagrama de caixa dos valores de *Fitness* para as diferentes técnicas de peso de inércia e $\alpha = 0,7$.

Figura 23 – Diagrama de Caixa - $\alpha = 0,7$



Fonte: A autora

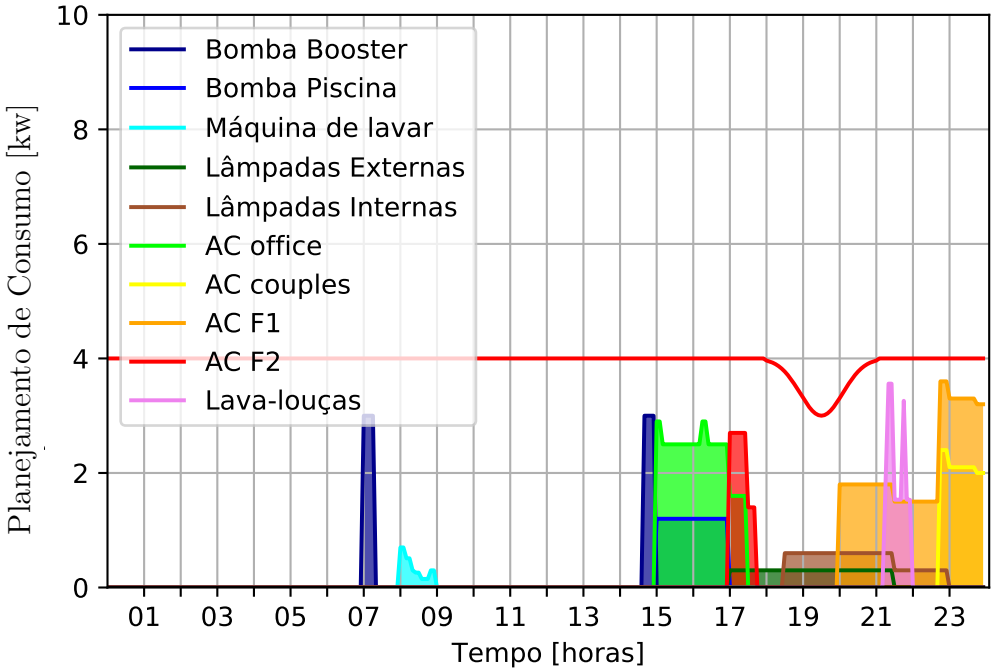
Pelo gráfico pode-se observar que as técnicas LDIW e EIW apresentam as maiores amplitudes, sendo que LDIW é ainda pior, pois apresenta amplitude maior e limite inferior mais baixo. As técnicas com menor amplitude são DIW e AIW. A técnica AIW apresenta uma medida discrepante (*outliers*), porém mesmo assim essa medida ainda é menor do que o limite inferior das outras técnicas e sua amplitude é a menor, mostrando resultados menos dispersos. Nesse caso, pelos resultados das tabelas e do gráfico, a técnica

AIW apresenta o melhor desempenho.

5.4.4 Gráficos de distribuição das Cargas ao longo do dia

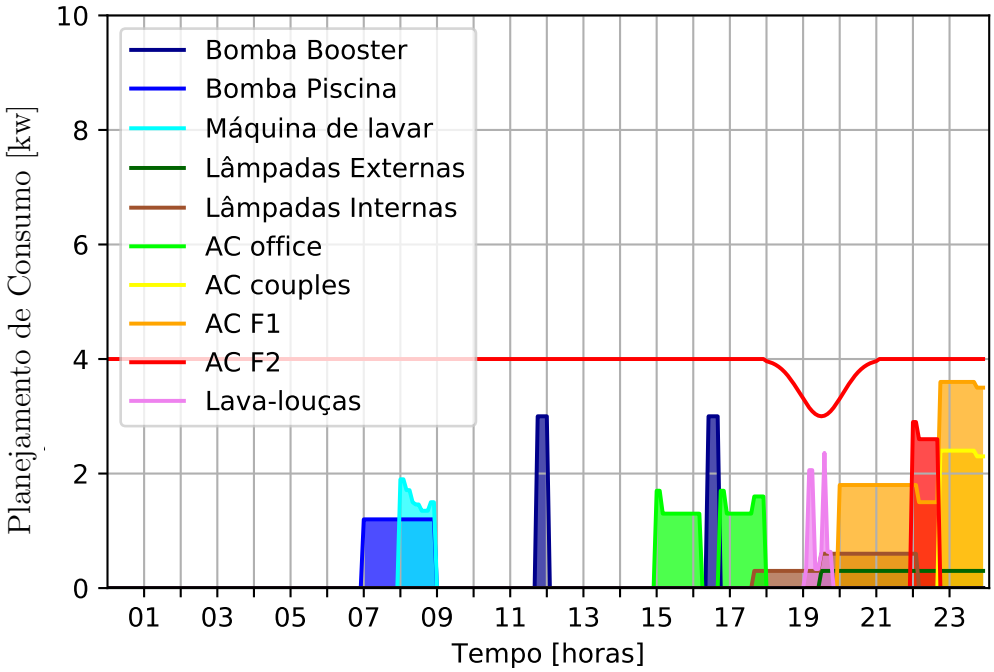
Nas Figuras 24, 25, 26 e 27 é mostrado o resultado do agendamento para as quatro técnicas diferentes, com o valor de $\alpha = 0{,}5$.

Figura 24 – Agendamento Cenário 2 - LDIW



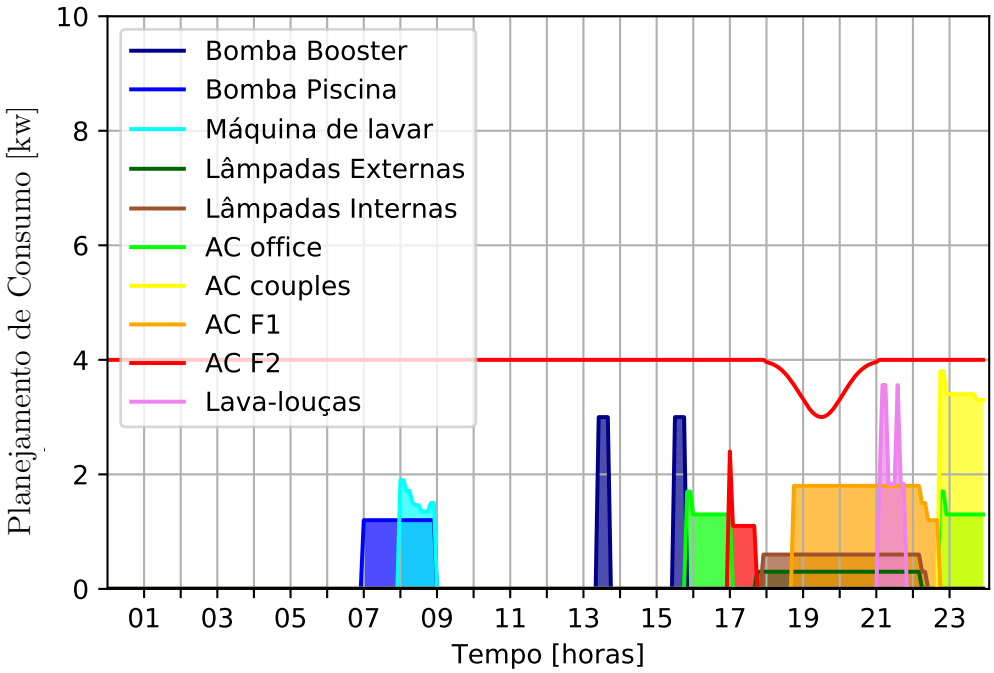
Fonte: A autora

Figura 25 – Agendamento Cenário 2 - DIW



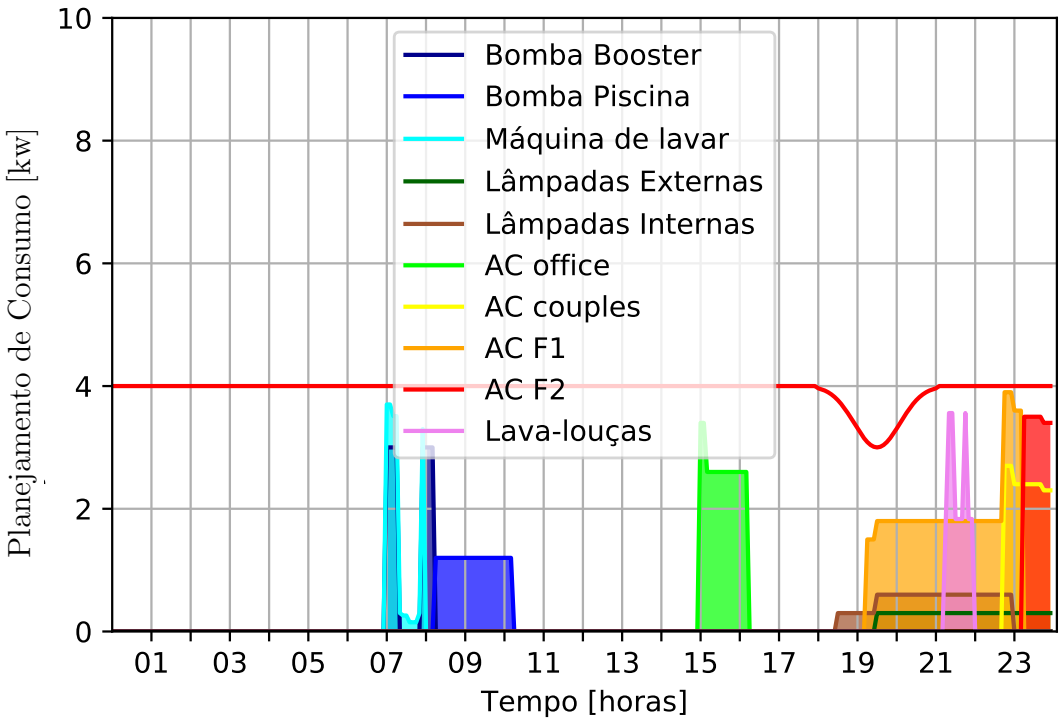
Fonte: A autora

Figura 26 – Agendamento Cenário 2 - AIW



Fonte: A autora

Figura 27 – Agendamento Cenário 2 - EIW



Fonte: A autora

5.5 Análises e Discussões

A Tabela 16 foi construída com base nos resultados dos valores médios de *Fitness*, economia e conforto, e também analisando o gráfico Diagrama de Caixa apresentados no Capítulo 5. Na Tabela é feita a relação das técnicas que tiveram pior e melhor desempenho em relação aos resultados médios e a estabilidade (menor dispersão) desses resultados.

Tabela 16 – Análise dos Resultados

Cenário	**Desempenho (Valores Médios)**		**Estabilidade (Menor Dispersão)**	
	Pior	Melhor	Pior	Melhor
Cen.1	LDIW	DIW	LDIW	EIW
Cen.2 $\alpha = 0.3$	EIW	DIW	LDIW	EIW
Cen.2 $\alpha = 0.5$	EIW	LDIW	DIW	LDIW
Cen.2 $\alpha = 0.7$	EIW	AIW	LDIW	AIW

Fonte: a autora.

Pelos resultados apresentados na Tabela 16 pode-se concluir que em relação aos valores médios de *Fitness*, economia e conforto, a técnica EIW foi a pior em todos os casos, porém nos dois primeiros casos (Cen.1 e Cen.2 $\alpha = 0,3$) mostra-se a mais estável. Já em relação à estabilidade dos resultados, a técnica LDIW se mostra a pior em três dos quatro

casos (Cen.1, Cen.2 $\alpha = 0,3$ e Cen.2 $\alpha = 0,7$). A técnica DIW se mostra a melhor técnica em dois dos quatro casos, porém não apresenta melhor estabilidade em nenhum dos casos e se mostra a menos instável em um dos casos (Cen.2 $\alpha = 0,5$). O tempo de execução é bastante parecido para todas as técnicas, porém a técnica DIW apresenta menor desvio padrão, pelos resultados apresentados na Seção 5 do Capítulo 5.

Pela análise dos resultados, pode-se concluir que, para cada cenário, uma técnica diferente se comporta de forma melhor. Dessa forma, a escolha da técnica de peso de inércia w do PSO estará vinculada ao valor de α que o usuário escolher, sendo escolhida a técnica que melhor se comporta para o dado valor de α. Logo, essas foram as técnicas escolhidas para cada cenário:

- Cenário 1: DIW
- Cenário 2 $\alpha = 0,3$: DIW
- Cenério 2 $\alpha = 0,5$: LDIW
- Cenário 2 $\alpha = 0,7$: AIW

5.6 Conclusões

Neste capítulo foram apresentados os resultados da solução proposta. Inicialmente, foram apresentadas as características do ambiente utilizado para as simulações, os parâmetros escolhidos e as características das cargas utilizadas. Por fim, foram apresentados os dois cenários de simulação com seus respectivos resultados e análises.

6 CONCLUSÕES

Neste trabalho foi apresentada uma arquitetura de um sistema residencial de gerenciamento de energia, que faz uso de um *Smart Home Controller* (SHC), que tem o objetivo de escolher o momento de início de execução de cada carga em uma residência de forma que os custos de execução do conjunto de cargas seja minimizado.

Foi realizado um levantamento do estado da arte sobre os modelos de SHC e suas respectivas técnicas de solução e sobre a técnica de meta-heurística *Particle Swarm Optimization* (PSO). Dessa forma, foi apresentado um modelo matemático de SHC baseado nos modelos apresentados na revisão de estado da arte. O modelo de SHC apresentado faz um *tradeoff* entre o conforto do usuário e o custo com a conta de energia elétrica, respeitando o limite de demanda máxima contratada utilizado.

A solução foi modelada e implementada utilizando o conceito de programação orientada a objetos na linguagem Python, de forma que possam ser adicionadas facilmente outras técnicas meta-heurísticas baseadas em população como alternativa ao PSO. O modelo também permite a adição de cargas flexíveis em relação à potência, apesar de terem sido utilizadas apenas cargas flexíveis em relação ao tempo.

O sistema foi validado utilizando cargas de uma residência inteligente real, por meio de simulações em dois cenários: tarifa com valor fixo e tarifa branca, variando a importância dada aos objetivos conflitantes (economia e conforto). O PSO foi testado com quatro técnicas diferentes do parâmetro peso de inércia (w) a fim de comparações. Pelos resultados apresentados, em cada caso, uma técnica se comporta de forma melhor. Sendo assim, para cada caso analisado, será utilizada a técnica que apresenta o melhor desempenho.

Os resultados alcançados pelo agendamento do SHC são apresentados de forma relacional (percentual) com o perfil preferencial, facilitando a compreensão em termos de comparação. Esse é um aspecto diferente da maioria da literatura, em que os resultados de conforto ou desconforto são apresentados com valores adimensionais e não relacionados em percentual. Os resultados alcançados nesta dissertação variam, de acordo com a preferência dada a cada objetivo (economia e conforto), estando em torno de 18,21 a 19,83% de economia financeira e 82,13 a 83,42% de conforto.

A solução do SHC apresentada está em fase de implementação. O algoritmo python está embarcado em uma Raspberry Pi. Também está em fase de desenvolvimento

uma aplicação WEB, que possibilitará ao usuário comunicar-se com o SHC por meio de qualquer dispositivo móvel ou computador.

6.1 Trabalhos Futuros

Como trabalhos futuros, espera-se que as seguintes atividades sejam realizadas:

- Ampliar o modelo para ambiente industrial: o modelo apresentado já leva em consideração um valor limite de pico como uma restrição do problema, o que viabiliza a utilização em ambientes industriais, que são taxados pela demanda máxima, para encontrar a melhor solução do ponto de vista do conforto que não ultrapasse os limites de pico contratados pelo usuário.
- Utilizar uma técnica de otimização multiobjetivo: apesar do problema ser multiobjetivo, o modelo está sendo solucionado com uma combinação linear das funções objetivo. Para uma solução mais robusta, pretende-se implementar um algoritmo multiobjetivo, como os apresentados na subseção 3.2.2 Capítulo 3.

REFERÊNCIAS

ALBUQUERQUE, P. U. B.; OHI, K. A. D.; PEREIRA, N. S.; PRATA, B. A.; BARROSO, G. C. Proposed architecture for energy efficiency and comfort optimization in smart homes. **JCAE - Journal of Control, Springer**, 2018.

AMOOZEGAR, M.; MINAEI-BIDGOLI, B. Optimizing multi-objective pso based feature selection method using a feature elitism mechanism. **Expert Systems With Applications**, p. 499–514, 2018.

ANEEL. **Tarifa Branca**. 2017. Disponível em: <http://www.aneel.gov.br/tarifa-branca>. Acesso em: 17 set. 2018.

ARASOMWAN, M. A.; ADEWUMI, A. **On adaptive chaotic inertia weights in particle swarm optimization**. IEEE Symposium, p. 72–79, 2013.

ARASOMWAN, M. A.; ADEWUMI, A. O. On the performance of linear decreasing inertia weight particle swarm optimization for global optimization. **The Scientific World Journal**, 2013.

BANSAL, J. C.; SINGH, K.; SARASWAT, M.; VERMA, A.; JADON, S. S.; ABRAHAM, A. **Inertia Weight Strategies in Particle Swarm Optimization**. IEEE, 2011.

BEZERRA FILHO, P. de T. F. **Proposição de um Con trolador de Cargas Inteligente Considerando Custo Energético e Conforto Baseado em Programação Linear Inteira.** 2016. 82 f. Dissertação (Mestrado em Engenharia Elétrica) — Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2016.

BEZERRA FILHO, P. de T. F.; PRATA, B. de A.; BARROSO, G. C. **Proposição de um controlador de cargas inteligente considerando custo energético e conforto baseado em programação linear inteira**. Simpósio Brasileiro de Automação Inteligente, 2015.

CLERC, M.; KENNEDY, J. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. **IEEE Trans Evol Comput**, v. 6, p. 58–72, 2002.

COELLO, C.; LECSUGA, M. **MOPSO: a proposal for multiple objective particle swarm optimization**. Proceedings of IEEE Congress on Evolutionary Computation, p. 1051–1056, 2002.

COMéRCIO, J. do. **Após um ano, tarifa branca tem adesão de menos de 1%**. 2019. Disponível em: <https://www.jornaldocomercio.com/_conteudo/cadernos/jc_logistica/2019/02/668475-apos-um-ano-tarifa-branca-tem-adesao-de-menos-de-1.html>. Acesso em: 12 jan. 2019.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. **A fast and elitist multi-objective genetic algorithm: NSGA-II**. IEEE Trans. Evol. Comput. 6, p. 182–197, 2002.

ESMIN, A. A. A. **Generating Fuzzy Rules from Examples Using the Particle Swarm Optimization Algorithm**. Seventh International Conference on Hybrid Intelligent Systems, 2007.

ESTHER, B. P.; KUMAR, K. S. A survey on residential demand side management architecture, approaches, optimization models and methods. **Renewable and Sustainable Energy Reviews**, v. 59, p. 342–351, 2016.

GELLINGS, C. W. **The Concept of Demand-Side Management for Electric Utilities**. Proceedings of the IEEE, v. 73, n. 10, 1985.

GELLINGS, C. W.; SAMOTYJ, M. Smart grid as advanced technology enabler of demand response. **Energy Efficiency**, p. 685–694, 2013.

GIORGIO, A. D.; PIMPINELLA, L. An event driven smart home controller enabling consumer economic saving and automated demand side management. **Applied Energy**, v. 96, p. 92–103, 2012.

GUDI, N.; WANG, L.; DEVABHAKTUNI, V.; DEPURU, S. S. S. R. **Demand Response Simulation Implementing Heuristic Optimization for Home Energy Management**. North American Power Symposium, 2010.

JAVAID, N.; AHMED, F.; ULLAH, I.; ABID, S.; ABDUL, W.; ALAMRI, A.; ALMOGREN, A. S. Towards cost and comfort based hybrid optimization for residential load scheduling in a smart grid. **Energies 2017**, v. 10, p. 1546, 2016.

JIAO, B.; LIAN, Z.; GU, X. A dynamic inertia weight particle swarm optimization algorithm. **Chaos Solitons Fractals**, v. 3, p. 698–705, 2008.

KENNEDY, J.; EBERHART, R. **Particle swarm optimization**. IEEE International Conference on Neural Networks, 1995.

KNOWLES, J.; CORNE, D. Approximating the nondominated front using the pareto archived evolution strategy. **Evol. Comput. 8**, p. 149–172, 2000.

LI, X. A nondominated sorting particle swarm optimizer for multiobjective optimization. p. 37–48, 2003.

LIN, Y.-H.; HU, Y.-C. Residential consumer-centric demand-side management based on energy disaggregation-piloting constrained swarm intelligence: Towards edge computing. **Sensors**, v. 18, p. 1365, 2018.

MA, K.; YAO, T.; YANG, J.; GUAN, X. Residential power scheduling for demand response in smart grid. **Electrical Power and Energy Systems**, v. 78, p. 320–325, 2016.

MANZOOR, A.; JAVAID, N.; ULLAH, I.; ABDUL, W.; ALMOGREN, A.; ALAMRI, A. An intelligent hybrid heuristic scheme for smart metering based demand side management in smart homes. **Energies**, v. 10, p. 1258, 2017.

NICKABADI, A.; EBADZADEH, M.; SAFABAKHSH, R. A novel particle swarm optimization algorithm with adaptive inertia weight. **Appl. Soft Comput**, v. 4, p. 3658–3670, 2011.

OGUNJUYIGBE, A.; AYODELE, T.; AKINOLA, O. User satisfaction-induced demand side load management in residential buildings with user budget constraint. **Applied Energy**, v. 187, p. 352–366, 2017.

RAHIM, S.; JAVAID, N.; AHMAD, A.; KHAN, S. A.; KHAN, Z. A.; ALRAJEH, N.; QASIM, U. Exploiting heuristic algorithms to efficiently utilize energy management controllers with renewable energy sources. **Energy and Buildings**, v. 129, p. 452–470, 2016.

SHI, Y.; EBERHART, R. **A modified particle swarm optimizer**. IEEE World Congress on Computational Intelligence, p. 69–73, 1998.

SIERRA, M.; COELLO, C. **Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance**. Proceedings of International Conference on Evolutionary Multi-Criterion Optimization, p. 505–519, 2005.

SMARTGRID.GOV. **Time Based Rate Programs**. 2013. Disponível em: <https://www.smartgrid.gov/recovery_act/time_based_rate_programs.html>. Acesso em: 17 set. 2018.

SPYDER. **The Scientific Python Development Environment**. 2018. Disponível em: <https://www.spyder-ide.org/>. Acesso em: 03 out. 2018.

TING, T.; SHI, Y.; CHENG, S.; LEE, S. **Exponential inertia weight for particle swarm optimization**. International Conference in Swarm Intelligence, Springer, v. 3, p. 83–90, 2012.

WACKS, K. P. Utility load management using home automation. **IEEE Transactions on Consumer Electronic**, v. 37, n. 2, 1991.

ZHANG, X.; ZHENG, X.; CHENG, R.; QIU, J.; JIN, Y. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. **Information Sciences**, p. 63–76, 2018.

ZHANG, Y.; GONG, D. W.; CHENG, J. Multi-objective particle swarm optimization approach for cost-based feature selection in classification. **IEEE/ACM Transactions on Computation Biology and Bioinformatics**, v. 14, 2017.

APÊNDICE A – ARTIGO CBA 2018

SMART HOME CONTROLLER: OTIMIZAÇÃO MULTI-OBJETIVO UTILIZANDO A META-HEURÍSTICA PSO

Stéphanie A. B. dos Santos,* Daniel K. de A. Ohi,* Pedro U. B. de Albuquerque,* José R. Bezerra,† Giovanni C. Barroso*

*Universidade Federal do Ceará
Fortaleza, Ceará, Brasil

†Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Fortaleza, Ceará, Brasil

Email: stephanie.abraga@gmail.com, kenji.ohi@alu.ufc.br, purbano@ifce.edu.br, jbroberto@yahoo.com.br, gcb@fisica.ufc.br

Abstract— This paper proposes a solution to the Smart Home Controller (SHC) Intelligent Load Controller model using the Particle Swarm Optimization (PSO) meta-heuristic technique. SHC has been modeled so that it can contemplate the energy cost, the comfort of the users or a combination of both. Conventional optimization techniques become very slow in relation to the convergence rate and when the number of devices is part of the demand response (DR), making them unviable for daily use. However, modern techniques based on heuristics have overcome these disadvantages. In this article we compare the results of the SHC solution with Linear Programming and PSO.

Keywords— Smart Home, Energy Efficiency, Optimization, Comfort, Demand Response

Resumo— O presente artigo propõe uma solução do modelo de um controlador de cargas inteligente, *Smart Home Controller* (SHC), utilizando a técnica de meta-heurística PSO (*Particle Swarm Optimization*). O SHC foi modelado de forma que é capaz de contemplar o custo energético, o conforto dos usuários ou uma combinação dos dois. Técnicas de otimização convencionais tornam-se muito lentas em relação à taxa de convergência e quando o número de aparelhos faz parte da resposta à demanda (DR), tornando-as inviáveis para o uso diário. Contudo, técnicas modernas baseadas em heurística superaram essas desvantagens. Neste artigo são comparados os resultados da solução do SHC com Programação Linear e PSO.

Palavras-chave— *Smart Home*, Eficiência Energética, Otimização, Conforto, Resposta à Demanda

Tabela 1: Lista de Símbolos

Símbolo	Descrição
M	Número de cargas planejáveis
$\bar{P_m}$	Vetor da potência média da m-ésima carga
$\hat{P_m}$	Vetor da potência de pico da m-ésima carga
N_m	Duração, em número de amostras, da m-ésima carga
I_{Sm}	Amostra no horário mínimo de início da m-ésima carga
I_{Em}	Amostra no horário máximo de término da m-ésima carga
S	Amostra assoc. ao início do período de planejamento
E	Amostra assoc. ao término do período de planejamento
u_{mi}	i-ésima variável de decisão da m-ésima carga
P_k	Limite de pico no $k-simo$ instante de tempo
C	Vetor do custo do consumo de energia elétrica no período

1 Introdução

O consumo de energia é um dos principais indicadores do desenvolvimento econômico e do nível de qualidade de vida de uma sociedade. Esse indicador reflete o ritmo de atividade dos setores industrial, comercial e de serviços, bem como a capacidade da população de adquirir bens e serviços tecnologicamente mais avançados, como automóveis, eletrodomésticos e eletroeletrônicos (ANEEL, 2008).

O sistema de energia moderno incorporado à rede inteligente gerencia a demanda de eletricidade selecionando as prioridades do usuário por meio da comunicação nos dois sentidos. Em *Smart Grid*, o uso extensivo de comunicação e automação tornam inteligentes tanto a rede de distribuição de energia quanto a *Demand Side Management* (DSM), que consiste nos usuários da energia elétrica realizar ações afim de equilibrar a demanda, principalmente em momentos de pico (Rehman et al., 2016).

Considerando o aumento da eficiência do uso de energia elétrica e se utilizando dos conceitos de *Smart Grids*, pode-se pensar em um controlador de cargas integrado à rede inteligente de energia, que possa alocar as cargas dos clientes para momentos de tarifa reduzida como uma estratégia de diminuição do consumo de energia elétrica nos mo-

mentos de pico.

Em (de T. F. B. Filho et al., 2015), os autores apresentaram um *Smart Home Controller* (SHC) modelado como um problema de otimização utilizando programação linear inteira que é capaz de contemplar o custo energético, o conforto dos usuários ou uma combinação dos dois.

Segundo (Rehman et al., 2016), no cenário de *Smart Home*, as técnicas de otimização convencionais, por exemplo, programação linear (Yong e Choi, 2014), programação não linear (Yong e Choi, 2014), programação convexa (Richard e Pistikopoulos, 2016), são muito lentas em relação à taxa de convergência e tornam-se muito demoradas quando o número de aparelhos faz parte da resposta da demanda, do inglês *Demand Response* (DR), tornando-as inviáveis para o uso diário. Contudo, técnicas modernas baseadas em heurística, como PSO (*Particle Swarm Optimization*), ACO (*Ant Colony optimization*) (Raka et al., 2016) e GA (*Genetic Algorithm*)(et al, 2013) superaram essas desvantagens.

O presente artigo propõe uma solução do modelo do SHC proposto em (de T. F. B. Filho et al., 2015), utilizando a técnica de metaheurística PSO, comparando o desempenho obtido com o apresentado em (de T. F. B. Filho et al., 2015).

O artigo está estruturado da seguinte forma: a Seção 2 contém uma explanação da metaheurística PSO, na Seção 3 está descrito o modelo do SHC, a Seção 4 descreve como foi solucionado o problema de otimização, a Seção 5 descreve os cenários de referência utilizados para as simulações, na Seção 6 estão os resultados das simulações, na Seção 7 está a conclusão e trabalhos futuros e, finalmente, os agradecimentos.

2 PSO (*Particle Swarm Optimization*)

O algoritmo (PSO) é um método de otimização baseado em população que encontra a solução ideal usando uma população de partículas (Kennedy e Eberhart, 1995). Cada enxame de PSO é uma solução possível no espaço da soluções. A definição de PSO é apresentado da seguinte forma:

- Cada partícula individual i tem as seguintes propriedades: uma posição atual no espaço de busca, x_i, a velocidade atual, v_i e uma melhor posição pessoal em espaço de busca, $pbest_i$.
- A melhor posição pessoal, $pbest_i$, corresponde à posição no espaço de busca onde a partícula i apresenta o menor erro conforme determinado pela função objetivo f.
- A melhor posição global, indicada por $gbest$, representa a posição que produz o menor erro entre todos os $pbest_i$.

As equações (1) e (2) definem como o $pbest_i$ e $gbest$ são atualizados no tempo t, respectivamente. É assumido que o enxame possui s partículas, em um problema de minimização da função f.

$$pbest_i(t+1) = \begin{cases} pbest_i(t) \ se \ f(pbest_i) \leq f(x_i(t+1))) \\ x_i(t+1) \ se \ f(pbest_i) > f(x_i(t+1))) \end{cases} \quad (1)$$

$$gbest(t+1) = min\{f(pbest), f(gbest)\} \quad (2)$$

$$pbest \in \{pbest_0, pbest_1, ..., pbest_s\}$$

A velocidade e posição atual das partículas são atualizadas como apresentado nas equações (3) e (4):

$$v_i(t+1) = wv_i(t) + c_1 r_1(pbest_i - x_i) + c_2 r_2(gbest - x_i) \quad (3)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

em que c_1 e c_2 são duas constantes positivas, r_1 e r_2 são dois números aleatórios dentro do intervalo [0, 1], e w é o peso de inércia. A equação (3) consiste em três partes. A primeira parte é a velocidade anterior do enxame, que mostra o estado atual do enxame; a segunda parte é o termo cognitivo, que expressa a experiência individual da partícula de onde a solução está; a terceira parte é o termo social, que representa a experiência da nuvem. As três partes em conjunto determinam a capacidade do espaço de pesquisa (Esmin, 2007).

3 Modelo do SHC

O modelo de SHC apresentado em (de T. F. B. Filho et al., 2015) baseia-se no modelo apresentado em (Giorgio e Pimpinella, 2013), cujo objetivo é escolher o momento de início de execução de cada carga em uma residência de forma que os custos de execução do conjunto de cargas seja minimizado. As decisões do SHC são tomadas baseadas nos seguintes parâmetros: custo das tarifas ao longo do dia, demanda máxima contratada, consumo de energia de cada carga, horário mínimo de início de cada carga (escolha do usuário), e horário máximo de término de cada carga (escolha do usuário).

Dado que o SHC é um sistema de tempo discreto que opera a uma certa taxa de amostragem T_s, que o dia é dividido em N amostras e que uma residência possui M cargas planejáveis, o modelo matemático do SHC, descrito como um problema de programação inteira, é apresentado a seguir. A notação dos símbolos utilizados está definida na Tabela 1.

A função-objetivo do controlador proposto por (Giorgio e Pimpinella, 2013) é definida pela

equação (5). Esta função define o custo total de execução de todas as cargas presentes no ambiente.

$$f_1(x) = min\{\sum_{m=1}^{M}\sum_{i=I_{sm}}^{I_{em}-N_m} (\sum_{n=1}^{i+(N_m-1)} \bar{P_m}[n-i]T_sC[n])u_{mi}\} \quad (5)$$

Sujeito às restrições:

$$\sum_{i=I_{sm}}^{I_{em}-N_m} u_{mi} = 1 \quad (6)$$

$$\sum_{m\in M_k} (\sum_{i=(k-(N_m-1))}^{k-(k-I_{Em}+N_m)} \hat{P_m}[k-i]u_{mi}) \leq P_k \quad (7)$$

$$u_{mi} \in \{0,1\} \quad (8)$$

A restrição definida pela equação (6) estabelece que, para cada carga, o somatório de todas as suas variáveis de decisão deve ser igual a unidade. Isto significa que para cada carga apenas uma variável de decisão poderá assumir o valor unitário enquanto as demais terão valor zero, garantindo que para cada carga a ser executada o controlador escolherá apenas um momento de início de acionamento. A restrição definida pela equação (7) estabelece que o consumo de pico em qualquer momento durante a execução de todas as cargas deve ser igual ou inferior à curva de limite de pico. E, a restrição definida pela equação (8) estabelece que as variáveis de decisão só podem assumir os valores zero e um, isto é, são variáveis de decisão binárias.

O modelo apresentado em (Giorgio e Pimpinella, 2013) é eficiente em minimizar os custos relacionados ao consumo de energia elétrica mas pode gerar um grande nível de desconforto aos usuários, se o horário de início de acionamento das cargas que estão associadas ao conforto do usuário for muito diferente do horário de início ideal definido, dado que a definição de conforto seja entendida como a distância entre o previsto acionamento da carga e o horário definido. (de T. F. B. Filho et al., 2015).

Para realizar análise sobre o nível de desconforto gerado pelo SHC, os seguintes parâmetros foram introduzidos ao problema por (de T. F. B. Filho et al., 2015).

- C_{Lm} : Nível de conforto da m-ésima carga. É definido como um valor real que varia entre 0, para cargas que não geram desconforto ao ter seu horário de início alterado, e 1 para cargas que geram grande impacto no conforto do usuário ao terem seus horários de início alterados.
- I_{Bm} : Amostra associada com o melhor horário de início da m-ésima carga. Indica ao controlador qual o horário ideal para o início da carga.
- Pesos de controle para otimização do custo e do conforto (w_1) e (w_2).

Estas novas variáveis têm seu valor definido pelo usuário, uma vez que não é possível para o controlador determinar o nível de conforto ou o melhor horário de início de cada carga.

A função-objetivo trabalha para minimizar o deslocamento do horário de início de cada carga em relação ao horário de início ideal definido pelo usuário, maximizando o nível de conforto do usuário. Na equação (9) é definida a função-objetivo.

$$f_2(x) = \min\{\sum_{m=1}^{M}\sum_{i=I_{sm}}^{I_{em}-N_m} (C_{Lm}|i-I_{Bm}|)u_{mi}\} \quad (9)$$

Com a função-objetivo que minimiza o desconforto do usuário definida, é necessário fazer o controlador minimizar a combinação das duas funções-objetivo apresentadas até aqui, isto é, o custo energético e o desconforto do usuário. Para alcançar este resultado o controlador proposto minimiza uma combinação linear das duas funções-objetivo descritas anteriormente, criando assim uma nova função-objetivo, como apresentado na equação (10). Novas restrições foram adicionadas ao problema e estão descritas nas equações (11), (12) e (13).

$$f(x) = w_1 f_1(x) + w_2 f_2(x) \quad (10)$$

onde

$$w_1 + w_2 = 1 \quad (11)$$

$$w_1 \geq 0 \quad (12)$$

$$w_2 \geq 0 \quad (13)$$

f_1 e f_2 foram definidas anteriormente nas equações (5) e (9), respectivamente. w_1 e w_2 são parâmetros de *trade-off*. O conjunto de restrições definidos pelas equações (11) a (13) estabelecem que a soma dos parâmetros w_1 e w_2 deve ser igual a unidade, e que ambos devem ser maior ou igual a zero. Ao variar w_1 e w_2, o usuário pode alcançar economia de energia agregado a um baixo impacto no seu conforto. Por exemplo, ao fazer $w_1 = 1$, o usuário determina que o nível de conforto não é importante para a execução das cargas e o controlador minimizará apenas os custos do consumo de eletricidade.Ao fazer $w_2 = 1$, o controlador obterá a solução que melhor mantém os níveis de conforto do usuário, obedecendo às restrições impostas.

4 Otimização Multi-Objetivo

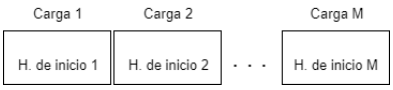
Nesta Seção é descrita a forma como o problema de otimização multi-objetivo foi modelado neste

artigo. Apesar do problema ser multi-objetivo, foi solucionado com uma técnica mono-objetivo, já que a função *Fitness* é um somatório das funções objetivo, ponderadas por parâmetros de *trade-off*. A técnica de meta-heurística PSO utiliza o conceito de população e de função *Fitness*, que estão descritos a seguir.

4.1 População de indivíduos

Como a solução ótima é um conjunto com os melhores horários de início de execução das M cargas, o indivíduo também é composto dessa forma. Na Figura 1 está ilustrado o modelo do indivíduo considerado.

Figura 1: Ilustração do indivíduo



O indivíduo é um vetor com tamanho M, em que cada posição é o horário de início de cada carga.

4.2 Função Fitness

Como o problema de otimização é um problema de minimização com restrições, fez-se necessário escolher uma função *Fitness* que leve em consideração o que se deseja minimizar e as restrições impostas.

O Modelo da função *Fitness* está na equação (14). Na primeira componente da função (f_1), pretende-se minimizar o custo total de execução de todas as cargas, obedecendo às restrições: o valor da potência de pico não exceder o limite de pico para cada amostra no tempo; e que os horários de início estejam entre o intervalo de horário selecionado pelo usuário. Já na segunda componente (f_2), pretende-se minimizar o desconforto gerado. w_1 e w_2 são parâmetros de *trade-off* entre (f_1) e (f_2) e obedecem as restrições já definidas pelas equações (11) a (13) . As equações (15) e (16) descrevem como foram calculadas as parcelas f_1 e f_2 da função *Fitness*, respectivamente.

$$fitness = w_1 f_1 + w_2 f_2 + f_3 \tag{14}$$

$$f_1 = \sum_{m=1}^{M} \sum_{i=I_{cm}}^{I_{cm}+N_m} (\bar{P_m}[i] T_s C[i]) \tag{15}$$

$$f_2 = \sum_{m=1}^{M} (C_{Lm} | I_{cm} - I_{Bm} |) \tag{16}$$

em que, I_{cm} é a hora de início escolhida para cada carga m. O restante dos termos já foram descritos anteriormente.

Para incorporar as restrições à função *Fitness*, criou-se a parcela f_3, que representa a não obediência às restrições. Calculou-se f_3 como sendo a soma da quantidade de vezes que a potência de pico excede, ao longo das amostras diárias. Esse valor foi multiplicado por 100, para dar um peso na função *Fitness*.

5 Cenários de Referência

Os cenários utilizados para as simulações foram os mesmos utilizados em (de T. F. B. Filho et al., 2015), a fim de comparar os resultados obtidos. O conjunto de cargas e suas características que são utilizadas nas simulações dos cenários estão descritas na Tabela 2. O intervalo de amostragem usado foi 5 minutos e o melhor horário de início para cada carga foi 18h. Os parâmetros hora mínima e máxima de início de cada carga, foram, respectivamente, 14h e 23h59 para todas as cargas.

O limite de pico utilizado nas simulações foi de 4,5 kWh. Para simular cargas não-planejáveis, isto é, cargas existentes no ambiente mas que seus horários de execução não podem ser alterados, este limite é reduzido utilizando duas gaussianas de 0,5 kWh de pico e uma gaussiana de 1,4 kWh de pico. As gaussianas estão centradas às 06h, 13h e 20h horas, respectivamente. Os perfis de tarifas utilizadas estão mostrados na Tabela 3.

5.1 Cenário 1 - Tarifa Única

Esse cenário utiliza a tarifa constante durante todo o dia (perfil 1 da Tabela 3), situação similar a como é realizada a tarifação residencial atualmente no Brasil. Neste tipo de situação o controlador trabalha para manter as cargas dentro do intervalo desejado pelo usuário e para garantir que o limite de pico não seja ultrapassado. Não há ganhos econômicos em acionar as cargas em momentos distintos do dia. Logo, para este tipo de cenário, qualquer solução que mantenha o conjunto de cargas abaixo do limite de pico e atenda aos requisitos do usuário de tempo de início e de término do acionamento de cada carga, é uma solução ótima do ponto de vista de economia energética.

5.2 Cenário 2 - Tarifa Dupla

Neste cenário é utilizado o perfil 2 da Tabela 3, que utiliza duas tarifas ao longo do dia. Portanto, do ponto de vista de economia energética, alocar cargas entre as 08h e 19h horas é menos vantajoso.

5.3 Cenário 3 - Tarifa Dupla e DSM

Neste cenário deseja-se simular uma situação com tarifa dupla juntamente com *Demand Response*, um tipo de DSM (*Demand Side Management*). A

Tabela 2: Características das cargas

Carga	Etapas	$\Delta t(min)$	$\overline{P}[kw]$	$\widehat{P}[kw]$	C_{Lm}
1	6	[5, 10, 15, 5, 5, 10]	[0.02, 1.96, 0.02, 0.02, 0.02, 0.05]	[0.15, 2.10, 0.15, 0.15, 0.20, 0.55]	1,0
2	1	[105]	[2.36]	[2.70]	0,9
3	7	[5, 25, 20, 5, 10, 10, 20]	[0.04, 1.99, 0.28, 0.06, 0.06, 0.06, 0.08]	[0.20, 2.10, 2.10, 0.20, 0.30, 0.25, 0.50]	0,8
4	6	[15, 30, 10, 5, 20, 50]	[0.07, 1.40, 0.10, 0.07, 2.02, 0.01]	[0.10, 2.10, 1.20, 0.10, 2.15, 0.02]	0,7
5	8	[25, 5, 60, 20, 10, 10, 10, 20]	[0.27, 0.05, 2.10, 0.11, 0.11, 0.10, 0.10, 0.26]	[2.10, 0.30, 2.20, 0.20, 0.60, 0.80, 0.80, 1.10]	0,6
6	6	[20, 15, 35, 10, 20, 50]	[0.07, 2.00, 0.07, 0.07, 1.80, 0.01]	[0.10, 2.10, 0.10, 0.25, 2.30, 0.02]	0,5
7	3	[50, 20, 50]	[0.80, 0.50, 1.00]	[1.00, 0.80, 1.20]	0,4
8	4	[20, 20, 10, 15]	[1.40, 0.50, 0.60, 1.00]	[1.60, 0.80, 0.60, 1.00]	0,3
9	3	[30, 20, 30]	[0.60, 0.70, 1.00]	[0.60, 0.70, 1.00]	0,2

Tabela 3: Preços Diários de Energia

	Tarifa	Período	Preço [cent/kwh]
1	Padrão	[00h - 24h]	18,00
2	Fora-Ponta	[00h - 08h] e [19h - 24h]	16,75
	Ponta	[08h - 19h]	21,22
3	Fora-Ponta	[00h - 08h] e [19h - 24h]	16,75
	Ponta	[08h - 16h] e [18h - 19h]	21,22
	DSM	[16h - 18h]	12,00

concessionária de energia elétrica envia uma mensagem aos seus usuários, utilizando, por exemplo, o conceito de redes inteligentes, informando sobre uma tarifa promocional. Com o controlador integrado à rede da concessionária este tipo de mensagem é recebida e pode então ser levada em consideração para os cálculos de otimização das cargas que ainda não iniciaram sua execução.

É utilizado o perfil 3 da Tabela 3, que utiliza três tarifas ao longo do dia. Portanto é mais vantajoso, do ponto de vista de economia energética, alocar o máximo de cargas no período de 16h às 18h e em seguida após as 19h.

5.4 Cenário 4 - Tarifa Dupla e DSM com Redução de Pico

Semelhante ao cenário 3, é utilizado o perfil 2 da Tabela 3. Neste cenário também deseja-se simular uma situação com tarifa dupla juntamente com DSM (*demand side management*), porém com uma redução no limite de pico. A concessionária de energia elétrica envia uma mensagem aos seus usuários, utilizando, por exemplo, o conceito de redes inteligentes, informando sobre uma redução de 1,5 kW no limite de pico entre as 21h e 22h. Com o controlador integrado à rede da concessionária, este tipo de mensagem é recebida e pode então ser levada em consideração para os cálculos de otimização das cargas que ainda não iniciaram sua execução.

6 Resultados e Discussões

Os parâmetros de simulação para PSO foram:

- Inicialização da População: aleatoriamente, dentro do intervalo escolhido pelo usuário, para hora inicial e final de início de carga;
- Tamanho da População: 20 indivíduos;
- Peso de inércia: $w = 0,5$;
- Parâmetro cognitivo: $c_1 = 1$;
- Parâmetro social: $c_2 = 4$;
- Critério de parada: número de iterações igual a 500;

Para se escolher esses parâmetros foram feitos testes e comparações, utilizando o cenário 2 como referência. Na Tabela 4 estão os valores obtidos da função *Fitness* com os diferentes valores de c_1 e c_2. O melhor valor de função *Fitness* encontrado foi quando utilizou-se $c_1 = 1$ e $c_2 = 4$.

Tabela 4: Relação dos Parâmetros PSO e Função *Fitness*

c_1	c_2	**Função *Fitness***
1	4	180,36
2	2	193,46
4	1	214,50

Com o algoritmo proposto (PSO), foram realizadas 4 simulações, uma para cada cenário apresentado, com todas as respectivas características citadas na Seção 5.

Da mesma forma, foi executado o algoritmo com programação linear, a fim de comparar os resultados. O computador utilizado para execução dos ambientes simulados possui as seguintes características:

- Processador: Intel Core i3-4005U @ 1,70 GHz;
- Memória RAM: 4,00 GB;
- Disco Rígido: HD 500 GB;
- Sistema Operacional: Microsoft Windows 8.1 64bits;

Nas Tabelas 5, 6, 7 e 8 estão os resultados obtidos para cada cenário, respectivamente, em comparação com os resultados em (de T. F. B. Filho et al., 2015). Cada simulação foi executada 3 vezes, variando os parâmetros de *trade-off* entre o custo e o desconforto (w_1 e w_2) da função *Fitness*:

a)$w_1 = 0,3$ e $w_2 = 0,7$

b)$w_1 = 0,5$ e $w_2 = 0,5$

c)$w_1 = 0,7$ e $w_2 = 0,3$

O custo está medido em cent e o tempo de execução em segundos.

Tabela 5: Resultados - Cenário 1

Cenário 1		**Custo**	**Desconforto**	**Tempo de Execução**
Prog. Linear	a	248,7	6,18	254,17
	b	248,7	6,18	251,48
	c	248,7	6,18	270,96
PSO	a	248,7	8,44	6,71
	b	248,7	11,00	5,25
	c	248,7	9,46	5,23

Tabela 6: Resultados - Cenário 2

Cenário 2		**Custo**	**Desconforto**	**Tempo de Execução**
Prog. Linear	a	241,07	9,99	465,37
	b	237,03	13,52	60,84
	c	237,03	13,52	26,04
PSO	a	260,56	10,22	5,05
	b	258,36	9,24	5,19
	c	257,00	11,00	5,04

No cenário 1, a tarifa é constante ao longo do dia, então não há economia em mover as cargas ao longo do dia. Logo, o custo nas duas técnicas foram os mesmos, porém na otimização com PSO houve um incremento do valor de desconforto. Em relação ao tempo de execução, o algoritmo com programação linear tem um tempo aproximadamente 50 vezes maior do que o algoritmo com PSO.

Tabela 7: Resultados - Cenário 3

Cenário 3		**Custo**	**Desconforto**	**Tempo de Execução**
Prog. Linear	a	201,28	7,09	69,26
	b	201,28	7,09	13,79
	c	199,79	9,99	4,62
PSO	a	222,9	10,66	5,17
	b	215,92	8,84	5,05
	c	217,20	9,94	5,09

Tabela 8: Resultados - Cenário 4

Cenário 4		**Custo**	**Desconforto**	**Tempo de Execução**
Prog. Linear	a	244,10	9,66	155,06
	b	247,36	9,54	85,59
	c	243,08	10,8	279,61
PSO	a	260,58	10,60	5,17
	b	260,50	10,76	4,95
	c	256,38	11,88	5,13

No cenário 2, os custos na otimização com PSO tiveram um incremento em média de 8,4%, porém o nível de conforto foi melhor, tendo um decremento de, em média, 13,3% no desconforto. Em relação ao tempo de execução, o algoritmo com programação linear apresenta um tempo de execução menor, se comparado ao cenário 1, porém o algoritmo com PSO mantém a mesma média de tempo de execução, continuando com o tempo de execução consideravelmente menor do que com programação linear.

Nos cenário 3, os custos tiveram um aumento de 8,9% na otimização com PSO e o nível de desconforto teve um aumento de 24,8%, em média. Já no cenário 4, os custos tiveram um aumento de 5,8% na otimização com PSO e o nível de desconforto teve um aumento de 10,8%, em média. Apesar do custo e nível de desconforto serem um pouco maiores no caso do PSO, o tempo de execução com PSO foi consideravelmente menor na maioria dos casos. Isso mostra que, apesar de não apresentar a solução ótima, PSO apresenta uma solução boa, com um menor tempo de execução.

Nas Figuras (2, 3, 4, 5) estão os gráficos da distribuição das cargas, ao longo do dia, para cada cenário. Foram considerados os valores de pesos de controle de $w_1 = 0,5$ e $w_2 = 0,5$.

7 Conclusões e Trabalhos Futuros

Neste artigo foi solucionado um problema de otimização do SHC proposto por (de T. F. B. Filho et al., 2015), utilizando a técnica de metaheurísticas PSO, comparando os desempenhos

Figura 2: Cenário 1 - PSO

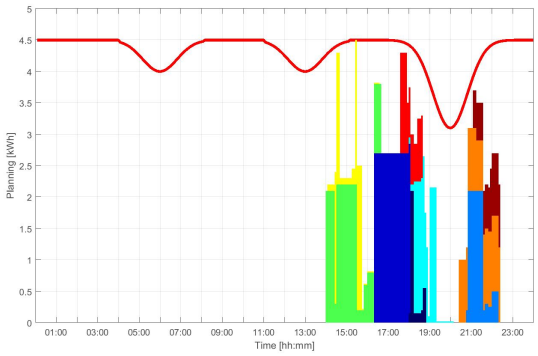


Figura 3: Cenário 2 - PSO

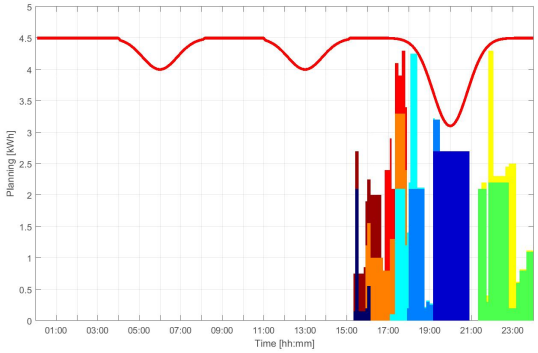


Figura 4: Cenário 3 - PSO

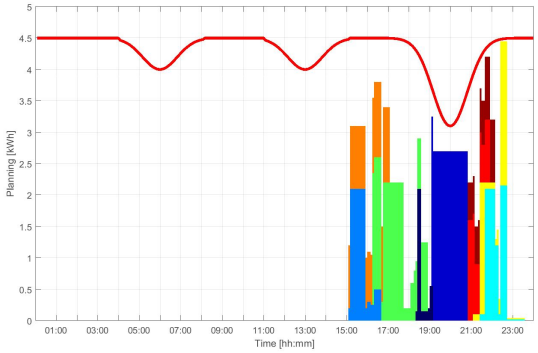
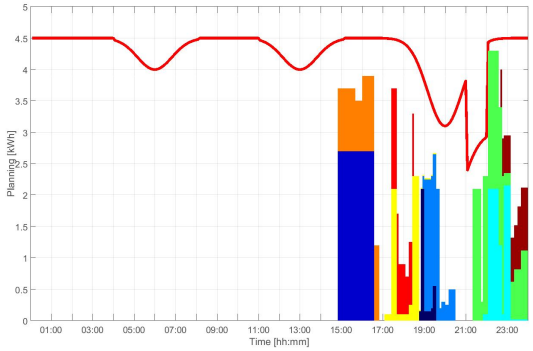


Figura 5: Cenário 4 - PSO



com a técnica utilizada no artigo referência (Programação Linear).

Pelos resultados obtidos, pôde-se observar que, em relação ao custo total de execução de todas cargas, o desempenho da otimização com PSO foi um pouco inferior ao artigo referência, com um aumento de aproximadamente 8,5%. Já em relação ao nível de desconforto, em alguns casos isolados a otimização aqui proposta (PSO) torna-se melhor, apresentando um desconforto menor, porém de forma geral, apresenta um desconforto, aproximadamente, 12% maior que o artigo referência. Porém, pelos resultados apresentados, em relação ao tempo de execução do algoritmo, o PSO é consideravelmente mais rápido do que a programação Linear.

Dessa forma, apesar da programação linear obter a resposta ótima, seu tempo de execução é bem maior do que o PSO, tornando-se inviável quando o número de cargas aumentar. No caso apresentado (com apenas 9 cargas), o algoritmo com PSO demora menos de 10 segundos, enquanto que com programação linear demora em média 3 minutos. A medida que o número de cargas aumentar, o tempo de execução também aumenta, dessa forma usar uma técnica de otimização com um tempo de execução grande, inviabiliza a ideia de embarcar a solução.

Como trabalhos futuros, pretende-se testar o algoritmo PSO em condomínio de residências e com múltiplos acionamentos das cargas, levando em consideração as diferentes etapas de cada uma.

Agradecimentos

Os autores agradecem a Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) pelo apoio financeiro.

Referências

ANEEL (2008). Atlas de energia elétrica no brasil 3a edição, `http://www2.aneel.gov.br/arquivos/pdf/atlas_par1_cap2.pdf`.

de T. F. B. Filho, P., de Albuquerque, P. U. B. e Bruno de A. Prata, G. C. B. (2015). Proposição de um controlador de cargas inteligente considerando custo energético e conforto baseado em programação linear inteira, *Simpósio Brasileiro de Autimação Inteligente* .

Esmin, A. A. A. (2007). Generating fuzzy rules from examples using the particle swarm optimization algorithm, *Seventh International Conference on Hybrid Intelligent Systems* .

et al, Z. Z. (2013). An optimal power scheduling method for demand response in home energy management system., *Smart Grid, IEEE Transactions* .

Giorgio, A. D. e Pimpinella, L. (2013). An event driven smart home controller enabling consumer economic saving and automated demand-side management, *Elsevier Applied Energy* .

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization, *IEEE International Conference on Neural Networks* .

Raka, J., Tuba, M. e Voss, S. (2016). An ant colony optimization algorithm for partitioning graphs with supply and demand., *Applied Soft Computing* .

Rehman, N. U., Rahim, H., Ahmad, A., Khan, Z. A., Qasim, U. e Javaid, N. (2016). Heuristic algorithm based energy management system in smart grid, *10th International Conference on Complex, Intelligent, and Software Intensive Systems* .

Richard, O. e Pistikopoulos, E. N. (2016). Multi-objective optimization with convex quadratic cost functions: A multi-parametric programming approach., *Computers Chemical Engineering* .

Yong, L. J. e Choi, S. G. (2014). "linear programming based hourly peak load shaving method at home area." advanced communication technology (icact), *16th International Conference on. IEEE* .

APÊNDICE B – CÓDIGOS-FONTES PYTHON

```
# -*- coding: utf-8 -*-
"""
# Universidade Federal do Ceará
# Mestrado em Engenharia Elétrica
# Autor:  Stéphanie A. Braga dos Santos
# Novo Modelo de SHC (Main)
"""
import pyrebase
import Load
import newPSOalgorithm
import DataPso
import numpy as np
import scipy as sp
from scipy import signal
import matplotlib.patches as patches
import matplotlib.pyplot as plt

import time

##############################################
#                CARGAS REAIS                #
##############################################

def cenarioPU():
    l1 = Load.TimeFlexLoad('Bomba booster', 1, 7, 17, [8,16], [20], True, [2], [3], 0.1)
    l2 = Load.TimeFlexLoad('Bomba Piscina', 1, 7, 17, [8], [120], False, [0.75], [1.2], 0.1)
    l3 = Load.TimeFlexLoad('maquina de lavar', 8, 7, 17, [8], [10, 10, 5, 10, 5, 5, 5, 10], False, [0.13,0.51,0.3,0.26,0.15,0.15,0.15,0.22], [0.7,0.51,0.3,0.26,0.15,0.15,0.15,0.3], 0.5)
    l4 = Load.TimeFlexLoad('lâmpadas externas', 1, 17, 24, [18], [270], False, [0.3], [0.3], 0.3);
    l5 = Load.TimeFlexLoad('lâmpadas internas', 1, 17, 23, [18], [270], False, [0.15],[0.3], 0.7);
    l6 = Load.TimeFlexLoad('AC office', 14, 15, 24, [16,20], [10, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5], True,
```

```
    [1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3],
    [1.7,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3], 1);
31  l7 = Load.TimeFlexLoad('AC couples', 7, 17, 24, [20], [30, 20, 5, 5,
    5, 5, 5], False, [2,2,2,2,2,2,2], [2.1,2.1,2.1,2.1,2,2,2], 1);
32  l8 = Load.TimeFlexLoad('AC F1', 1, 17, 24, [20], [240], False, [1.1],
     [1.2], 1);
33  l9 = Load.TimeFlexLoad('AC F2', 7, 17, 24, [20], [10, 10, 5, 5, 5, 5,
     5], False, [0.9,0.9,0.9,0.9,0.9,0.9,0.9],
    [1.1,1.1,1.1,1.1,1.1,1.1,1.1], 1);
34  l10 = Load.TimeFlexLoad('lava-louças', 5, 18, 22, [21], [5, 10, 15,
    5, 10], False, [0.033,1.76,0.033,1.76,0.033],
    [0.033,1.76,0.033,1.76,0.033], 0.3);
35
36  LOADS = [l1, l2, l3, l4, l5, l6, l7, l8, l9, l10]
37  return LOADS
38
39
40 #==================== INICIO DO MAIN
       ===================================
41
42 ################################################
43 #     PARÂMETROS/VALORES PRÉ-DEFINIDOS         #
44 ################################################
45 MINUTS_IN_ONE_DAY = (60*24) #Minutos em 1 dia
46 SAMPLE_INTERVAL = 5 # Intervalo de amostragem (Minutos)
47 ONE_HOUR_IN_SAMPLES = (60/SAMPLE_INTERVAL) # 1 Hora em Amostas
48 N_SAMPLES = (MINUTS_IN_ONE_DAY / SAMPLE_INTERVAL) # Amostras em 1 dia
49 FIRST_SAMPLE = 1
50 LAST_SAMPLE = N_SAMPLES
51
52 ############ TARIFAS ##############
53 cambio = 1./3.2
54 TARIFF_P = np.hstack(([0.36*np.ones(int(16.5*ONE_HOUR_IN_SAMPLES))], #
       Fora ponta
55 [1.08*np.ones(int(1*ONE_HOUR_IN_SAMPLES))], # Ponta
56 [1.44*np.ones(int(3*ONE_HOUR_IN_SAMPLES))], # DSM
57 [1.08*np.ones(int(1*ONE_HOUR_IN_SAMPLES))], # Ponta
58 [0.36*np.ones(int(2.5*ONE_HOUR_IN_SAMPLES))]))*cambio # Fora ponta
59
```

```
60
61 ##TARIFA BRANCA
62 TARIFF_B = np.hstack(([0.56355*np.ones(int(17*ONE_HOUR_IN_SAMPLES))], #
       Fora ponta
63 [0.88144*np.ones(int(1*ONE_HOUR_IN_SAMPLES))], # Ponta
64 [1.42294*np.ones(int(3*ONE_HOUR_IN_SAMPLES))], # DSM
65 [0.88144*np.ones(int(1*ONE_HOUR_IN_SAMPLES))], # Ponta
66 [0.56355*np.ones(int(2*ONE_HOUR_IN_SAMPLES))])) # Fora ponta
67
68 ##TARIFA CONSTANTE
69 TARIFF_C = np.hstack(([0.722495*np.ones(int(12*ONE_HOUR_IN_SAMPLES))
       ],[0.722495*np.ones(int(12*ONE_HOUR_IN_SAMPLES))]))
70
71 TARIFF = TARIFF_C
72
73 ########## LIMITE DE PICO ############
74
75 # Limite de pico (4.0Kwh) durante o dia
76 PEAK_LIMIT = np.array(4.0*np.ones(int(N_SAMPLES)))
77 # O consumo das cargas monitoráveis e detectaveis é modelado como
78 # uma gaussiana centrada as 19:00 e com amplitude de 1.0kwh
79 X = np.arange(0,3*ONE_HOUR_IN_SAMPLES + 1)
80 Y = sp.signal.gaussian(np.size(X), 7)
81 Y = 1.0*Y# Pico de 1kwh
82 PEAK1 = np.hstack(([np.zeros(int(18*ONE_HOUR_IN_SAMPLES))],
83 [Y],
84 [np.zeros(int(3.0*ONE_HOUR_IN_SAMPLES - 1))]))
85
86 PEAK_LIMIT = PEAK_LIMIT - PEAK1
87 PEAK_LIMIT_BACKUP = PEAK_LIMIT
88
89
90 #
       ================================================================
91 #--- Cria os objetos das cargas com suas respectivas características ---#
92 Loads = cenarioPU()
93
94 for l in Loads:
```

```
95      l.fillPowerInSamples(SAMPLE_INTERVAL)
96
97  """
98  ################################################
99  #                  ALGOTIRMO PSO               #
100 ################################################
101 #--- Cria o objeto para execução do algoritmo PSO ---#
102 #  w = peso de inércia
103 #  c1 = parâmetro cognitivo
104 #  c2 = parâmetro social
105 #  alpha = parâmetro de treadoff
106 """
107
108 w=0.4
109 c1=2
110 c2=2
111 ALPHA = 0
112
113 nTimes = 20
114 COST = np.zeros((nTimes))
115 CONF = np.zeros((nTimes))
116 tempo = np.zeros((nTimes))
117 gbest = np.zeros((nTimes))
118
119 #Inicializa objeto pso
120 pso = newPSOalgorithm.PsoAlgorithmMultiAcio(w, c1, c2)
121
122 #Calcula custo do cenário ideal escolhido pelo usuário
123 target = []
124 for l in Loads:
125     tg=DataPso.Population()
126     tg.startTime = l.bestTimeInSamples
127     target = np.append(target,tg)
128
129 targetCost = pso.calcCost(target, Loads, SAMPLE_INTERVAL, TARIFF)
130
131 #Loop com nTimes execuções
132 for k in range(nTimes):
133
```

```
134     start_time = time.time()
135
136     #--- Inicializa População e Velocidades ---#
137     pso.initPopulation(10, Loads)
138
139     pso.executeWithGar(targetCost, Loads, SAMPLE_INTERVAL, TARIFF,
        PEAK_LIMIT, ALPHA)
140     SOLUTION = pso.gBest
141
142     tempo[k] = (time.time() - start_time)
143     fitn, COST[k], DISCOMF,T,U, timeValue_solution = pso.calcFitness(
        targetCost, SOLUTION, Loads, SAMPLE_INTERVAL, TARIFF, PEAK_LIMIT,
        ALPHA)
144     CONF[k] =  DISCOMF/sum(T)
145     gbest[k] = fitn
146     plt.plot(pso.gbestToPlot)
147
148
149 #==================    PLOT DOS RESULTADOS
        ==================================
150 best = np.max(gbest)
151 imi = np.nonzero(gbest==best)[0]
152
153 print("--- W COM TÉCNICA LDIW")
154 print("--- FIM COM %s EXECUÇÕES e ALPHA=%s---"% (nTimes, ALPHA))
155 print("--- MÉDIA SOLUTION: %.7s ---"% ( np.mean(gbest)))
156 print("--- MELHOR SOLUTION: %.7s ---"% ( np.max(gbest)))
157 print("--- DESVIO PADRÃO SOLUTION: %.5s ---"% ( np.std(gbest)))
158 print("--- MÉDIA CUSTO: %.7s ---"% ( np.mean(COST)))
159 print("--- ECONOMIA: %.5s %%---"% ( (targetCost - np.mean(COST))/
    targetCost*100 ))
160 print("--- MÉDIA CONFORTO: %.5s %% ---"% (np.mean(CONF)*100))
161 print("--- MÉDIA TEMPO: %.5s seg---"% (np.mean(tempo)))
162
163 ############ PLOT GRÁFICOS #############
164 xmin = 0
165 xmax = 288 + 1
166 X = np.arange(0,N_SAMPLES)
167 xlabel = 'Tempo [horas]'
```

```
168 ylabel = 'Planejamento de Consumo [kWh]'
169 xticks1 = np.hstack(([1*ONE_HOUR_IN_SAMPLES, 2*ONE_HOUR_IN_SAMPLES, 3*
        ONE_HOUR_IN_SAMPLES, 4*ONE_HOUR_IN_SAMPLES,
170 5*ONE_HOUR_IN_SAMPLES, 6*ONE_HOUR_IN_SAMPLES, 7*ONE_HOUR_IN_SAMPLES, 8*
        ONE_HOUR_IN_SAMPLES,
171 9*ONE_HOUR_IN_SAMPLES, 10*ONE_HOUR_IN_SAMPLES, 11*ONE_HOUR_IN_SAMPLES,
        12*ONE_HOUR_IN_SAMPLES,
172 13*ONE_HOUR_IN_SAMPLES, 14*ONE_HOUR_IN_SAMPLES, 15*ONE_HOUR_IN_SAMPLES,
        16*ONE_HOUR_IN_SAMPLES,
173 17*ONE_HOUR_IN_SAMPLES, 18*ONE_HOUR_IN_SAMPLES, 19*ONE_HOUR_IN_SAMPLES,
        20*ONE_HOUR_IN_SAMPLES,
174 21*ONE_HOUR_IN_SAMPLES, 22*ONE_HOUR_IN_SAMPLES, 23*ONE_HOUR_IN_SAMPLES]))
175 xticks2 = ('01','','03','','05','','07','','09','','11','',
176 '13','','15','','17','','19','','21','','23', '')
177
178 color = [ ['darkblue','darkblue'],
179           ['blue'],
180           ['aqua'],
181           ['darkgreen'],
182           ['sienna'],
183           ['lime', 'lime'],
184           ['yellow'],
185           ['orange'],
186           ['red'],
187           ['violet']]
188
189 lab = ['Bomba Booster',
190           'Bomba Piscina',
191           'Máquina de lavar',
192           'Lâmpadas Externas',
193           'Lâmpadas Internas',
194           'AC office',
195           'AC couples',
196           'AC F1',
197           'AC F2',
198           'Lava-louças']
199
200 #=============================================
201 # TARIFAS (Descomentar Se quiser)
```

```
202 #==========================================
203
204 #fig = plt.figure(1)
205 #fig, ax = plt.subplots()
206 #plt.axis([xmin,xmax,0,max(TARIFF_B[0]) + 1])
207 #plt.grid(True)
208 #
209 #plt.xlabel(xlabel)
210 #plt.ylabel('Tarifas [R$/kwh]')
211 #plt.xticks(xticks1, xticks2 )
212 #
213 #plt.plot(X,TARIFF_B[0], 'grey', dashes=[6, 2], label='Tarifa Branca')
214 #plt.plot(X,TARIFF_C[0], 'grey', label='Tarifa Constante')
215 #plt.legend(loc='best')
216 #fig.savefig('Tarifas.pdf')
217
218 #==========================================
219 # LIMITE DE PICO (Descomentar Se quiser)
220 #==========================================
221
222 #fig = plt.figure(1)
223 #plt.axis([xmin,xmax,0,6])
224 #plt.grid(True)
225 #
226 #plt.plot(X,PEAK_LIMIT[0][:],'-r',label='Demanda Máxima')
227 #
228 #plt.xlabel(xlabel)
229 #plt.ylabel('Consumo [kwh]')
230 #plt.xticks(xticks1, xticks2 )
231 #
232 #plt.legend(loc='best')
233 #fig.savefig('LimitePico.pdf')
234
235 #==========================================
236 #Perfil ORIGINAL (Descomentar Se quiser)
237 #==========================================
238
239 #fig = plt.figure(1)
240 #fig, ax = plt.subplots()
```

```
241 #plt.axis([xmin,xmax,0,10])
242 #plt.grid(True)
243 #
244 #plt.plot(X,PEAK_LIMIT[0][:],'-r','LineWidth',3)
245 #
246 #plt.xlabel(xlabel)
247 #plt.ylabel(ylabel)
248 #plt.xticks(xticks1, xticks2 )
249 #
250 #
251 #TIME_VALUE = np.zeros(int(N_SAMPLES))
252 #TIME_VALUE_aux = np.zeros(int(N_SAMPLES))
253 #c=0
254 #for load in Loads:
255 #    startPosition = load.bestTimeInSamples
256 #    nAcio = np.size(load.bestTimeInSamples)
257 #    for s in range(nAcio):
258 #        aux=0
259 #        DURATION = np.sum(load.durationInSamples)
260 #        for sample in range(int(DURATION)):
261 #            TIME_VALUE[int(startPosition[s]) + aux] = TIME_VALUE[int(
    startPosition[s])+aux] + load.peakPowerInSamples[sample]
262 #            TIME_VALUE_aux[int(startPosition[s]) + aux] = TIME_VALUE[int
    (startPosition[s]) + aux]
263 #            aux = aux+1
264 #
265 #    plt.plot(X,TIME_VALUE_aux, color[c][0],label=lab[c])
266 #    ax.stackplot(X,TIME_VALUE_aux, colors=color[c], alpha=0.7)
267 #    TIME_VALUE_aux = np.zeros(int(N_SAMPLES))
268 #    c=c+1
269
270 #plt.plot(X,TIME_VALUE_aux, 'black')
271 #plt.legend(loc='best')
272 #fig.savefig('perfilOriginal.pdf')
273
274 #=============================================
275 #########     PERFIL OTIMIZADO
276 #=============================================
277
```

```
278 fig = plt.figure(1)
279 fig, ax = plt.subplots()
280 plt.axis([xmin,xmax,0,10])
281 plt.grid(True)
282
283 plt.plot(X,PEAK_LIMIT[0][:],'-r',3)
284
285 plt.xlabel(xlabel)
286 plt.ylabel(ylabel)
287 plt.xticks(xticks1, xticks2 )
288
289 TIME_VALUE2 = np.zeros(int(N_SAMPLES))
290 TIME_VALUE_aux = np.zeros(int(N_SAMPLES))
291
292 sol = (SOLUTION[0])
293 for a in range(len(sol)):
294     startPosition = sol[a].startTime
295     nAcio = np.size(Loads[a].bestTimeInSamples)
296     for s in range(nAcio):
297         aux=0
298         DURATION = np.sum(Loads[a].durationInSamples)
299         for sample in range(int(DURATION)):
300             TIME_VALUE2[int(startPosition[s]) + aux] = TIME_VALUE2[int(
    startPosition[s])+aux] + Loads[a].peakPowerInSamples[sample]
301             TIME_VALUE_aux[int(startPosition[s]) + aux] = TIME_VALUE2[int
    (startPosition[s]) + aux]
302             aux = aux+1
303
304     plt.plot(X,TIME_VALUE_aux, color[a][0],label=lab[a])
305     ax.stackplot(X,TIME_VALUE_aux, colors=color[a], alpha=0.7)
306     TIME_VALUE_aux = np.zeros(int(N_SAMPLES))
307
308 plt.plot(X,TIME_VALUE_aux, 'black')
309
310 #plt.plot(X,TARIFF[0], 'grey')
311 plt.legend(loc='best')
312 fig.savefig('perfilOtimizado.pdf')
313
314 #ESCREVE NO ARQUIVO
```

```
315 texto=[]
316 texto.append("--- FIM COM " + str(nTimes) + " EXECUÇÕES e ALPHA= " + str(
    ALPHA))
317 texto.append("--- \nMÉDIA SOLUTION: " + str( np.mean(gbest)))
318 texto.append("--- \nMELHOR SOLUTION: " + str( np.max(gbest)))
319 texto.append("--- \nPIOR SOLUTION: " + str( np.min(gbest)))
320 texto.append("--- \nDESVIO PADRÃO SOLUTION: "+ str( np.std(gbest)))
321 texto.append("\n-----\n")
322
323 texto.append("--- \nMÉDIA CUSTO: " + str( np.mean(COST)))
324 texto.append("--- \nECONOMIA MEDIA: " + str( (targetCost - np.mean(COST))
    /targetCost*100 ) + "%")
325 texto.append("--- \nMELHOR CUSTO: " + str( np.min(COST)))
326 texto.append("--- \nECONOMIA MELHOR: " + str( (targetCost - np.min(COST))
    /targetCost*100 ) + "%")
327 texto.append("--- \nPIOR CUSTO: " + str( np.max(COST)))
328 texto.append("--- \nECONOMIA PIOR: " + str( (targetCost - np.max(COST))/
    targetCost*100 ) + "%")
329 texto.append("\n-----\n")
330
331 texto.append("--- \nMÉDIA CONFORTO: " + str(np.mean(CONF)*100) + "%")
332 texto.append("--- \nMELHOR CONFORTO: " + str(np.max(CONF)*100) + "%")
333 texto.append("--- \nPIOR CONFORTO: " + str(np.min(CONF)*100) + "%")
334 texto.append("\n-----\n")
335
336 texto.append("--- \nMÉDIA TEMPO: " + str(np.mean(tempo)))
337 texto.append("--- \nMELHOR TEMPO: " + str(np.min(tempo)))
338 texto.append("--- \nPIOR TEMPO: " + str(np.max(tempo)))
339
340 texto.append("\n-----\n")
341 texto.append("\nCUSTO: " + str(COST) )
342 texto.append("\nCONF: " + str(CONF) )
343 texto.append("\nFITNESS: " + str(gbest) )
344 texto.append("\nTEMPO: " + str(tempo))
345 texto.append("--- \nDESVIO PADRÃO tempo: "+ str( np.std(tempo)))
346
347 arq = open('/home/stephanie/Área de Trabalho/MESTRADO/c1-EIW.txt', 'w')
348 arq.writelines(texto)
349 arq.close()
```

```
# -*- coding: utf-8 -*-
import numpy as np
"""
Classe que representa uma Carga

    ...

    Attributes
    ----------
    name : str
        nome da carga
    nPhases: int
        número de fases de operação (ciclos)
    minTime:np.array
        mínimo instante para início da carga
    maxTime: np.array
        máximo instante para início da carga
    bestTime: np.array
        melhor instante para início da carga
    duration: np.array
        duração de cada ciclo de operação (em minutos)
    isMultiAcio: boolean
        se a carga é multiacionamento
    avgPower: np.array
        potência média de cada operação da carga
    peakPower: np.array
        potência de pico de cada operação
    comfortLevel: float
        nível de conforto da carga

    Methods
    -------
    fillPowerInSamples(sampleInterval):
        Preenche cada vetor referente a um dia, de acordo com a taxa de
    amostragem escolhida
"""
class Load:
     def __init__(self, name, nPhases, minTime, maxTime,  bestTime,
```

```
    duration, isMultiAcio, avgPower, peakPower, comfortLevel):
38          self.name = np.array(name)
39          self.nPhases = nPhases
40          self.minTime = np.array(minTime)
41          self.maxTime = np.array(maxTime)
42          self.bestTime = np.array(bestTime)
43          self.duration = np.array(duration)
44          self.isMultiAcio = np.array(isMultiAcio)
45          self.avgPower = np.array(avgPower)
46          self.peakPower = np.array(peakPower)
47          self.comfortLevel = np.array(comfortLevel)
48
49      def fillPowerInSamples(self, sampleInterval):
50          ONE_HOUR_IN_SAMPLES = (60/sampleInterval) # 1 Hora em Amostas
51          N_PHASES = self.nPhases
52          N_SAMPLES = (60*24 / sampleInterval) # Amostras em 1 dia
53
54          # Amostra associada ao tempo minimo de inicio da carga
55          self.minTimeInSamples = self.minTime*ONE_HOUR_IN_SAMPLES
56
57          # Amostra associada ao tempo máximo de término
58          self.maxTimeInSamples = self.maxTime*ONE_HOUR_IN_SAMPLES
59
60          # Amostra associada ao tempo ideal de inicio da carga
61          self.bestTimeInSamples = self.bestTime * ONE_HOUR_IN_SAMPLES
62
63          # Duração total da carga em numero de amostras
64          self.durationInSamples = self.duration/sampleInterval
65
66          TOTAL_DURATION = np.sum(self.durationInSamples)
67
68          PEAK = np.zeros((int(TOTAL_DURATION)))
69          AVG = np.zeros((int(TOTAL_DURATION)))
70          aux = 0
71
72          for phase in range(N_PHASES):
73              PEAK = PEAK + np.concatenate(( np.zeros((aux)), (self.
    peakPower[phase] * np.ones((int(self.durationInSamples[phase])))), np.
    zeros((int(TOTAL_DURATION) - aux - int(self.durationInSamples[phase]))
```

```
    )),0)
74              AVG = AVG + np.concatenate(( np.zeros((aux)), (self.avgPower[
    phase] * np.ones((int(self.durationInSamples[phase])))), np.zeros((int
    (TOTAL_DURATION) - aux - int(self.durationInSamples[phase])))),0)
75
76              aux = aux + int(self.durationInSamples[phase])
77
78          self.peakPowerInSamples = np.concatenate( (PEAK , np.zeros((int(
    N_SAMPLES - TOTAL_DURATION))) ), 0)
79          self.avgPowerInSamples =  np.concatenate( (AVG , np.zeros((int(
    N_SAMPLES - TOTAL_DURATION))) ), 0)
80
81  """
82  Classe que representa uma Carga Flexível na potência
83  """
84  class TimeFlexLoad(Load):
85
86      def __init__(self, name, nPhases, minTime, maxTime,  bestTime,
    duration, isMultiAcio, avgPower, peakPower, comfortLevel):
87          Load.__init__(self, name, nPhases, minTime, maxTime,  bestTime,
    duration, isMultiAcio, avgPower, peakPower, comfortLevel)
88
89  """
90  Classe que representa uma Carga Flexível na potência
91
92      ...
93
94      Attributes
95      ----------
96      minPower: np.array
97          potência mínima em que a carga pode operar
98      maxPower: np.array
99          potência máxima em que a carga pode operar
100 """
101 class PowerFlexLoad(Load):
102
103     def __init__(self, name, nPhases, minTime, maxTime,  bestTime,
    duration, isMultiAcio, avgPower, peakPower, minPower, maxPower,
    comfortLevel):
```

```
104         super(Load, self).__init__(name, nPhases, minTime, maxTime,
    bestTime, duration, isMultiAcio, avgPower, peakPower, comfortLevel)
105         self.minPower = minPower
106         self.maxPower = maxPower
```

Código-fonte 1 – Código Principal

```
1  # -*- coding: utf-8 -*-
2  import numpy as np
3
4  """
5  Classe que representa uma Solução do SHC
6
7      ...
8
9      Attributes
10     ----------
11     startTime: array
12         vetor de soluções (instantes de inicio das cargas)
13     opPower: array
14         potência de operação de cada carga
15     v1: array
16         velocidade 1 (para atualização de startTime)
17     v2: array
18         velocidade 2 (para atualização de opPower)
19 """
20 class DataPso:
21     def __init__(self):
22         pass
23
24 class Population(DataPso):
25     def __init__(self):
26         DataPso.__init__(self)
27         self.startTime = np.array([])
28         self.opPower = np.array([])
29         self.v1 = np.array([])
30         self.v2 = np.array([])
```

Código-fonte 2 – Classe Load

```
# -*- coding: utf-8 -*-
"""
# Classe para Algortimo PSO
"""

import numpy as np
import DataPso
import copy

class PsoAlgorithmMultiAcio():
    def __init__(self, w, c1, c2):
        self.w = w
        self.c1 = c1
        self.c2 = c2
        self.nOfIterations = 500

    def initPopulation(self, popLength,loads):
        self.popLength = popLength
        nSubjects = len(loads)

        for i in range(popLength):
            pop = []
            for n in range(nSubjects):
                nAcio = np.size(loads[n].bestTimeInSamples)

                pu = DataPso.Population()

                MIN_TIME = loads[n].minTimeInSamples
                MAX_TIME = loads[n].maxTimeInSamples
                DURATION = loads[n].durationInSamples

                MIN_POWER = loads[n].avgPower
                MAX_POWER = loads[n].avgPower

                init = MIN_TIME
                fin = MAX_TIME - sum(DURATION)

                proib = []
```

```
39                  h = np.zeros((nAcio))
40
41                  for j in range(nAcio):
42                      flag = True
43                      while(flag):
44                          h[j] = round(init + (fin - init)*np.random.rand()
    )
45
46                          ind = np.nonzero(proib==h[j])
47                          if np.size(ind) == 0:
48                              flag = False
49
50                      proib = np.concatenate((proib, np.linspace(h[j]-sum(
    DURATION)+1,h[j]+sum(DURATION)-1,2*sum(DURATION)-1)), 0)
51
52                  h = np.sort(h)
53
54                  multFact = (MIN_POWER + (MAX_POWER-MIN_POWER)*np.random.
    rand())
55
56                  pu.startTime = h - 1
57                  pu.opPower = np.round(multFact*100)/100;
58                  pu.v1 = np.zeros((nAcio))
59                  pu.v2 = np.zeros((nAcio))
60
61                  pop = np.append(pop,pu)
62
63              if i==0:
64                  population = pop
65              else:
66                  population = np.vstack([population,pop])
67
68          self.population = population
69
70      def execute(self, loads, sampleInterval, tariff, peakLimit, alpha):
71
72          (self.fitness, COST, DISCOMF, dmax, rest) = self.calcFitness(
    self.population, loads, sampleInterval, tariff, peakLimit, alpha)
73
```

```
74          gbestFit = np.min(self.fitness)
75          imin = np.nonzero(self.fitness==gbestFit)[0]
76          self.gBest = self.population[imin][:]
77          self.pBest = self.population
78          pbestFit = self.fitness
79          #cost = COST[imin]
80          REST = rest[imin]
81
82          it=1
83          conv=0
84          #Laço até convergência
85          while ((it < self.nOfIterations and conv!=self.popLength) or
    REST!=0):
86              for j in range(self.popLength):
87                  #Comparação dos fitness com gbest e pbest
88                  if self.fitness[j] < gbestFit:
89                      self.gBest = [self.population[j][:]]
90                      gbestFit = self.fitness[j]
91                      #cost = COST[j]
92                      #discomf = DISCOMF[j]
93                      REST = rest[j]
94
95                  if self.fitness[j] < pbestFit[j]:
96                      self.pBest[j][:] = self.population[j][:]
97                      pbestFit[j] = self.fitness[j]
98
99                  r1= np.random.rand()
100                 r2= np.random.rand()
101
102                 nLoads = np.size(loads)
103                 for i in range(nLoads):
104                     duration = loads[i].durationInSamples
105                     #Atualização da v1 e startTime
106                     self.population[j][i].v1 = self.w *self.population[j
    ][i].v1 + self.c1*r1*(self.pBest[j][i].startTime - self.population[j][
    i].startTime) + self.c2*r2*(self.gBest[0][i].startTime - self.
    population[j][i].startTime)
107                     self.population[j][i].startTime = np.round(self.
    population[j][i].startTime + self.population[j][i].v1 )
```

```
108
109                     #Limitação de startTime
110                     limitMaxV1 = loads[i].maxTimeInSamples - sum(
    duration)
111
112                     limitIndexMinV1 = np.nonzero( self.population[j][i].
    startTime < loads[i].minTimeInSamples)
113
114                     for k in range(np.size(limitIndexMinV1)):
115                         self.population[j][i].startTime[limitIndexMinV1
    [0][k]] =  np.array([loads[i].minTimeInSamples])
116
117                     limitIndexMaxV1 = np.nonzero( self.population[j][i].
    startTime > limitMaxV1 )
118                     for k in range(np.size(limitIndexMaxV1)):
119                         self.population[j][i].startTime[limitIndexMaxV1
    [0][k]] =  np.array([limitMaxV1])
120
121
122                 #calcula fitness e atualiza variável
123                 (self.fitness, COST, DISCOMF, dmax, rest) = self.
    calcFitness(self.population, loads, sampleInterval, tariff, peakLimit,
     alpha)
124
125             #Gbest(1,it) = gbestFit
126             it = it+1 #incrementa it
127             conv = np.shape(np.nonzero(np.round(self.fitness[0]*100)==np
    .round(self.fitness*100)))[1]
128
129         self.gbestFit = gbestFit
130         self.rest = REST
131
132     def ldiw(self, wstart, wend, it, T):
133         w = wstart -it *(wstart - wend)/T
134         return w
135
136     def diw(self, wstart, u, it):
137         w = wstart*pow(u, -it)
138         return w
```

```
139
140     def aiw(self, wstart, wend, Si, N):
141         S = np.sum(Si)/N
142         w = (wstart - wend)*S + wend
143         return w
144
145     def eiw(self, wstart, it, T):
146         w = wstart * np.exp(-it/T)
147         return w
148
149     def executeWithGar(self, targetCost, loads, sampleInterval, tariff,
    peakLimit, alpha):
150
151         (self.fitness, COST, DISCOMF, dmax, rest, tv) = self.calcFitness
    (targetCost, self.population, loads, sampleInterval, tariff, peakLimit
    , alpha)
152         T = self.nOfIterations
153         Si = np.zeros((self.popLength))
154         self.gbestToPlot = []
155         limitConver = 200
156         restart=1
157
158         gbestFit = np.max(self.fitness)
159         imin = np.nonzero(self.fitness==gbestFit)[0]
160         self.gBest = self.population[imin][:]
161         self.pBest = self.population
162         pbestFit = self.fitness
163         REST = rest[imin]
164         self.lastIt = 0
165         it=1
166         sair=1
167         c1=self.c1
168         c2=self.c2
169         #Laço até convergência
170         while ((it <  self.nOfIterations  and sair==1) or REST!=1):
171
172             #w = self.ldiw(0.9, 0.4, it, T)
173             #w = self.diw(0.3, 1.00002, it)
174             #w = self.aiw(1, 0.1, Si, self.popLength)
```

```
175             w = self.eiw(0.9, it, T)
176
177             for j in range(self.popLength):
178                 #Comparação dos fitness com gbest e pbest
179                 if self.fitness[j] > gbestFit:
180                     self.gBest = copy.deepcopy([self.population[j][:]])
181                     gbestFit = copy.deepcopy(self.fitness[j])
182                     self.lastIt = it
183                     cost = COST[j]
184                     #discomf = DISCOMF[j]
185                     REST = rest[j]
186
187                 if self.fitness[j] > pbestFit[j]:
188                     self.pBest[j][:] = copy.deepcopy(self.population[j
    ][:])
189                     pbestFit[j] = copy.deepcopy(self.fitness[j])
190                     Si[j]=1
191                 else:
192                     Si[j] = 0
193
194
195                 r1= np.random.rand()
196                 r2= np.random.rand()
197
198                 nLoads = np.size(loads)
199                 for i in range(nLoads):
200                     duration = loads[i].durationInSamples
201                     #Atualização da v1 e startTime
202                     self.population[j][i].v1 = w*self.population[j][i].
    v1 + c1*r1*(self.pBest[j][i].startTime - self.population[j][i].
    startTime) + c2*r2*(self.gBest[0][i].startTime - self.population[j][i
    ].startTime)
203                     self.population[j][i].startTime = np.round(self.
    population[j][i].startTime + self.population[j][i].v1 )
204
205                     #Limitação de startTime
206                     limitMaxV1 = loads[i].maxTimeInSamples - sum(
    duration)
207
```

```
208                     limitIndexMinV1 = np.nonzero( self.population[j][i].
    startTime < loads[i].minTimeInSamples)
209 
210                     for k in range(np.size(limitIndexMinV1)):
211                         self.population[j][i].startTime[limitIndexMinV1
    [0][k]] =  np.array([loads[i].minTimeInSamples])
212 
213                     limitIndexMaxV1 = np.nonzero( self.population[j][i].
    startTime > limitMaxV1 )
214                     for k in range(np.size(limitIndexMaxV1)):
215                         self.population[j][i].startTime[limitIndexMaxV1
    [0][k]] =  np.array([limitMaxV1])
216 
217                 #calcula fitness e atualiza variável
218                 (self.fitness, COST, DISCOMF, dmax, rest, tv) = self.
    calcFitness(targetCost, self.population, loads, sampleInterval, tariff
    , peakLimit, alpha)
219 
220             #Gbest(1,it) = gbestFit
221             it = it+1 #incrementa it
222 #             if it == 100:
223 #                 print("---it---")
224 
225             conv = np.shape(np.nonzero(np.round((self.fitness[0])*1000)
    ==np.round(self.fitness*1000)))[1]
226 
227             if((it-self.lastIt)>=100):
228                 if it<=limitConver and restart==1:
229                     self.initPopulation(self.popLength, loads)
230                     (self.fitness, COST, DISCOMF, dmax, rest, tv) = self
    .calcFitness(targetCost, self.population, loads, sampleInterval,
    tariff, peakLimit, alpha)
231                     indexBest = np.argmin(self.fitness)
232                     self.population[indexBest] = copy.deepcopy(self.
    gBest[0])
233                     self.lastIt = it
234                     restart=0
235                     #rint("--- reiniciei ---")
236                     c1=1
```

```
237                         c2=3
238                     else:
239                         sair=0
240 #
241 #                 if conv == 10 and it>150:
242 #                     sair=0
243
244                 if(conv>0.5*self.popLength and REST!=1):
245                     self.initPopulation(self.popLength, loads)
246                     (self.fitness, COST, DISCOMF, dmax, rest, tv) = self.
    calcFitness(targetCost, self.population, loads, sampleInterval, tariff
    , peakLimit, alpha)
247
248                     gbestFit = np.max(self.fitness)
249                     imin = np.nonzero(self.fitness==gbestFit)[0]
250                     self.gBest = self.population[imin][:]
251                     self.pBest = self.population
252                     pbestFit = self.fitness
253                     print("--- reiniciei ---")
254                     REST = rest[imin]
255                     limitConver = limitConver + it
256
257                 self.gbestToPlot.append(gbestFit)
258
259             self.gbestFit = gbestFit
260             self.rest = REST
261             self.cost = cost
262             print("--- FIM COM %s it ---"% (it))
263
264         def calcFitness(self, targetCost, population, LOADS, Ts, C,
    peakLimit, alpha):
265             (sizePop, n_loads) = np.shape(population)
266             n_samples = (24*60)/Ts
267
268             TIME_VALUE = np.zeros((int(n_samples), int(sizePop)))
269             exec_cost = np.zeros((int(sizePop), 1))
270             discomf = np.zeros((int(sizePop), 1))
271
272             for a in range(sizePop):
```

```
273             #---------- Calcula o valor da potência de pico pra cada amostra
274             for load in range(n_loads):
275                 startPosition = population[a][load].startTime
276 
277                 nAcio = np.size(LOADS[load].bestTimeInSamples)
278                 for s in range(nAcio):
279                     aux=0
280                     #print(a)
281                     DURATION = np.sum(LOADS[load].durationInSamples)
282                     for sample in range(int(DURATION)):
283                         TIME_VALUE[int(startPosition[s]) + aux][a] = TIME_VALUE[int(startPosition[s])+aux][a] + LOADS[load].peakPowerInSamples[sample]
284                         aux = aux+1
285 
286             exec_cost[a] = self.calcCost(population[a][:], LOADS, Ts, C)
287             (discomf[a], DMAX) = self.calcConfort(population[a][:],LOADS)
288 #=============================================================================
289 
290         #Comparação com o pico limite
291         f3 = np.zeros((sizePop,1))
292         for a in range(sizePop):
293             exceed = np.nonzero(TIME_VALUE[:,a] > peakLimit[0])
294             for e in exceed[0]:
295                 f3[a] = f3[a] + (TIME_VALUE[e,a] - peakLimit[0][e])
296 
297         f1 =(targetCost- exec_cost)/targetCost
298         f2 = (discomf/np.sum(DMAX))
299         f2 = f2/3.5
300         f3 = 1/(1+f3)
301 
302         Apt = (alpha)*f1 + (1-alpha)*f2 + f3
303 
304         return Apt, exec_cost, discomf, DMAX, f3, TIME_VALUE
305 
306     def calcCost(self, subject, LOADS, Ts, C):
307         N_LOADS = np.shape(LOADS)[0]
```

```
308
309         exec_cost = 0
310         for load in range(N_LOADS):
311
312             startPosition = subject[load].startTime
313             aux=0
314             nAcio = np.size(LOADS[load].bestTimeInSamples)
315             for s in range(nAcio):
316                 DURATION = np.sum(LOADS[load].durationInSamples);
317                 finalPosition = startPosition[s] + (DURATION);
318                 load_cost = 0
319
320                 #n=startPosition[s]
321                 for n in range(int(startPosition[s]),int(finalPosition))
    :
322                     load_cost = load_cost +  LOADS[load].
    avgPowerInSamples[int(n-startPosition[s])]* (Ts/60) * C[0][n]
323
324                 aux = aux + load_cost
325
326             exec_cost = exec_cost + aux
327         return exec_cost
328
329     def calcConfort(self, population, LOADS):
330         N_LOADS = np.shape(LOADS)[0]
331
332         DMAX = np.zeros(N_LOADS)
333
334         confort = 0
335         for load in range(N_LOADS):
336             conf = 0
337             #nAcio = np.size(LOADS[load].bestTimeInSamples)
338
339             Dmax = np.maximum( np.abs(LOADS[load].minTimeInSamples -
    LOADS[load].bestTimeInSamples), np.abs(LOADS[load].maxTimeInSamples -
    LOADS[load].bestTimeInSamples) )
340             conf = Dmax - LOADS[load].comfortLevel * np.abs( population[
    load].startTime - LOADS[load].bestTimeInSamples )
341
```

```
342             #conf = sum(conf)/nAcio;
343             conf = np.sum(conf)
344
345             confort = confort + conf
346             DMAX[load] = np.sum(Dmax)
347
348         return confort, DMAX
349
350
351     def calcConfortTaguchi(self, population, LOADS):
352         N_LOADS = np.shape(LOADS)[0]
353
354         DMAX = np.zeros(N_LOADS)
355
356         confort = 0
357         for load in range(N_LOADS):
358             conf = 0
359             #nAcio = np.size(LOADS[load].bestTimeInSamples)
360
361             Dmax = np.maximum( pow(LOADS[load].minTimeInSamples - LOADS[
    load].bestTimeInSamples,2), pow(LOADS[load].maxTimeInSamples - LOADS[
    load].bestTimeInSamples,2) )
362
363             conf = Dmax - LOADS[load].comfortLevel * pow( population[
    load].startTime - LOADS[load].bestTimeInSamples ,2)
364
365             #conf = sum(conf)/nAcio;
366             conf = np.sum(conf)
367
368             confort = confort + conf
369             DMAX[load] = np.sum(Dmax)
370
371         return confort, DMAX
```

Código-fonte 3 – Classe DataPso