**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE TECNOLOGIA**

**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

**ÍCARO JONAS BATISTA**

# AGENDAMENTO DE CARGAS RESIDENCIAIS OTIMIZADO UTILIZANDO EVOLUÇÃO DIFERENCIAL

**FORTALEZA**

**2018**

ÍCARO JONAS BATISTA

# AGENDAMENTO DE CARGAS RESIDENCIAIS OTIMIZADO UTILIZANDO EVOLUÇÃO DIFERENCIAL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas de Engenharia Elétrica.

Orientador: Prof. Dr. Luiz Henrique Silva Colado Barreto.

Coorientador: Prof. Dr. Paulo Peixoto Praça

FORTALEZA

2018

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)

---

B337a Batista, Ícaro Jonas.
Agendamento de Cargas Residenciais Otimizado Utilizando Evolução Diferencial / Ícaro Jonas Batista. – 2018.
92 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2018.
Orientação: Prof. Dr. Luiz Henrique Silva Colado Barreto.
Coorientação: Prof. Dr. Paulo Peixoto Praça.

1. Agendamento de Cargas. 2. Evolução Diferencial. 3. Sistema de Gerenciamento de Energia. 4. Gerenciamento pelo lado da Demanda. 5. Otimização. I. Título.

CDD 621.3

ÍCARO JONAS BATISTA

# AGENDAMENTO DE CARGAS RESIDENCIAIS OTIMIZADO UTILIZANDO EVOLUÇÃO DIFERENCIAL

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas de Engenharia Elétrica.

Aprovada em: 23/07/2018.

BANCA EXAMINADORA

_____

Prof. Dr. Luiz Henrique Silva Colado Barreto (Orientador)

Universidade Federal do Ceará (UFC) - DEE

_____

Prof. Dr. Paulo Peixoto Praça (Coorientador)

Universidade Federal do Ceará (UFC) - DEE

_____

Prof. Dr. Wilkley Bezerra Correia

Universidade Federal do Ceará (UFC) - DEE

_____

Prof. Dr. Jarbas Aryel Nunes da Silveira

Universidade Federal do Ceará (UFC) - DETI

## AGRADECIMENTOS

À minha família por me ensinar desde pequeno valores e exemplos essenciais para meu desenvolvimento, por me apoiar e por me ajudar a alcançar todos os meus objetivos.

Ao meu orientador, Prof. Dr. Luiz Henrique Colado Barreto, por acompanhar meu desenvolvimento desde a graduação na UFC, por sua eficiência e responsabilidade durante minha orientação e por me incentivar a sempre seguir em frente.

Ao Prof. Dr. Paulo Peixoto Praça, por sempre ser extremamente solícito, por suas conversas e sugestões que vão além do desenvolvimento deste trabalho.

Ao Prof. Dr. Wilkley Bezerra Correia e ao Prof. Dr. Jarbas Aryel Nunes da Silveira por disponibilizarem seu tempo para participação como membros da banca e por suas valiosas sugestões durante a revisão deste trabalho.

A todos os outros professores do Departamento de Engenharia Elétrica da Universidade Federal do Ceará (DEE-UFC), obrigado por seus ensinamentos e por exercerem uma profissão tão digna e tão importante para a nossa sociedade.

Agradeço, também, aos colegas do DEE-UFC pela companhia e amizade. Em especial, aos meus colegas do GPEC, Juliano Pacheco, Marcus Anderson, Francisco Barbosa, Welton Lima, Jorge Wattes, Breno Chaves, Janaína Almada, Bruno Ricardo e Allan Uchoa. Obrigado pelo companheirismo.

À minha noiva Mariana Lira, por sua paciência e apoio incondicional.

À todas as pessoas que de forma direta ou indireta contribuíram e fizeram parte dessa etapa da minha vida.

À FUNCAP, pelo apoio financeiro durante o projeto de pesquisa que culminou nesse trabalho.

“The light works," he said, indicating the window, "the gravity works," he said, dropping a pencil on the floor. "Anything else we have to take our chances with.”

Douglas Adams, Dirk Gently's Holistic Detective Agency.

## RESUMO

Este trabalho descreve uma abordagem para a realização de um agendamento de cargas elétricas ótimo utilizando técnicas de otimização baseada em estratégia computacional evolucionária. A realização da busca e da otimização do problema descrito no trabalho é realizado através da concepção e implementação do algoritmo Evolução Diferencial. A proposta investiga a aplicação de modalidades tarifárias diferentes, em especial, do tipo *Time-Of-Use* e dos benefícios que o agendamento, quando unido à automação residencial pode trazer ao usuário comum. A solução apresenta a formulação matemática do problema através de uma função multiobjetivo que procura, além da minimização do custo total do consumo de energia elétrica, a otimização do nível de conforto e da relação entre o consumo máximo e médio do sistema (fator de carga) ao longo de um período de um dia. A implementação e os resultados das simulações mostram que o sistema de agendamento proposto pode prover uma redução nos custos de energia elétrica enquanto é capaz de considerar o nível de inconveniência e do fator de carga do agendamento proposto em níveis satisfatórios.



**Palavras-chave**: Agendamento de Cargas, Evolução Diferencial, Sistema de Gerenciamento de Energia, Gerenciamento pelo lado da Demanda, Otimização.

## ABSTRACT

This work describes an approach to perform load scheduling optimization using evolutionary strategy. The utilized method optimizes the problem by searching the space iteratively using Differential Evolution. The proposal investigates the different types of pricing methods, in particular, the application of Time-Of-Use Pricing and the benefits that it can return to the average user when combined with optimum load scheduling and home automation. The mathematical formulation of the problem makes practical use of a multiobjective function which is utilized to minimize the overall electricity cost, to optimize the comfort and to increase the peak-to-average ratio of the electricity system during the day. Simulation results demonstrate the applicability of the proposed solution. It is shown that the electricity cost is reduced taking comfort and peak-to-average ratio under consideration.



**Keywords**: Load Scheduling, Differential Evolution, Energy Management System, Demand Side Management, Optimization.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *AMI* | *Advanced Metering Infrastructure* |
| *BAs* | *Battery-assisted Appliances* |
| $CO_2$ | *Dióxido de Carbono* |
| *CPP* | *Critical Peak Pricing* |
| *DLC* | *Direct Load Control* |
| *DR* | *Demand Response* |
| *DSM* | *Demand Side Management* |
| *ED* | *Evolução Diferencial* |
| *FDP* | *Função Densidade de Probabilidade* |
| *I/C* | *Interruptable/Curtainable* |
| *kWh* | kiloWatt-hora |
| *RED* | Recursos Energéticos Distribuídos |
| *RTP* | *Real-Time Pricing* |
| *SG* | *Smart Grid* |
| *TIC* | Tecnologia da Informação e Comunicação |
| *ToU* | *Time-of-Use* |

## LISTA DE SÍMBOLOS

$M$ *Quantidade de blocos de tempo*

$N$ *Quantidade total de cargas*

$R$ *Quantidade total de ciclos de operação*

$k$ *Índice do bloco de tempo em M*

$i$ *Índice da carga em N*

$j$ *Índice do ciclo de operação*

$T_p^i$ *Horário confortável de i*

$T_{sw}^i$ *Início da janela para i*

$T_{ew}^i$ *Fim de janela para i*

$T_d^i$ *Duração da carga i*

$C_k$ *Valor da tarifa em k*

$P_{ij}^k$ *Perfil de carga de i em j*

$P_{max}^k$ *Valor de energia máxima durante o dia*

$w_i$ *Coeficiente de conforto de i*

**SUMÁRIO**

# 1 INTRODUÇÃO

Desde a revolução industrial iniciada do século XVIII, a rápida ascensão da população mundial e o consumo vertiginoso dos combustíveis fósseis tornam-se questões desafiadoras para o desenvolvimento da sociedade moderna. Em especial, o elevado índice de $CO_2$ (dióxido de carbono) na atmosfera do nosso planeta tem inclinado governos e cientistas ao redor do mundo em busca de alternativas energéticas ecológicas, seguras e renováveis.

Nesse âmbito, o modelo de produção de energia elétrica através de grandes usinas (térmicas, hidroelétricas, etc) vem sendo transformado e o clássico paradigma da produção de energia elétrica centralizada cede espaço a um sistema de geração distribuída. Todavia, a inserção de novas fontes de energia renováveis, como a solar e a eólica, apresentam limitações. Devido a sua enorme dependência de fatores climáticos — nível de irradiação solar, velocidade do vento, nebulosidade, etc — a geração de energia através dessas fontes renováveis é flutuante e pode acarretar instabilidade ao sistema elétrico. O mantimento do balanço entre geração e demanda em um sistema distribuído exige o desenvolvimento de novas abordagens acerca do monitoramento e do controle da produção que, tradicionalmente, é vinculado a sistemas de geração em larga escala.

A busca pelas soluções dos desafios impostos para implementação de um sistema de produção de energia distribuída traça um novo perfil de rede elétrica. Utilizando Tecnologia da Informação e Comunicação (TIC) para englobar aspectos técnicos e econômicos em tempo real, essa nova estrutura é atualmente referida como *Smart Grid* (SG) — em português, Rede Elétrica Inteligente. Ao dotar o sistema elétrico da capacidade de novos tipos de gerenciamento e controle de demanda, a SG permite maior eficiência na utilização dos componentes e equipamentos do sistema elétrico. Dessa maneira, a sua realização possibilita a busca por soluções economicamente viáveis frente a investimentos onerosos em infraestrutura.

Com a incorporação de TICs no contexto das redes inteligentes, emerge-se a possibilidade da participação do consumidor como parte capaz de atuação em operações sobre o sistema elétrico. Essa participação, estimulada através de recompensas e incentivos financeiros, torna-se o ponto de partida para uma série de metodologias chamadas de *Demand Response* (DR) — em português, resposta da demanda — que prevê métodos e tecnologias que permitam ao consumidor a atuação em operações do sistema elétrico.

O equilíbrio entre geração e demanda é realizado majoritariamente através do uso de estimativas para a previsão da demanda e de operações rebuscadas nas grandes usinas de geração de energia. De fato, o custo da energia elétrica varia ao longo do dia e depende da localização das unidades de geração e do estado de utilização do sistema elétrico de potência. Com tarifas de valores fixos e a falta de incentivos para a utilização eficiente do consumo de energia elétrica, o perfil de consumo dos usuários residenciais não se correlaciona com as necessidades instantâneas da infraestrutura e dos recursos do sistema elétrico. Assim, a DR se apresenta como uma solução alternativa à minimização do estresse do sistema elétrico em certos períodos do dia e ao controle não somente através dos grandes centros de geração, mas da realização do gerenciamento pelo lado da demanda. Na literatura, o último método é comumente referido como *Demand Side Management (DSM)*.

Com o estabelecimento do DSM e a redução nos valores de sensores e medidores inteligentes, o gerenciamento de energia e a automação residencial tem, cada vez mais, atraído o interesse de pesquisadores. Várias técnicas e algoritmos propostos na literatura buscam reduzir o custo total da conta de energia correspondente a tarifas e incentivos oferecidos pelas concessionárias. Ademais, outro tópico de grande interesse trata da produção de energia residencial através de fontes de energia renováveis onde o consumidor é dotado da capacidade de armazenar energia para utilização em outros períodos do dia com o repasse de um eventual excesso de produção para a rede elétrica, visando, assim, obter créditos da concessionária (QUAYYUM et al, 2015).

Mediante esse cenário, a utilização de técnicas de otimização em sistemas automatizados, de modo a obter redução nos custos do consumo de energia elétrica, torna-se objeto de estudo de extrema relevância. Em particular, o uso de técnicas de minimização sobre o custo total da conta de energia realizado através do agendamento de cargas se apresenta como um método de implementação acessível. Além disso, essa estratégia se mostra uma contribuição relevante no âmbito do gerenciamento pelo lado da demanda e, consequentemente, das SGs.

## 1.1 Motivação

No Brasil, segundo a Agência Nacional de Energia Elétrica (ANEEL), a partir do dia 1º de Janeiro de 2018, todas as concessionárias do país deverão atender pedidos de adesão da modalidade tarifa branca, inicialmente com consumidores de média mensal de 500 kWh. Em 2019, unidades com consumo de 250 kWh e, em 2020, para consumidores de baixa tensão, qualquer que seja o consumo (ANEEL, 2015). Esse novo tipo de tarifa, pertencente à modalidade tarifária do tipo *Time-of-Use* (ToU), ainda que seja uma estratégia benéfica para o sistema elétrico, pode não representar uma vantagem imediata para usuários comuns.

Tarifas do tipo ToU pertencem ao grupo de tarifas que variam o seu valor para diferentes períodos do dia visando refletir situações típicas da utilização do sistema elétrico. Com a implementação de tarifas variáveis a possibilidade de otimização do consumo de energia elétrica se torna expressiva, considerando não somente a utilização consciente dos equipamentos elétricos, mas, também, do horário em que eles são utilizados.

A determinação do horário ideal de utilização das cargas elétrica, contudo, pode se tornar uma tarefa exaustiva. A complexidade do problema pode crescer exponencialmente com a adição de novas cargas, o que torna inviável sua realização manualmente. Fazendo uso de TIC e da automação residencial, o agendamento e o controle das diferentes cargas podem ser realizados sem interferência humana, utilizando como base as informações em tempo real advindas do usuário e da concessionária de energia.

Dessa maneira, este trabalho visa a produção de um conteúdo acerca do desenvolvimento de um sistema de otimização para o agendamento de cargas realizado de modo acessível, rápido e de baixo custo computacional para inserção em residências e pequenas unidades comerciais.

## 1.2 Objetivos

Os objetivos dessa dissertação são pautados da seguinte maneira:

- Investigar o problema do agendamento de cargas elétricas no contexto da aplicação de tarifas de energia variáveis no tempo visando a execução de uma estratégia para a redução da conta de energia.
- Formular matematicamente as funções custo que definam os processos de otimização dos aspectos relacionados ao custo total da conta de energia elétrica, ao conforto do usuário e ao fator de carga de uma instalação.
- Propor e realizar o desenvolvimento de um otimizador baseado em técnicas de computacional evolucionária, especificamente, Evolução Diferencial (ED).
- Realizar o desenvolvimento de uma ferramenta *open-source* que integre o otimizador para a aplicação em cenários reais, possibilitando a geração de planos de agendamentos eficazes para processos DSM.
- Ratificar o sistema desenvolvido através de simulações com eventuais cenários verossímeis à realidade brasileira.

## 1.3 Organização do conteúdo

Os demais capítulos dessa dissertação estão organizados como se segue:

No segundo capítulo descreve-se o gerenciamento pelo lado da demanda e os principais métodos de resposta da demanda.

No terceiro capítulo os principais componentes para a constituição do trabalho são apresentados. Neste capítulo discute-se sobre as especificações das diferentes categorias de cargas residenciais, da constituição do problema de agendamento e do algoritmo utilizado.

No capítulo quatro detalha-se a formulação matemática do problema e do desenvolvimento do solucionador baseado em evolução diferencial.

No quinto capítulo são apresentados os resultados das simulações em diferentes cenários com a minimização do custo, da elevação do fator de carga e do conforto do usuário.

No sexto capítulo destacam-se os principais aspectos do problema e de sua resolução durante o desenvolvimento do trabalho. Apresentam-se, também, os principais resultados obtidos e suas implicações. Por fim, expõe-se direcionamentos futuros.

# 2 GERENCIAMENTO PELO LADO DA DEMANDA

O gerenciamento pelo lado da demanda, ou DSM, engloba uma série de técnicas orientadas ao consumidor para a modificação dos padrões do consumo de forma a auxiliar o planejamento e as operações do sistema elétrico.

A utilização das estratégias envolvendo o gerenciamento pelo lado da demanda não é uma abordagem moderna. No início da década de 1970, a compreensão da necessidade do gerenciamento de cargas foi, em parte, motivado pelo advento de grandes sistemas de ares-condicionados que acarretavam baixos valores de fator de carga e de fator de potência no sistema elétrico (CAPPERS et al, 2009). Inicialmente, devido à indisponibilidade de equipamentos de comunicação eficientes e de baixo custo, as práticas de DSM eram, quando raramente utilizadas, implementadas de forma manual. Atualmente, o surgimento das SGs, o que inclui a união de TICs com a rede elétrica, proporciona novas formas de execução do DSM. Múltiplos objetivos como conforto, confiabilidade e índices de qualidade são atualmente incorporados ao DSM. A Figura 1 ilustra os principais componentes do gerenciamento pelo lado da demanda no presente.

Figura 1 - Divisões do gerenciamento pelo lado da demanda


Fonte: Adaptado de (CASTRO (2017))

Através da adoção desses artifícios o gerenciamento pelo lado da demanda pode alavancar benefícios que incluem desde fatores econômicos até ambientais. A redução dos picos de demanda se destaca como um dos principais benefícios econômicos alcançados através da utilização do DSM. Isso se deve ao fato de que, embora picos de demanda de energia sejam eventos momentâneos, seus impactos econômicos são significativos. A

necessidade de se manter uma estrutura de produção e distribuição de energia mais robusta para o fornecimento dessa alta momentânea na demanda exige grandes investimentos. Além disso, considera-se o aumento nos custos devido às perdas durante a transmissão e a distribuição. Outros benefícios obtidos através do DSM são: aumento da confiabilidade da rede, gerenciamento dos gastos com energia, favorecimento da geração distribuída e aumento da utilização e o fator de carga das unidades.

## 2.1 Resposta da Demanda

De modo geral, os programas de DSM baseados no conceito de resposta da demanda envolvem métodos aplicados à modificação da curva de demanda das companhias de energia elétrica. Os programas de DR dividem-se basicamente em métodos diretos e métodos indiretos. Essa divisão refere-se a motivação dada ao consumidor para sua adesão aos programas. Ambas as estratégias possuem inúmeras variantes.

### *2.1.1 Métodos indiretos de Resposta da Demanda*

Métodos indiretos de DR, também referidos como critérios de preço, envolvem estratégias que buscam modificar o modo de utilização da energia elétrica pelo usuário através da alteração nos valores das tarifas. Sobretudo, esses métodos dividem-se em: *Time-of-Use Pricing*, *Real-Time Pricing* e *Critical Peak Pricing* (CPP) (DOE, 2006).

A tarifação ToU baseia-se na utilização de valores distintos de tarifa em determinados momentos do dia objetivando reproduzir situações típicas do sistema elétrico. Usualmente os valores da tarifação ToU são definidos como: horário fora de ponta, horário intermediário e horário de ponta. Tarifas ToU são tradicionalmente mandatórias para grandes consumidores comerciais e industriais. No Brasil, por exemplo, são disponíveis as modalidades tarifárias ToU do tipo *horossazonal azul* e *horossazonal verde* para os grandes consumidores (alimentados em alta tensão). Um exemplo mais recente da adesão de tarifas do tipo ToU em território nacional refere-se a implementação da *tarifa branca* que utiliza o conceito para aplicação em consumidores alimentados em baixa tensão.

Programas de DR baseados em tarifação do tipo RTP, contudo, buscam refletir o custo da produção e distribuição da energia de hora em hora. Nesse tipo de programa os consumidores são sinalizados com os valores de RTP previamente com intervalos de tempo de

uma hora ou até um dia de antecedência. Embora essa estratégia seja capaz de reproduzir precisamente os valores agregados à produção de energia elétrica, a necessidade de uma infraestrutura de TIC de alto nível, somado a grande quantidade de decisões com relação ao uso da energia tornam o RTP, atualmente, uma solução mais complexa quando comparada a ToU.

Programas do tipo CPP são híbridos entre tarifas ToU e RTP. Nesse tipo de tarifação a estrutura básica é do tipo ToU, porém, com o acontecimento de eventos relacionados a confiabilidade no fornecimento ou na produção de energia, os valores de pico são reajustados para valores mais representativos. Ainda que não seja caracterizado como um programa do tipo CPP, a utilização das bandeiras tarifárias no Brasil utiliza um conceito similar onde se aplica uma taxa variável relacionada com fatores como períodos de seca e nível de água nos reservatórios das hidroelétricas.

### *2.1.2 Métodos diretos de Resposta da Demanda*

Métodos direto de DR, ou métodos de DR por incentivo, são programas que permitem à concessionária de energia o controle direto sobre as cargas dos usuários em troca do pagamento de incentivos ou créditos. A redução das cargas é realizada conforme a requisição do operador e geralmente depende de condições de comprometimento do sistema e/ou de preços elevados para a produção de energia. Os programas de DR baseados em incentivos financeiros dividem-se em: controle direto da demanda (DLC), interrupção consentida da demanda (I/C), redução da demanda de emergência e oferta de redução da demanda (LEVY, 2013).

No programa de DLC o operador é habilitado para, remotamente, desligar máquinas e equipamentos (ar condicionado, aquecedores de água, etc) do consumidor sendo o cliente avisado previamente. Esse tipo de programa é voltado para pequenos consumidores comerciais e residenciais.

No serviço de interrupção consentida, um esquema de créditos ou descontos é estabelecido para a redução da demanda em momentos de incidentes sobre o sistema. Esse tipo de programa exige um contrato e pode aplicar multas caso o consumidor não seja capaz de cumpri-lo no momento da requisição.

Os participantes do programa de oferta de redução da demanda de emergência

são beneficiados com pagamentos na necessidade da redução de cargas durante contingências da reserva de energia do sistema. Esse tipo de programa é realizado de forma voluntária e penalidades não são aplicadas caso o consumidor não responda ao pedido das concessionárias.

A oferta de redução da demanda (*Demand Side Bidding*) é geralmente designada para consumidores de grande porte que passam a oferecer reduções de cargas baseados no preço do mercado atacadista. Assim, tornando as curvas de demanda dos consumidores um elemento de formação de preço no mercado.

# 3 OTIMIZAÇÃO DA DEMANDA DE ENERGIA ELÉTRICA

Embora a aplicação de estratégias de resposta da demanda possuam grandes vantagens com relação ao modelo de precificação e a inserção dos consumidores na formação do preço da energia elétrica, a falta de ferramentas de suporte à decisão impõe grande dificuldade para obtenção de seus benefícios. A aquisição do máximo potencial das técnicas de DR costuma ser prejudicado especialmente pela incapacidade do usuário final de responder aos eventos do sistema elétrico e a variação no valor da tarifa.

Em um cenário residencial automatizado onde o agendamento de cargas é realizado como parte de uma infraestrutura de medição avançada — comumente referido como *Advanced Metering Infrastructure* ou AMI — os sinais de programas de DR concedidos pela concessionária de energia podem ser utilizados para inferência de um consumo ótimo, respeitando parâmetros como, por exemplo, custo e conforto. Nesse contexto, o agendamento das cargas impõe a utilização de técnicas computacionais de otimização. Visto que esse problema pode possuir inúmeras variáveis e que naturalmente será aplicada em sistemas computacionais de capacidade limitada, o desempenho do algoritmo utilizado é de extrema relevância.

De modo geral, o agendamento de cargas consiste em uma lista diária de tarefas pré-programadas com o seu respectivo horário de início e término. No gerenciamento automatizado das cargas elétricas, a otimização deve considerar, além do perfil de consumo de cada carga, o tipo de carga que ela representa com relação ao seu ciclo típico de trabalho e a sua disponibilidade para atuação.

## 3.1 Tipos de Cargas

Tipicamente o cenário doméstico envolve a utilização de cargas elétricas de diferentes características, como, por exemplo: congeladores, máquinas de lavar roupas, sistemas de iluminação, sistemas de escritório, entretenimento, etc. Para a realização do gerenciamento dessas cargas, a identificação das características dos equipamentos — que incluem duração, modos de operação, capacidade de antecipar ou postegar ciclos de operações e possibilidade de intervenção direta ou indireta — são essenciais para a escolha adequada da ação a ser realizada (SOARES et al, 2012).

Dessa maneira, as preferências do usuário final, a qualidade do serviço fornecido e as restrições técnicas definem grupos de cargas que podem ser classificados como:

1. Cargas não-controláveis: cargas que não podem ser submetidas ao gerenciamento autônomo, geralmente são cargas que não possuem ciclos de operação bem definidos.
2. Cargas parametrizáveis: controladas por termostato e que possuem ponto de operação configurável.
3. Cargas interrompíveis: cargas que podem ser desligadas por um período de tempo, dependendo das preferências do usuário.
4. Cargas deslocáveis: possuem flexibilidade com relação ao horário de início de operação.

A Tabela 1 apresenta uma lista dos equipamentos domésticos típicos e sua classificação de acordo com a sua categoria e capacidade de controle.

Tabela 1 - Tipos de cargas residenciais.

| ***Categoria*** | ***Equipamentos*** |
|---|---|
| *Não-controlável* | *Forno elétrico, equipamento de escritório, equipamentos de entretenimento, iluminação.* |
| *Parametrizável* | *Ares-condicionados, aquecedores, chuveiro elétrico.* |
| *Interrompível* | *Geladeiras, congeladores, sistemas de aquecimento de água* |
| *Deslocável* | *Máquina de lavar roupa, lava-louça, secadora de roupas, bombas de água.* |

Fonte: (SOARES et al, 2012).

Em adição a categorização clássica proposta acima, em sistemas de Recursos Energéticos Distribuídos (RED), sistemas de baterias e carros elétricos são considerados, em algumas vertentes, como um novo tipo de carga. Pelo fato desses sistemas serem capazes de operar tanto como uma fonte de energia ou como uma carga, esses equipamentos são classificados como sendo equipamentos assistidos por bateria — ou Battery-Assisted Appliances, BAs (TSUI e CHAN, 2012). A principal característica desse tipo de carga é sua capacidade de armazenamento de energia, que pode ser utilizada para operações de DR ou uso do equipamento de forma autônoma.

## 3.2 Otimização

O processo de otimização envolve, de maneira geral, a busca pela maximização de propriedades relevantes de um sistema. Por conseguinte, esse processo também incorpora a minimização de características indesejadas. Matematicamente, esse método é realizado através da busca por soluções ótimas de funções que modelam e são capazes de caracterizar o sistema e suas propriedades. Essa função recebe o nome de função objetivo quando deseja-se maximizar o seu valor e função custo, quando o interesse é a realização de um problema de minimização.

Após a realização da modelagem matemática do sistema, ou seja, após a obtenção das funções custo, a escolha de um otimizador adequado deve ser realizada considerando as características do problema. De forma genérica, os otimizadores operam de forma diferente dependendo do número de pontos (vetores) e do modo que os mesmos varrem o espaço *N* dimensional do problema. A Tabela 2 mostra a classificação de alguns algoritmos clássicos de otimização e do modo que eles operam com relação à quantidade de pontos em paralelo e o método de busca que utilizam.

Tabela 2 - Classificação e exemplos de otimizadores.

| | ***Único vetor*** | ***Multi-vetores*** |
|---|---|---|
| ***Busca em espaços contínuos (baseados em derivadas)*** | *Método do gradiente*<br>*Hill climbing*<br>*Método Quase Newton* | *Random-restart hill climbing* |
| ***Livres de derivadas*** | *Passeio aleatório*<br>*Método de Hooke-Jeeves* | *Nelder-Mead*<br>*Algoritmos evolucionários* |

Fonte: (PRICE et al, 2005).

Ainda que a divisão da Tabela 2 seja eficiente na representação das características das estratégias utilizadas, alguns algoritmos não possuem distinção bem definida e não podem ser alocados a uma categoria específica. Por exemplo, a classificação não pode ser bem atribuída no caso da estratégia de s*imulated annealing* (KIRKPATRICK et al, 1983) pois ela pode ser aplicada para quaisquer problemas de busca direta (livres de derivadas). Do mesmo modo, a classificação não é bem definida no caso de técnicas de agrupamento (*clustering*) que comumente são combinadas com otimizadores baseados em derivadas.

Neste trabalho, para a resolução do problema de otimização do agendamento de cargas, desenvolve-se um otimizador baseado na utilização de uma estratégia computacional

evolucionária. Especificamente, elabora-se a construção de um otimizador utilizando o método de Evolução Diferencial. Inicialmente, essa escolha se deve pela complexidade do problema — a otimização do agendamento de cargas é um problema de decisão do tipo NP-difícil — que exige a otimização de múltiplas variáveis em espaço multidimensional discreto.

### *3.2.1* Trabalhos Relacionados

Com a rápida modernização e implementação de diferentes estratégias de gerenciamento pelo lado da demanda, diferentes técnicas e abordagens para otimização do custo e da implementação de métodos de resposta da demanda tem sido propostas.

No trabalho de SIEBERT et al (2012) o gerenciamento do consumo de energia elétrica de uma residencia é realizado através do agendamento de dispositivos quando aplicado uma tarifa ToU. Algoritmos genéticos são utilizados para a realização da otimização no deslocamento das cargas. A otimização é realizada considerando apenas o custo total.

Em QUAYYUM et al (2015) apresenta-se o conceito de um controlador de cargas inteligente capaz de realizar o agendamento de cargas levando em consideração a exportação de energia através de geração fotovoltaica. Neste trabalho o problema é formulado em termos de custo total de eletricidade. A formulação do problema e a otimização é determinada através da utilização do *toolbox* YALMIP no MATLAB e utiliza-se da técnica de programação linear para minimização do custo.

AYODELE et al (2016) apresenta uma estratégia DSM para o agendamento de cargas definindo três postulados que permitem quantizar o nível de satisfação do usuário. Além disso, o padrão de uso das cargas é gerado através do algoritmo que toma como base um valor máximo predeterminado que o usuário pode pagar pelo uso de energia durante o dia.

Em VERAS et al (2017) um modelo de resposta da demanda para otimização do agendamento de cargas residenciais é aplicado utilizando algoritmos genéticos. Nessa abordagem são considerados os critérios de custo e conforto. Perfis típicos de consumidores são considerados para diferentes regiões do Brasil levando em consideração a diferença entre o clima e a utilização de diferentes tipos de cargas.

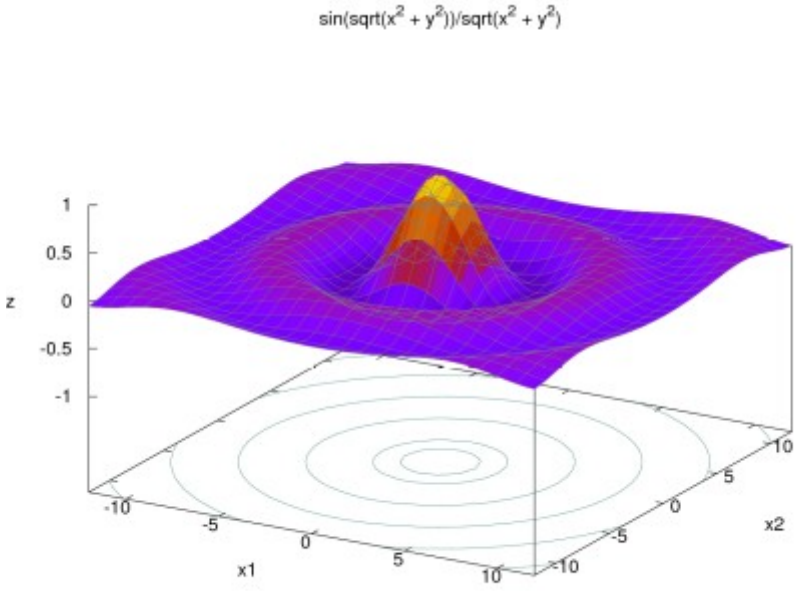
### *3.2.2 Evolução diferencial*

Idealizado por Storn e Price (1996), ED é uma estratégia evolucionária baseada em populações que varrem o espaço amostrando possíveis soluções em múltiplos pontos. Tal método busca melhorar soluções candidatas iterativamente a cada geração através da orientação dos seus protótipos, balanceando explotação e exploração, com base em outras possíveis soluções.

O método ED classifica-se como sendo um algoritmo metaheurístico, ou seja, a estratégia é responsável pela geração, busca e seleção de soluções que podem prover resultados suficientemente bons para o problema de otimização. Dessa forma, ainda que a solução ótima não seja necessariamente encontrada, uma solução satisfatória é descoberta fazendo-se poucas suposições sobre o problema. Essa característica torna o algoritmo extremamente versátil e permite grandes possibilidades para o aprimoramento do algoritmo.

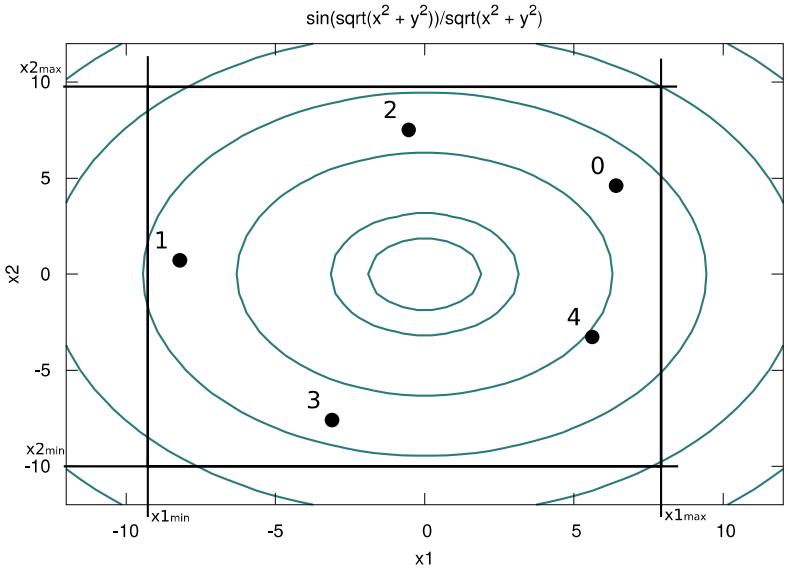
A implementação do algoritmo de ED baseia-se, inicialmente, em um conjunto de *Np* vetores que são alocados de maneira aleatória sobre o espaço da função. O exemplo da Figura 3 ilustra a inicialização de vetores sobre contornos da função $sen(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$ exibida na Figura 2. Observa-se que a população contém cinco vetores (*Np = 5)* e que os vetores são indexados com valores de 0 a *(Np – 1)*.

Figura 2 - Representação de uma função objetivo tridimensional e suas linhas de contorno.




Fonte: elaborado pelo autor.

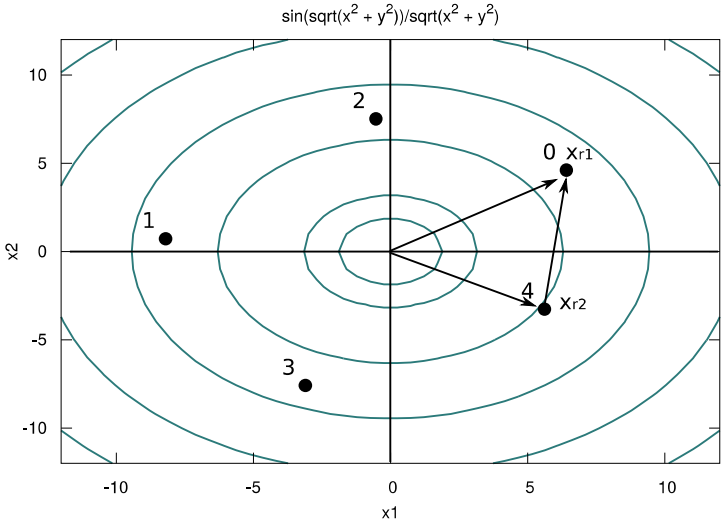
Figura 3 - Inicialização da população delimitada por uma região $x_{1min}/x_{2min}/x_{1max}/x_{2max}$.




Fonte: elaborado pelo autor.

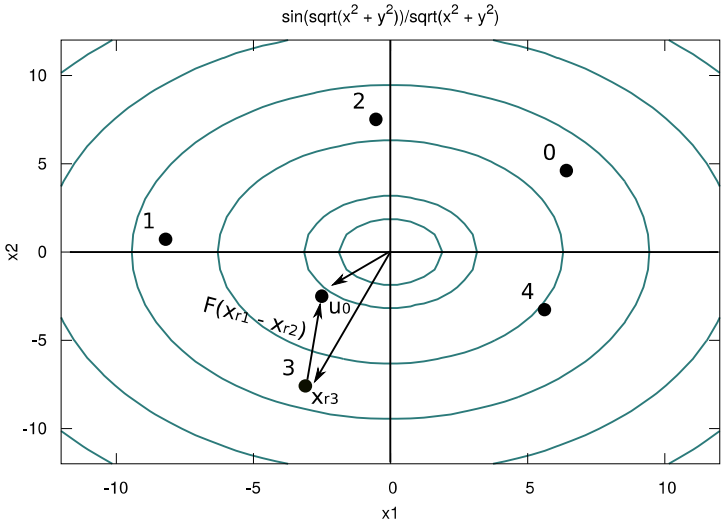
Em ED, assim como em outros métodos baseados em populações, novos vetores são gerados a partir de pertubações. Conforme a Figura 4, essa pertubação é gerada pela diferença escalonada entre dois outros vetores selecionados aleatoriamente dentro do conjunto da população. Esse vetor diferença, quando adicionado a outro vetor — que também é aleatoriamente selecionado — produz um novo elemento $u_i$. A Figura 5 ilustra esse processo.

Figura 4 - Cálculo da pertubação.




Fonte: elaborado pelo autor.

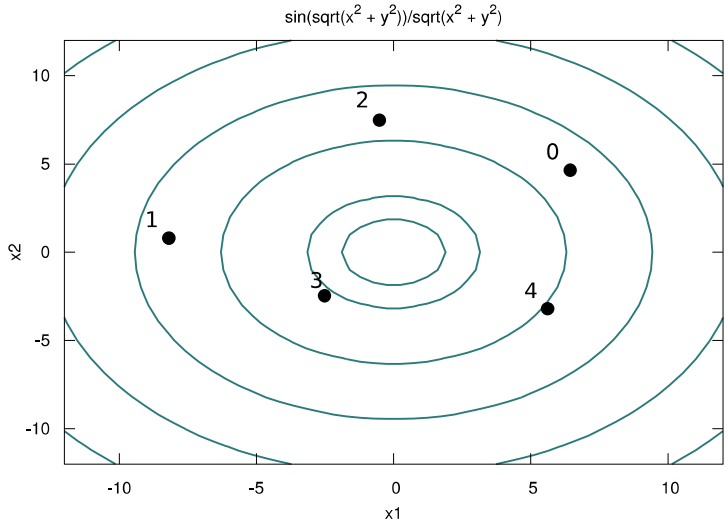
Figura 5 - Geração de novo vetor.




Fonte: elaborado pelo autor.

Durante o processo de seleção o novo vetor $u_i$ compete com o vetor antigo de mesmo índice. Caso o valor amostrado no novo ponto seja uma solução mais viável, ele é repassado para a nova geração (ver Figura 6). Após a comparação de todos os vetores $u_i$, os pontos selecionados tornam-se membros da nova população. O processo é então repetido até que um critério de parada seja atingido.

Figura 6 - Devido ao seu maior valor, o novo vetor é selecionado para a nova geração.




Fonte: elaborado pelo autor.

Embora o esquema apresentado constitua um método capaz de gerar bons resultados, o desempenho do algoritmo pode ser melhorado através de metodologias que são capazes de ajustar o método ao tipo de problema de otimização aplicado de maneira específica. Essas metodologias dividem o algoritmo em etapas bem definidas que podem ser decompostas como: inicialização, mutação, *crossover* e seleção.

### *3.2.2.1 Inicialização*

Usualmente a inicialização da população utilizando evolução diferencial é determinada por uma região previamente delimitada no espaço da função objetivo. De modo geral, definem-se vetores que especificam o limite inferior $\boldsymbol{L_i}$ e superior $\boldsymbol{L_s}$ para cada dimensão das quais os indivíduos da população inicial $\boldsymbol{x_i}$ poderão povoar. Após a delimitação dos vetores de limites, uma variável contínua aleatória é destinada a cada parâmetro $j$ dos vetores. A fórmula definida na Equação (3.1) define o processo de atribuição dos valores iniciais dos parâmetros de índice *j* para os indivíduos $\boldsymbol{x_i}$ da população inicial.

$$\boldsymbol{x_{i,j}} = aleatório\,(0,1) \times (\boldsymbol{L_{s,j}} - \boldsymbol{L_{i,j}}) + \boldsymbol{L_{i,j}} \qquad (3.1)$$

Durante o desenvolvimento do otimizador para este trabalho, um gerador de números pseudoaleatórios baseado na técnica *Mersenne Twister* (MATSUMOTO e NISHIMURA) foi utilizado para geração dos valores retornados pela função $aleatório(0,1)$. O gerador de números aleatórios é utilizado para produção de valores com base em uma função de densidade de probabilidade uniforme.

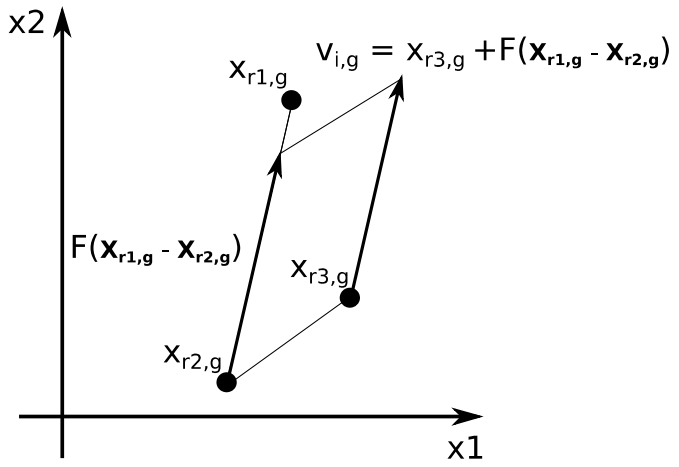
### *3.2.2.2 Mutação*

Após a inicialização, os indivíduos da população são submetidos a uma mutação. A mutação recombina indivíduos com o objetivo de produzir novos vetores que amostrem o espaço da função objetivo . Esse processo é realizado adicionando a um vetor aleatoriamente selecionado $\boldsymbol{x}_{r_{3,g}}$ a diferença escalonada entre dois outros vetores ( $\boldsymbol{x}_{r_{1,g}}, \boldsymbol{x}_{r_{2,g}}$ ), conforme a fórmula mostrada na Equação (3.2).

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{r_{3,g}} + F\left(\boldsymbol{x}_{r_{1,g}} - \boldsymbol{x}_{r_{2,g}}\right) \qquad (3.2)$$

O coeficiente de escalonamento *F* determina um valor multiplicador do vetor diferença que será utilizado para a pertubação. Tipicamente, atribui-se um valor entre 0 e 1, porém, não há um valor máximo determinado para o limite superior de *F*.

Figura 7 - Realização da mutação diferencial.




Fonte: elaborado pelo autor.

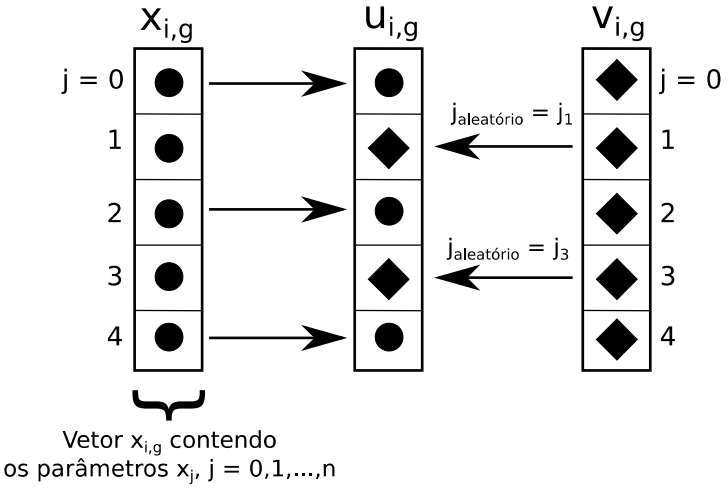
*3.2.2.3 Crossover*

O processo de *crossover* — também chamado de recombinação discreta (PRICE et al, 2005) — auxilia o algoritmo de ED na elevação do grau de entropia causado pela pertubação aplicada aos vetores mutantes. Esse processo é realizado através da utilização de uma taxa de probabilidade de *crossover* $C_r \in [0,1]$ que determina as chances de recombinação entre os parâmetros *j* do vetor $\boldsymbol{x}_{i,g}$ e do vetor mutante $\boldsymbol{v}_{i,g}$, conforme Equação (3.3).

$$\boldsymbol{u}_{i,g} = \boldsymbol{u}_{j,i,g} = \begin{cases} \boldsymbol{v}_{i,j}\ , \text{se } aleatório\,(0,1) \leqslant C_r \text{ ou } j = j_{aleatório} \\ \boldsymbol{x}_{i,j}\ , \text{caso contrário} \end{cases} \qquad (3.3)$$

Dessa maneira, caso o valor obtido através do gerador de números aleatórios da função *aleatório* seja menor que a taxa de controle *Cr*, o parâmetro da dimensão *j* do vetor mutante $\boldsymbol{v}_{i,g}$ é repassado para o indivíduo da próxima geração, caso contrário, o parâmetro do vetor original $\boldsymbol{x}_{i,g}$ é repassado. A Figura 8 exemplifica esse processo graficamente.

Figura 8 - Representação do processo de crossover para vetores de dimensão n = 5.




Fonte: adaptado de (PRICE e STORN, 1996).

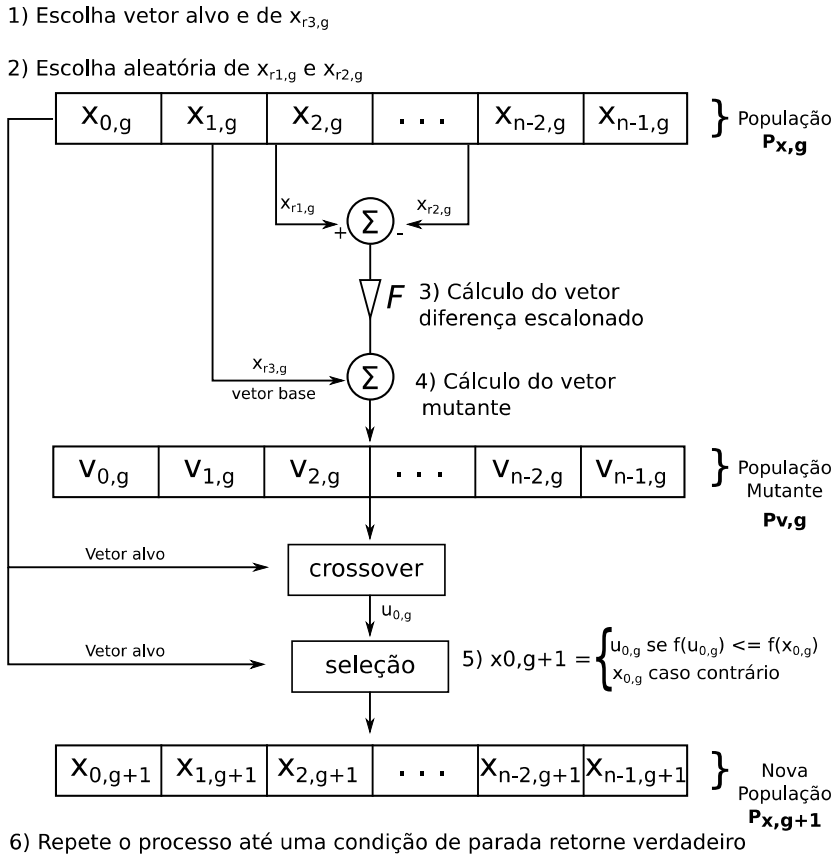
### *3.2.2.4 Seleção*

Por fim, para que o processo de convergência em direção ao valor ótimo seja melhor orientado e que o tamanho da população seja constante ao longo do processo de otimização, uma seleção dos vetores mais adequados é realizada. Esse processo é efetuado através da comparação do valor da função no ponto $\boldsymbol{x_{i,g}}$ com relação ao vetor obtido após o processo de mutação e *crossover* $\boldsymbol{u_{i,g}}$, conforme a Equação (3.4).

$$\boldsymbol{x_{i,g+1}} = \begin{cases} \boldsymbol{u_{i,g}}, \text{ se } f(\boldsymbol{u_{i,g}}) \leq f(\boldsymbol{x_{i,g}}) \\ \boldsymbol{x_{i,g}}, \text{ caso contrário} \end{cases} \tag{3.4}$$

Após a execução da seleção dos vetores, os processos de mutação, *crossover* e seleção são repetidos até que um critério de parada seja atingido. Os critérios de paradas são usualmente definidos como sendo a conclusão do número máximo de iterações pré-definida ou um valor de erro mínimo. A Figura 9 mostra o fluxograma com todas as etapas do algoritmo.

Figura 9 - Fluxograma do algoritmo de evolução diferencial.




Fonte: adaptado de (PRICE et al, 2005).

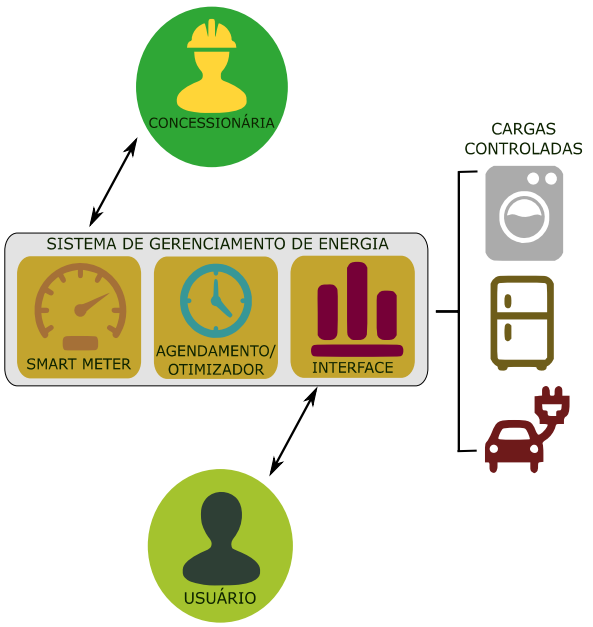
# 4 FORMULAÇÃO DO PROBLEMA E SOLUÇÃO

## 4.1 Sistema de gerenciamento de energia

A Figura 10 ilustra a utilização do sistema de otimização e agendamento de cargas inseridos no sistema de gerenciamento de energia localizado na residência do consumidor.

Nesse cenário, dados referentes as tarifas e incentivos são fornecidos pela concessionária de energia. Informações acerca das preferências e dados sobre as cargas são inseridas no sistema pelo usuário através de uma interface. Considera-se também que as cargas constituintes do sistema são passíveis de controle através de automação, ou seja, as cargas devem ser cargas capazes de atuar ligando ou desligando o fornecimento de energia através da sinalização dada pelo sistema de gerenciamento de energia. Uma outra possibilidade é que essas cargas são conectadas na rede através de tomadas inteligentes que desempenham a função de atuação.

Figura 10 - Modelo do sistema de gerenciamento de energia com agendamento de cargas.




Fonte: elaborado pelo autor.

É importante observar que, embora o módulo de agendamento e otimização apresentando na Figura acima se encontre inserido dentro do sistema de gerenciamento de energia, esse módulo pode ser uma componente do medidor inteligente.

## 4.2 Definição do problema

O problema do agendamento de cargas é definido em termos de intervalos de tempo discreto. Uma quantidade de *M* blocos de $T_s$ minutos dividem um período de 24 horas. Cada unidade de tempo é indicada pelo índice *k*. Informações acerca do preço da tarifa fornecida pela concessionária são denominadas por $C_k$, onde $C_k$ representa o custo em termos de valores monetários para cada kWh no instante *k*.

Um conjunto de *N* cargas elétricas são definidas com seus correspondentes dados definidos pelo usuário acerca do tempo de início preferível $T_i^p$, do tempo de início de janela $T_{sw}^i$ e do tempo de término de janela $T_{ew}^i$ para cada carga $n_i$ sendo *i = 1, 2, 3, ... N*. A fim de garantir o funcionamento de $n_i$ dentro da janela de tempo especificada, a restrição da Equação (4.1) é definida.

$$T_{sw}^i \leq T_{st}^k + T_d^k \leq T_{ew}^i \qquad (4.1)$$

Cada carga $n_i$ contém o seu respectivo coeficiente de conforto $w_i$. O coeficiente de conforto é utilizado como peso indicando ao agendamento o grau de prioridade para a respectiva carga iniciar sua operação no tempo de horário de preferência $T_p^i$. De maneira oposta, o coeficiente de conforto $w_i$ também pode ser utilizado com valor nulo caso o usuário não se importe com o horário de início específico para a carga $n_i$.

O perfil de carga $P_{ij}^k$, definido em kW, representa a potência consumida pela carga *i* no ciclo de operação *j*. Cada ciclo de operação *j* representa o consumo médio da carga em $T_s$ minutos após ser realizado o acionamento da carga no instante $T_{st}$.

A Tabela 3 sumariza os símbolos e notações utilizadas ao longo da dissertação para a formulação do problema.

Tabela 3 - Lista de Símbolos

| ***Símbolo*** | ***Definição*** |
|---|---|
| $M$ | *Quantidade de blocos de tempo* |
| $N$ | *Quantidade total de cargas* |
| $R$ | *Quantidade total de ciclos de operação* |
| $k$ | *Índice do bloco de tempo em M* |
| $i$ | *Índice da carga em N* |
| $j$ | *Índice do ciclo de operação* |
| $T_p^i$ | *Horário confortável de i* |
| $T_{sw}^i$ | *Início da janela para i* |
| $T_{ew}^i$ | *Fim de janela para i* |
| $T_d^i$ | *Duração da carga i* |
| $C_k$ | *Valor da tarifa em k* |
| $P_{ij}^k$ | *Perfil de carga de i em j* |
| $w_i$ | *Coeficiente de conforto de i* |

Fonte: elaborado pelo autor.

## 4.3 Funções custo

### *4.3.1 Minimização do custo*

O primeiro objetivo do módulo de otimização do agendamento de cargas é a minimização do custo total de eletricidade durante um período de 24 horas quando aplicada sobre o consumidor tarifas variáveis no tempo.

Seguindo a convenção da Tabela 3, considerando o valor $C_k$ [reais/kWh] da tarifa para o instante *k*, o custo total pode ser minimizado através da Equação (4.2).

$$F_{CT} = min\left\{\left(\frac{T_s}{60}\right)\sum_{k=1}^{M} C_k\left(\sum_{i=1}^{N}\sum_{j=1}^{R} P_{ij}^k\right)\right\} \qquad (4.2)$$

A função custo $F_{CT}$ apresentada acima define o custo total da eletricidade consumida iterando os *M* blocos de tempo de $T_s$ minutos através do primeiro somatório de índice *k*. Consequentemente, o segundo somatório itera todos os equipamentos inseridos no agendamento para que seja possível a realização do cálculo do produto entre a energia $P_{ij}^{k}$ [kWh] do equipamento $i$ durante o ciclo *j* de operação — representado no somatório mais à direita — e o valor da tarifa $C_k$. O termo constante $T_s/60$ é utilizado para normalização das unidades.

### *4.3.2 Maximização das preferências do agendamento*

A maximização das preferências do agendamento define-se como sendo a capacidade do usuário de definir um horário específico para o início do ciclo de operação de um equipamento. A preferência do agendamento relaciona-se diretamente com o nível de conforto do usuário, pois, retrata a utilização das cargas sem a inconveniência de adiantamentos ou atrasos. Dessa maneira, a maximização das preferências do agendamento pode ser tratada de forma intercambiável como a maximização do conforto do usuário.

A função custo que considera o critério da maximização das preferências do agendamento é apresentada na Equação (4.3).

$$F_{PA} = min\left\{\sum_{i=1}^{N} w_i \sqrt{\left(T_p^i - T_{st}^i\right)^2}\right\} \qquad (4.3)$$

Na equação acima o problema da maximização da preferência de agendamento de cada carga é reformulado e tratado como um problema de minimização das distâncias entre o instante de tempo de início de operação $T_{st}^i$ e o instante de tempo de preferência $T_p^i$ definido pelo usuário. O coeficiente de preferência $w_i \in [0,1]$ define valores de prioridade para o agendamento. Dessa maneira, o valor do peso $w_i$ associado à i-ésima carga indica a relevância da distância entre os horários de início e preferência no cálculo do valor da função custo. Caso $w_i = 0$, a i-ésima carga associada ao peso não terá influência no cálculo da função e, portanto, poderá possuir um horário de início diferente do horário de preferência associado.

### *4.3.3 Maximização do fator de carga*

Segundo a resolução normativa nº 414 de 9 de setembro de 2010 da ANEEL, o fator de carga é definido como sendo "a razão entre a demanda média e a demanda máxima da unidade consumidora ocorrida no mesmo intervalo de tempo especificado" (ANEEL, 2010). O fator de carga é, portanto, um índice adimensional com valores que variam de 0 a 1 que permite identificar o quanto de energia está sendo utilizada de forma racional.

Dessa maneira, a maximização do fator de carga representa, além da redução de picos de demanda, uma maior eficiência na utilização da instalação elétrica. A Equação (4.4) apresenta a função custo considerada para a maximização dessa característica.

$$F_{FC} = min\left\{1 - \frac{\sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{R} P_{ij}^{k}}{M \times P_{max}^{k}}\right\} \qquad (4.4)$$

Similarmente a função custo $F_{PA}$, o problema de maximização do fator de carga é reformulado para um problema de minimização. Por se tratar de um índice de valor máximo unitário, a função custo $F_{FC}$ trata do cálculo da minimização do complemento do fator de carga, fazendo, portanto, com que o fator de carga seja maximizado. Nessa equação, subtrai-se do valor máximo a razão entre o calculo da média da demanda de energia e do valor máximo de energia durante o período de tempo considerado.

### *4.3.4 Função custo com múltiplos objetivos*

Para que as funções custo apresentadas possam ser utilizadas para a geração de um agendamento que considere os efeitos das três funções apresentadas com seus efeitos associados, uma única função custo é gerada como uma combinação linear dessas funções, conforme a Equação (4.5).

$$F_L = \alpha \times F_{CT} + \beta \times F_{PA} + \gamma \times F_{FC} \qquad (4.5)$$

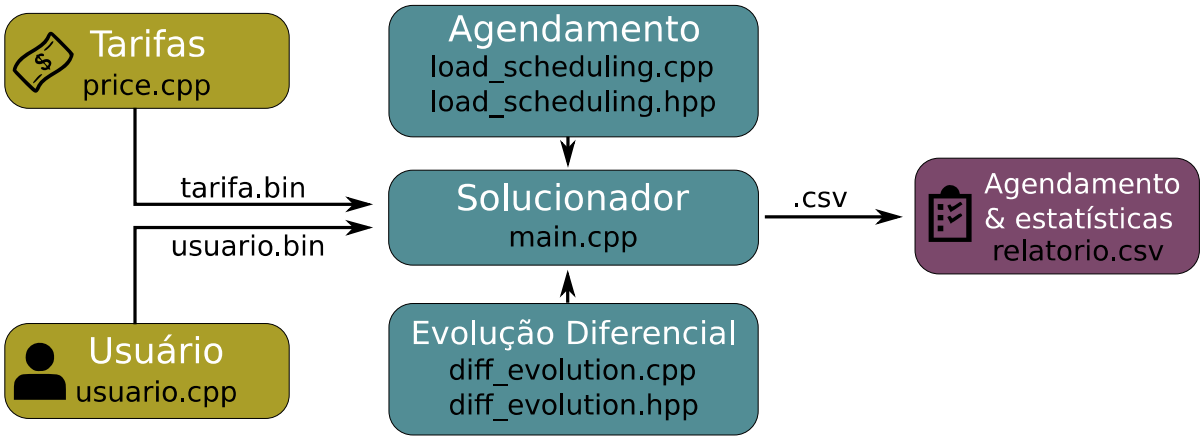
Com esse artifício, os coeficientes da Equação (4.5) refletem o tamanho da parcela de contribuição de sua função custo associada. No caso em que se pretende, por exemplo, a obtenção de uma solução na qual apenas o custo total e o conforto é otimizado, pode-se utilizar os coeficientes $\alpha = \beta = 1$ e $\gamma = 0$. Nota-se que a variação dos coeficientes constitui uma ferramenta importante para a geração de diversos modos de solução do problema, podendo, assim, combinar às três características definidas pelas funções previamente apresentadas e permitindo ao usuário indicar o nível de importância de cada uma delas, individualmente.

Embora cada coeficiente possa assumir qualquer valor numérico real, objetivando uma melhor leitura e organização do problema, os valores de cada um dos coeficientes podem assumir qualquer valor no intervalo $[0,1]$. Dessa maneira, caso o valor associado a característica seja unitário, essa característica será considerada com o maior peso possível. De modo inverso, a característica pode ser ignorada associando o valor nulo ao seu coeficiente. Qualquer outro valor no intervalo indicará um nível de consideração intermediário.

## 4.4 Módulos do software

Nesta seção descreve-se o arranjo e desenvolvimento do conjunto de ferramentas, escrito em linguagem C++, que compreende a geração dos arquivos de precificação, do desenvolvimento do solucionador e do módulo de agendamento de cargas residenciais. O esquema da Figura 11 apresenta os diferentes componentes e interfaces do programa.

Figura 11 - Arquivos do sistema de agendamento.




Fonte: elaborado pelo autor.

Conforme o diagrama da Figura 11, estão representadas à esquerda as duas interfaces para geração dos arquivos contendo informações sobre os valores dinâmicos da tarifa e sobre as informações acerca dos equipamentos e preferências do usuário. No centro, duas classes foram desenvolvidas para a formulação do problema e para a implementação do otimizador baseado em evolução diferencial. Cada classe é responsável por uma série de funcionalidades que auxiliam e abstraem parte do processo necessário para produção do agendamento. A organização e o agrupamento para montagem do agendamento, por sua vez, é elaborado no arquivo *main.cpp* definindo a sequência de leitura das informações, execução do otimizador e saída dos dados. Por fim, após a completa execução do programa, um arquivo csv (*comma-separated values*) é gerado contendo as informações do novo agendamento e dos índices de custo, conforto e fator de carga da nova configuração.

### *4.4.1 Sinais de tarifa e informações do usuário*

Conforme o cenário apresentado na Figura 10, sinais de valor de tarifa fornecidos pela concessionária de energia são repassados para o sistema de gerenciamento de energia através de uma infraestrutura de comunicação. Similarmente, os dados relativos aos diferentes tipos de cargas e preferências do usuário são introduzidas no sistema através de uma interface. Buscando fornecer uma aproximação da arquitetura real da AMI, os módulos de tarifação e de interface do usuário auxiliam na representação do sistema de agendamento. Ambos os módulos auxiliam na inserção dos dados advindos da concessionária e do usuário dentro do sistema sem que seja necessária uma intervenção direta no agendador.

O desenvolvimento do módulo de tarifação permite a produção de um arquivo binário contendo uma estrutura de dados com valores de tarifa para cada período de $T_s$ minutos durante o dia considerado. Dessa maneira, as informações relativas ao custo energético podem ser repassadas para o módulo do agendamento de cargas para realização dos cálculos de custo. Nesse sentido, esse arranjo permite a produção de novos agendamentos conforme o recebimento de sinais diários de tarifa, característica relevante em cenários com tarifas dinâmicas, como no caso das tarifas do tipo RTP, por exemplo.

O módulo de interface do usuário é constituído por uma série de comandos que definem a criação de um arquivo binário contendo configurações de preferências e carga que são utilizadas como entrada para o otimizador.

### *4.4.2 Solucionador*

O módulo solucionador compreende as interfaces de leitura e saída dos dados, validação do formato das informações e a produção de novas soluções utilizando o algoritmo de otimização. Inicialmente as informações relativas aos valores de tarifas advindos do arquivo *tarifa.bin* são lidos e alocados a um vetor para futuro cálculo do custo energético. De maneira similar, a leitura do arquivo *usuario.bin* retorna dados acerca das preferências do usuário e das cargas consideradas para o agendamento. As informações sobre as cargas constituem, então, uma estrutura de dados chamada *equipamento_t*, conforme Figura 12.

Figura 12 - Estrutura contendo dados das cargas.

```
typedef struct {
  std::string nome;
  double coef_conforto;
  unsigned int hora_inicio;
  unsigned int hora_preferencia;
  unsigned int hora_inicio_janela;
  unsigned int hora_termino_janela;
  std::vector<double> vetor_energia;
} equipamento_t;
```

Fonte: elaborado pelo autor.

A declaração desse novo tipo de dados é essencial para a realização da otimização pois cada nova instância de *equipamento_t* compõe um novo indivíduo de uma população considerada no algoritmo de ED. O Algoritmo 1 mostra em forma de pseudocódigo a construção do solucinador. Nessa descrição não é detalhado o desenvolvimento das duas classes *load_scheduling* e *diff_evolution* que auxiliam na formulação do problema e na construção do algoritmo, respectivamente.

Algoritmo 1 - Pseudocódigo do solucionador.

***Constantes:***

*numérico Tamanho da População, Tamanho do Passo, Probabilidade de Crossover, Maxímo de Iterações, Número de Cargas, Minutos por Slot;*

***Variáveis:***

*numérico* Tarifa[24 * 60 / Minutos por Slot], Contador;

*equipamento_t* Equipamentos[Número de Cargas];

***Início***

*Leitura do arquivo tarifa.bin* e armazenamento dos valores em Tarifa;

Leitura do arquivo *usuario.bin* e composição do vetor Equipamentos*;*

Inicialização do agendamento e da População Atual;

**Para** i ← 0 *até* Tamanho da População:
    População Atual ← novo LoadScheduling(Equipamentos, Número de Cargas);
**Fim**

Inicialização do otimizador ED;

**Enquanto** Contador < Máximo de Iterações*:*
    Geração de uma nova população;
    Operação de seleção;
    Contador ← Contador + 1;
**Fim**

Solução ← Seleção do melhor agendamento

**Se** criação de um novo arquivo *relatório.csv* é verdadeiro:
    Escreve Solução em relatório.csv;
**Senão**
    Exibe mensagem de erro;
**Fim**

Imprime na tela Solução;

***Fim***

***Retorna:*** *Verdadeiro caso fim do programa;*

# 5 ESTUDO DE CASO

A avaliação do programa de agendamento de cargas para a minimização dos fatores relacionados à função de custo multiobjetivo da Equação 4.5 é realizada nessa seção através da sua execução em diferentes cenários de tarifas, tipos de cargas e preferências do usuário. Seus diferentes perfis de consumo são baseados no trabalho (QUAYYUM et al, 2015).

Toda simulação é realizada através da utilização exclusiva dos aplicativos desenvolvidos — o código-fonte dos arquivos em C++ utilizados estão disponíveis no apêndice deste trabalho — que são utilizados para geração dos arquivos de tarifa, dos arquivos de cargas e preferências, computação do algoritmo de otimização e geração dos arquivos de saída contendo os resultados. Os gráficos apresentados são representações do valor de consumo de energia elétrica ao longo de todo o dia após o agendamento. Todos os gráficos são gerados a partir dos pontos fornecidos pela saída do otimizador utilizando a aplicação Gnuplot 5.0.

Os testes foram realizados utilizando um computador pessoal contendo como *hardware* de processamento e memória um processador Core i5-6200 (2.3 GHz), memória RAM DDR4 de 8 GB e 1000 GB HDD.

## 5.1 Cenário 01: Tarifa constante

Nesse cenário o programa é executado com a utilização de uma tarifa monômia fixa, ou seja, uma tarifa na qual o valor monetário é aplicado unicamente ao consumo de energia elétrica ativa com valor constante ao longo dia. No Brasil esse tipo de tarifa, até o momento da publicação deste trabalho, é considerada o modelo padrão de tarifação de consumidores residenciais de baixa tensão (tipo B).

As Tabelas 4 e 5 mostram os perfis de consumo das cargas e as preferências do usuário considerados para esse cenário. Os parâmetros do otimizador são mostrados na Tabela 6. O valor da tarifa é de R$ 0,72956/kWh e é invariável ao longo do dia. As Figuras 13 e 14 mostram diferentes resultados para os cenários de otimização do conforto e, em seguida, dos fatores de conforto e fator de carga combinados. É importante observar que, nesse cenário, o custo energético não é considerado para otimização, pois, se trata de uma tarifa de valor constante, e portanto, não existindo diferença entre valores devido à alocação das cargas.

Tabela 4 - Perfil de consumo dos equipamentos para o primeiro cenário.

| ***Nº*** | ***Equipamento*** | ***Duração (slots)*** | ***Duração (min)*** | $P_{i,j}^{k}$ ***(kWh)*** |
|---|---|---|---|---|
| *1* | *Máquina de Lavar* | *8* | *120* | *[0,5; 0,5; 0,5; 0,1; 3,0; 3,0; 3,0; 0,5]* |
| *2* | *Refrigerador* | *95* | *1440* | *[0,34; 0,34; 0,34 ... 0,34; 0,34]* |
| *3* | *A/C* | *16* | *240* | *[0,15; 0,35; 0,35; 0,15; 0,35; 0,35 ... 0,35; 0,35; 0,15]* |
| *4* | *Carro Elétrico I* | *10* | *150* | *[3,0; 3,0; 3,0; 3,0 ... 3,0; 3,0; 3,0]* |
| *5* | *Carro Elétrico II* | *10* | *150* | *[3,1; 2,9; 3,0; 3,0 ... 3,0; 3,0; 3,0]* |
| *6* | *Fogão Elétrico* | 3 | *45* | *[1,3; 0,8; 1,1]* |
| *7* | *Lava-louças* | 5 | *75* | *[1,2; 1,2; 0,3; 0,7; 0,6]* |
| *8* | *Bomba de água* | 2 | *30* | *[1,0; 1,0]* |

Fonte: elaborado pelo autor.

Tabela 5 - Preferências do usuário para o primeiro cenário.

| ***Nº*** | ***Equipamento*** | ***Horário de Preferência*** | ***Horário de Início de Janela*** | ***Horário de Término de Janela*** | $w_i$ |
|---|---|---|---|---|---|
| *1* | *Máquina de Lavar* | *17:30* | *00:00* | *20:00* | *1,0* |
| *2* | *Refrigerador* | *00:00* | *00:00* | *23:45* | *1,0* |
| *3* | *A/C* | *14:00* | *14:00* | *18:00* | *1,0* |
| *4* | *Carro Elétrico I* | *10:00* | *10:00* | *14:00* | *1,0* |
| *5* | *Carro Elétrico II* | *20:00* | *18:00* | *23:45* | *1,0* |
| *6* | *Fogão Elétrico* | 12:00 | *12:00* | *12:45* | *1,0* |
| *7* | *Lava-louças* | 13:00 | *13:00* | *16:00* | *1,0* |
| *8* | *Bomba de água* | 18:00 | *17:30* | *21:00* | *1,0* |

Fonte: elaborado pelo autor.

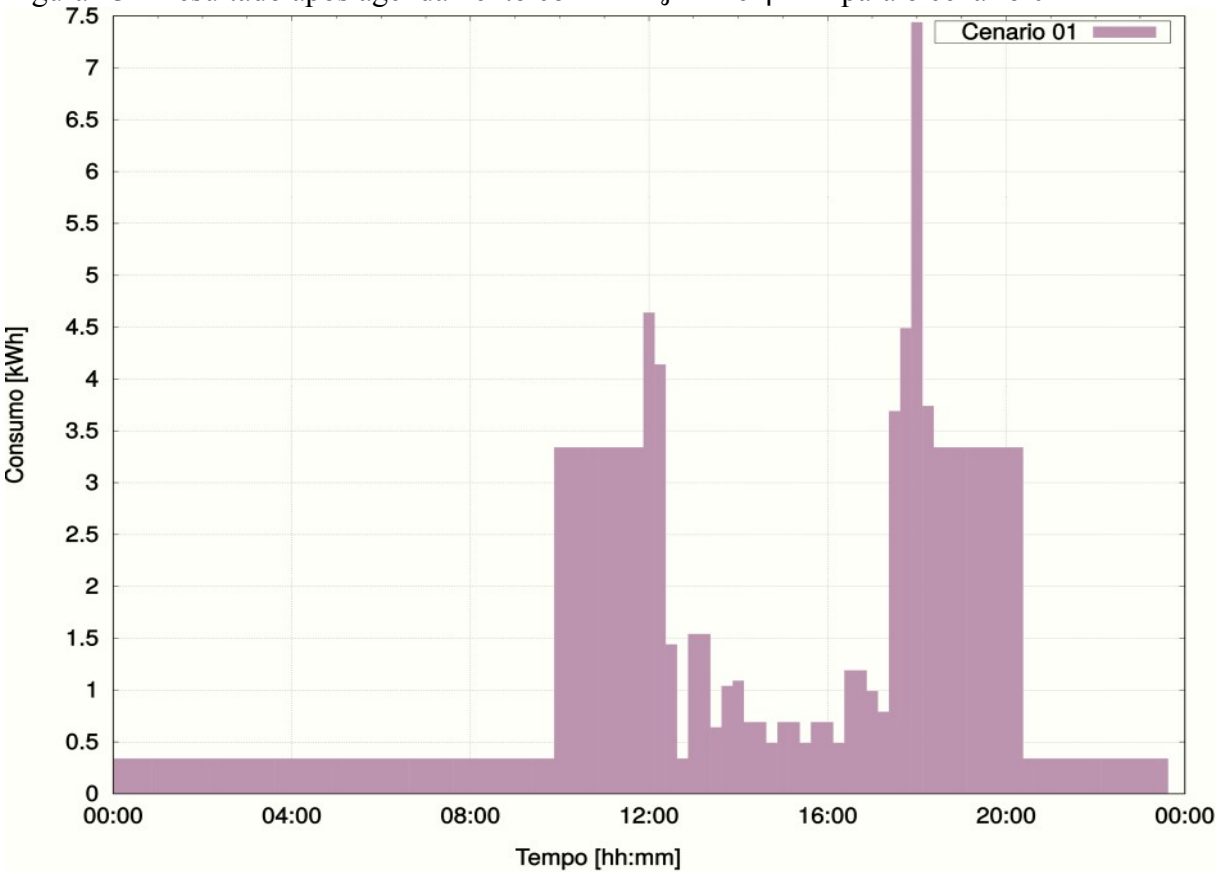
Tabela 6 - Parâmetros do otimizador para o cenário 01.

| ***Símbolo*** | ***Parâmetro*** | ***Valor*** |
|---|---|---|
| *MAX_ITE* | *Máximo de Iterações* | *3000* |
| $C_r$ | *Probabilidade de Crossover* | *0,6* |
| *F* | *Tamanho do Passo* | *0,1* |
| $T_s$ | *Minutos por Slot* | *15* |

Fonte: elaborado pelo autor.

A Figura 13 e a Tabela 7 exibem o resultado do agendamento de carga para a execução da simulação considerando apenas o nível de conforto do usuário.

Figura 13 - Resultado após agendamento com $\alpha = \gamma = 0$ e $\beta = 1$ para o cenário 01




Fonte: elaborado pelo autor.

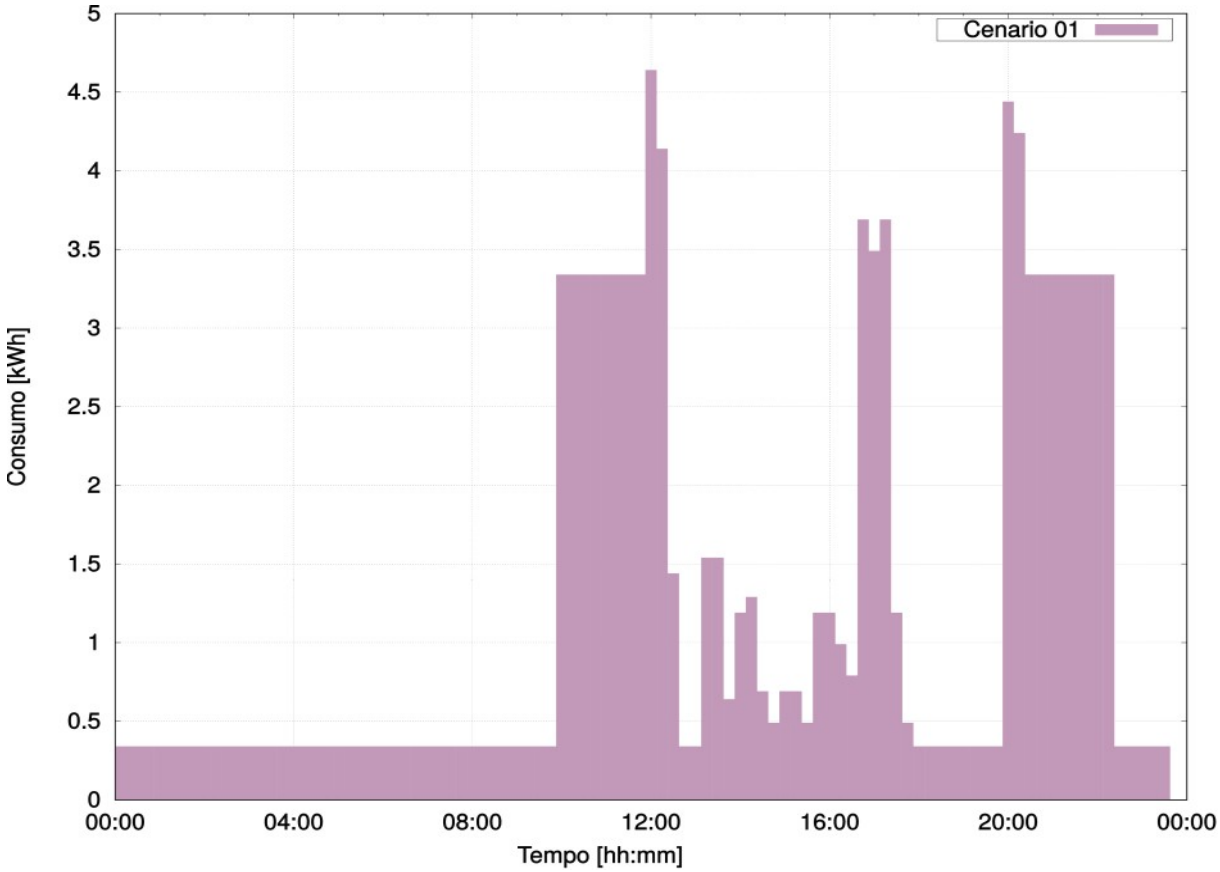
Tabela 7 - Resultados após agendamento com $\alpha = \gamma = 0$ e $\beta = 1$ para o cenário 01.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *85,3585* |
| *Nível de Conforto* | *0,904138* |
| *Fator de Carga* | *0,16381* |
| *Tempo de Execução (ms)* | *8378,12* |

Fonte: elaborado pelo autor.

A seguir, os resultados do agendamento para a execução do conforto do usuário e do fator de carga são exibidos na Figura 14 e na Tabela 8.

Figura 14 - Resultado após agendamento com $\alpha=0$ e $\beta=\gamma=1$ para o cenário 01.




Fonte: elaborado pelo autor.

Tabela 8 - Resultados após agendamento com $\alpha=0$ e $\beta=\gamma=1$ para o cenário 01.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *85,3585* |
| *Nível de Conforto* | *0,865278* |
| *Fator de Carga* | *0,262662* |
| *Tempo de Execução (ms)* | *8419,43* |

Fonte: elaborado pelo autor.

Neste cenário, devido à ocorrência de sobreposição no horário de funcionamento das cargas número 1 e 8, ao considerar no primeiro teste apenas o conforto nota-se que o fator de carga, quando comparado com o segundo teste, diminui. Isso ocorre pois ambas as cargas (1 e 8) contribuem para a criação de um pico de demanda por volta das 18:00 horas. No segundo teste, ao computar o valor da função custo da Equação 4.4, o algoritmo determina um novo agendamento que evita a sobreposição dos horários de funcionamento de ambas as cargas. Dessa maneira, essa nova alocação das cargas provoca a elevação do fator de carga, uma vez que evita a formação do pico de demanda devido ao funcionamento das cargas 1 e 8 em paralelo. Contudo, devido à maior diferença do horário de preferência, o nível de conforto do usuário é reduzido.

É importante apontar que, embora o cenário de tarifa fixa seja um dos mais simples — pois o custo total do consumo de energia elétrica não necessita de otimização — o agendamento ótimo das cargas torna-se necessário quando se objetiva a elevação do fator de carga. A sobreposição no horário de funcionamento de cargas, como apresentado no cenário 1, é um comportamento ordinário para consumidores residenciais e provoca a geração de picos de demanda indesejados que reduzem o valor do fator de carga. Conforme apresentado com os resultados acima, observa-se que o otimizador é capaz de realizar um novo agendamento considerando o conforto e o fator de carga equilibrando os fatores de modo a produzir uma alocação adequada em um tempo de processamento bem reduzido (aproximadamente 8,5 segundos para um conjunto de oito cargas).

## 5.2 Cenário 02: tarifa branca

Neste cenário os equipamentos utilizados e seus respectivos perfis de cargas são mostrados na Tabela 9. Os dados acerca das preferências do usuário são exibidos na Tabela 10. Os parâmetros do otimizador utilizados permanecem constantes e são definidos conforme o cenário anterior (Tabela 6).

Para a verificação do agendamento utilizando uma tarifa variável no tempo, adotou-se como modelo tarifário a tarifa branca. Esse tipo de tarifa, conforme já descrito na seção 2.1.1, projeta-se na realidade brasileira como uma nova modalidade tarifária residencial tornando-se, dessa maneira, objeto de grande interesse para análise. Os valores adotados para o teste são definidos como:

- Tarifa de ponta (TPT): R$ 1,42294/kWh entre 18:30-20:30.
- Tarifa intermediária (TINT): R$ 0,88144/kWh entre 17:30-18:30 e 20:30-21:30.
- Tarifa fora de ponta (TFP): R$ 0,56355/kWh no restante do tempo.

Tabela 9 - Perfil de consumo dos equipamentos para o primeiro cenário.

| ***Nº*** | ***Equipamento*** | ***Duração (slots)*** | ***Duração (min)*** | $P_{i,j}^{k}$ ***(kWh)*** |
|---|---|---|---|---|
| *1* | *Máquina de Lavar* | *8* | *120* | *[0,5; 0,5; 0,5; 0,1; 3,0; 3,0; 3,0; 0,5]* |
| *2* | *Refrigerador* | *95* | *1440* | *[0,34; 0,34; 0,34 ... 0,34; 0,34]* |
| *3* | *A/C I* | *16* | *240* | *[0,15; 0,35; 0,35; 0,15; 0,35; 0,35 ... 0,35; 0,35; 0,15]* |
| *4* | *A/C II* | *16* | *240* | *[0,15; 0,35; 0,35; 0,15; 0,35; 0,35 ... 0,35; 0,35; 0,15]* |
| *5* | *Carro Elétrico I* | *10* | *150* | *[3,0; 3,0; 3,0; 3,0 ... 3,0; 3,0; 3,0]* |
| *6* | *Carro Elétrico II* | *10* | *150* | *[3,1; 2,9; 3,0; 3,0 ... 3,0; 3,0; 3,0]* |
| *7* | *Fogão Elétrico* | 3 | *45* | *[1,3; 0,8; 1,1]* |
| *8* | *Lava-louças* | 5 | *75* | *[1,2; 1,2; 0,3; 0,7; 0,6]* |
| *9* | *Bomba de água* | 3 | *45* | *[1,0; 1,0; 1,0]* |
| *10* | *Bomba de recalque* | 2 | *30* | *[1,0; 1,0]* |

Fonte: elaborado pelo autor.

Tabela 10 - Preferências do usuário para o primeiro cenário.

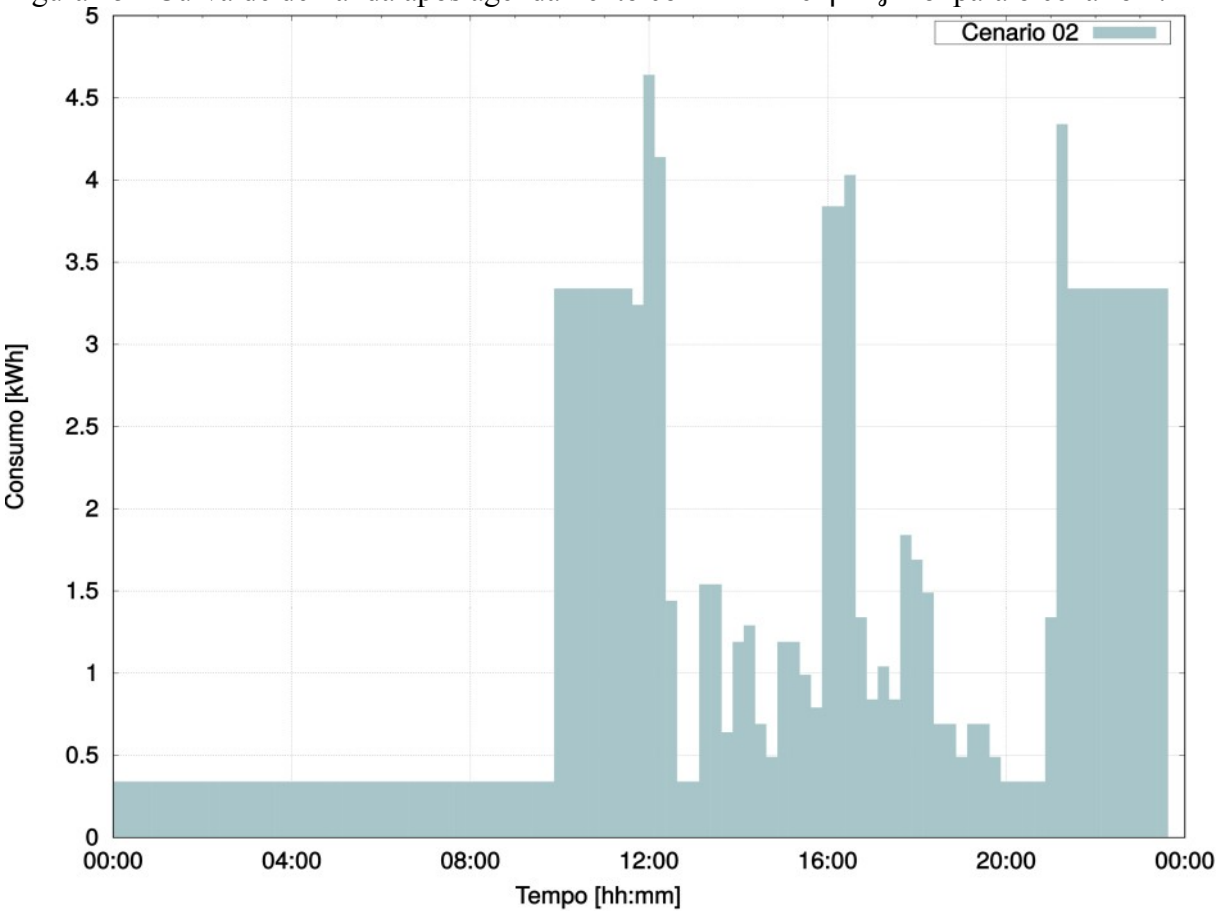
| ***Nº*** | ***Equipamento*** | ***Horário de Preferência*** | ***Horário de Início de Janela*** | ***Horário de Término de Janela*** | $w_i$ |
|---|---|---|---|---|---|
| *1* | *Máquina de Lavar* | *03:00* | *00:00* | *20:00* | *0,5* |
| *2* | *Refrigerador* | *00:00* | *00:00* | *23:45* | *1,0* |
| *3* | *A/C I* | *14:00* | *14:00* | *18:00* | *1,0* |
| *4* | *A/C II* | *16:00* | *16:00* | *20:00* | *1,0* |
| *5* | *Carro Elétrico I* | *10:00* | *08:00* | *18:00* | *1,0* |
| *6* | *Carro Elétrico II* | *20:00* | *16:00* | *23:45* | *1,0* |

| | | | | | |
|---|---|---|---|---|---|
| *7* | *Fogão Elétrico* | 12:00 | *12:00* | *12:45* | *1,0* |
| *8* | *Lava-louças* | 13:00 | *13:00* | *16:00* | *1,0* |
| *9* | *Bomba de água* | 18:00 | *17:00* | *21:00* | *0,5* |
| *10* | *Bomba de recalque* | 18:00 | *17:00* | *23:00* | *0,5* |

Fonte: elaborado pelo autor.

As Figuras 15, 16, 17 e 18, acompanhadas por suas referentes tabelas, mostram os testes considerando diferentes valores para os coeficientes da função custo da Equação 4.5. Os resultados são apresentados para os casos considerando, respectivamente, o custo total, o nível de conforto, o fator de carga e todos os fatores combinados.

Figura 15 - Curva de demanda após agendamento com $\alpha=1$ e $\beta=\gamma=0$ para o cenário 2.



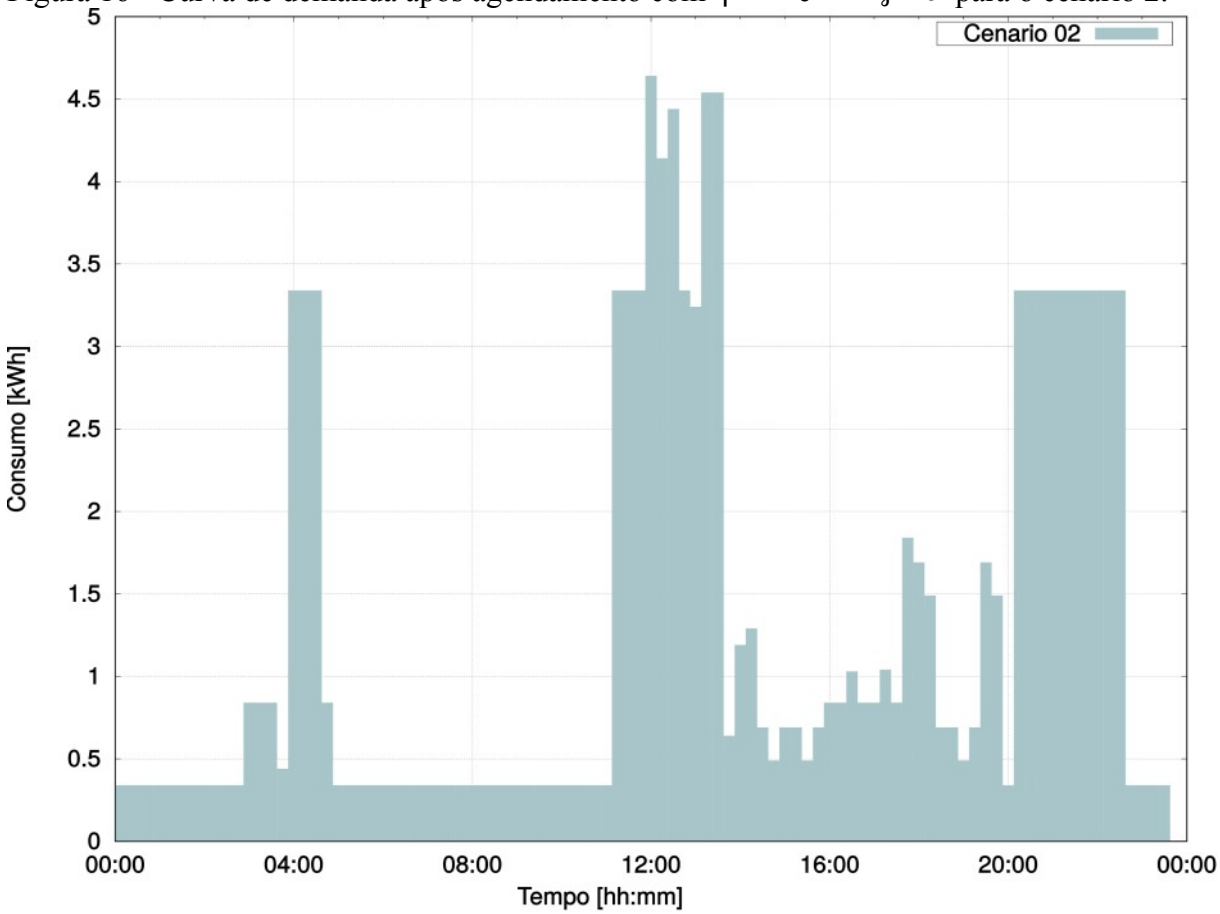

Fonte: elaborado pelo autor.

Tabela 11 - Resultados após agendamento com $\alpha=1$ e $\beta=\gamma=0$ para o cenário 2.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *77,7267* |
| *Nível de Conforto* | *0,877024* |
| *Fator de Carga* | *0,279027* |
| *Tempo de Execução (ms)* | *9746,27* |

Fonte: elaborado pelo autor.

Figura 16 - Curva de demanda após agendamento com $\beta=1$ e $\alpha=\gamma=0$ para o cenário 2.



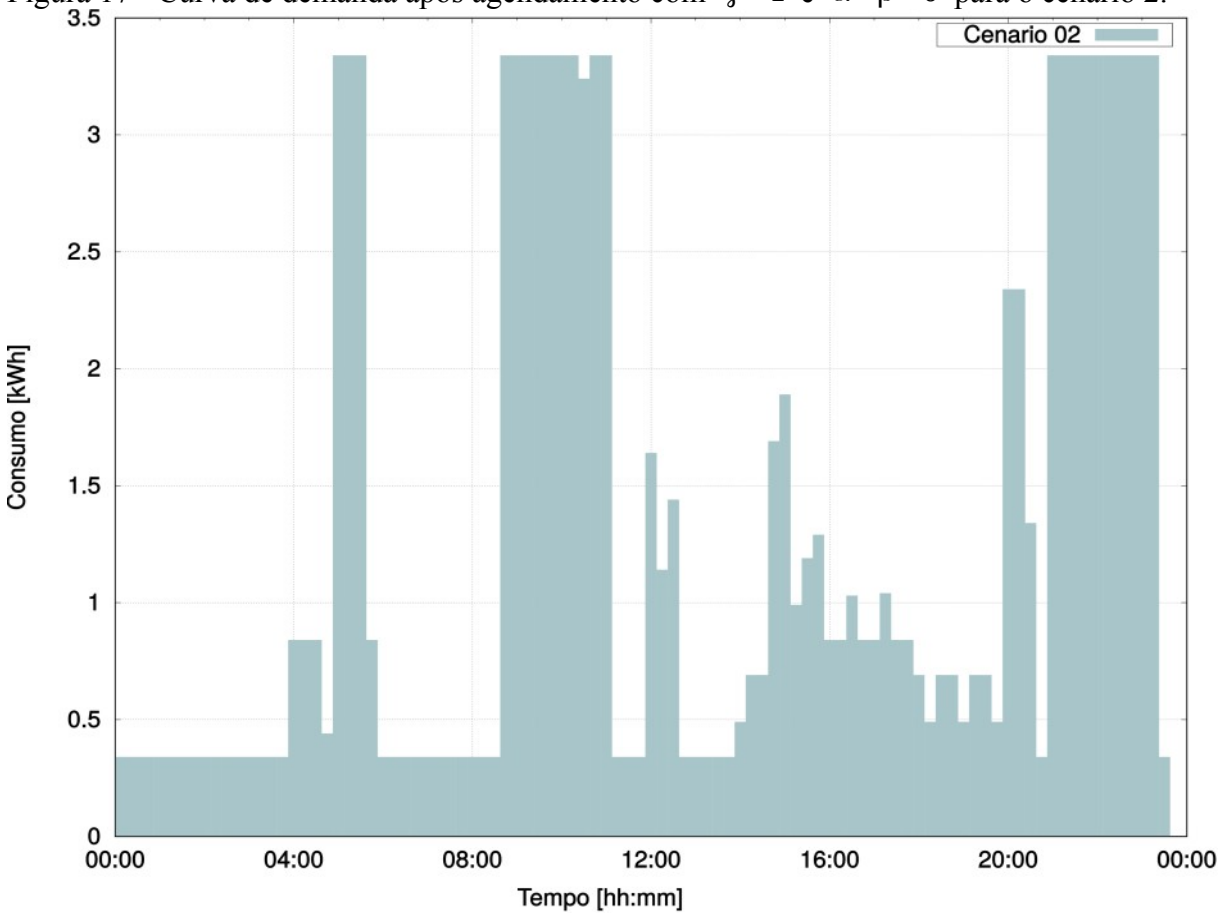

Fonte: elaborado pelo autor.

Tabela 12 - Resultados após agendamento com $\beta=1$ e $\alpha=\gamma=0$ para o cenário 2.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *84,2489* |
| *Nível de Conforto* | *0,918039* |
| *Fator de Carga* | *0,279027* |
| *Tempo de Execução (ms)* | *9732,13* |

Fonte: elaborado pelo autor.

Figura 17 - Curva de demanda após agendamento com $\gamma = 1$ e $\alpha = \beta = 0$ para o cenário 2.



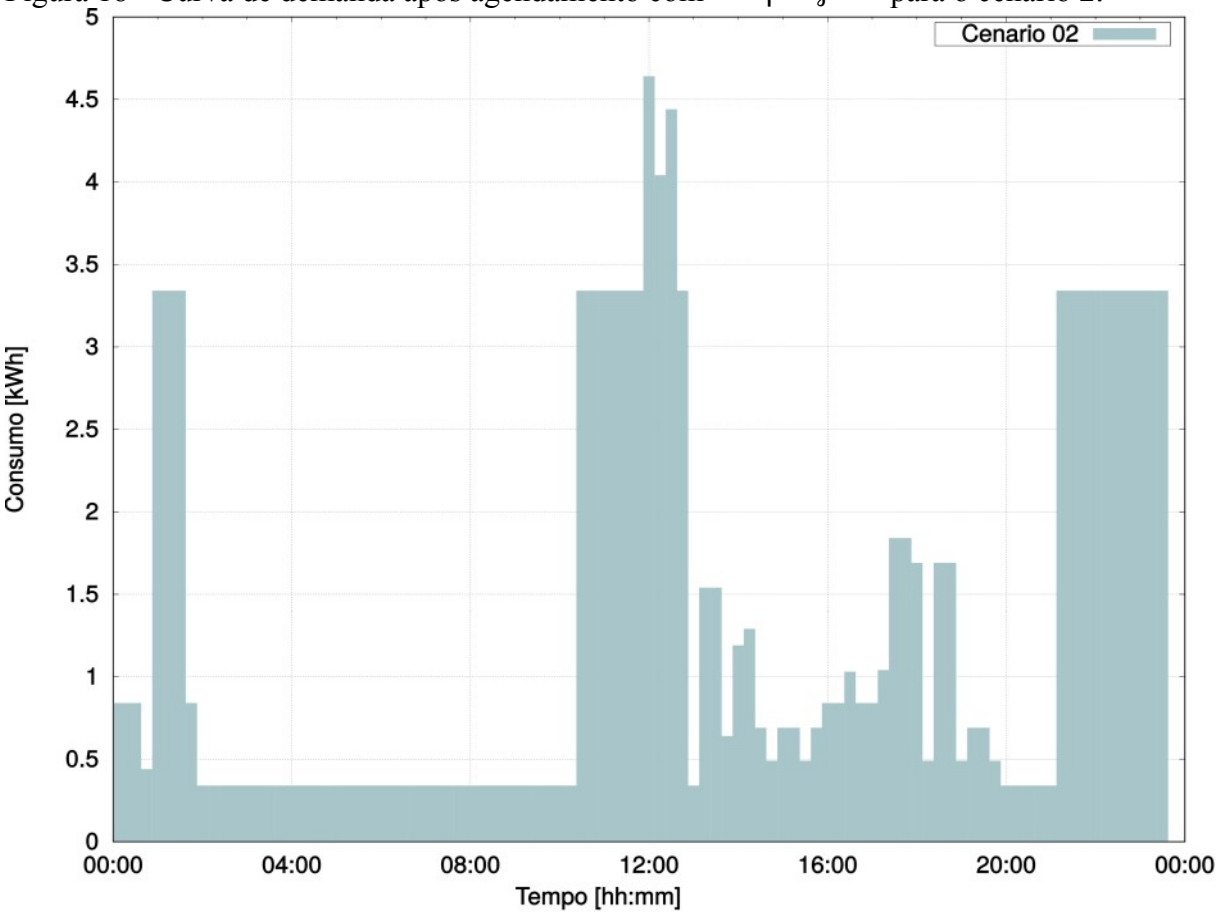

Fonte: elaborado pelo autor.

Tabela 13 - Resultados após agendamento com $\gamma = 1$ e $\alpha = \beta = 0$ para o cenário 2.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *80,8464* |
| *Nível de Conforto* | *0,791243* |
| *Fator de Carga* | *0,387631* |
| *Tempo de Execução (ms)* | *9700,91* |

Fonte: elaborado pelo autor.

Figura 18 - Curva de demanda após agendamento com $\alpha=\beta=\gamma=1$ para o cenário 2.




Fonte: elaborado pelo autor.

Tabela 14 - Resultados após agendamento com $\alpha=\beta=\gamma=1$ para o cenário 2.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *78,8097* |
| *Nível de Conforto* | *0,92585* |
| *Fator de Carga* | *0,279027* |
| *Tempo de Execução (ms)* | *9813,59* |

Fonte: elaborado pelo autor.

A partir da observação das Tabelas 11 e 12 nota-se que, ao realizar a otimização apenas a partir do custo, obteve-se um valor final de R$ 77,7267 enquanto no segundo teste, considerando apenas o conforto, obteve-se um valor de R$ 84,2489. Uma diferença de aproximadamente R$ 7,00, o que representa um acréscimo em torno de 9% em relação ao custo otimizado.

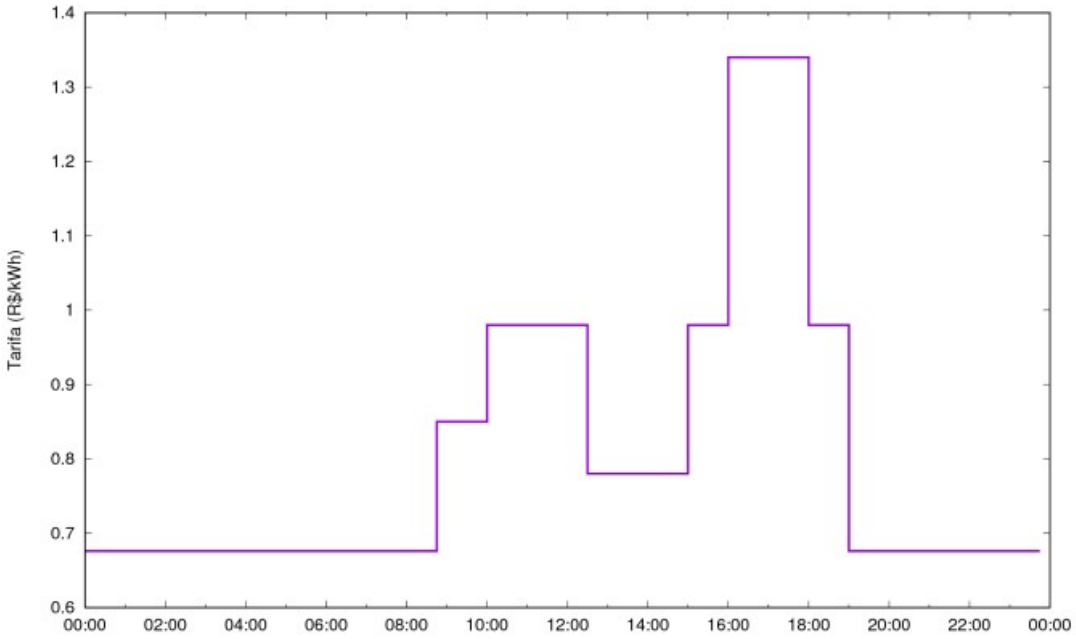
No terceiro teste, considerando a otimização do fator de carga, observa-se um acréscimo de 10,86% com relação ao restante dos testes. Contudo, nesse teste o nível de conforto do usuário é o menor registrado com o valor de 0,791243.

No último teste, onde todos os fatores são igualmente considerados, o custo total de eletricidade é de R$ 78,8097, representando um acréscimo de R$ 1,083 quando comparado com primeiro teste. Considerando o nível de conforto, verificou-se um acréscimo de 0,78%, ou seja, o nível de conforto nesse caso permaneceu praticamente constante quando comparado com o agendamento do segundo teste onde apenas o nível de conforto é considerado. Contudo, não se notou nenhuma elevação com relação ao fator de carga. Isso se deve a inflexibilidade entre a realização da alocação das cargas com relação ao nível de conforto e ao fator de carga. Dessa maneira, caso a elevação do fator de carga seja prioritária, faz-se necessário uma redução do valor da constante $\beta$. Nesse teste, nota-se, também, um pequeno acréscimo no tempo de execução do agendamento (cerca de 90 milissegundos) com relação aos outros testes.

## 5.3 Tarifa RTP

Neste cenário uma tarifa variável do tipo RTP é aplicada sobre as cargas do cenário anterior. A Figura 19 mostra o perfil tarifário aplicado para as simulações.

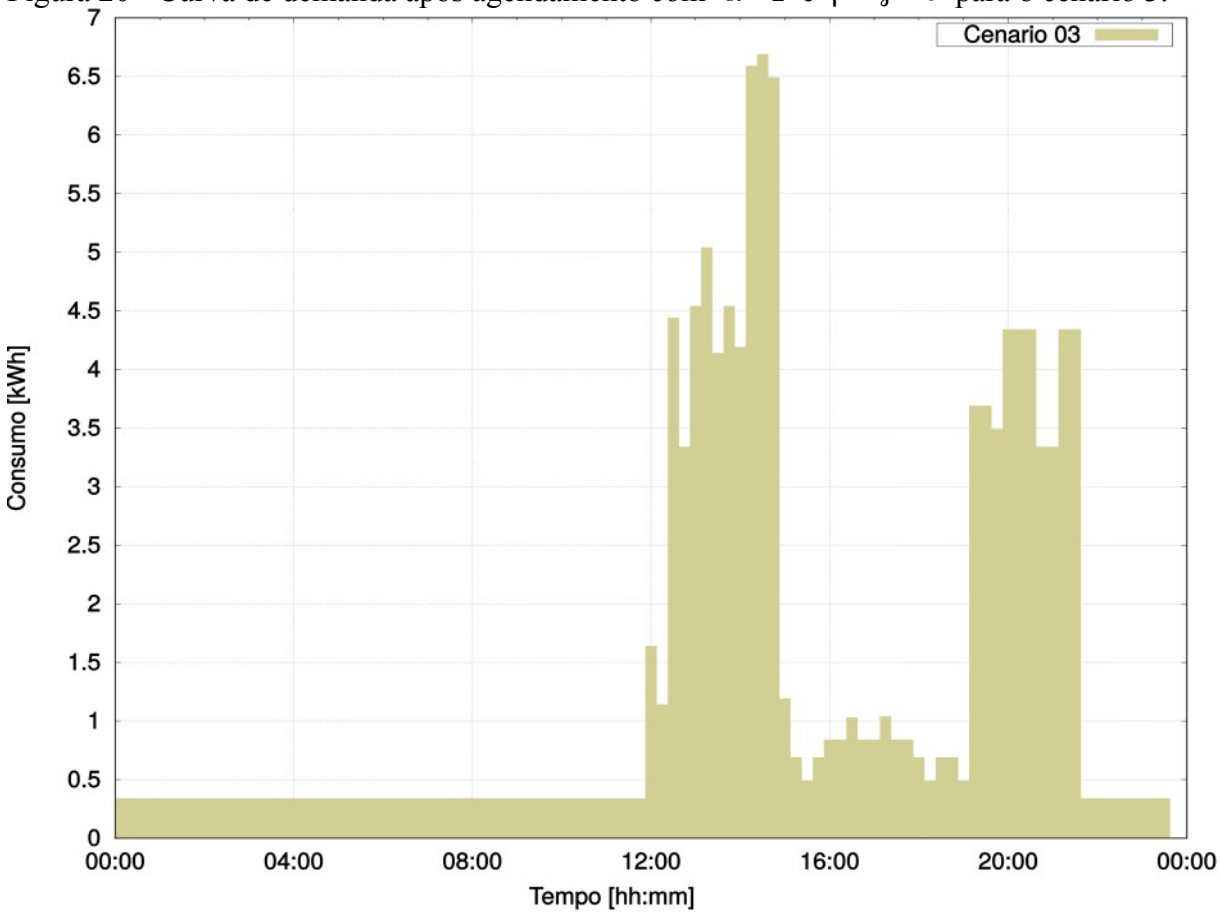
Figura 19 - Tarifa considerada para o cenário 3




Fonte: elaborado pelo autor.

As Figuras 20, 21, 22 e 23 juntamente das Tabelas 15, 16, 17 e 18 mostram os resultados após o agendamento no cenário 03 considerando, respectivamente, a otimização do custo total, conforto, fator de carga e dos três fatores combinados.

Figura 20 - Curva de demanda após agendamento com $\alpha=1$ e $\beta=\gamma=0$ para o cenário 3.



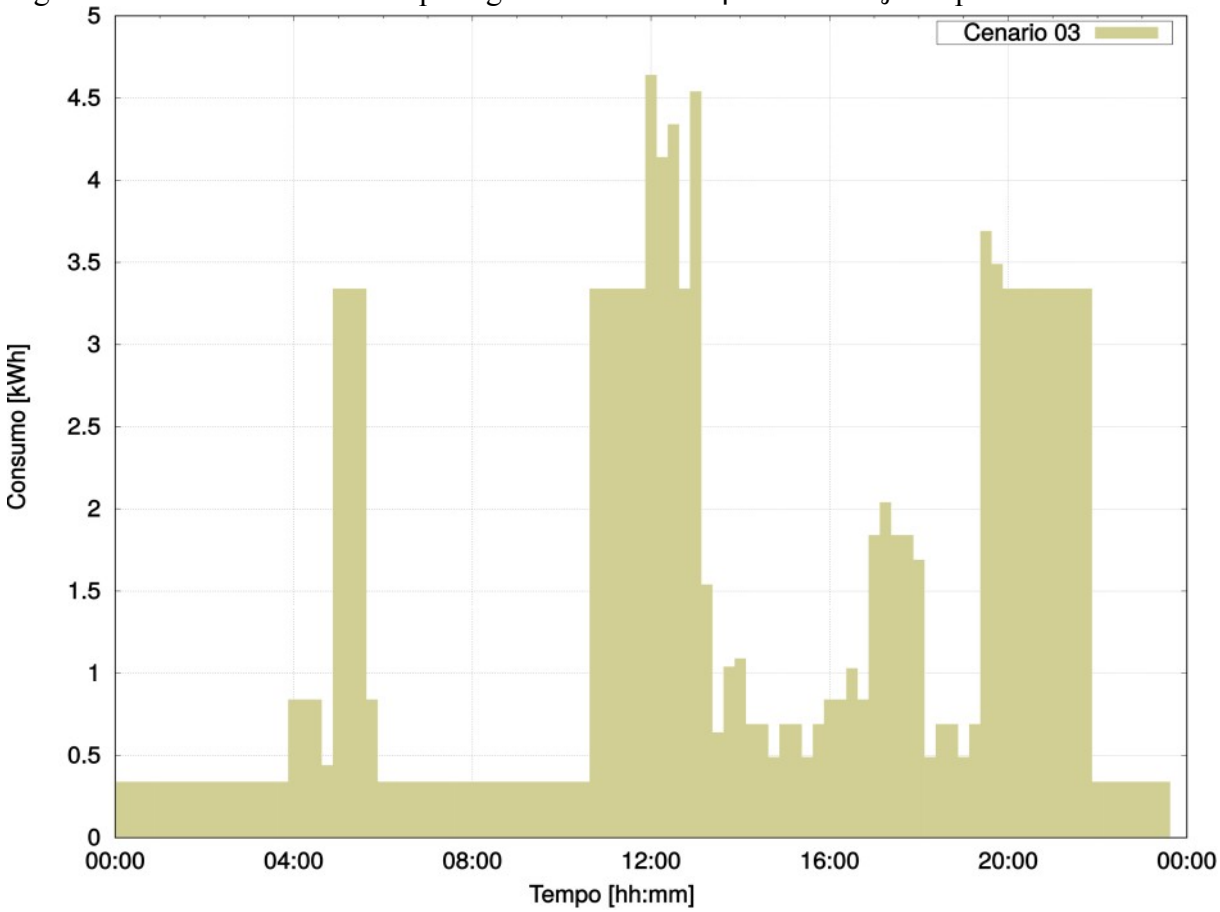

Fonte: elaborado pelo autor.

Tabela 15 - Resultados após agendamento com $\alpha=1$ e $\beta=\gamma=0$ para o cenário 3.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *97,6174* |
| *Nível de Conforto* | *0,81112* |
| *Fator de Carga* | *0,193526* |
| *Tempo de Execução (ms)* | *9760,52* |

Fonte: elaborado pelo autor.

Figura 21 - Curva de demanda após agendamento com $\beta=1$ e $\alpha=\gamma=0$ para o cenário 3.



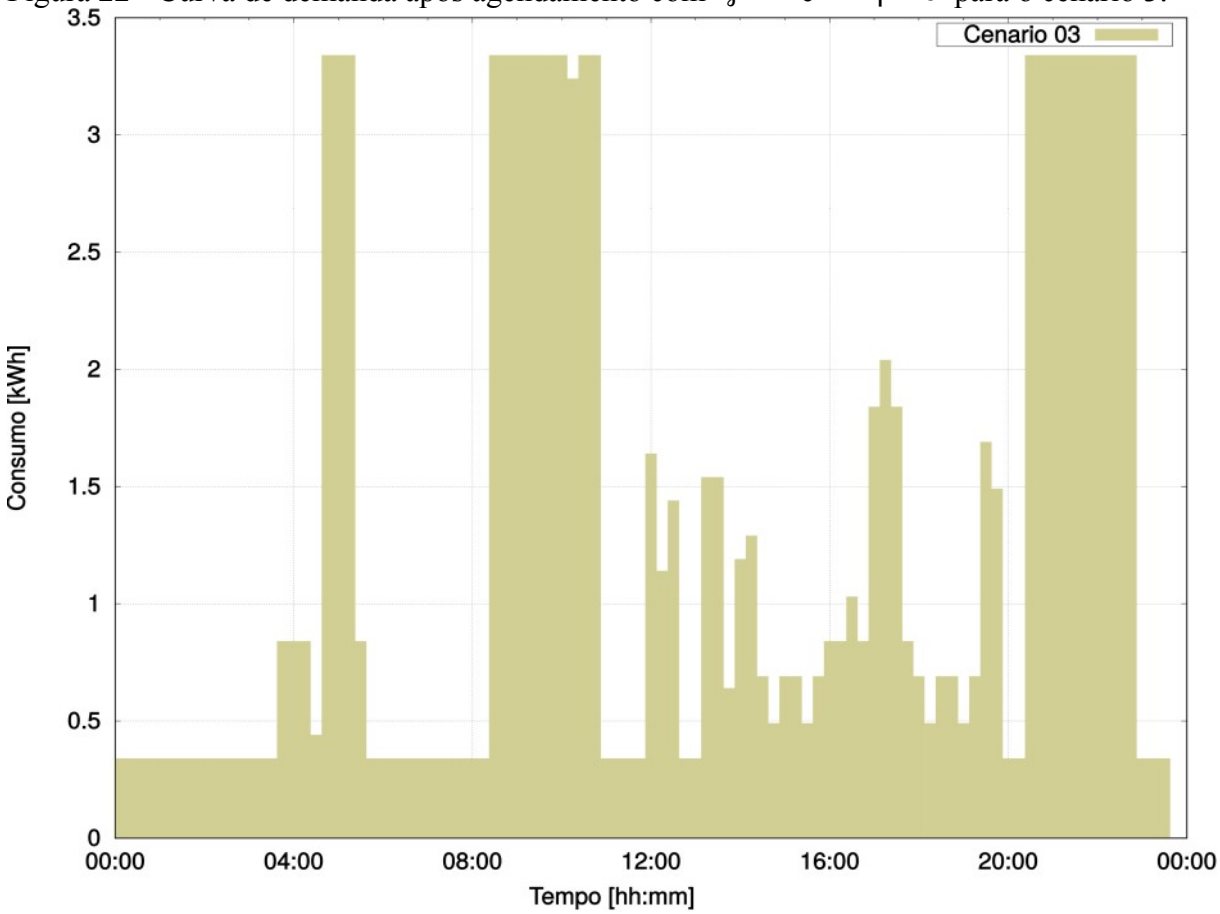

Fonte: elaborado pelo autor.

Tabela 16 - Resultados após agendamento com $\beta=1$ e $\alpha=\gamma=0$ para o cenário 3.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *103,523* |
| *Nível de Conforto* | *0,948337* |
| *Fator de Carga* | *0,279027* |
| *Tempo de Execução (ms)* | *9893,52* |

Fonte: elaborado pelo autor.

Figura 22 - Curva de demanda após agendamento com $\gamma = 1$ e $\alpha = \beta = 0$ para o cenário 3.



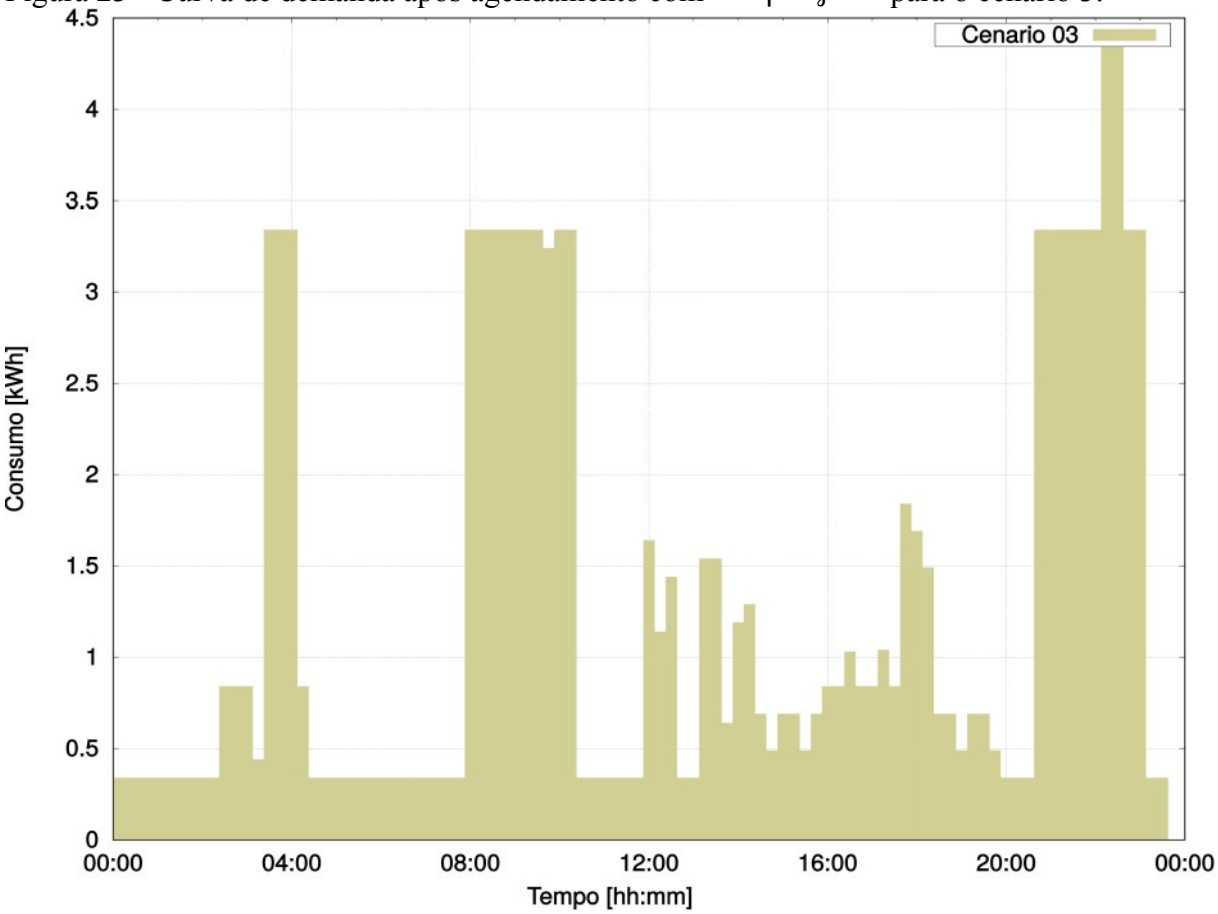

Fonte: elaborado pelo autor.

Tabela 17 - Resultados após agendamento com $\gamma = 1$ e $\alpha = \beta = 0$ para o cenário 3.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *101,473* |
| *Nível de Conforto* | *0,907777* |
| *Fator de Carga* | *0,387631* |
| *Tempo de Execução (ms)* | *9810,27* |

Fonte: elaborado pelo autor.

Figura 23 - Curva de demanda após agendamento com $\alpha=\beta=\gamma=1$ para o cenário 3.




Fonte: elaborado pelo autor.

Tabela 18 - Resultados após agendamento com $\alpha=\beta=\gamma=1$ para o cenário 3.

| ***Parâmetro*** | ***Valor*** |
|---|---|
| *Custo Total (R$)* | *98,942* |
| *Nível de Conforto* | *0,893788* |
| *Fator de Carga* | *0,298315* |
| *Tempo de Execução (ms)* | *9942,18* |

Fonte: elaborado pelo autor.

Observa-se a partir do resultados que a otimização do custo total obteve um valor de R$ 97,6174 no teste 1, enquanto, no teste 2, o valor total foi de R$ 103,523. Isso representa uma redução de aproximadamente 5,5% no custo total de eletricidade.

No terceiro teste obteve-se o valor máximo de 0,387631 para o fator de carga. Conforme já visto no cenário 02, esse valor é o valor máximo possível considerando as cargas do teste.

No quarto teste, onde todos os fatores foram considerados igualmente, obteve-se um acréscimo de apenas R$ 1,3246 quando comparado com o custo total do primeiro teste. O nível de conforto, comparado com o segundo teste, reduziu em 5,45%. Considerando o fator de carga do terceiro teste, obteve-se uma redução de 8,93%. Dessa maneira, é possível mostrar que, mesmo com a rigidez e alta inter-relação entre os coeficientes — como, por exemplo, a elevação do nível de conforto implicar em uma redução do fator de carga ou outros fatores — é possível notar que o otimizador realiza um agendamento mantendo um bom equilíbrio entre os fatores considerados.

# 6 CONCLUSÃO

O processo de otimização do agendamento de cargas controladas através de um sistema de automação apresenta-se com o objetivo de implementar uma estratégia de resposta da demanda capaz de maximizar a eficiência na utilização de equipamentos residenciais. No modelo descrito no trabalho, as funções compreendem três critérios: a minimização dos custos, a maximização das preferências do usuário e a maximização do fator de carga. Pesos relativos a cada uma das três funções são especificados através das preferências do usuário de forma a apontar ao otimizador o objetivo final do agendamento. Um usuário com desejo de reduzir a conta de energia, por exemplo, pode especificar um valor maior na primeira função, um outro usuário buscando otimizar conforto e custo inclinaria-se a otimização de uma combinação entre duas funções e assim por diante.

Em síntese, o problema apresenta-se com o recebimento dos sinais de tarifação advindos da concessionária, das informações acerca dos ciclos de operação e dos tipos de cargas consideradas e das preferências do usuário. O otimizador utilizado baseia-se no algoritmo de metaheurística Evolução Diferencial. A aplicação de um otimizador baseado em heurística apresenta grandes vantagens computacionais — conforme observado nos resultados — permitindo que uma grande parte do espaço de soluções seja explorado de forma eficiente.

Com respeito a arquitetura do projeto utilizado para a implementação do solucionador, são apresentados durante o trabalho um conjunto de módulos escritos em linguagem C++ que compõe o sistema integralmente. O sistema compõe-se de duas interfaces, de um módulo de otimização e de um módulo de saída. As duas interfaces são responsáveis pela geração e validação de arquivos binários que informam ao otimizador os dados com relação aos valores de tarifa ao longo do dia e das preferências do usuário. O módulo de otimização auxilia na implementação das diferentes etapas para realização do algoritmo de Evolução Diferencial. Já o módulo de saída, é responsável pela geração do histórico contendo os resultados dos horário de agendamento para cada carga e de estatísticas globais com relação ao agendamento.

Por fim, foram realizadas diversas simulações com diferentes combinações entre os parâmetros da função custo global em três cenários distintos de tarifação: tarifa fixa, tarifa branca e tarifa RTP. Os três cenários foram escolhidos por sua maior aplicação em cenários reais, em especial, as duas primeiras modalidades que já fazem parte da realidade brasileira. No cenário com tarifação fixa mostrou-se que o agendamento é útil para uma elevação do fator carga e de nível de conforto, uma vez que, por apresentar um valor constante, não faz

sentido otimizar o custo. Todavia, nos cenários de tarifas variáveis, apresentaram-se agendamentos mais complexos devido ao maior número de cargas e de características consideradas. De maneira geral os resultados mostraram que o sistema proposto de fato proporciona ganhos tanto para os consumidores como para as concessionárias e integra-se como uma estratégia DSM efetiva.

Como proposta para futuras investigações, espera-se que o otimizador possa ser incorporado em sistemas computacionais menores, como computadores embarcados SBC e microcontroladores. A utilização de soluções através de projetos diretos em *hardware* — por exemplo, com a utilização de soluções em FPGA — também pode representar grandes vantagens para a realização do algoritmo e o tratamento dos indivíduos de forma paralela.

Indica-se, também, a investigação de valores médios dos coeficientes $\alpha$, $\beta$ e $\gamma$ para diferentes perfis de usuários com a possível utilização de informações com base em dados históricos e de previsão de carga.

Ademais, ainda que a apresentação da otimização do fator de carga seja descrita nesse trabalho, seus impactos econômicos ainda não significativos para esse cenário. Dessa forma, a investigação em cenários industriais, onde existe um limite de demanda, apresenta-se como uma nova possibilidade, podendo fornecer um agendamento ótimo minimizando ao máximo a demanda contratada.

# REFERÊNCIAS

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL). **ANEEL aprova tarifa branca, nova opção para os consumidores a partir de 2018.** 2018. Disponível em: <http://www.aneel.gov.br/sala-de-imprensa-exibicao/-/asset_publisher/XGPXSqdMFHrE/content/aneel-aprova-tarifa-branca-nova-opcao-para-os-consumidores-a-partir-de-2018/656877>. Acesso em: 1 abr. 2018.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA (ANEEL). **RN N° 414:** RESOLUÇÃO NORMATIVA Nº 414,. Brasília, 2010. 156 p.

OGUNJUYIGBE, A. S. O.; AYODELE, T. R.; AKINOLA, O. A.. User satisfaction-induced demand side load management in residential buildings with user budget constraint. **Applied Energy**, [s.l.], v. 187, p.352-366, fev. 2017. Elsevier BV. http://dx.doi.org/10.1016/j.apenergy.2016.11.071.

CAPPERS, Peter; GOLDMAN, Charles; KATHAN, David. **Demand response in U.S. electricity markets: empirical evidence:** Empirical Evidence. Berkeley: Ernest Orlando Lawrence Berkeley National Laboratory, 2009. 29 p.

CASTRO, Roberto. **Resposta da Demanda.** São Paulo: CCEE, 2017. 16 slides, color. Disponível em: <http://az545403.vo.msecnd.net/uploads/2017/06/roberto-castro-.pdf>. Acesso em: 1 Abr. 2018.

U.S Department of Energy (DOE). **Benefits of Demand Response in Electricity Markerts and Recommendations for Achieving Them:** Pursuant To Section 1252 of the Energy Policy Act of 2005, 2006.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P.. Optimization by Simulated Annealing. **Science**, [s.l.], v. 220, n. 4598, p.671-680, 13 maio 1983. American Association for the Advancement of Science (AAAS). http://dx.doi.org/10.1126/science.220.4598.671.

LEVY, Beatriz Nogueira. **Influência de Programas de Resposta da Demanda na Rede de Distribuição.** 2013. 90 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013

MATSUMOTO, Makoto; NISHIMURA, Takuji. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. **Acm Transactions On Modeling And Computer Simulation**, [s.l.], v. 8, n. 1, p.3-30, 1 jan. 1998. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/272991.272995.

PRICE, Kenneth; STORN, Rainer M.; LAMPINEN, Jouni A.. **Differential Evolution: A Practical Approach to Global Optimization.** Berlin: Springer, 2005.

STORN, Rainer; PRICE, Kenneth. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. **Journal Of Global Optimization**, [s.l.], v. 11, n. 4, p.341-359, 1997. Springer Nature. http://dx.doi.org/10.1023/a:1008202821328.

QAYYUM, F. A. et al. Appliance Scheduling Optimization in Smart Home Networks. **Ieee Access**, [s.l.], v. 3, p.2176-2190, 2015. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/access.2015.2496117.

SIEBERT, L . C . et al. **Gerenciamento Pelo Lado da Demanda em Redes Inteligentes Utilizando Algoritmos Genéticos**. Simpósio Brasileiro de Sistemas Elétricos, 2012, Goiás. SBSE 2012.

SOARES, A.; GOMES, A.; ANTUNES, C. H.. Domestic load characterization for demand-responsive energy management systems. **2012 Ieee International Symposium On Sustainable Systems And Technology (issst)**, maio 2012. IEEE. http://dx.doi.org/10.1109/issst.2012.6227976.

TSUI, K. M.; CHAN, S. C.. Demand Response Optimization for Smart Home Scheduling Under Real-Time Pricing. **IEEE Transactions On Smart Grid**, v. 3, n. 4, p.1812-1821, dez. 2012. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tsg.2012.2218835.

UNIVERSIDADE FEDERAL DO CEARÁ. Biblioteca **Universitária**. **Guia de normalização de trabalhos acadêmicos da Universidade Federal do Ceará**. Fortaleza, 2013.

VERAS, Jaclason M. et al. A demand response optimization model for home appliances load scheduling. **2017 IEEE International Conference On Systems, Man, And Cybernetics (smc)**, out. 2017. IEEE. http://dx.doi.org/10.1109/smc.2017.8123070. .

# APÊNDICE A – CLASSE LOADSCHEDULING

## load_scheduling.hpp

```
/*! \file load_scheduling.h
   \brief Define uma classe para o agendamento de cargas.

   More details.
*/

#pragma once

#include <iostream>
#include <vector>
#include <random>

#define LS_START_TIME 0
#define LS_END_TIME 1

typedef struct
{
 std::string nome;
 double coef_conforto;
 unsigned int hora_inicio;
 unsigned int hora_preferencia;
 unsigned int hora_inicio_janela;
 unsigned int hora_termino_janela;
 std::vector<double> vetor_energia;
} equipamento_t;

class LoadScheduling
{
private:
 std::vector<equipamento_t> m_equipamentos;    // Vetor equipamentos
 const unsigned int MINUTES_PER_DAY = 24 * 60; // Quantidade de minutos por dia
 const unsigned int N_SLOTS;                   // Quantidade de slots de tempo m por dia
 const unsigned int N_APPLIANCES;              // Quantidade de equipamentos
 const unsigned int MIN_PER_SLOT;              // Minutos por slot
 double m_alpha = 1;                           // Peso do preco total
 double m_beta = 1;                            // Peso do conforto total
 double m_gamma = 1;                           // Peso do fator de carga
 std::vector<double> m_energyPrice;            // Vetor com o valor de tarifa para cada slot de tempo [k]
 std::mt19937 m_mersenne_engine;

public:
 /*! \fn LoadScheduling(const unsigned int min, equipamento_t *equipamentos, const unsigned int n);
```

```
    \brief Inicializa a classe \a LoadScheduling.

    \param min Resolucao, em minutos, de cada slot de tempo ao longo de 24h. Por exemplo, 15 minutos retornam 96 divisoes do dia.

    \param equipamentos Referencia do vetor de estruturas contendo cada carga e suas informacoes.

    \param n Quantidade de cargas do vetor equipamentos.

  */

 LoadScheduling(const unsigned int min, equipamento_t *equipamentos, const unsigned int n);

 /*! \fn void setRandomStartTimesForAppliances()

    \brief Gera valores de tempos aleatorios dentro da regiao configurada de inicio e termino do agendamento para cada carga

    \param nenhum
  */

 void setRandomStartTimesForAppliances();

 /*! \fn void setNewStartTimesForAppliances(std::vector<unsigned int> newStartTime)

    \brief Seta novo valor para o horario de inicio das cargas.

    \param newStartTime Vetor contendo os novos horarios para cada carga em ordem.
  */

 void setNewStartTimesForAppliances(std::vector<unsigned int> newStartTime);

 /*! \fn void setEnergyPriceVector(std::vector<double> energyPrice)

    \brief Inicializa o vetor de tarifas para a classe.

    \param &energyPrice vetor com os valores da tarifa para cada timeSlot ao longo do dia (deve ser de tamanho N_SLOTS).
  */

 void setEnergyPriceVector(std::vector<double> &energyPrice);

 /*! \fn bool setEnergyPrice(std::string filename)

    \brief Le arquivo binario gerado por 'energyPrice' com os dados de tarifa ToU e passa para setEnergyPriceVector().

    \param filename string com o path nome do arquivo binario.
  */
```

```
bool setEnergyPrice(std::string filename);

/*! \fn unsigned int getApplianceStartSlot()

    \brief Retorna uma referencia para o slot de tempo de inicio agendado para a carga.

    \param index Indice da carga.
 */

unsigned int getApplianceStartSlot(unsigned int index);

/*! \fn void setApplianceStartSlot(unsigned int slot, unsigned int index)

    \brief Define um novo tempo de inicio para a carga.

    \param slot Valor do slot de inicio

    \param index Indice da carga.
 */

void setApplianceStartSlot(unsigned int slot, unsigned int index);

/*! \fn double getTotalElectricityCost()

    \brief Retorna o valor total do custo por dia

    \param nenhum
 */

double getTotalElectricityCost();

/*! \fn std::vector<unsigned int> getWindowSlot(unsigned int opt)

    \brief Retorna configuracoes de preferencia do usuario

    \param opt LS_START_TIME ou LS_END_TIME.

    \param index Indice da carga.
 */

unsigned int getWindowSlot(unsigned int opt, unsigned int index);

/*! \fn unsigned int getDurationOfAppliance(unsigned int index)

    \brief Retorna a duracao de uma carga em quantidade de slots

    \param index Indice da carga
 */
```

```
unsigned int getDurationOfAppliance(unsigned int index);

/*! \fn double getElectricityCostNormalized()
    \brief Retorna o custo de energia normalizado

    \param nenhum
 */

double getElectricityCostNormalized();

/*! \fn double getComfortCost()
    \brief Retorna o nivel de conforto associado a carga de indice 'index'

    \param nenhum
 */

double getComfortCost();

/*! \fn double getLoadFactor()

    \brief Retorna o fator de carga num periodo de 24h

    \param nenhum
 */
double getLoadFactor();

/*! \fn void changeLinearWeights(double price, double comfort);

    \brief Muda os valores dos pesos da funcao custo, tipicamente entre 0 e 1.

    \param price Valor do peso do preco total

    \param comfort Valor do peso do conforto

    \param loadFactor Valor do peso do fator de carga
 */
void changeLinearWeights(double price, double comfort, double loadFactor);

/*! \fn double costFunction(LoadScheduling Ls)

    \brief Calcula o valor da funcao custo

    \param nenhum
 */
double costFunction();

/*! \fn bool saveAsCSV(std::string filename)
```

```
    \brief Salva o agendamento em um arquivo CSV na pasta output

    \param filename nome do arquivo CSV
  */
 bool saveAsCSV(std::string filename);

 /*! \fn std::string LoadScheduling::convertToTimeString(double t)

    \brief Converte o parametro t em uma string no formato HH:MM

    \param t Entrada da hora em formato decimal
  */
 static std::string convertToTimeString(double t);

 /*! \fn unsigned int uniformIntDist(unsigned int lowerBound, unsigned int upperBound)

    \brief Retorna um numero aleatorio (uniformemente distribuido) entre lowerBound e upperBound

    \param nenhum
  */
 unsigned int uniformIntDist(unsigned int lowerBound, unsigned int upperBound);

 /*! \fn ~LoadScheduling()

    \brief Destructor
  */
 ~LoadScheduling()
 {
 }
};
```

**load_scheduling.cpp**

```
#include <iostream>
#include <fstream>
#include <cassert>
#include <cstdint>
#include <vector>
#include <random>
#include <algorithm>
#include <string>
#include <cmath>

#include "load_scheduling.hpp"

LoadScheduling::LoadScheduling(const unsigned int min, equipamento_t *equipamentos, const unsigned int
n) :
MIN_PER_SLOT(min),
N_SLOTS(MINUTES_PER_DAY / min),
N_APPLIANCES(n),
m_mersenne_engine(std::random_device{}())
{
   // Copia os dados originais para um novo individuo
   for(unsigned int i = 0; i < n; ++i)
   {
      equipamento_t equipamento;
      equipamento.nome = equipamentos[i].nome;
      equipamento.coef_conforto = equipamentos[i].coef_conforto;
      equipamento.hora_inicio = equipamentos[i].hora_inicio;
      equipamento.hora_preferencia = equipamentos[i].hora_preferencia;
      equipamento.hora_inicio_janela = equipamentos[i].hora_inicio_janela;
      equipamento.hora_termino_janela = equipamentos[i].hora_termino_janela;
      for(auto val: equipamentos[i].vetor_energia)
      {
         (equipamento.vetor_energia).push_back(val);
      }
      m_equipamentos.push_back(equipamento);
   }
   // Tempo de inicio de cargas setados aleatoriamente no intervalo delimitado
   LoadScheduling::setRandomStartTimesForAppliances();
}


void LoadScheduling::setRandomStartTimesForAppliances()
{
   for(unsigned int i = 0; i < N_APPLIANCES; ++i)
   {
      unsigned int inicio_janela = m_equipamentos[i].hora_inicio_janela / MIN_PER_SLOT;
```

```
        unsigned int termino_janela = m_equipamentos[i].hora_termino_janela / MIN_PER_SLOT;
        termino_janela = termino_janela - LoadScheduling::getDurationOfAppliance(i);
        m_equipamentos[i].hora_inicio = uniformIntDist(inicio_janela, termino_janela);
    }
}


void LoadScheduling::setNewStartTimesForAppliances(std::vector<unsigned int> newStartTime)
{
    for(unsigned int i = 0; i < N_APPLIANCES; ++i)
        m_equipamentos[i].hora_inicio = newStartTime[i];
}


void LoadScheduling::setEnergyPriceVector(std::vector<double> &energyPrice)
{
    assert((energyPrice.size() == N_SLOTS) && "O conjunto de valores de tarifa nao representam um dia
inteiro!");

    m_energyPrice = energyPrice;
}


bool LoadScheduling::setEnergyPrice(std::string filename)
{
    std::ifstream price_file;
    price_file.open(filename);

    if(!price_file.is_open())
    {
        std::cout << "Erro ao ler o arquivo " << filename << std::endl;
        return false;
    }

    std::vector<double> energyPrice;
    size_t v_len;
    price_file.read(reinterpret_cast<char *>(&v_len), sizeof(v_len));
    energyPrice.resize(v_len);
    price_file.read(reinterpret_cast<char *>(&energyPrice[0]), v_len * sizeof(energyPrice[0]));
    price_file.close();

    LoadScheduling::setEnergyPriceVector(energyPrice);
    return true;
}


unsigned int LoadScheduling::getApplianceStartSlot(unsigned int index)
{
    // std::vector<unsigned int> start_time;
```

```
    // for(unsigned int i = 0; i < N_APPLIANCES; ++i)
    //     start_time.push_back(m_equipamentos[i].hora_inicio);
    // return start_time;
    return m_equipamentos[index].hora_inicio;
}


void LoadScheduling::setApplianceStartSlot(unsigned int slot, unsigned int index)
{
    m_equipamentos[index].hora_inicio = slot;
}

double LoadScheduling::getTotalElectricityCost()
{
    double totalCost = 0;
    unsigned int j = 0;

    for(unsigned int k = 0; k < N_SLOTS; ++k) // indice de tempo k
    {
        for(unsigned int i = 0; i < N_APPLIANCES; ++i)  // indice de equipamento i
        {
            j = k - m_equipamentos[i].hora_inicio;  // indice do perfil de carga

            if (k >= m_equipamentos[i].hora_inicio && j < LoadScheduling::getDurationOfAppliance(i))
            {
                totalCost += m_energyPrice[k] * m_equipamentos[i].vetor_energia[j];
            }
        }
    }

    return totalCost;
}


unsigned int LoadScheduling::getWindowSlot(unsigned int opt, unsigned int index)
{
    std::vector<unsigned int> ret;
    if(opt == LS_START_TIME)
    {
        return (m_equipamentos[index].hora_inicio_janela);
    }
    else if(opt == LS_END_TIME)
    {
        return (m_equipamentos[index].hora_termino_janela);
    }
    else
    {
        std::cout << "[ATENCAO] Especifique alguma opcao para getPrefferenceSettings!" << std::endl;
    }
```

```
    return 0;
}


unsigned int LoadScheduling::getDurationOfAppliance(unsigned int index)
{
    return (m_equipamentos[index].vetor_energia).size();
}


double LoadScheduling::getElectricityCostNormalized()
{
    double maxTotalCost = 0;
    double totalCost = 0;
    double maxEnergyPrice = *std::max_element(m_energyPrice.begin(), m_energyPrice.end());
    unsigned int j = 0;

    for (unsigned int k = 0; k < N_SLOTS; ++k) // indice de tempo k
    {
        for (unsigned int i = 0; i < N_APPLIANCES; ++i) // indice de equipamento i
        {
            j = k - m_equipamentos[i].hora_inicio; // indice do perfil de carga

            if (k >= m_equipamentos[i].hora_inicio && j < LoadScheduling::getDurationOfAppliance(i))
            {
                maxTotalCost += maxEnergyPrice * m_equipamentos[i].vetor_energia[j];
                totalCost += m_energyPrice[k] * m_equipamentos[i].vetor_energia[j];
            }
        }
    }

    return (totalCost / maxTotalCost);
}


double LoadScheduling::getComfortCost()
{
    int deltaMax, deltaStartWindow, deltaEndWindow;
    double deltaT = 0;
    for (unsigned int i = 0; i < N_APPLIANCES; i++)
    {
        deltaStartWindow = m_equipamentos[i].hora_inicio_janela - m_equipamentos[i].hora_inicio;
        deltaEndWindow = m_equipamentos[i].hora_termino_janela - m_equipamentos[i].hora_inicio;
        deltaStartWindow = std::abs(deltaStartWindow);
        deltaEndWindow = std::abs(deltaEndWindow);
        deltaMax = std::max(deltaStartWindow, deltaEndWindow);
        if (deltaMax == 0) continue;
        int distance = m_equipamentos[i].hora_inicio - m_equipamentos[i].hora_preferencia;
        deltaT += m_equipamentos[i].coef_conforto * fabs(static_cast<int>(distance) /
```

```
static_cast<double>(deltaMax));
    }
    return deltaT / N_APPLIANCES;
}


double LoadScheduling::getLoadFactor()
{
    std::vector<double> sumPowerPerSlot;
    double sum;
    unsigned int j;

    for (unsigned int k = 0; k < N_SLOTS; ++k) // indice de tempo k
    {
        sum = 0;
        for (unsigned int i = 0; i < N_APPLIANCES; ++i) // indice de equipamento i
        {
            j = k - m_equipamentos[i].hora_inicio; // indice do perfil de carga

            if (k >= m_equipamentos[i].hora_inicio && j < LoadScheduling::getDurationOfAppliance(i))
            {
                sum += m_equipamentos[i].vetor_energia[j];
            }
        }
        sumPowerPerSlot.push_back(sum);
    }

    double avgLoad = 0;
    double maxLoad = sumPowerPerSlot[0];

    for (unsigned int i = 0; i < sumPowerPerSlot.size(); ++i)
    {
        avgLoad += sumPowerPerSlot[i];
        maxLoad = std::max(sumPowerPerSlot[i], maxLoad);
    }

    // std::cout << "max: " << maxLoad << std::endl;
    // std::cout << "sum: " << avgLoad << std::endl;
    avgLoad /= N_SLOTS;
    // std::cout << "avg: " << avgLoad << std::endl;

    return (avgLoad / maxLoad);
}


void LoadScheduling::changeLinearWeights(double price, double comfort, double loadFactor)
{
    m_alpha = price;
    m_beta = comfort;
```

```
    m_gamma = loadFactor;
}


double LoadScheduling::costFunction()
{
    return (m_alpha * LoadScheduling::getElectricityCostNormalized() + m_beta *
LoadScheduling::getComfortCost() + m_gamma * (1 - LoadScheduling::getLoadFactor()));
}


bool LoadScheduling::saveAsCSV(std::string filename)
{
    // sumario_*.csv
    std::ofstream csvFile;
    csvFile.open("./sumario_" + filename + ".csv");
    if(csvFile.is_open())
    {
        // Cabecalho
        csvFile << "id,nome,inicio,duracao,desejado\n";
        for(unsigned int i = 0; i < N_APPLIANCES; ++i)
        {
            csvFile << i + 1 << ',';                                    // indice
            csvFile << m_equipamentos[i].nome << ',';                   // nome
            csvFile << MIN_PER_SLOT * m_equipamentos[i].hora_inicio << ','; // inicio
            csvFile << MIN_PER_SLOT * LoadScheduling::getDurationOfAppliance(i) << ','; // duracao
            csvFile << MIN_PER_SLOT * m_equipamentos[i].hora_preferencia << ','; // preferencia
            csvFile << '\n';
        }
        csvFile << "Custo Total:," << LoadScheduling::getTotalElectricityCost() << '\n'; // Valor De Custo Total
        csvFile << "Conforto:," << 1 - LoadScheduling::getComfortCost() << '\n';          // Nivel De Conforto
        csvFile << "Fator de Carga:," << LoadScheduling::getLoadFactor() << '\n';         // Fator de Carga
    }
    else
    {
        return false;
    }

    csvFile.close();

    // plt_*.dat
    std::ofstream pltFile;
    pltFile.open("./plt_"+ filename + ".dat");
    double sum;
    unsigned int j;
    if(pltFile.is_open())
    {
        for (unsigned int k = 0; k < N_SLOTS; ++k) // indice de tempo k
        {
```

```
            pltFile << LoadScheduling::convertToTimeString(k * MIN_PER_SLOT / 60.0) << ' ';
            sum = 0;
            for (unsigned int i = 0; i < N_APPLIANCES; ++i) // indice de equipamento i
            {
               j = k - m_equipamentos[i].hora_inicio; // indice do perfil de carga

               if (k >= m_equipamentos[i].hora_inicio && j < LoadScheduling::getDurationOfAppliance(i))
               {
                  sum += m_equipamentos[i].vetor_energia[j];
               }
            }
            pltFile << sum << '\n';
         }
      }

      pltFile.close();

      return true;
}


std::string LoadScheduling::convertToTimeString(double t)
{
      double h, m;
      m = std::modf(t, &h);
      m = m * 60.0;
      std::string s = std::to_string(static_cast<unsigned int>(h)) + ':' + std::to_string(lround(m));
      return s;
}


unsigned int LoadScheduling::uniformIntDist(unsigned int lowerBound, unsigned int upperBound)
{
      std::uniform_int_distribution<unsigned int> dist(lowerBound, upperBound);
      return dist(m_mersenne_engine);
}
```

# APÊNDICE B – CLASSE DIFFEVOLUTION

## diff_evolution.hpp

```
#pragma once

#include "load_scheduling.hpp"

#include <vector>
#include <random>

class DiffEvolution
{
 private:
   const unsigned int POPULATION_SIZE;          // Tamanho da populacao
   const unsigned int DIM_SIZE;                 // Quantidade de dimensoes do problema
   const unsigned int CR;                       // Taxa de crossover [valores de 0 - 100]
   const double F;                              // Taxa de escalonamento do vetor diferenca DE
   std::vector<LoadScheduling> &m_population;   // REFERENCIA para a populacao original do problema
   std::vector<LoadScheduling> m_trialPopulation; // Populacao candidata da nova geracao
   std::mt19937 m_mersenne_engine;              // mersenne twister engine

 public:
   /*! \fn DiffEvolution()

       \brief Inicializa a classe DiffEvolution.

       \param pop Vetor com as variaveis do problema

       \param dim Numero de dimensoes das variaveis a serem otimizadas (no caso, qnt de equipamentos)

       \param cr Crossover-rate (valores entre 0 - 100 [%])

       \param f Valor para escalonamento do passo (geralmente entre 0 e 1)
   */

   DiffEvolution(std::vector<LoadScheduling> &pop, unsigned int dim, unsigned int cr, double f);

   /*! \fn void generateTrialPopulation()

       \brief Gera uma populacao candidata para substituir a populacao original. Cria o vetor m_trialPopulation e
povoa com os candidatos modificados da geracao atual.

       \param nenhum
   */

   void generateTrialPopulation();
```

```
    /*! \fn void selection()

       \brief Seleciona individuos mais aptos

       \param nenhum

    */

    void selection();

    /*! \fn unsigned int uniformIntDist(unsigned int lowerBound, unsigned int upperBound)

       \brief Retorna um numero aleatorio (uniformemente distribuido) entre lowerBound e upperBound

       \param nenhum
    */

    unsigned int uniformIntDist(unsigned int lowerBound, unsigned int upperBound);

    /*! \fn ~DiffEvolution()

       \brief Destructor

    */
    ~DiffEvolution()
    {
    }
};
```

## diff_evolution.cpp

```
#include "diff_evolution.hpp"
#include "load_scheduling.hpp"

#include <iostream>
#include <random>
#include <vector>

DiffEvolution::DiffEvolution(std::vector<LoadScheduling> &pop, unsigned int dim, unsigned int cr, double f)
:
POPULATION_SIZE(pop.size()),
DIM_SIZE(dim),
CR(cr),
F(f),
m_population(pop)
{
   // inicializa trialPopulation
   for(unsigned int i = 0; i < pop.size(); ++i)
      m_trialPopulation.push_back(m_population[i]);
}

void DiffEvolution::generateTrialPopulation()
{
   unsigned int r0, r1, r2;          // r0: prototipo base; r1 e r2: prototipo p/ calculo da diferenca
   unsigned int ti, t0, t1, t2;      // vetores com o horario de inicio de cada carga para r0, r1 e r2

   for(unsigned int i = 0; i < POPULATION_SIZE; ++i) // i: individuo
   {

      do
      {
         r0 = DiffEvolution::uniformIntDist(0, POPULATION_SIZE - 1);
      } while (r0 == i);

      do
      {
         r1 = DiffEvolution::uniformIntDist(0, POPULATION_SIZE - 1);
      } while (r1 == r0 || r1 == i);

      do
      {
         r2 = DiffEvolution::uniformIntDist(0, POPULATION_SIZE - 1);
      } while (r2 == r1 || r2 == r0 || r2 == i);


      for (unsigned int j = 0; j < DIM_SIZE; ++j) // j: equipamento
```

```
        {
            ti  = m_population[i].getApplianceStartSlot(j);
            t0 = m_population[r0].getApplianceStartSlot(j);
            t1 = m_population[r1].getApplianceStartSlot(j);
            t2 = m_population[r2].getApplianceStartSlot(j);
            unsigned int val;
            if(DiffEvolution::uniformIntDist(0, 100) <= CR)
            {

                val = t0 + F * (t1 - t2);

            }
            else
            {
                val = ti;
            }

            // Checa se o valor esta fora dos limites de satisfacao (janela) do usuario
            if(val < m_trialPopulation[i].getWindowSlot(LS_START_TIME, j))
                val = m_trialPopulation[i].getWindowSlot(LS_START_TIME, j);

            if(val > m_trialPopulation[i].getWindowSlot(LS_END_TIME, j))
                val = m_trialPopulation[i].getWindowSlot(LS_END_TIME, j);

            // insere valor
            m_trialPopulation[j].setApplianceStartSlot(val, j);

        }
    }
}

void DiffEvolution::selection()
{

    double currentCostFunction = 0;
    double trialCostFunction = 0;
    for(unsigned int i = 0; i < POPULATION_SIZE; ++i)
    {
        currentCostFunction = m_population[i].costFunction();

        trialCostFunction = m_trialPopulation[i].costFunction();

        if (currentCostFunction > trialCostFunction)
        {
            for(unsigned int j = 0; j < DIM_SIZE; ++j) // atualiza o valor do inicio para cada carga
            {
                unsigned int slot = m_trialPopulation[i].getApplianceStartSlot(j);
                m_population[i].setApplianceStartSlot(slot, j);
            }
```

```
        }

    }
}

unsigned int DiffEvolution::uniformIntDist(unsigned int lowerBound, unsigned int upperBound)
{
    std::uniform_int_distribution<int> dist(lowerBound, upperBound);
    return dist(m_mersenne_engine);
}
```

# APÊNDICE C – CÓDIGO DO SOLUCIONADOR

## main.cpp

```
//
// Libraries
//
#include "load_scheduling.hpp"
#include "diff_evolution.hpp"

#include <iostream>
#include <vector>
#include <fstream>
#include <chrono>

//
// Defines
//
#define TAM_POPULACAO (50)
#define MINUTOS_SLOT (5)
#define N_SLOTS ((24 * 60) / MINUTOS_SLOT)

#define PROB_CROSSOVER (60)
#define TAM_PASSO (0.1)
#define MAX_ITERACOES (3000)

//
// Function declaration
//
void imprimeAgendamento(LoadScheduling &pop_atual);
void setApplianceData(std::string filename);

//
// Global vars
//
unsigned int N_CARGAS;
std::vector<equipamento_t> equipamentos;

//
// Main code
//
int main(int argc, char *argv[])
{
   auto start = std::chrono::steady_clock::now();
   setApplianceData("usuario.bin");
   // Cria a populacao de agendamentos
   std::vector<LoadScheduling> pop_atual;
```

```
for(unsigned int i = 0; i < TAM_POPULACAO; ++i)
{
   LoadScheduling ind_agendamento(MINUTOS_SLOT, &equipamentos[0], N_CARGAS);
   ind_agendamento.setEnergyPrice("./tarifas/tarifa.bin");
   ind_agendamento.changeLinearWeights(0, 0, 1);
   pop_atual.push_back(ind_agendamento);
   imprimeAgendamento(ind_agendamento);
}

// unsigned int pMax = 0;
// for(unsigned int i = 0; i < TAM_POPULACAO; ++i)
// {
//    if (pop_atual[i].getComfortCost() > pop_atual[pMax].getComfortCost())
//       pMax = i;
// }
//    pop_atual[pMax].saveAsCSV("inicial");
// DE
DiffEvolution ClassicalDE(pop_atual, N_CARGAS, PROB_CROSSOVER, TAM_PASSO);

unsigned int contador = 0;
while(contador < MAX_ITERACOES)
{
   ClassicalDE.generateTrialPopulation();
   ClassicalDE.selection();
   ++contador;

   std::cout << '\r' << " >> Passo: " << contador << " / " << MAX_ITERACOES;
}

std::cout << std::endl << "----------------------" << std::endl << std::endl;

// imprimeAgendamentoPopulation(pop_atual);

std::cout << ">>>> MELHOR SOLUCAO <<<<" << std::endl;

double indice_min = 0;
double valor_min = pop_atual[indice_min].costFunction();
for (unsigned int i = 0; i < TAM_POPULACAO; ++i)
{
   if(pop_atual[i].costFunction() < valor_min)
   {
      indice_min = i;
      valor_min = pop_atual[indice_min].costFunction();
   }
}

imprimeAgendamento(pop_atual[indice_min]);

if(!pop_atual[indice_min].saveAsCSV("resultado"))
```

```
        std::cout << "Erro ao salvar o resultado!" << std::endl;
    else
        std::cout << "Arquivo exportado com sucesso! " << std::endl;

    auto end = std::chrono::steady_clock::now();
    auto diff = end - start;
    std::cout << "Tempo de execucao: "  << std::chrono::duration<double, std::milli> (diff).count() << " ms" <<
std::endl;
    return 0;
}

// Print simples varrendo a populacao atual
void imprimeAgendamento(LoadScheduling &pop_atual)
{
    for (unsigned int i = 0; i < N_CARGAS; ++i)
    {
        unsigned int startSlot = pop_atual.getApplianceStartSlot(i);
        std::cout << equipamentos[i].nome << " - Inicio: " << LoadScheduling::convertToTimeString(startSlot *
MINUTOS_SLOT / 60.0);
        std::cout << " - tempo desejado: " <<
LoadScheduling::convertToTimeString(static_cast<double>(equipamentos[i].hora_preferencia / 60.0))<<
std::endl;
    }
    std::cout << "Custo Total: " << pop_atual.getTotalElectricityCost() << std::endl;
    std::cout << "Nivel de Desconforto: " << pop_atual.getComfortCost() << std::endl;
    std::cout << "Fator de Carga: " << pop_atual.getLoadFactor() << std::endl;
    std::cout << std::endl;
}

// le o arquivo dos equipamentos
void setApplianceData(std::string filename)
{
    std::ifstream user_file;
    user_file.open(filename);


    unsigned int equipamentos_size;
    user_file.read(reinterpret_cast<char *>(&equipamentos_size), sizeof(equipamentos_size));
    equipamentos.resize(equipamentos_size);

    N_CARGAS = equipamentos_size;


    for (unsigned int i = 0; i < equipamentos_size; ++i)
    {
        // nome
        unsigned int nome_size;
        user_file.read(reinterpret_cast<char *>(&nome_size), sizeof(nome_size));
        char *temp = new char[nome_size + 1];
```

```
        user_file.read(temp, nome_size);
        temp[nome_size] = '\0';
        equipamentos[i].nome = temp;
        delete[] temp;

        // coef de conforto
        user_file.read(reinterpret_cast<char *>(&equipamentos[i].coef_conforto),
sizeof(equipamentos[i].coef_conforto));

        // horario preferencia
        user_file.read(reinterpret_cast<char *>(&equipamentos[i].hora_preferencia),
sizeof(equipamentos[i].hora_preferencia));

        // horario inicio janela
        user_file.read(reinterpret_cast<char *>(&equipamentos[i].hora_inicio_janela),
sizeof(equipamentos[i].hora_inicio_janela));

        // horario termino janela
        user_file.read(reinterpret_cast<char *>(&equipamentos[i].hora_termino_janela),
sizeof(equipamentos[i].hora_termino_janela));

        // vetor energia
        unsigned int vetor_energia_size = equipamentos[i].vetor_energia.size();
        user_file.read(reinterpret_cast<char *>(&vetor_energia_size), sizeof(vetor_energia_size));
        equipamentos[i].vetor_energia.resize(vetor_energia_size);
        user_file.read(reinterpret_cast<char *>(&equipamentos[i].vetor_energia[0]), vetor_energia_size *
sizeof(equipamentos[i].vetor_energia[0]));
    }

    user_file.close();
}
```

# APÊNDICE D – CÓDIGO TARIFA

## tarifa.cpp

```
// libs
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <cstdio>

// def
#define MIN_DAY 24 * 60

// fnc
int convertToSlot(std::string str, int min);
bool writeVectorToFile(std::string filename, std::vector<double> &v, int tSlot);
bool readVectorFromFile(std::string filename, std::vector<double> &v);

// vars
double TFdP, TInt, TP;
int min, timeSlot, slotStart, slotEnd;
std::string tStart, tEnd;
std::string nome_arquivo;


// main fc
int main(int argc, char* argv[])
{
   // resolucao
   do {
   std::cout << "Entre com o valor da resolucao de tempo 'r' (em minutos): ";
   std::cin >> min;
   std::cin.ignore(256, '\n');
   if (MIN_DAY % min != 0)
      std::cout << "60 nao eh divisivel pelo valor entrado, por favor, insira novo valor." << std::endl;
   } while(MIN_DAY % min != 0);

   timeSlot = MIN_DAY / min;

   // inicio do horario de pico
   do {
      std::cout << "Entre com a hora e minutos do INICIO do horario de ponta (no formato HH:MM): ";
      std::getline(std::cin, tStart);
      slotStart = convertToSlot(tStart, min);
      if(slotStart < 0)
         std::cout << "Formato invalido, por favor, reentre o horario no formato HH:MM e multiplo de 'r'";
```

```
} while(slotStart < 0);

// termino do horario de pico
do {
   std::cout << "Entre com a hora e minutos do TERMINO do horario de ponta (no formato HH:MM): ";
   std::getline(std::cin, tEnd);
   slotEnd = convertToSlot(tEnd, min);
   if (slotEnd < 0)
      std::cout << "Formato invalido, por favor, reentre o horario no formato HH:MM e multiplo de 'r'";
} while (slotEnd < 0);

// valores das tarifas
std::cout << "Entre com o valor da tarifa fora de ponta (TFdP): ";

std::cin >> TFdP;
std::cin.ignore(256, '\n');

std::cout << "Entre com o valor da tarifa intermediaria (TInt): ";

std::cin >> TInt;
std::cin.ignore(256, '\n');

std::cout << "Entre com o valor da tarifa de ponta (TP): ";

std::cin >> TP;
std::cin.ignore(256, '\n');

// cria o vetor de tarifas

std::vector<double> energyPriceVector;
int slotHora = 60 / min;

for(int i = 0; i < timeSlot; ++i)
{
   if(i >= slotStart - slotHora && i < slotStart || i >= slotEnd && i <= slotEnd + slotHora - 1)
      energyPriceVector.push_back(TInt);
   else if(i >= slotStart && i < slotEnd)
      energyPriceVector.push_back(TP);
   else
      energyPriceVector.push_back(TFdP);
}

// salva vetor em arquivo binary

std::cout << "Entre com o nome do arquivo para salvar: ";
std::getline(std::cin, nome_arquivo);

writeVectorToFile(nome_arquivo, energyPriceVector, min);
```

```
    std::vector<double> readVector;
    readVectorFromFile(nome_arquivo, readVector);

    for(int i = 0; i < readVector.size(); i++)
        std::cout << readVector[i] << " ";

    std::cout << std::endl;

    return 0;
}

// converte HH:MM em timeSlot
int convertToSlot(std::string str, int min)
{
    unsigned int h, m;
    std::sscanf(str.c_str(), "%d:%d", &h, &m);
    int t = h * 60 + m;

    if(t % min != 0) return -1;

    return t / min;

}

bool writeVectorToFile(std::string filename, std::vector<double> &v, int tSlot)
{
    std::ofstream binaryFile;
    binaryFile.open(filename + ".bin", std::ios::binary | std::ios::out);
    if(!binaryFile.is_open())
    {
        std::cout << "Erro ao criar arquivo " << filename << ".bin " << std::endl;
        return false;
    }

    size_t v_len = v.size();
    binaryFile.write(reinterpret_cast<char *>(&v_len), sizeof(v_len));
    binaryFile.write(reinterpret_cast<char *>(&v[0]), v_len * sizeof(v[0]));

    binaryFile.close();

    std::ofstream datFile;
    datFile.open(filename + ".dat", std::ios::out);
    if (!datFile.is_open())
    {
        std::cout << "Erro ao criar arquivo " << filename << ".dat " << std::endl;
        return false;
    }

    for(int i = 0; i < v.size(); ++i)
```

```
        datFile << static_cast<int>(i * tSlot / 60) << ":" << static_cast<int>(i * tSlot % 60) << " " << v[i] <<
std::endl;

    datFile.close();

    return true;

}

bool readVectorFromFile(std::string filename, std::vector<double> &v)
{
    std::ifstream inputFile;
    inputFile.open(filename + ".bin", std::ios::binary | std::ios::in);
    if (!inputFile.is_open())
    {
        std::cout << "Erro ao ler arquivo " << filename + ".bin" << std::endl;
        return false;
    }

    size_t v_len;

    inputFile.read(reinterpret_cast<char *>(&v_len), sizeof(v_len)); // read length

    v.resize(v_len); // resize vector

    inputFile.read(reinterpret_cast<char *>(&v[0]), v_len * sizeof(v[0])); // read vector

    inputFile.close();

    return true;
}
```

# APÊNDICE E – CÓDIGO USUÁRIO

## usuario.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cstdio>

// evita padding entre os membros da struct
#pragma pack(push, 1)
typedef struct
{
   std::string nome;
   double coef_conforto;
   unsigned int hora_inicio;
   unsigned int hora_preferencia;
   unsigned int hora_inicio_janela;
   unsigned int hora_termino_janela;
   std::vector<double> vetor_energia;
} equipamento_t;
#pragma pack(push, 1)

std::vector<equipamento_t> equipamentos;

// Converte string no formato HH:MM em valores de minutos
unsigned int timeInMinutes(std::string &time)
{
   unsigned int h, m;
   std::sscanf(time.c_str(), "%d:%d", &h, &m);
   return h*60 + m;
}


bool generateCSVfile()
{
   std::ofstream binaryFile;

   binaryFile.open("usuario.bin", std::ios::binary | std::ios::out);
   if (!binaryFile.is_open())
   {
      std::cout << "Erro ao criar arquivo usuario.bin " << std::endl;
      return false;
   }

   unsigned int equipamentos_size = equipamentos.size();
```

```
    binaryFile.write(reinterpret_cast<char *>(&equipamentos_size), sizeof(equipamentos_size));

    for (int i = 0; i < equipamentos_size; ++i)
    {
        // nome
        unsigned int nome_size = equipamentos[i].nome.size();
        binaryFile.write(reinterpret_cast<char *>(&nome_size), sizeof(nome_size));
        binaryFile.write(equipamentos[i].nome.c_str(), nome_size);

        // coef de conforto
        double coef = equipamentos[i].coef_conforto;
        binaryFile.write(reinterpret_cast<char *>(&coef), sizeof(coef));

        // horario preferencia
        unsigned int hora_pref = equipamentos[i].hora_preferencia;
        binaryFile.write(reinterpret_cast<char *>(&hora_pref), sizeof(hora_pref));

        // horario inicio janela
        unsigned int hora_inicio_jan = equipamentos[i].hora_inicio_janela;
        binaryFile.write(reinterpret_cast<char *>(&hora_inicio_jan), sizeof(hora_inicio_jan));

        // horario termino janela
        unsigned int hora_termino_jan = equipamentos[i].hora_termino_janela;
        binaryFile.write(reinterpret_cast<char *>(&hora_termino_jan), sizeof(hora_termino_jan));

        // vetor energia
        unsigned int vetor_energia_size = equipamentos[i].vetor_energia.size();
        binaryFile.write(reinterpret_cast<char *> (&vetor_energia_size), sizeof(vetor_energia_size));
        binaryFile.write(reinterpret_cast<char *>(&equipamentos[i].vetor_energia[0]), vetor_energia_size *
sizeof(equipamentos[i].vetor_energia[0]));
        }

    binaryFile.close();

    return true;
}


int main()
{
    int nCargas;

    // Entrada do numero de cargas
    std::cout << "Entre com o numero de cargas: ";
    std::cin >> nCargas;
    std::cin.ignore(256, '\n');

    equipamentos.resize(nCargas);
```

```
// Entrada de cada carga
for(int i = 0; i < nCargas; ++i)
{
    std::cout << "Entre com os dados referentes a carga #" << i + 1 << std::endl;

    // nome
    std::cout << '#' << i + 1 << " > Nome da carga: ";
    std::getline(std::cin, equipamentos[i].nome);

    // coeficiente de conforto
    std::cout << '#' << i + 1 << " > Coeficiente de conforto [0-1]: ";
    double cdc;
    std::cin >> cdc;
    std::cin.ignore(256, '\n');
    equipamentos[i].coef_conforto = cdc;

    // preferencia de inicio
    std::cout << '#' << i + 1 << " > Horario de preferencia [HH:MM]: ";
    std::string hp;
    unsigned int hp_min;
    std::getline(std::cin, hp);
    hp_min = timeInMinutes(hp);
    equipamentos[i].hora_preferencia = hp_min;

    // inicio de janela
    std::cout << '#' << i + 1 << " > Horario inicio de janela [HH:MM]: ";
    std::string hij;
    unsigned int hij_min;
    std::getline(std::cin, hij);
    hij_min = timeInMinutes(hij);
    equipamentos[i].hora_inicio_janela = hij_min;

    // termino de janela
    std::cout << '#' << i + 1 << " > Horario termino de janela [HH:MM]: ";
    std::string htj;
    unsigned int htj_min;
    std::getline(std::cin, htj);
    htj_min = timeInMinutes(htj);
    equipamentos[i].hora_termino_janela = htj_min;

    // vetor de consumo da carga
    std::cout << '#' << i + 1 << " > Especifique quantidade de ciclos: ";
    unsigned int ciclos;
    std::cin >> ciclos;
    std::cin.ignore(256, '\n');
    std::vector<double> ciclos_equip;

    std::cout << "Insira os valores [kWh]" << std::endl;
    for(int j = 0; j < ciclos; ++j)
```

```
        {
            std::cout << "Ciclo #" << j + 1 << ": ";
            double val;
            std::cin >> val;
            std::cin.ignore(256, '\n');
            equipamentos[i].vetor_energia.push_back(val);
        }

        std::cout << std::endl;

    }

    generateCSVfile();

    return 0;
}
```