NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

RUI FILIPE PINTO ESTEVES DA SILVA

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

OTIMIZAÇÃO DO CONSUMO DE ENERGIA POR RECURSO A TÉCNICAS DE COMPUTAÇÃO EVOLUCIONÁRIA

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Novembro, 2021

DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES

OTIMIZAÇÃO DO CONSUMO DE ENERGIA POR RECURSO A TÉCNICAS DE COMPUTAÇÃO EVOLUCIONÁRIA

RUI FILIPE PINTO ESTEVES DA SILVA

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Orientador: Pedro Miguel Ribeiro Pereira
Prof. Doutor, NOVA School of Science and Technology

Júri:

Presidente: João Paulo Branquinho Pimentão
Prof. Doutor, NOVA School of Science and Technology

Arguente: Rui Manuel Leitão Tavares
Prof. Doutor, NOVA School of Science and Technology

Orientador: Pedro Miguel Ribeiro Pereira
Prof. Doutor, NOVA School of Science and Technology

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Novembro, 2021

Otimização do Consumo de Energia por recurso a Técnicas de Computação Evolucionária

Este documento foi gerado utilizando o processador (pdf/Xe/Lua)LaTeX, com base no template NOVAthesis, desenvolvido no Dep. Informática da FCT–NOVA por João M. Lourenço. Lourenço, 2021

Agradecimentos

No término deste percurso de 5 anos é importante, de alguma forma, reconhecer a ajuda que me foi prestada por várias pessoas, em ambiente académico e não só. Felizmente, estas seriam difíceis de enumerar, ainda assim, as seguintes pessoas contribuíram de forma muito significativa para esta dissertação.

O meu agradecimento ao Professor Pedro Pereira, meu orientador, pela oportunidade de desenvolver esta dissertação sobre temas pelos quais muito me interesso, bem como pela sua disponibilidade e ajuda valiosas.

À Faculdade de Ciências e Tecnologias, instituição que me proporcionou uma formação excepcional e aos docentes do Departamento de Engenharia Eletrotécnica e de Computadores, pelo conhecimento transmitido e pela partilha da sua visão.

Aos meus pais, pelos valores transmitidos e esforço incansável, sem os quais não teria chegado aqui. À minha irmã, interlocutora principal de triunfos e infortúnios.

Aos meus amigos, companheiros de todas as horas, pelos quais tenho a sorte de ser apoiado em todos os desafios.

"Sempre chegamos ao sítio aonde nos esperam." (José Saramago)

Resumo

Os edifícios, residenciais ou associados a serviços, compõem uma grande porção do consumo de eletricidade em Portugal e continuam a ter um grande potencial para o aumento de eficiência energética. Por outro lado, as alterações climáticas são um assunto cada vez mais premente e, como tal, surgem vários esforços na tentativa de diminuir os efeitos negativos que a sociedade tem sobre o ambiente. Um dos fatores que contribui para esta diminuição é a implementação de medidas que visam o aumento da eficiência energética.

Com vista a aumentar a eficiência energética pela diminuição do consumo de energia elétrica, foi implementado na presente dissertação, um sistema de controlo de iluminação por recurso a técnicas de computação evolucionária, pela sua capacidade de resolução de problemas multiobjetivo. Este sistema permitiu alcançar vários objetivos simultaneamente, nomeadamente a diminuição do consumo de energia e a manutenção do conforto para o utilizador.

O sistema foi implementado no Departamento de Engenharia Eletrotécnica e de Computadores da NOVA School of Science and Technology, onde teve a capacidade de controlar a iluminação artificial num determinado espaço, de modo a obter consumos mais baixos, mantendo o nível de iluminância necessário de acordo com as necessidades do mesmo.

Palavras-chave: Sistema de Controlo, Eficiência Energética, Iluminação, Computação Evolucionária.

Abstract

Buildings, residential or service related, make up a large share of electricity consumption in Portugal while still having a great potential for increased efficiency. On the other hand, climate change is an increasingly pressing issue and, as such, various efforts are being made to reduce the negative effects that society has on the environment. One of the factors contributing to this decrease is the implementation of energy efficiency measures.

In order to increase energy efficiency by reducing consumption, in this dissertation a lighting control system was developed, using evolutionary computing techniques, due to its multiobjective problem solving capability. This system allows the achievement of several objectives simultaneously, namely the reduction of energy consumption and mantaining comfort for the user.

This system was implemented within the Department of Electrical and Computer Engineering at NOVA School of Science and Technology, where it had the capability to control the lamps in a certain space, in order to obtain lower electricity consumption while maintaining the necessary level of illuminance according to its needs.

Keywords: Control System, Lighting, Energy Efficiency, Evolutionary Computing.

Índice

Índice de Figuras

Índice de Tabelas

Siglas e Acrónimos

ACO	*Ant Colony Optimization*
AE	Algoritmos Evolucionários
AG	Algoritmo Genético
AGS	Algoritmo Genético Simples
CART	*Classification and Regression Trees*
ED	Evolução Diferencial
HVAC	*Heating, ventilation, and air conditioning*
LCS	*Learning Classifier System*
MQTT	*Message Queuing Telemetry Transport*
MS	*Michigan Style (Learning Classifier System)*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
nZEB	*Nearly Zero Energy Building*
PE	Programação Evolucionária
PG	Programação Genética
PSO	*Particle Swarm Optimization*
SBC	*Single Board Computer*
WSAN	*Wireless Sensor and Actuator Network*

1

Introdução

O capítulo de introdução é composto por quatro subcapítulos. No primeiro é feito o enquadramento da dissertação no panorama energético atual, como uma possível solução para o aumento da eficiência energética em edifícios. São também apresentados os objetivos que se pretendem alcançar com o sistema a desenvolver, bem como contribuições originais da dissertação. Por fim, é descrita a estrutura deste documento.

1.1 Enquadramento

A eficiência energética é um dos temas mais emergentes no que diz respeito à resposta à atual crise energética e ambiental. Os edifícios, residenciais e não residenciais, continuam a ter um enorme potencial para aumento de eficiência e são uma das grandes apostas da União Europeia, onde surgem esforços no sentido de tornar todas as novas construções em edifícios com necessidades quase nulas de energia (nZEB, do inglês *Nearly Zero Energy Buildings*).

No ano 2017 o setor residencial foi responsável por 26% do consumo total de energia em Portugal e o setor não doméstico por 25% ("Consumo de energia eléctrica: total e por tipo de consumo", s.d.). Na União Europeia 57.2% do consumo de energia elétrica residencial é devido a iluminação e aparelhos ("Energy consumption in households", s.d.) e nos E.U.A. 20% da eletricidade consumida pelo setor de serviços em 2013 foi para iluminação ("Electricity Customers", s.d.).

No Plano Nacional Energia e Clima 2030 (PNEC2030) do governo português é proposta a redução de consumos em 70% no setor dos serviços em relação ao ano 2005 bem como a redução em 35% de consumo no setor residencial face ao mesmo ano ("Resolução do Conselho de Ministros n.º 53/2020", 2020). A utilização de lâmpadas mais eficientes, como a tecnologia LED, e a proibição em 2018 da venda de lâmpadas de halogéneo ineficientes, foram grandes passos na diminuição do consumo energético por iluminação artificial.

Para além do esforço na utilização de lâmpadas mais eficientes, o controlo de sistemas de iluminação tem sido uma resposta importante para uma poupança de energia ainda

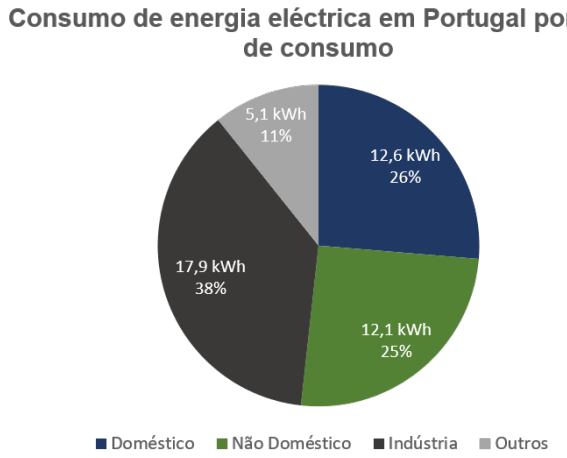


Figura 1.1: Consumo de energia eléctrica em Portugal. Obtido de ("Consumo de energia eléctrica: total e por tipo de consumo", s.d.).

mais significativa. Estes sistemas podem atingir poupanças de energia elétrica elevadas, na ordem dos 60% (Galasiu, Newsham, Suvagau & Sander, 2013), face a iluminação ligada durante todo o período de trabalho, dependendo do tipo de sistema e da altura do ano.

Outras preocupações que estes sistemas devem ter em conta são a simplicidade de instalação, havendo sempre preferência por instalações que não exijam a remodelação do espaço, por razões financeiras e de praticidade, bem como a preocupação com o conforto visual do utilizador (Iacomussi, Radis, Rossi & Rossi, 2015) e o efeito que este pode ter na sua produtividade. (Haynes, 2008)

1.2 Objetivos

Com esta tese pretende-se desenvolver e documentar a implementação de um sistema inteligente de controlo de iluminação artificial no interior de um edifício não residencial para otimizar o consumo de energia, recorrendo a técnicas de Computação Evolucionária. O sistema terá as seguintes capacidades:

- Atuação nas luminárias dependendo da necessidade de iluminância e das condições medidas por vários sensores, tendo em conta a eficiência energética;
- Aproveitamento da luz natural, considerando a sua influência na luminosidade do espaço;
- Diagnóstico de falhas no sistema de iluminação;
- Melhorar a experiência do utilizador simultaneamente à poupança de energia;

1.3 Contribuição

Esta dissertação contribuirá com um sistema funcional capaz de ser aplicado em vários contextos, com pequenas modificações, para controlo de iluminação, promovendo um aumento da eficiência energética. O sistema de controlo inteligente será de fácil instalação e baixo custo, de modo a permitir a replicação em edifícios residenciais ou públicos de qualquer dimensão, através do uso de uma rede sem fios de sensores e atuadores (WSAN), usando computação evolucionária para fazer o balanço entre menores consumos energéticos e um maior conforto para os utilizadores do sistema.

1.4 Estrutura da Dissertação

Este documento encontra-se dividido em 7 capítulos, organizados da seguinte forma:

Capítulo 2 - Conceitos Fundamentais : Contém uma introdução à computação evolucionária, expondo as componentes de um algoritmo evolucionário bem como alguns exemplos de algoritmos evolucionários e de inteligência coletiva.

Capítulo 3 - Estado da Arte : São analisados cinco projetos semelhantes ao desta dissertação, que aplicam técnicas de computação evolucionária na área de controlo inteligente de iluminação, para perceber as vantagens e desvantagens de cada implementação.

Capítulo 4 - Arquitetura do Sistema : Descreve a arquitetura a adotar no desenvolvimento da dissertação. Clarifica o objetivo e as componentes necessárias para o alcançar, na camada física do sistema, bem como na camada de decisão. Expõe a forma como estes componentes devem interagir.

Capítulo 5 - Implementação do Sistema : Apresenta a implementação das diferentes partes do sistema. Caracteriza a instalação, descreve os equipamentos escolhidos para o sistema, bem como os programas e código envolvidos no processo de decisão e controlo.

Capítulo 6 - Resultados : Relata os resultados obtidos na dissertação. Os resultados provêm de duas etapas: recolha de informação e treino dos modelos de decisão e posteriormente teste do sistema de controlo de iluminação. Avalia-se o impacto da tese no consumo de energia e no conforto do utilizador.

Capítulo 7 - Conclusão : Aborda as conclusões e observações finais. Propõe sugestões para trabalhos futuros, que permitem ultrapassar algumas limitações do trabalho.

2

Conceitos Fundamentais

Os Algoritmos Evolucionários (AE) tendem a ser usados na resolução de problemas multiobjetivo, nos quais são considerados múltiplos objetivos, que podem ser incompatíveis. Um bom exemplo deste tipo de problemas é o conhecido problema "*travelling salesman*", em que se procura diminuir a distância percorrida por um vendedor ambulante, que deve visitar um determinado número de cidades antes de regressar à cidade de onde partiu (Collette & Siarry, 2003). Assim, tipicamente, não existe uma única solução, mas sim um conjunto de soluções que otimiza os vários objetivos, de acordo com pesos que devem ser definidos para cada um.

A vantagem da utilização de AE na resolução de problemas multiobjetivo está na eficiência e rapidez dos mesmos, já que em vez de validar todas as possíveis soluções, estes algoritmos criam um conjunto de soluções aleatórias que iterativamente evoluem. Assim, encontram um determinado número de soluções que respeitam parâmetros previamente definidos.

2.1 Computação Evolucionária

A computação evolucionária é um ramo da computação que se inspira no processo de evolução natural biológica segundo Darwin. Embora existam vários grupos de algoritmos evolucionários, todos têm o mesmo princípio de funcionamento: vários organismos competem num ambiente com recursos limitados e apenas os mais aptos podem sobreviver. Existem também mutações (alterações no fenótipo) que alteram a aptidão do organismo e podem ou não ser transmitidas à sua descendência após o processo de recombinação. A junção destes fatores resulta numa melhoria na aptidão da população.

Para além dos algoritmos evolucionários, existe outra categoria de algoritmos que, apesar de ser considerada parte da computação evolucionária, baseia-se maioritariamente no comportamento social de animais. Estes algoritmos são classificados como algoritmos de inteligência coletiva, pois o sucesso da espécie depende da cooperação entre os vários indivíduos.

2.1.1 Algoritmos Evolucionários

A aplicação da teoria da evolução de Darwin na solução de problemas data do fim dos anos 1940 (Eiben & Smith, 2007). Desde então, surgiram vários algoritmos, que se diferenciam principalmente pela forma como implementam as componentes do algoritmo.

Na composição de qualquer algoritmo evolucionário estão várias componentes que permitem a existência de evolução. De seguida, são resumidas e representadas na figura 2.1 as mais relevantes: Representação, Avaliação, Seleção de Ascendentes, Mutação, Recombinação e Seleção de Sobreviventes. Para cada componente é dado um exemplo da sua implementação, tipicamente em Algoritmos Genéticos Simples (*SGA*), já que é constituído por versões simples de cada componente.

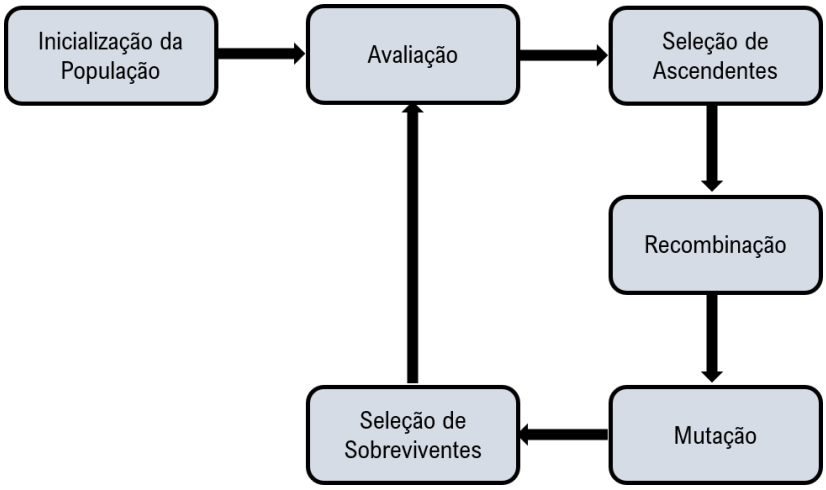


Figura 2.1: Componentes de um Algoritmo Evolucionário.

Representação Um indivíduo no contexto de algoritmos evolucionários é representado em dois níveis: o fenótipo e o genótipo. De forma semelhante à da biologia, o fenótipo é a característica observável do gene no ambiente, isto é, a expressão dos seus genes, como a cor dos olhos ou o peso. Por outro lado, o genótipo é a codificação dos genes, no caso da biologia, uma cadeia de cromossomas (Fonseca, 1995). A representação é a relação entre os fenótipos e os genótipos que os representam, embora o termo também seja usado para referir à estrutura em que estes genótipos são implementados (Eiben & Smith, 2007). Entre os tipos de representação possíveis estão os vetores binários, vetores de números inteiros, vetores de números reais ou árvores.

Numa representação vetorial, como a da Figura 2.2, cada elemento do vetor representa um gene, sendo o seu índice chamado de *locus* (ou localização) e o valor correspondente de alelo (Yu & Gen, 2010).

Avaliação A avaliação da aptidão da população é feita através de uma função objetivo (ou *fitness function*), que representa o problema e os requisitos que a população deve preencher. Com esta função os indivíduos são avaliados em relação ao quão aptos são,

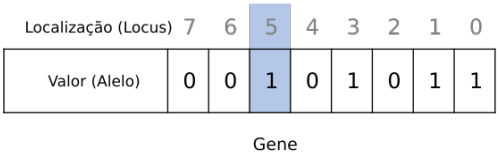


Figura 2.2: Representação de um Cromossoma no Algoritmo SGA. Adaptado de (Yu & Gen, 2010).

dado o ambiente em que estão inseridos. É assim, através da função objetivo, que deverão ser definidos os parâmetros mais relevantes para o problema em questão e o peso que cada variável deverá ter na procura pela solução.

Um exemplo, muito básico, de uma função objetivo num algoritmo genético simples seria avaliar a quantidade de genes cujo valor binário é 1, como demonstrado na Figura 2.3.

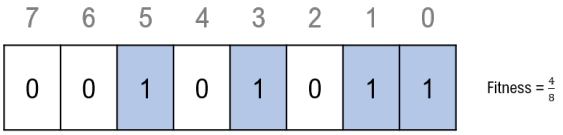


Figura 2.3: Exemplo de Avaliação para Representação Binária.

Seleção de Ascendentes Neste passo é feita a escolha dos ascendentes, ou progenitores, para se proceder à obtenção de novas combinações de genes. Para isto, interessa selecionar os indivíduos mais aptos, podendo esta escolha ser feita de vários modos.

Quando a seleção é probabilística, isto é, os mais aptos têm uma probabilidade superior aos restantes de produzir descendência, é importante que estes últimos ainda retenham alguma probabilidade de modo a evitar a convergência para uma solução ótima local (Eiben & Smith, 2007). Existem várias possibilidades de executar o processo de seleção de ascendentes:

- **Seleção Proporcional à Adaptação**: a probabilidade de um indivíduo ser escolhido para progenitor é proporcional ao seu resultado absoluto da função objetivo face ao dos restantes. Pode resultar em convergência prematura, quando muitos dos primeiros cromossomas têm avaliações baixas (Eiben & Smith, 2007). Um exemplo é o método da roleta, em que indivíduos com maior aptidão ocupam porções maiores de uma roleta, como na Figura 2.4.

- **Seleção por *Ranking***: deste modo, os indivíduos são colocados em posições com base na sua avaliação e só então é distribuída a probabilidade de se tornarem progenitores, com base na sua posição.

- **Seleção por Torneio**: usada quando a população tem uma grande dimensão ou a modelação do problema o requer, consiste na comparação direta de pares de indivíduos.
- **Seleção Uniforme**: todos os cromossomas têm a mesma probabilidade de gerar descendência, este método deve ser aliado a uma seleção de sobreviventes mais forte.

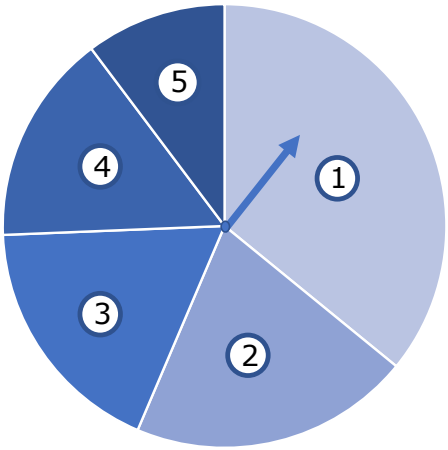


Figura 2.4: Exemplo de Seleção pelo Método da Roleta.

Mutação e Recombinação A mutação e recombinação permitem a criação de novos indivíduos a partir de outros já existentes, esta introdução de inovação no sistema é o que permite a melhoria da aptidão.

A mutação consiste na alteração de uma pequena porção do genótipo, mantendo-se a grande parte deste igual ao resultante da recombinação de progenitores. A probabilidade de mutação é chamada *mutation rate* ou *mutation step size*, dependendo da implementação. A recombinação (ou *crossover*, em certos casos) dá-se quando existe o cruzamento dos genótipos entre dois ou mais organismos, resultando num ou mais organismos com características dos anteriores. Esta componente é aplicada conforme uma probabilidade de recombinação ou *crossover rate*.

A forma como se implementam estas componentes depende da forma como os indivíduos são representados no contexto do problema.

- **Representação Binária**
  - **Mutação por Troca de Bit**: um bit aleatório do vetor é invertido, como exemplificado na Figura 2.5.
  - **Recombinação por *One Point Crossover***: separa os vetores dos progenitores no mesmo ponto aleatório e gera um vetor descendente composto pela cabeça e cauda de cada um dos vetores progenitores, do modo descrito na Figura 2.6.

Poderão ser escolhidos mais pontos de divisão e o descendente receberá um conjunto de genes alternadamente de cada parente, nesse caso teremos *n Point Crossover*.

– **Recombinação por *Crossover* Uniforme (Direta)**: em cada gene é decidido aleatoriamente de qual progenitor será herdado o valor do alelo.

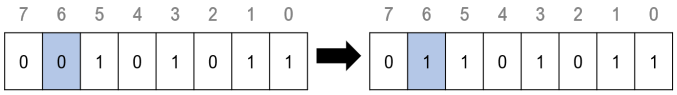


Figura 2.5: Mutação em Representação Binária por Troca de Bit.

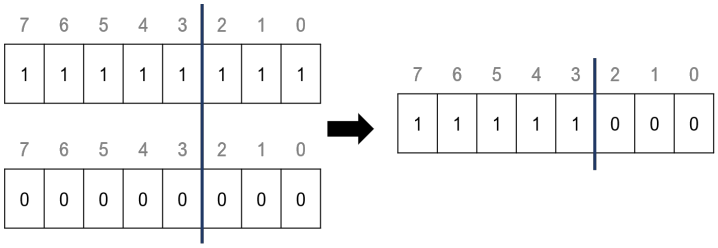


Figura 2.6: Recombinação Binária por *One Point Crossover*.

- **Representação Real**

  – **Mutação por *Random Resetting***: cada gene tem uma probabilidade definida de ser alterado para um número aleatório.

  – **Mutação *Creep***: adiciona um valor baixo a cada gene, com uma determinada probabilidade.

  – **Recombinação**: nesta representação podem ser usados os mesmos métodos que na representação binária, por serem suficientes (Eiben & Smith, 2007).

  – **Recombinação por *Crossover* Uniforme (Intermediária)**: em cada gene é aplicada a média do valor dos ascendentes.

- **Representação em Árvore**

  – **Mutação**: é escolhido aleatoriamente um nó da árvore para que o seu valor seja alterado. A mutação em algoritmos que usam árvores para representação tende a ser baixa ou nula (Eiben & Smith, 2007).

  – **Recombinação**: são criadas 4 subárvores a partir de dois indivíduos, dividindo cada um num nó aleatório, depois são geradas duas árvores descendentes, sendo cada uma constituída por uma parte de cada progenitor.

Seleção de Sobreviventes Dado que o número de organismos num algoritmo evolucionário costuma ser constante (dimensão da população), deve ser feita uma seleção em relação a que indivíduos continuarão a fazer parte da população.

Esta seleção pode ser feita tendo em conta vários fatores, como a avaliação por uma função objetivo (de modo semelhante à seleção de ascendentes) ou a seleção com base na idade dos indivíduos. Nos algoritmos genéticos mais simples toda a população é substituída pelos novos descendentes (Fonseca, 1995).

Em alguns algoritmos é implementado o conceito de elitismo, que consiste em preservar o indivíduo com a melhor classificação de toda a população, quando este participa no processo de reprodução só é substituído no caso da descendência ser igualmente ou melhor classificada (Eiben & Smith, 2007).

- **Substituição Geracional**: os elementos descendentes substituem imediatamente os seus progenitores.
- **Substituição por Idade**: a avaliação dos indivíduos não é tida em conta, mas sim a idade, isto é, indivíduos que façam parte da população há um dado número de iterações, serão eliminados e substituídos por uma nova geração.
- **Substituição por Avaliação**: os piores n elementos de uma dada população são substituídos pela totalidade n de descendentes, de modo demonstrado na Figura 2.7.
- **Substituição por Torneio**: de maneira semelhante à implementada na seleção de ascendentes, porém cada indivíduo é comparado a vários.
- **Seleções** $(\mu + \lambda)$ **e** (μ, λ): utilizadas em Estratégias Evolutivas, na primeira os ascendentes e descendentes são avaliados juntamente e os piores descartados, na segunda a população é inteiramente substituída pelos melhores elementos da nova geração.

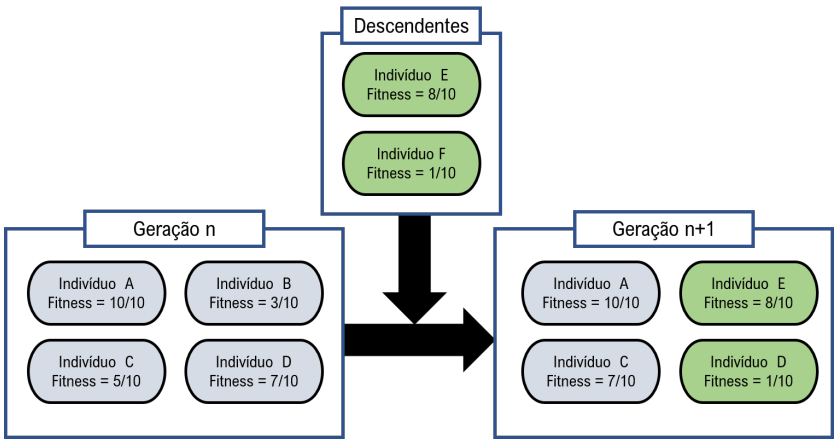


Figura 2.7: Exemplo de Seleção de Sobreviventes por Avaliação.

Inicialização e Condição de Término Tipicamente a inicialização de um algoritmo consiste apenas na criação de uma população aleatória, definindo um número de indivíduos.

Em relação ao término do programa, sendo que este, por ser estocástico, pode nunca atingir uma solução ótima para o problema, é necessária a definição de condições para o fim da sua execução. Estas poderão ser definidas com base na junção de várias condições como exceder tempo de processamento ou atingir um nível baixo de diversidade no sistema (Eiben & Smith, 2007).

Nos capítulos seguintes, de 2.1.1.1 a 2.1.1.4, são apresentados quatro tipos de algoritmos evolucionários, sendo eles: Algoritmos Genéticos, Estratégias Evolutivas, Programação Genética e Algoritmo de Classificação.

2.1.1.1 Algoritmos Genéticos

Os algoritmos genéticos (AG) são os mais comuns no ramo da computação evolucionária. Os primeiros e mais básicos algoritmos genéticos, Algoritmos Genéticos Simples (AGS), são caracterizados pela representação simples dos genes, usando apenas um vetor de *bits* para este efeito (representação binária).

A componente de mutação é feita por troca de bit num determinado gene do cromossoma e a recombinação consiste em *One Point Crossover* aplicado aos vetores de dois progenitores. A seleção de progenitores é feita proporcionalmente à adaptação e através do método da roleta, como tal, quanto maior for a sua adaptação, maior é a probabilidade de um cromossoma ser escolhido para progenitor. A nova geração substitui na totalidade a geração que a precedeu. Um esquema simplificado de algoritmo genético é ilustrado na Figura 2.8.

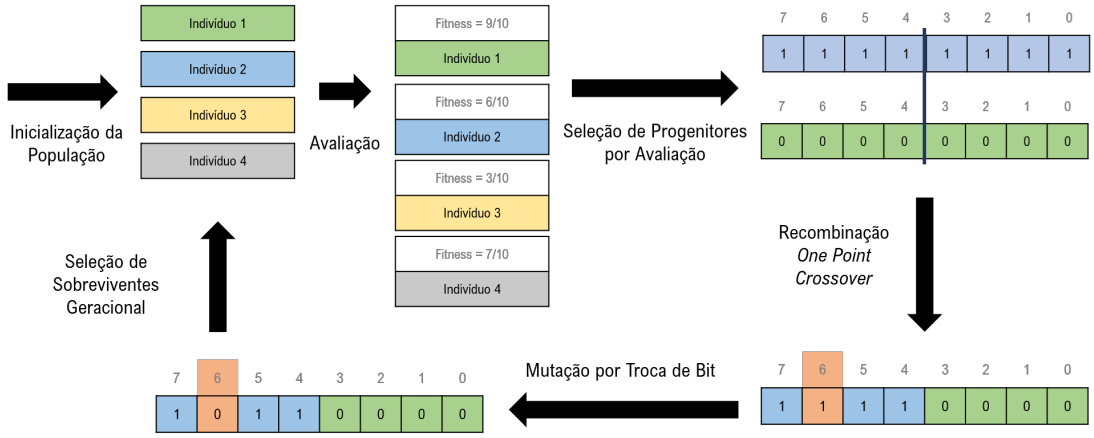


Figura 2.8: Algoritmo Genético.

Novas iterações do algoritmo, como o célebre *Non-dominated Sorting Genetic Algorithm II* (NSGA-II), diferenciam-se bastante desta noção inicial, notavelmente incluindo a ideia de elitismo (Eiben & Smith, 2007). Outro algoritmo genético é denominado Algoritmo Memético e incorpora a noção de meme de Richard Dawkins, que é análogo a um gene mas contendo ideias ou práticas que se vão disseminando. Assim, incorporam *Local Search*

no algoritmo, permitindo que os indivíduos ganham experiência antes de entrarem no processo evolutivo (Elbeltagi, Hegazy & Grierson, 2005).

2.1.1.2 Estratégias Evolutivas

Os algoritmos Estratégias Evolutivas diferem dos algoritmos genéticos simples em vários sentidos, começando pela sua representação no espaço vetorial real. A descendência é, neste caso, obtida por recombinação direta ou intermediária dos genes dos progenitores e mutação *creep*, por adição de um valor baixo proveniente de uma distribuição normal, cujo desvio padrão é chamado *mutation step size*.

A seleção dos indivíduos que dará origem a descendência é uniforme, com probabilidade igual para todos os indivíduos. No que toca à seleção de sobreviventes, como mencionado anteriormente, existem dois modos de operação: $(\mu + \lambda)$, em que a população existente é avaliada juntamente com a descendência e apenas os mais aptos sobrevivem, de modo representado na figura 2.9 e (μ, λ) em que os descendentes substituem inteiramente a geração anterior, sendo este último o mais usado. Para além disto, cada cromossoma inclui também parâmetros que controlam a evolução, assim, estes evoluem paralelamente com as soluções e dão lugar a *self-adaptation* (Eiben & Smith, 2007).

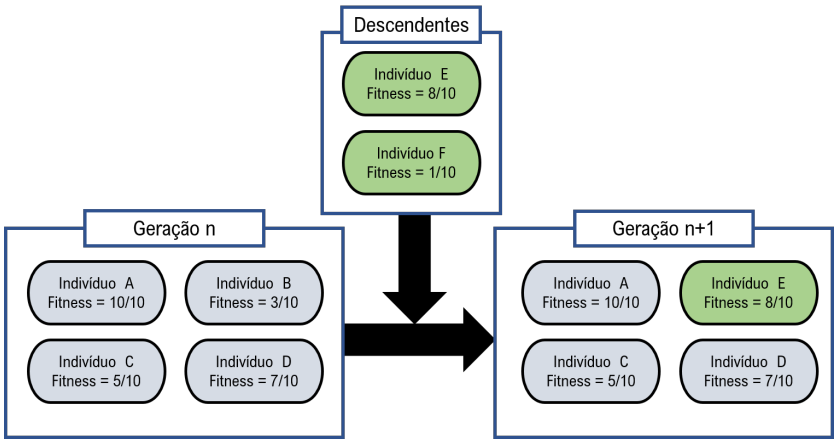


Figura 2.9: Exemplo de Seleção de Sobreviventes $(\mu + \lambda)$.

Um outro grupo de algoritmos evolucionários, a Programação Evolutiva (PE), pode ser visto como uma subcategoria de Estratégias Evolutivas, sendo que o vetor real na representação deste primeiro é visto como uma máquina de estados. Para além disto, o algoritmo *PE* conta apenas com a mutação como operador de variação. Cada progenitor gera apenas um descendente e a totalidade da população compete em torneios *round-robin* pela sobrevivência (Eiben & Smith, 2007).

Existe ainda um algoritmo mais recente, Evolução Diferencial, introduzido em 1996, que é em muitos aspetos igual a EE, exceto na componente da mutação, que é feita diferencialmente. Na mutação diferencial, um vetor de perturbação (que consiste na diferença entre dois outros vetores da população) é adicionado ao vetor cromossoma (Eiben & Smith, 2007).

2.1.1.3 Programação Genética

A grande diferença do algoritmo Programação Genética (PG), quando comparado com os descritos anteriormente (que evoluíram paralelamente no mesmo espaço temporal), é o facto de os indivíduos serem representados como árvores, como na Figura 2.10. Isto permite que estes algoritmos encontrem soluções para problemas que requerem aprendizagem e otimização de estrutura, enquanto que os algoritmos genéticos, por exemplo, são mais indicados para problemas de otimização (Yu & Gen, 2010). Desta maneira, os algoritmos PG podem ser vistos como o processo de evolução dos programas de computador em si (Eiben & Smith, 2007).

Estes algoritmos representam um programa como uma árvore, cujos nós serão operadores (matemáticos ou lógicos) ou variáveis, e podem ser convertidas em sintaxe da linguagem LISP. A reprodução em GP tende a usar apenas mutação ou recombinação, e sendo a representação do cromossoma feita através de árvores, a recombinação passa pela troca de partes da árvore. A mutação, que tem um papel muito diminuto nestes algoritmos, consiste em substituir um nó aleatório da árvore por um gerado também aleatoriamente, sendo depois a estrutura do programa verificada para garantir a possibilidade de execução (Eiben & Smith, 2007).

Um algoritmo que surge como resposta a PG é o *Gene Expression Programming*, que implementa uma representação linear das árvores, sendo a dimensão dos cromossomas constante, que resolve alguns dos problemas no algoritmo original (Ferreira, 2001).

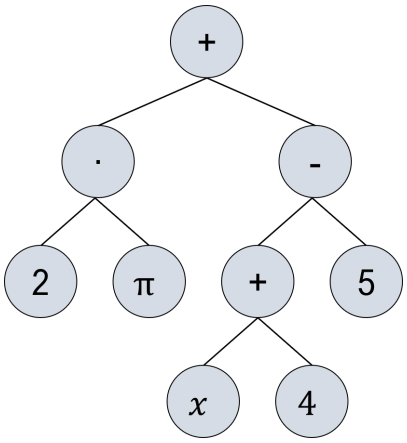


Figura 2.10: Representação por Árvore. Adaptado de (Eiben & Smith, 2007).

2.1.1.4 Algoritmo de Classificação

Os Algoritmo de Classificação, ou *Learning Classifier System* (LCS), adoptam outro método para a representação, baseado em regras (Dam, Abbass, Lokan & Yao, 2007). A aplicação destes métodos é usada quando se pretende evoluir um sistema que responda ao estado atual do ambiente em que se situa.

Assim, os *LCS* representam os cromossomas com regras compostas por Condição e Ação. A condição é um estado do ambiente e a ação (ou predição, dependendo do contexto) é aquela que é desbloqueada quando se verifica a condição. A junção das regras constitui um modelo que representa todos os estados possíveis e os seus *outputs* (Eiben & Smith, 2007). O primeiro *LCS* surgiu em 1976 e foi criado por John Henry Holland como extensão do algoritmo genético, o seu esquemático está representado na Figura 2.11.

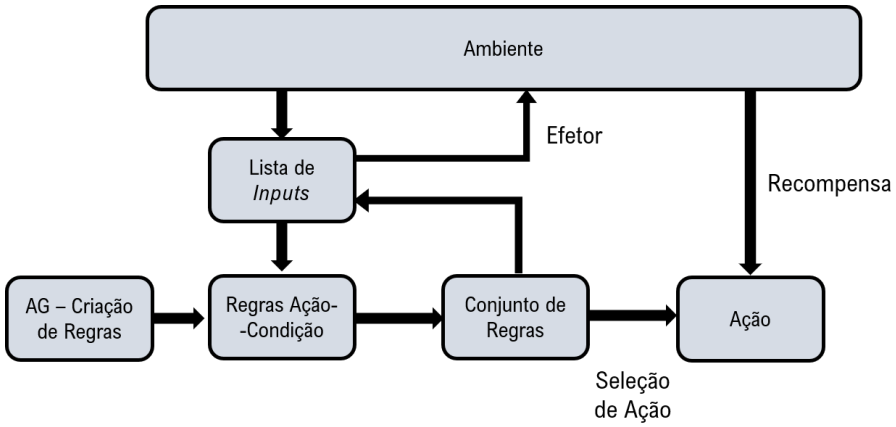


Figura 2.11: Esquemático do Classificador de Holland. Adaptado de (Bull, 2004).

Existem duas variações deste algoritmo: os classificadores *Michigan Style* (*MS*), em que a solução é representada através de um único conjunto de regras com vários indivíduos e *Pittsburgh Style* em que a população consiste em vários conjuntos de regras de várias dimensões e a solução é um destes conjuntos. No primeiro, as regras são avaliadas individualmente e no último a função objetivo avalia a precisão média do conjunto de regras.

Na variação *MS*, a recombinação dá-se através de *one point crossover* para os valores de condição e ação e a mutação através da redefinição das mesmas, podendo estas tomar um valor X (nem 0 nem 1), indicando *don't care*.

Um classificador mais recente, da variante Michigan, o algoritmo *XCS*, usa para representação tuplos do tipo *condition:action:payoff,accuracy*.

2.1.2 Inteligência Coletiva

Para além dos algoritmos evolucionários, existe outra categoria de algoritmos que também se enquadra no tópico da evolução computacionária, embora não sejam inspirados no processo de evolução natural.

Os algoritmos de inteligência coletiva, ou *swarm intelligence*, baseiam-se no comportamento de conjuntos de animais visto como um todo, inspirando-se em animais altamente sociais como insetos, pássaros ou peixes.

Neste tipo de algoritmos, existem dois conceitos relevantes: (Selvi & Umarani, 2010)

- **Homogeneidade**: todos os indivíduos do grupo são modelados de forma igual e não existem líderes definidos.
- **Comportamento Local**: um indivíduo só é influenciado pelos que estão nos seus arredores.

2.1.2.1 Algoritmo Enxame de Partículas

O Algoritmo Enxame de Partículas, em inglês *Particle Swarm Optimization* (PSO), baseia-se no comportamento de um bando de aves, embora a terminologia usada seja baseada em partículas. Cada indivíduo é representado por um par de vetores: a solução que propõe, que também pode ser vista como a sua posição, e um vetor de propagação associado, considerada a velocidade. Pode então considerar-se uma representação por trios compostos pela posição, velocidade e ainda a melhor posição (solução) que o indivíduo já obteve (Eiben & Smith, 2007). Na figura 2.12 está representada a dinâmica de convergência do algoritmo PSO.

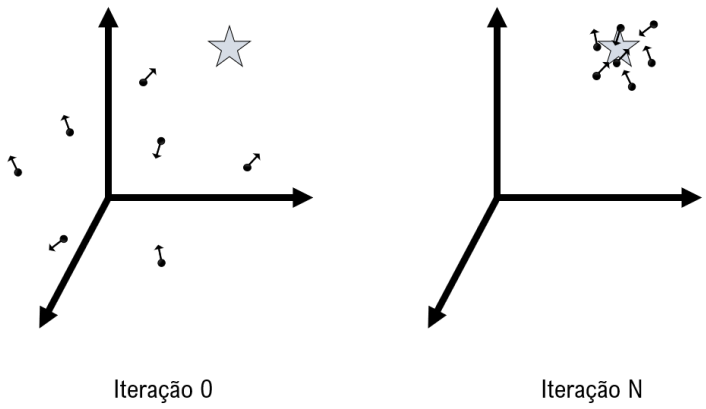


Figura 2.12: Algoritmo PSO. Adaptado de ("Particle Swarm Optimization (PSO)", s.d.).

Ao contrário dos algoritmos explorados anteriormente, não existem ascendentes e descendentes, porém cada indivíduo evolui o seu comportamento social. O algoritmo calcula o novo vetor perturbação ou velocidade para cada partícula com base na sua posição atual (existindo um conceito de inércia), na diferença entre a sua melhor posição e a atual, e ainda na diferença entre a melhor posição que qualquer partícula já obteve e a sua posição atual, possuindo estes três termos ponderações diferentes no cálculo do vetor (Selvi & Umarani, 2010). Em cada nova iteração deste algoritmo, cada um dos 3 vetores de cada partícula é substituído por uma versão alterada do mesmo, de acordo com as regras mencionadas.

2.1.2.2 Otimização por Colónia de Formigas

O algoritmo Otimização por Colónia de Formigas, em inglês *Ant Colony Optimization* (ACO), é também baseado no comportamento de uma espécie, neste caso simulando

uma colónia de formigas. O conceito base é a forma como estes insetos são capazes de encontrar o trajeto mais curto entre a sua colónia e a fonte de alimento, comunicando de forma indireta através do depósito de feromonas.

Cria-se desta maneira uma forma de *feedback* positivo, já que as formigas inicialmente escolhem um trajeto aleatório, porém as que retornam com alimento voltarão pelo trajeto inicia, depositando assim o dobro da quantidade de feromonas (Elbeltagi et al., 2005), conforme ilustrado na Figura 2.13.

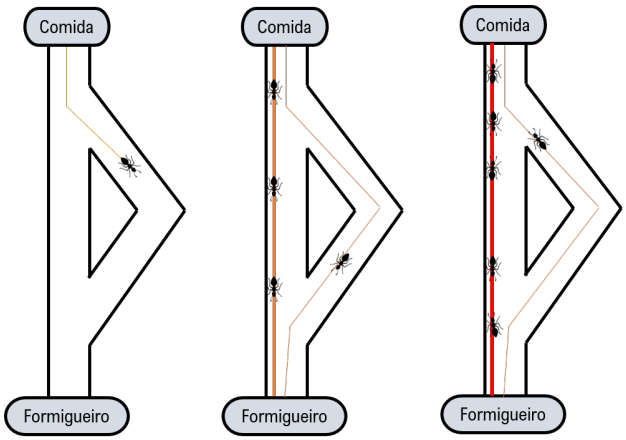


Figura 2.13: Comportamento das formigas. Adaptado de (“Wikimedia”, s.d.).

A representação de cada indivíduo é feita através de um vetor com S variáveis e cada variável armazena um conjunto de possibilidades de trajeto bem como o valor da concentração de feromonas associado a essa escolha. Uma formiga é, portanto, descrita pelo trajeto que percorre.

O algoritmo começa por inicializar um valor definido de indivíduos que são avaliados quanto à sua adaptação. Para cada formiga é determinada a melhor posição e assim é descoberto o melhor indivíduo a nível global e o trajeto de feromona atualizado.

2.2 Considerações Finais

Na tabela 2.1 estão resumidas as principais características de cada algoritmo, sendo assim possível compará-los e verificar as principais diferenças no que diz respeito a cada componente.

Verifica-se que as principais diferenças resultam da utilização de formas de representação diferentes, que condicionam os as componentes de mutação e recombinação. Além disto, os métodos de seleção de ascendentes e sobreviventes variam para alguns algoritmos.

Conclui-se assim, que a decisão sobre o algoritmo evolucionário ou de inteligência coletiva a utilizar para um determinado projeto depende bastante da aplicação destes algoritmos, nomeadamente na representação que é necessária, bem como o poder de computação disponível face ao tempo pretendido até que se dê a convergência. É muito

Tabela 2.1: Comparação de Algoritmos Evolucionários.

Algoritmo	**Algoritmo Genético (Simples)**	**Estratégias Evolutivas**	**Programação Genética**	**Algoritmo de Classificação**
Representação	Vetor Binário	Vetor Real	Árvores	Tuplos {condition :action: payoff, accuracy}
Recombinação	*One Point Crossover*	Uniforme: Discreta ou Intermediária	Troca de Subárvores	*One Point Crossover*
Mutação	Troca de Bit	Não Uniforme com *Self-Adaptation*	Redefinição Aleatória de Nó	Redefinição do Ternário
Seleção de Ascendentes	Proporcional à Adaptação	Uniforme	Proporcional à Adaptação	Proporcional à Adaptação
Seleção de Sobreviventes	Geracional	$(\mu+\lambda)$ e (μ,λ)	Geracional	Proporcional à Adaptação

pouco provável que se encontre um algoritmo que produza bons resultados em todas as aplicações. (Yuen, Chow, Zhang & Lou, 2016)

É também importante ter em conta que a maioria dos algoritmos utilizados atualmente, são novas iterações dos algoritmos expostos neste trabalho que, apesar de manterem o mesmo princípio de funcionamento, podem variar em alguns aspetos na implementação do algoritmo.

De um modo geral, os Algoritmos Genéticos são vistos como ferramentas de optimização, sendo aplicados em problemas de otimização com restrições, multimodal (que consiste na procura de todas, ou da maioria das soluções), combinatorial (como é o caso do conhecido problema "*travelling salesman*") mas também, como anteriormente referido, problemas multiobjetivo. (Slowik & Kwasnicka, 2020).

A característica mais apelativa dos algoritmos de Estratégias Evolutivas talvez seja a sua capacidade de *self-adaptation*, através da qual o algoritmo evolui ao mesmo tempo que o conjunto de soluções, o que torna estes algoritmos mais flexíveis e ideais para aplicações que requerem uma capacidade de resposta dinâmica da parte do algoritmo que procura encontrar a solução (Bäck, 2002).

Os algoritmos de Programação Genética têm como objetivo a evolução dos programas de computador em si, pela sua representação de árvore que permite a variação de cada etapa do programa. Por esta razão, estes algoritmos são usados como ferramentas de *machine learning* e podem até ser vistos como ferramentas de programação automática. Em aplicações nas quais a forma ou dimensão da solução não seja conhecida, a representação em árvore pelos algoritmos GP é ideal. (Poli, Langdon & Mcphee, 2008)

Tal como o nome indica, os Algoritmos de Classificação são usados como ferramentas de classificação, havendo também bastantes aplicações nas áreas de *data mining* (por produzirem regras fáceis de compreender), modelação e controlo (na implementação de estratégias de controlo, com regras de resposta para cada condição observada).(Bull, 2004)

No tópico de algoritmos de inteligência coletiva, os algoritmos *Particle Swarm Optimization* são fáceis de implementar, possuindo poucos parâmetros para alteração e ainda assim possibilitam uma capacidade de memória mais eficiente, já que cada indivíduo sabe e expõe o seu melhor valor. O algoritmo PSO é eficiente no que toca a manter a diversidade da população dado que cada partícula tem a capacidade de se melhorar.(Slowik & Kwasnicka, 2018)

A primeira aplicação do algoritmo *Ant Colony Optimization* foi para resolução de um problema de escolha de itinerário, e é também nessa dinâmica que surge a inspiração do mesmo, contudo o algoritmo tem sido aplicado também em problemas de escalonamento e otimização e usado para resolver problemas industriais e em redes de telecomunicações. (Dorigo, Birattari & Stützle, 2006) Uma vantagem deste algoritmo é o facto de poder alterar a solução dinamicamente de acordo com alterações no ambiente (Selvi & Umarani, 2010).

3

Estado da Arte

A computação evolucionária pode ser aplicada às mais diversas áreas, sendo que nesta dissertação o interesse é focado sobre a área da energia. Encontram-se várias aplicações desta tecnologia em tópicos inerentes à mesma. Segundo (Slowik & Kwasnicka, 2020), na base de dados *Web of Science*, a área sobre a qual existem mais artigos de investigação no tópico de aplicação de algoritmos evolucionários, é a engenharia elétrica e eletrónica, conforme ilustrado na Figura 3.1.

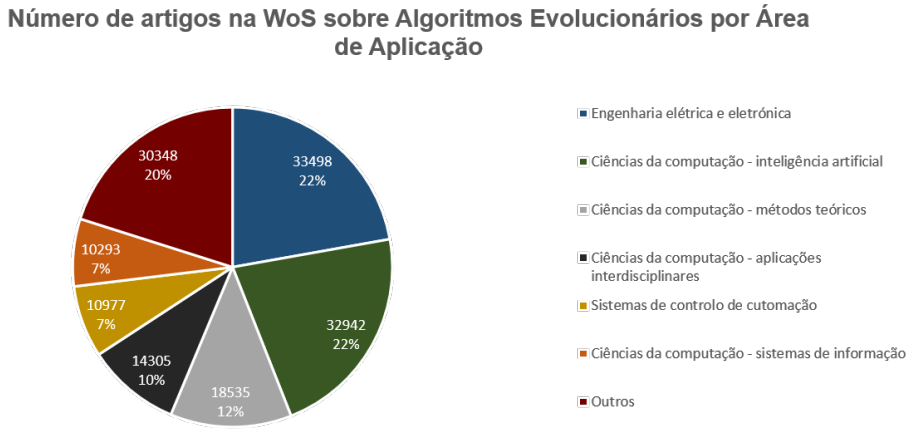


Figura 3.1: Número de artigos na base de dados WoS sobre Algoritmos Evolucionários por Área de Aplicação. Obtido de (Slowik & Kwasnicka, 2020).

Na área de *Power Systems*, em (Silva & Abrao, 2002) é resumida a utilização de algoritmos evolucionários para escalonamento de geração, modelação de carga e previsão de consumo, entre outras aplicações, referindo que existiram ainda poucos testes em grande escala, mas também que o futuro da aplicação de AE nesta área deve contemplar a combinação de vários algoritmos.

A computação evolucionária já mostrou também as suas capacidades na área de desenho de projeto, sendo (Gómez-Lorente, Rabaza, Espín & Peña-García, 2013; Gómez-Lorente, Rabaza, Estrella & Peña-García, 2013) exemplos de utilização no desenho e cálculo de iluminação para vias de trânsito, que obtém resultados superiores aos de um

programa de desenho de instalação de luminárias, já que sendo usado um algoritmo multiobjetivo é possível ter em conta o parâmetro de eficiência energética, como indicador financeiro e ambiental. Em (Tamilselvi, Baskar, Anandapadmanaban, Karthikeyan & Rajasekar, 2018) uma aplicação no dimensionamento de transformadores usa vários algoritmos, alcançando, com um destes, redução de perdas energéticas bem como um desenho melhor para o ambiente.

Nas energias renováveis e geração distribuída, em (Kusiak & Zheng, 2010) é usado um algoritmo evolucionário para otimizar a potência ativa, mas também o fator de potência de uma turbina eólica através das definições de controlo da mesma. Em (Wilson et al., 2018) é usada para otimizar o *layout* de um parque eólico.

No que toca ao transporte e distribuição de energia, é usada em (Lezama, Soares & Vale, 2019) para uso em leilões no mercado energético ao nível local, comparando vários algoritmos. O algoritmo mais eficiente demora mais tempo a convergir, porém obtém melhores resultados. Em (Lezama, Soares, Sucar, Cote & Vale, 2017) é usada Evolução Diferencial no planeamento de *smart grids* dada a natureza não linear desses problemas, estes algoritmos mostram vantagens quando comparados com outros do ramo evolucionário. O algoritmo DE obtém, contudo, resultados piores quando comparado com um algoritmo determinístico, apesar de este último demorar mais tempo.

Até aos casos de uso em controlo de equipamentos com foco na poupança de energia, como é o caso de sistemas *HVAC* (Caldas & Norford, 2003), (Kusiak, Tang & Xu, 2011) bem como na iluminação de edifícios, explorado no subcapítulo 3.1 com exemplos relacionados.

3.1 Exemplos Relacionados

Neste capítulo pretende-se expor e analisar outros projetos na área do controlo de iluminação artificial com recurso a computação evolucionária.

A aplicação de técnicas de controlo inteligente aos sistemas de iluminação não é um conceito recente e, como tal, encontram-se inúmeros artigos de investigação, bem como produtos comercializados com o objetivo de aumentar a eficiência energética e conforto na utilização de iluminação artificial. A aplicação de computação evolucionária no controlo de iluminação surge mais recentemente, sendo que a maioria dos trabalhos encontrados foram desenvolvidos nos últimos cinco anos.

Nos seguintes parágrafos são apresentados cinco artigos de investigação no tópico deste projeto e posteriormente tenta-se perceber quais as vantagens e inovações que cada solução traz.

3.1.1 Sistema de Controlo de Iluminação para Museus

Em (Viani, Polo, Anselmi, Salucci & Giarola, 2017) é desenvolvido um sistema de controlo de iluminação para um museu com o propósito de reduzir o consumo energético,

mantendo o conforto do visitante e o tendo em conta a conservação da arte. Para este fim, são recolhidas informações através de uma rede *WSAN* (*TI SensorTag*, com especificações *Zigbee*) de baixo custo em que cada *node* tem um medidor de iluminância bem como um atuador que varia a intensidade de lâmpadas *Mi-Light* com função de ajuste de luminosidade incorporada. A arquitetura do sistema implementado está descrita na Figura 3.2.

Esta informação é então enviada para um coordenador (*Raspberry Pi*) que a processa de acordo com um algoritmo evolucionário do tipo *PSO* (*Particle Swarm Optimization*), que tem em vista a minimização de uma função custo.

Este algoritmo PSO, multiobjetivo, tem em conta os níveis de iluminância desejados, que são corrigidos de acordo com a iluminância natural medida no exterior, bem como os níveis de consumo que se pretendem atingir. O utilizador deve definir 2 parâmetros que são usados como pesos para os objetivos do algoritmo - o nível de iluminância e o consumo energético.

Ultimamente é conseguida uma poupança relevante de energia, de 37%, face ao consumo que as lâmpadas teriam se fossem alimentadas à sua potência nominal.

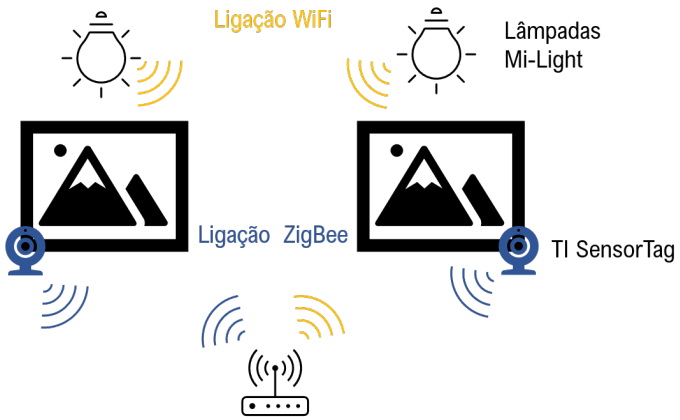


Figura 3.2: Arquitetura do Sistema de Sensores. Adaptado de (Viani, Polo, Anselmi, Salucci & Giarola, 2017).

3.1.2 Sistema de Controlo para Iluminação de um Escritório

Um sistema de controlo para poupança de energia na iluminação de um escritório é desenvolvido em (Si, Ogai, Li & Hirai, 2013), aliando redes neuronais a um algoritmo PSO para obter uma poupança de energia, tendo em conta as necessidades e o conforto do utilizador. Numa primeira etapa, várias redes neuronais calculam a contribuição das luminárias para cada área do escritório, posteriormente, essa informação é utilizada pelo algoritmo PSO para otimizar a intensidade de cada luminária face à iluminância pretendida para cada área, tendo em conta se esta está ocupada ou não.

Para o treino das redes neuronais (uma por cada lâmpada), que pode ser feito fisicamente ou através de um programa de simulação de iluminação, é definido um sensor

por cada posição em que se pretende um certo nível de iluminância. Disto resulta uma função de consumo de energia que tem em conta a contribuição de cada lâmpada para o nível de iluminância num determinado sensor, de acordo com o rácio de energia que lhe é fornecida, bem como a iluminância natural.

O funcionamento do algoritmo *Particle Swarm Optimization* é relativamente simples e está descrito na Figura 3.3. Cada partícula representa uma lâmpada, sendo o seu vetor posição o rácio de energia que lhe é fornecido e o vetor velocidade a variação instantânea para este valor. Cada partícula tem conhecimento do seu melhor valor bem como do melhor valor global. Posteriormente o algoritmo avalia cada partícula, tendo em conta uma função de custo 3.1 que será menor quanto menor for a diferença entre a iluminância em cada sensor e a pretendida, penalizando as partículas que resultem numa iluminância inferior à pretendida em alguma zona, às quais atribui um custo muito maior.

$$Custo = \sum_{i=1}^{m} |E_i - T| \times Penalty \tag{3.1}$$

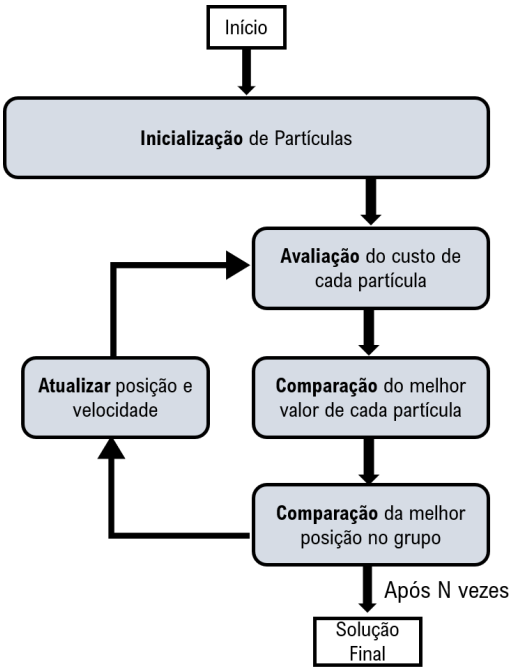


Figura 3.3: Fluxograma do algoritmo PSO. Adaptado de (Si, Ogai, Li & Hirai, 2013).

Feita a avaliação, cada partícula atualiza o seu melhor resultado caso tenha melhorado e os seus valores são comparados para atualizar o melhor valor global. Por fim os valores de posição e velocidade são atualizados e o algoritmo volta ao estado de avaliação, repetindo-se um determinado número de vezes.

O método de controlo foi testado por simulação, sendo que os valores de teste foram obtidos através de um *software* de desenho de iluminação e destas simulações resultou um valor de poupança de energia médio de 44% face à situação sem controlo de intensidade

das lâmpadas, graças ao algoritmo PSO, que manteve sempre o nível de iluminância pretendido.

3.1.3 Controlo de Iluminação para Espaços Públicos

Os autores do artigo (Petrinska, Georgiev & Ivanov, 2018) propõem igualmente uma estratégia de poupança energética baseada na consideração e aproveitamento da influência que a luminosidade natural tem na luminosidade da divisão, citando para além de uma poupança económica, também benefícios ao nível da saúde, conforto e produtividade. Utilizam para este efeito um sistema de controlo de iluminação com recurso à função de ligar/desligar, também com a possibilidade de ajuste de luminosidade (embora não tenha sido utilizado na sua simulação). Para possibilitar a medição de iluminância vertical bem como horizontal, são usados dois fotosensores.

O algoritmo é construído com o objetivo de manter a iluminância a um valor constante e definido no plano de trabalho. A iluminância necessária da parte das luminárias é calculada subtraindo ao valor pretendido, o valor da iluminação exterior (corrigida por certos fatores dependentes da altura do ano). É, então, calculado quanto cada uma das lâmpadas deve contribuir para atingir este objetivo.

No caso de um número de sensores/zonas de trabalho igual ao de luminárias (controladas individualmente), existiria apenas uma solução para atingir o objetivo, mas dado existirem mais locais de trabalho do que circuitos, é usado um algoritmo genético para encontrar soluções.

O sistema de controlo de iluminação, cuja estrutura está presente na Figura 3.4, é também auxiliado por uma base de dados, onde é armazenado o historial de controlo e estado das luminárias, bem como uma aplicação de controlo instalada num controlador físico.

Conforme a informação proveniente dos sensores, o controlador escolhe, a cada meia hora, um cenário de iluminação de entre centenas - dependentes do mês do ano, meteorologia e hora do dia. O controlador gere o sistema por apenas 6 horas diariamente, do princípio ao fim do dia de trabalho, sendo o controlo manual nas restantes horas do dia.

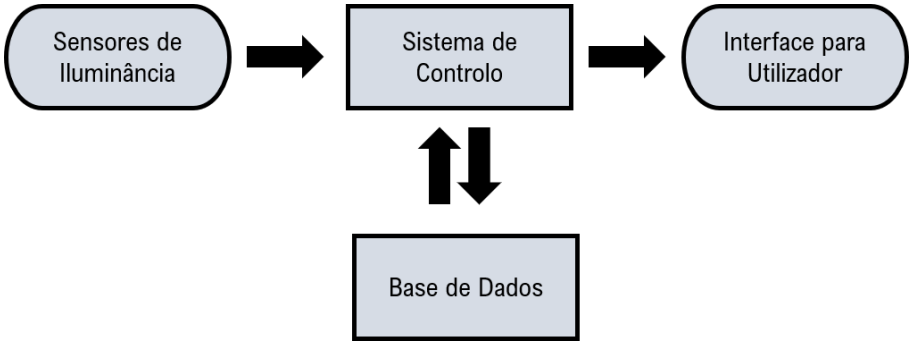


Figura 3.4: Estrutura do Sistema de Controlo. Adaptado de (Petrinska, Georgiev & Ivanov, 2018).

No caso deste trabalho, o espaço onde se deu a simulação sofreu uma remodelação que possibilitou o controlo individual, estando a cada luminária atribuído um circuito, havendo ainda assim a possibilidade de controlo manual por grupos. Contudo, os autores frisam que, sendo usadas tecnologias sem fios, é possível atingir o mesmo efeito com menor gasto financeiro.

3.1.4 Sistema de Controlo de Iluminação que considera a Luz Natural

Um sistema de iluminação através da otimização dos padrões de acendimento das luminárias é descrito em (Ngo, Nguyen, Duong & Nguyen, 2019), permitindo uma poupança de energia aliada a uma maior satisfação do utilizador. O algoritmo usado é baseado num algoritmo genético e calcula os padrões de acendimento das luminárias tendo em conta a iluminância exterior bem como a das luminárias e o nível pretendido pelo utilizador. Os autores mencionam o tempo rápido de processamento e os resultados precisos como vantagens do algoritmo genético, que não precisa de informação exata sobre a estrutura do ambiente para solucionar o problema.

Para recolha de informação sensorial são usados sensores de presença humana e de iluminância, conectados a um dispositivo central através de protocolos *WSAN* como *ZigBee* ou *Echonet Lite*. Este dispositivo é também responsável por receber do servidor ordens de ligar/desligar luminárias e enviá-las para as mesmas. Também conectado ao servidor está o controlador de iluminação que, através do algoritmo genético, calcula o nível de iluminação artificial necessário. Existe ainda uma aplicação onde o utilizador pode alterar a sua preferência de iluminação e também verificar o estado do sistema.

O algoritmo considera M luminárias com C níveis diferentes de iluminação bem como N áreas, cada área está associada a um sensor de iluminação e um de presença. São calculadas a iluminância natural numa área bem como o efeito de cada luminária nessa área.

O processo é dividido em duas partes: pré-otimização e otimização. Na primeira parte o sistema não tem conhecimento sobre o efeito de uma luminária sobre o nível de luminosidade de uma área, como tal, ao detetar a presença de alguém liga sucessivamente as várias lâmpadas até atingir o valor determinado, fazendo o cálculo da variação de luminosidade causada por cada luminária em determinada área.

Quando todos estes valores estão determinados, o sistema entra na segunda parte do processo, descrita pelo fluxograma na Figura 3.5, em que é calculada a iluminância natural de cada área.

O algoritmo genético é usado para encontrar uma solução que determina o valor de iluminância para cada lâmpada que vá ao encontro dos objetivos definidos. Os vários indivíduos gerados são avaliados por uma função objetivo 3.2 que devem minimizar.

$$F = \sum_{i=0}^{M-1} E(i) \times x(i) + \sum_{j=0}^{N-1} Pen(j) \tag{3.2}$$

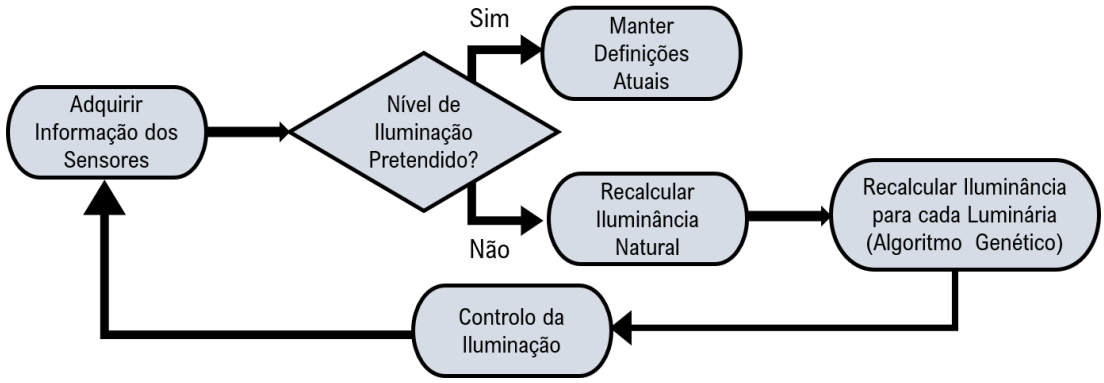


Figura 3.5: Fluxograma do Sistema implementado. Adaptado de (Ngo, Nguyen, Duong & Nguyen, 2019).

Em que o primeiro termo representa a energia despendida pelo sistema ao ligar a lâmpada i ao nível $x(i)$ e o segunda uma penalização por exceder o nível de iluminância pretendido para cada uma das áreas. Esta função objetivo beneficia um nível de iluminação que vá de encontro às preferências do utilizador em prol da poupança de energia.

Neste algoritmo cada gene de um cromossoma representa o nível de iluminação de uma luminária, sendo estes gerados aleatoriamente e dependendo da ordem de acendimento das lâmpadas no início do processo. A seleção de ascendentes é feita pela função F. Existe ainda um processo de *crossover* e mutação em que os descendentes são também avaliados e dão entrada na população no caso de serem dos melhores classificados. O processo é recursivo e termina caso após duas iterações o cromossoma mais apto for o mesmo.

O sistema é então testado, numa casa modelo de plástico com 6 kits de iluminação (luminárias com 5 níveis de controlo, sensores e módulos *Wi-Fi*) sendo o dispositivo central um *Raspberry Pi*. Uma aplicação é usada para definir o nível de iluminância pretendido por área.

Devido a limitações relacionadas com o número baixo de luminárias os autores dizem não ter sido possível testar perfeitamente o sistema, pelo que foi efetuada uma simulação que comparou o seu algoritmo a um de *Exaustive Search*, no qual comprovaram que o tempo de execução era melhor usando *AG*, principalmente quando a complexidade do sistema aumenta (mais luminárias ou níveis de iluminância), obtendo consumos de energia iguais ao segundo algoritmo.

3.1.5 Algoritmo para Controlo Inteligente de Iluminação

Em (Čongradac, Milosavljevic, Velickovic & Prebiracevic, 2012) é descrita a implementação de um algoritmo genético de controlo, utilizando o *software Matlab*. O programa é composto pela inicialização do sistema com as características da divisão e do modelo bem como parâmetros do algoritmo e também funções relacionadas com o algoritmo evolucionário (para mutação, recombinação, avaliação) e ainda uma função para controlo

ligar/desligar das luminárias.

O algoritmo inicia-se com uma população gerada aleatoriamente, sendo que cada cromossoma contém a intensidade de uma das luminárias, variando de 0 a 100%. A seleção é feita através de torneios e os indivíduos são avaliados comparando a iluminância horizontal em que resulta o seu padrão face à iluminância pretendida.

Para proceder à simulação são usados quatro fotosensores, colocados ao nível das zonas de trabalho, para a medição da iluminância média horizontal das 9 luminárias instaladas.

O algoritmo é inicializado com uma população de 50 indivíduos e corre por 500 iterações com valores de *crossing threshold* de 70% e de *mutation rate* de 30%, determinados experimentalmente, havendo mudanças a cada 10 minutos desde as oito às dezoito horas, tendo em vista um nível de 500 lux para a iluminância da divisão.

Os autores concluem que o algoritmo genético resulta numa poupança de 63% face à utilização normal do sistema de iluminação, contudo resulta numa poupança ligeiramente inferior à que seria obtida por um sistema ligar/desligar. O algoritmo teve um erro de regulação médio de 8 lux, o que é bastante satisfatório.

Com vista em melhorar o modelo em até 20%, é recomendada a instalação de sensores de presença dada a relevância para o algoritmo de saber se estão presentes pessoas na divisão.

3.2 Considerações Finais

Em relação aos artigos apresentados, estão na tabela 3.1 resumidas e agregadas as principais características de cada um, de modo a auxiliar a comparação entre estes.

Tal como na maioria da literatura encontrada, a tabela 3.1 reflete o quão mais prolíficos os algoritmos genéticos são no que toca as aplicações de computação evolucionária na resolução de problemas reais. Segundo (Slowik & Kwasnicka, 2020), cerca de 80% dos artigos que aplicam computação evolucionária na área de engenharia elétrica e eletrónica fazem-no através de algoritmos genéticos.

Uma característica relevante nos artigos acima expostos é a forma como obtêm dados e como recebem o *feedback* das alterações que provocam no ambiente. Apenas o último opta por fazê-lo através de um modelo da divisão, o que torna o sistema menos prático de instalar noutra localização, bem como aumenta o poder de computação necessária para o modelo da divisão.

A maioria destes artigos refere como motivação o objetivo de manter ou aumentar o conforto do utilizador do sistema paralelamente à poupança de energia, este é, como já foi referido anteriormente, um fator muito relevante no dimensionamento destes sistemas não só para que seja útil mas também tendo em conta os efeitos que poderá ter na produtividade e saúde do utilizador.

Outra característica, que em todos os artigos é implementada ou planeada para futuros trabalhos, é a do aproveitamento de luz natural, considerando a sua influência na

Tabela 3.1: Comparação de Artigos Mencionados.

Artigo	**Algoritmo Utilizado**	**Recolha de Informação**	**Tipo de Controlo**	**Poupança Energética**
Secção 3.1.1 (Viani, Polo, Anselmi, Salucci & Giarola, 2017)	PSO	Sensores de Iluminância	Variação de Intensidade	37%
Secção 3.1.2 (Si, Ogai, Li & Hirai, 2013)	Redes Neuronais + PSO	Sensores de Presença e Iluminância	Variação de Intensidade	44%
Secção 3.1.3 (Petrinska, Georgiev & Ivanov, 2018)	AG	Sensores de Iluminância	Ligar/Desligar	—
Secção 3.1.4 (Ngo, Nguyen, Duong & Nguyen, 2019)	AG	Sensores de Presença e Iluminância	Ligar/Desligar	—
Secção 3.1.5 (Čongradac, Milosavljevic, Velickovic & Prebiracevic, 2012)	AG	Sensores de Iluminância	Variação de Intensidade e Ligar/Desligar	63%

luminosidade no interior do edifício, sendo que esta não tem custo e se for tida em conta pelos algoritmos, pode permitir um menor consumo em iluminação artificial.

4

Arquitetura do Sistema

Pretende-se neste capítulo descrever a arquitetura a adotar no processo de desenvolvimento do sistema, apresentando as opções a considerar.

O objetivo desta dissertação é automatizar o controlo dos circuitos de luminárias de um determinado espaço, de modo otimizado, pretendendo-se reduzir o consumo de energia, mantendo um nível de luminosidade na divisão de acordo com o pretendido pelo utilizador.

De modo a alcançar o objetivo, o sistema deve ser composto por uma rede de sensores para aquisição de dados e atuadores para controlo dos circuitos. Relativamente à componente de decisão, o sistema deve possuir um algoritmo genético, bem como algoritmos de previsão que permitam a este avaliar as possíveis soluções para controlo do sistema, fornecendo dados de previsão acerca do efeito de variáveis como a taxa de presença e nível de luminosidade.

Paralelamente, deverá ser implementado um sistema de monitorização, que detete possíveis problemas nos circuitos de iluminação.

Os traços gerais da arquitetura do sistema estão expostos na Figura 4.1, que inclui as várias componentes do sistema, organizadas pelas três categorias de camada física, camada de decisão e ainda interface com o utilizador, bem como a troca de informação bidirecional entre as várias categorias.

4.1 Camada Física

Nesta secção pretende-se expor a arquitetura física do sistema, no que diz respeito ao tipo de dispositivos a usar, como sensores e atuadores, mas também os protocolos a utilizar para permitir a comunicação entre estes e o controlador. Na Figura 4.2 está exposta uma representação simplificada desta arquitetura.

4.1.1 Sensores e Atuadores

Na constituição do sistema de aquisição de dados, é necessária a instalação de sensores de movimento, que recolham informação binária relativamente à presença de pessoas

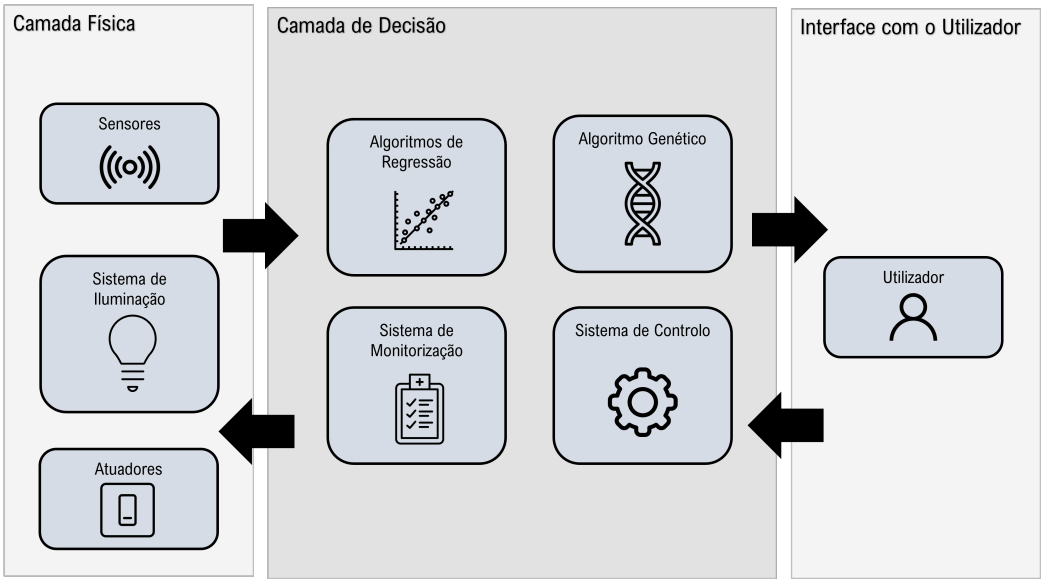


Figura 4.1: Arquitetura do sistema.

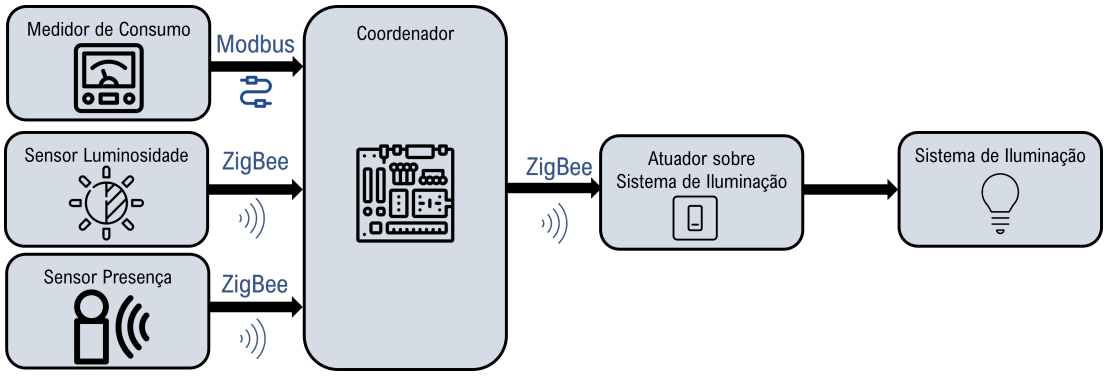


Figura 4.2: Arquitetura da Camada Física do Sistema.

no espaço, bem como sensores de luminosidade. Para obter informação em relação ao consumo de energia, devem ser instalados medidores do consumo de energia, capazes de medir grandezas como energia e potência de cada um dos circuitos de iluminação.

De modo a controlar as luminárias, são necessários atuadores, e é preferencial que estes se conectem ao restante sistema através de uma ligação sem fios, de modo a tornar a sua instalação mais acessível. Estes atuadores não necessitam de ter a capacidade de regulação de intensidade, visto que as lâmpadas no espaço não possuem este tipo de funcionalidade.

4.1.2 Protocolos e Especificações de Comunicação

Para efetuar a transferência de dados entre estes diferentes sensores e o computador responsável pela recolha dos dados, bem como a comunicação deste computador com os atuadores que controlam a iluminação, é necessário estabelecer canais de comunicação entre os vários dispositivos. Para atingir este fim, podem ser usados vários protocolos de

comunicação, como é o caso dos protocolos Zigbee, MQTT e ainda Modbus, descritos nos parágrafos seguintes.

Zigbee

O protocolo de comunicação sem fios Zigbee, estabelecido pela norma IEEE 802.15.4, e criado pela Zigbee Alliance, tem como propósito a comunicação de baixo custo e baixo consumo energético entre vários nós de uma rede, a curtas distâncias. Numa rede deste protocolo, os nós podem ser dispositivos como sensores ou atuadores.(Wang, He & Wan, 2011)

Numa rede Zigbee existe um coordenador, vários clientes e possivelmente, *routers*. São possíveis três topologias de rede: estrela, árvore e malha, conforme ilustrado na Figura 4.3. Numa topologia em estrela, existe um dispositivo chamado coordenador que configura a ligação com os restantes nós, que comunicam diretamente com ele. Nas redes em topologia de árvore ou malha, existem dispositivos chamados *routers*, que permitem extender a rede. (Alliance, s.d.)

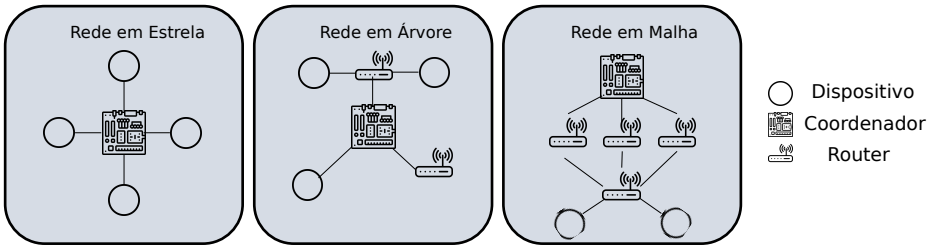


Figura 4.3: Topologias Zigbee. Adaptado de (Wang, He & Wan, 2011).

A especificação Zigbee funciona tipicamente na frequência de 2.4GHz e é usada extensivamente em sistemas de monitorização e aquisição de dados, pelas características descritas acima.

O protocolo Zigbee tem bastantes vantagens em relação a outros conhecidos protocolos de comunicação, como Wi-Fi, Bluetooth, LoRa, ou Z-Wave, sendo uma delas o custo, já que os dispositivos Zigbee estão disponíveis no mercado a custos acessíveis, bem como pela sua utilização extensiva neste contexto de monitorização e controlo, o que permite o acesso a um maior leque de dispositivos.

MQTT

O protocolo de mensagens MQTT, sigla de *Message Queuing Telemetry Transport*, que funciona numa lógica publicador/subscritor, é um protocolo leve e código aberto, tipicamente usado em comunicação máquina para máquina num contexto industrial, ou entre dispositivos no contexto de Internet das Coisas.

No protocolo MQTT, que tipicamente corre sobre redes TCP-IP, cada cliente pode publicar ou subscrever a determinados tópicos, que funcionam como canais com um

endereço específico, conforme ilustrado na Figura 4.4. A entidade responsável por receber a informação enviada pelos publicadores e por entregá-la aos subscritores de um determinado tópico, chama-se *MQTT Broker*. (OASIS, s.d.)

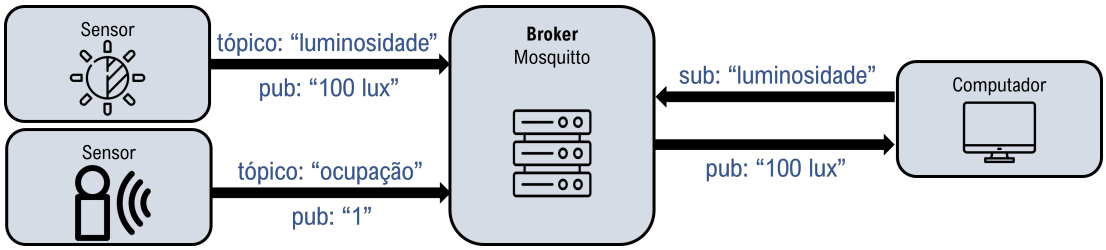


Figura 4.4: Estrutura publicador/subscritor do protocolo MQTT.

Existe, no entanto, a possibilidade de utilizar o protocolo MQTT sobre Zigbee, de modo a garantir uma comunicação fiável entre os sensores de ocupação e luminosidade e os interruptores com o computador.

Modbus

O protocolo de comunicação Modbus, criado em 1979 pela Modicon (agora Schneider Electric), estabelece uma relação cliente-servidor entre dois dispositivos inteligentes. (“Modbus”, s.d.)

Existem dois tipos de comunicação possíveis com o protocolo Modbus, sendo um deles série, utilizando a norma de comunicação RS-485, para conectar os dispositivos a um controlador principal (mestre), mas também Modbus para redes TCP/IP. Funcionando numa lógica Mestre-Escravo, o mestre faz um pedido de informação ao dispositivo escravo, que a produz e envia.

Este protocolo é principalmente usado no contexto industrial, sendo, mais recentemente, também utilizado em sistemas de monitorização e controlo em edifícios. (Ahmad, Mourshed, Mundow, Sisinni & Rezgui, 2016)

4.2 Camada de Decisão

Nesta secção pretende-se expor a arquitetura de decisão do sistema, no que toca aos algoritmos de inteligência artificial a utilizar para obter dados que permitam uma maior eficiência, bem como o controlo do sistema de iluminação. A arquitetura da camada de decisão está representada na Figura 4.5.

O sistema de decisão deve receber como entrada as seguintes variáveis, que utiliza para, em cada intervalo do dia, obter a melhor solução de controlo para as luminárias:

- Mês
- Dia da semana
- Hora do dia

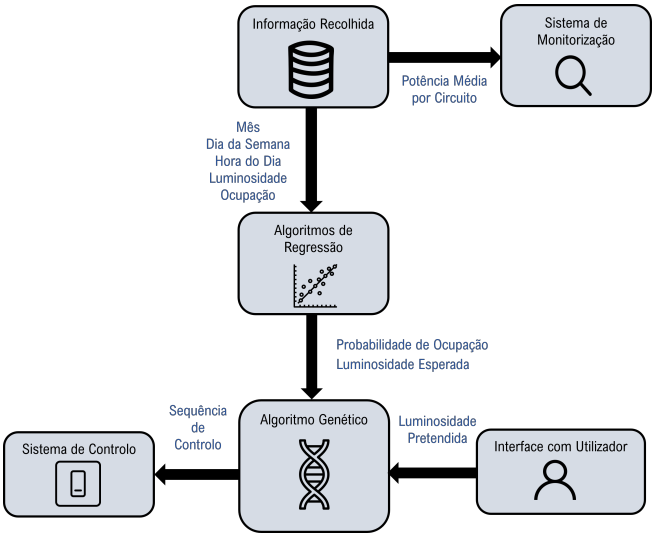


Figura 4.5: Arquitetura da camada de decisão do sistema.

- Luminosidade
- Rácio da ocupação face à ocupação máxima na última semana
- Número de circuitos de iluminação ligados
- Potência média
- Luminosidade pretendida para períodos diurnos e nocturnos

4.2.1 Algoritmos de Regressão

Numa primeira etapa, devem existir dois algoritmos de regressão, a processar os dados históricos recolhidos, das variáveis mencionadas acima. Estes algoritmos, do tipo Árvore de Decisão, devem gerar dois modelos:

- **Previsão da Ocupação:** A partir dos dados históricos de ocupação, prevê a ocupação esperada num determinado intervalo de tempo, tendo em conta o dia da semana.
- **Previsão da Luminosidade:** Este modelo é treinado com dados relativos ao mês do ano, hora do dia, circuitos de iluminação e luminosidade. Permite fazer uma estimativa da luminosidade numa determinada hora, considerando o número de circuitos ligado e o mês do ano.

4.2.2 Algoritmo Evolucionário

O algoritmo evolucionário, cujo fluxograma está representado na Figura 4.6, tem como objetivo a otimização das luminárias ligadas em cada intervalo de tempo. Para este efeito, utilizará informação relativa à ocupação e à luminosidade, proveniente dos modelos citados acima.

Devido à estrutura do problema, que consiste em definir o estado de vários circuitos de iluminação ao longo de um dia, o mais indicado é optar-se por utilizar um algoritmo genético de representação binária, pela sua estrutura simples e adequada.

Cada indivíduo deve ser definido como um vetor booleano, cujo número de índices será igual ao número de circuitos de iluminação associados ao sistema, considerados genes. Cada gene pode ter o valor de 0 ou 1 e isto representa o facto de um determinado circuito estar ou não ligado.

4.2.3 Sistema de Controlo de Iluminação

Ao princípio de cada hora, o vetor solução do algoritmo genético será lido, sendo esta componente responsável por ligar ou desligar os circuitos conforme a informação produzida pelo algoritmo genético. No caso de ser detetada a presença de alguém no espaço durante o período noturno definido, e caso o vetor de controlo indique que todos os circuitos de iluminação devem estar desligados, um dos circuitos deverá ser ligado, de modo a garantir o conforto do utilizador. Na Figura 4.7, está representado o fluxograma do sistema de controlo.

4.2.4 Sistema de Monitorização

O sistema deverá ainda possuir uma componente de monitorização, pretendendo-se que esta reconheça possíveis avarias nos circuitos de iluminação. O processo de deteção de avarias consiste na comparação de dados referentes ao consumo energético de cada um dos circuitos num determinado intervalo, com o valor esperado de consumo. No caso de deteção de falha, o sistema deve gerar uma mensagem, que poderá ser visualizada na secção "Avisos", da interface com o utilizador. O arquitetura do sistema de monitorização está explicitada na forma de fluxograma na Figura 4.8.

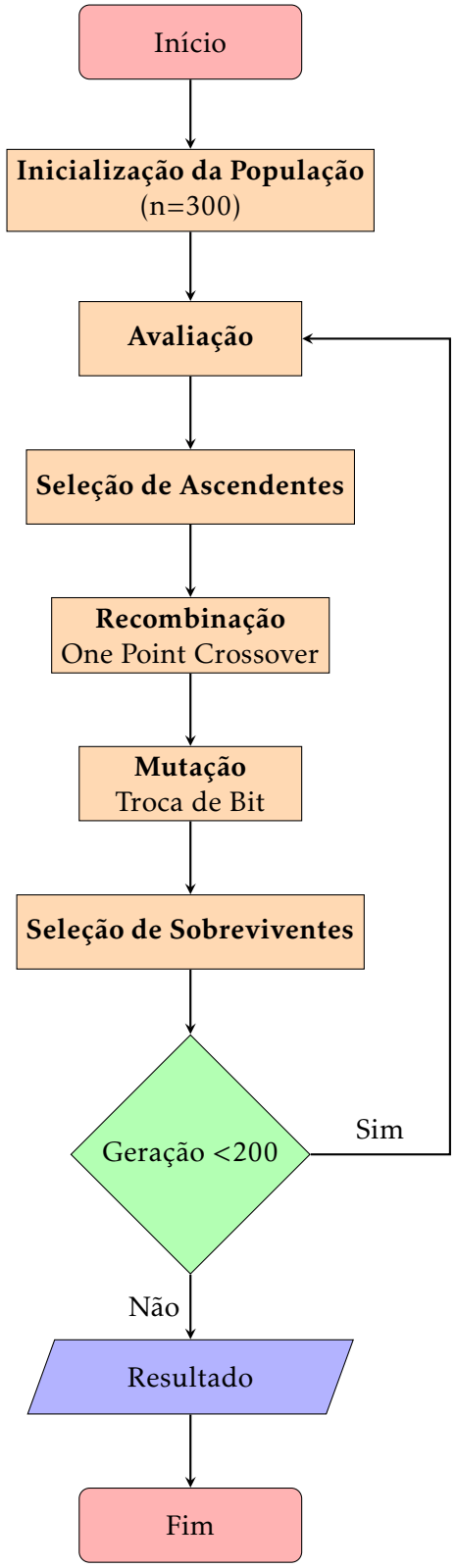


Figura 4.6: Fluxograma do algoritmo genético.

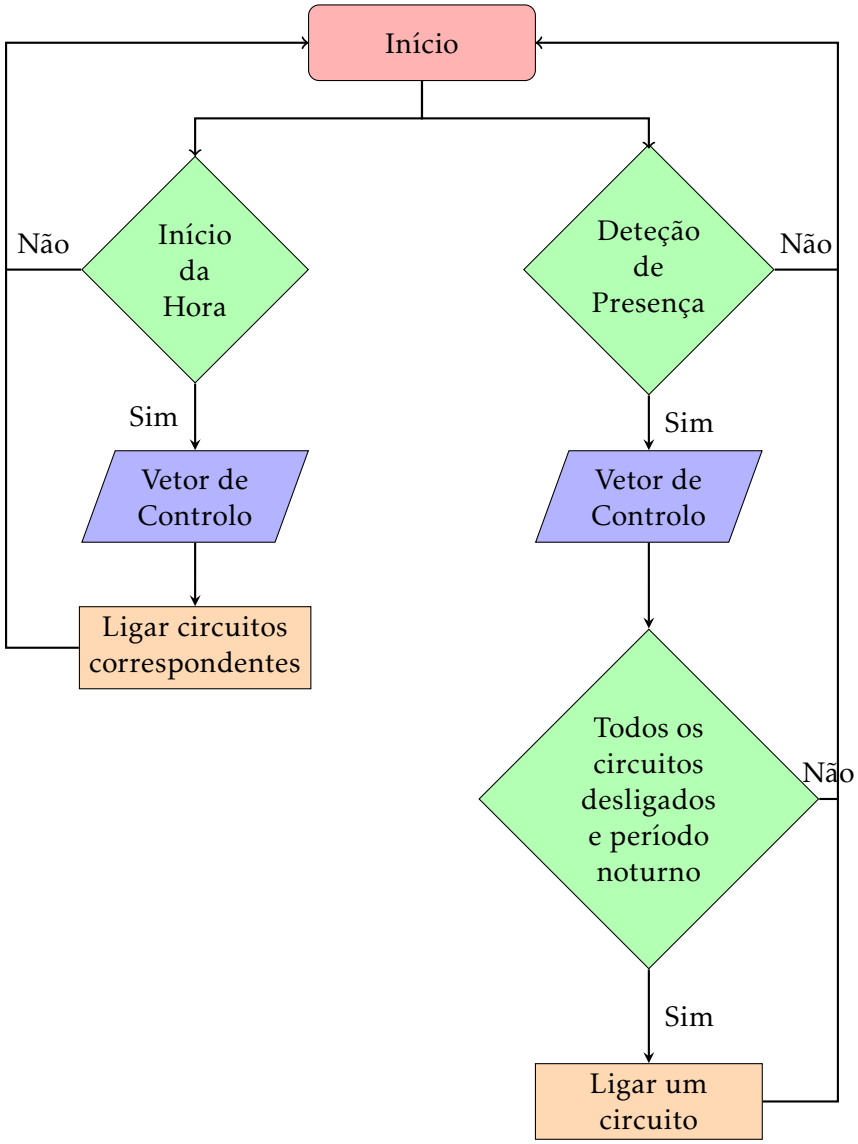


Figura 4.7: Fluxograma do sistema de controlo.

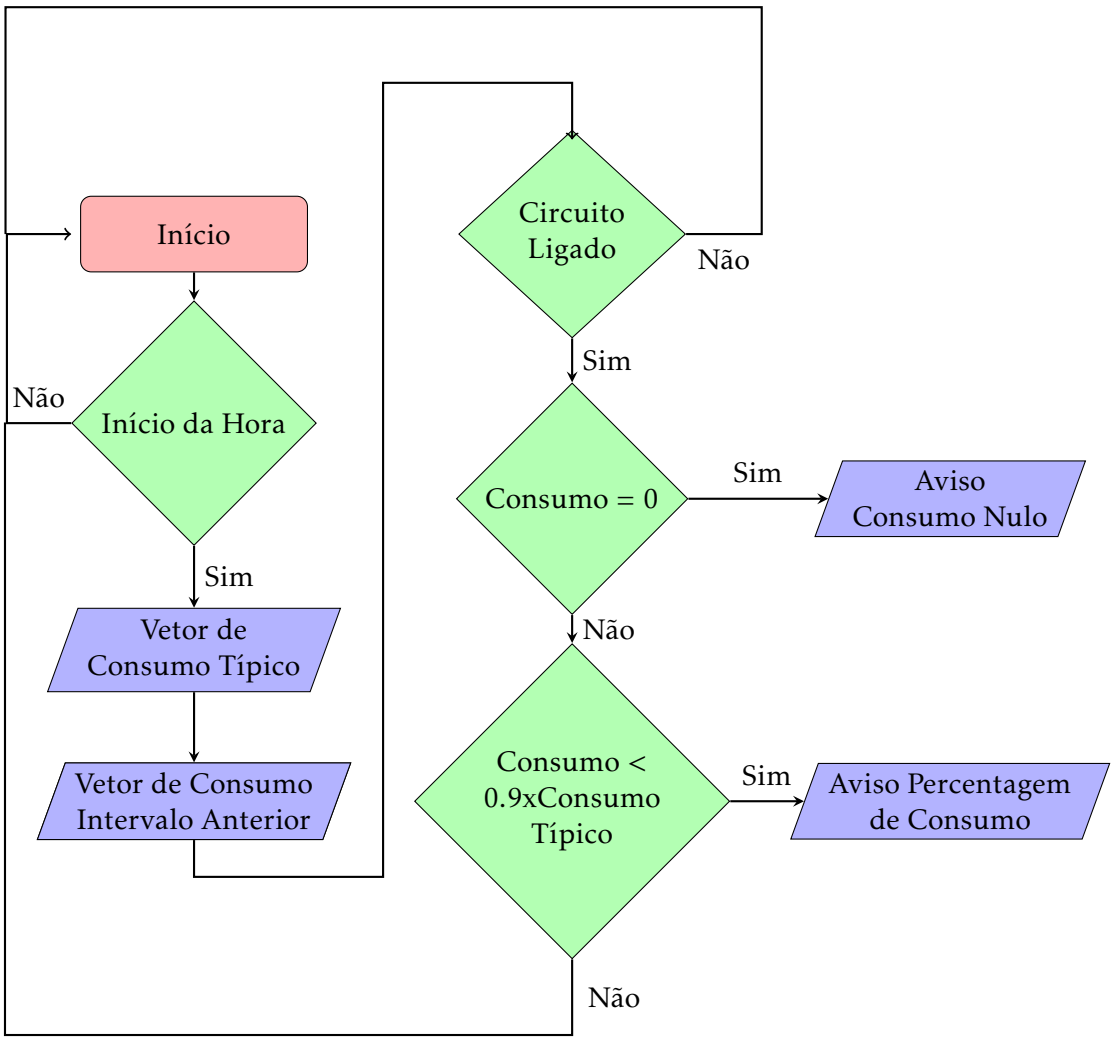


Figura 4.8: Fluxograma do sistema de monitorização.

5

Implementação do Sistema

Neste quinto capítulo, é descrita a implementação das diferentes partes do sistema. Este é implementado utilizando um computador Raspberry Pi, onde é instalado um sistema operativo orientado à automação doméstica, que permite a integração de diferentes tecnologias, oferecendo um maior número de possíveis funcionalidades. Os sensores de iluminação, presença e consumo de energia, bem como os interruptores sem fios estão conectados a este computador, onde correm os programas necessários para as funcionalidades do sistema.

Num primeiro subcapítulo, a instalação onde o sistema foi implementado e testado é descrita. De seguida é feita a descrição da implementação de *hardware*, contemplando os sensores, atuadores e coordenador Zigbee do sistema. O terceiro subcapítulo foca-se no desenvolvimento da camada de software, englobando o sistema operativo, os algoritmos *machine learning* de regressão, o algoritmo evolucionário a partir do *framework* DEAP e ainda os sistemas de controlo e monitorização, bem como a interface gráfica com o utilizador.

5.1 Caracterização da Instalação

A instalação onde a dissertação foi desenvolvida é o corredor do segundo piso do Departamento de Engenharia Eletrotécnica e de Computadores da NOVA School of Science and Technology. O computador, bem como os interruptores e os medidores de consumo são instalados adjacentes a um dos quadros elétricos deste piso.

O corredor possui quatro pontos de entrada ou saída e é servido por 54 luminárias equipadas com lâmpadas de tecnologia LED, distribuídas por 4 circuitos de iluminação, divididos em pares por dois quadros elétricos. Com estas luminárias não existe a possibilidade de controlo da luminosidade e a distribuição de circuitos é feita de forma alternada, não havendo um circuito por zona.

Anteriormente, o controlo de iluminação era feito por um temporizador, que entretanto deixou de estar funcional, sendo o controlo manual até à instalação do sistema. Os

dois extremos do corredor foram convertidos em pequenos espaços de estudo, com algumas mesas e cadeiras, pelo que é relevante preservar uma boa qualidade de iluminação, principalmente nestes espaços.

Apesar do sistema ter sido inicialmente dimensionado para três circuitos de iluminação, foi posteriormente adaptado para servir dois circuitos, estando assim associado a apenas um quadro elétrico.

5.2 Equipamentos Utilizados

Neste subcapítulo é descrita a implementação do sistema em relação aos equipamentos. Nesta categoria estão incluídos os sensores para formar uma rede de aquisição de informação, bem como os atuadores sobre as luminárias e o coordenador da rede.

Na Figura 5.1 está uma imagem dos equipamentos posteriormente instalados junto ao quadro elétrico, designadamente: o Raspberry Pi, os relés Zigbee e os medidores de consumo, conectados por RS-485 ao computador. Na Figura 5.2 esta montagem adjacente ao quadro elétrico está esquematizada.

Figura 5.1: Instalação dos equipamentos adjacentes ao quadro elétrico.

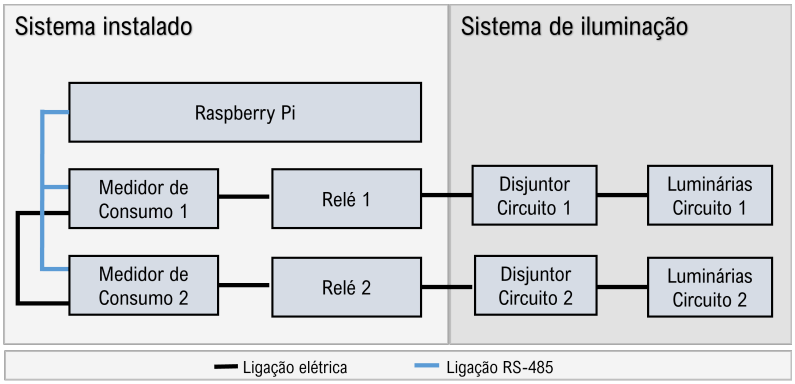


Figura 5.2: Esquemático das ligações físicas dos equipamentos adjacentes ao quadro elétrico.

5.2.1 Luminárias

O corredor é servido por 54 luminárias, equipadas com lâmpadas LED. As luminárias estão distribuídas por 4 circuitos, estando estes divididos em pares por 2 quadros elétricos. As lâmpadas são alimentadas a tensão de 230V.

Apenas 2 dos circuitos, pertencentes a um dos quadros elétricos, são afetados pelo sistema de controlo, mantendo-se os restantes sob controlo manual.

5.2.2 Raspberry Pi

O Raspberry Pi Model 3B+, na Figura 5.3, é um computador de placa única (ou *Single Board Computer - SBC*), sendo assim uma opção de baixo custo e alta praticabilidade para projetos em fase de protótipo. Neste sentido, foi usado nesta tese como o coordenador da rede Zigbee, albergando também o sistema operativo *Home Assistant*, responsável pela integração dos vários equipamentos e onde correm os sistemas de controlo e monitorização.

Figura 5.3: Raspberry Pi Model 3B+. Retirado de ("Wikimedia: Raspberry Pi", s.d.).

Ao Raspberry Pi são conectados dois tipos de periféricos: um dongle Zigbee (Sonoff C2531), que permite a interação do computador com os dispositivos Zigbee, criando uma rede local nesse protocolo, e ainda dois adaptadores RS-485 para USB, necessários para a transmissão de informação dos medidores de consumo de energia para o coordenador, criando 2 redes de protocolo Modbus.

5.2.3 Sensor de Luminosidade

Para aquisição de informação sobre a luminosidade na divisão, foram usados sensores de luminosidade *Mi Light Detection*, exposto na Figura 5.4. Este sensor mede periodicamente a quantidade de Lux no espaço e comunica esta informação, através do protocolo Zigbee ao coordenador da rede. Foram usados, no corredor onde a tese foi desenvolvida, 4 destes sensores.

Figura 5.4: Sensor Mi Light Detection. Retirado de ("PcDiga: DetetorLux", s.d.).

5.2.4 Sensor de Presença

De modo a analisar a presença de pessoas no corredor, e deste modo estimar a afluência em cada intervalo do dia, foram usados sensores de presença de tecnologia infravermelhos. O modelo usado foi um modelo *Xiaomi Mi Motion Sensor*, como demonstrado na Figura 5.5. Estes sensores são também colocados ao longo do corredor e passam a informação recolhida, de modo binário, para o coordenador Zigbee.

Figura 5.5: Sensor de Movimento Xiaomi Mi Motion Sensor. Retirado de ("PcDiga: Sensor de Presença", s.d.).

5.2.5 Medidor de Consumo Energético

Adjacente ao quadro elétrico, ligado aos circuitos referentes à iluminação, é usado um medidor de consumo energético. Este sensor é capaz de fazer várias leituras de grandezas elétricas, nomeadamente de potência e energia. O sensor escolhido foi o modelo Orno OR-WE-504, representado na Figura 5.6. Este medidor de consumo monofásico, utiliza o protocolo RS485 para comunicação, pelo que é conectado ao sistema de aquisição de dados através de um conversor USB para RS485.

Figura 5.6: Medidor digital de consumo de energia. Retirado de ("Orno: Medidor de Consumo", s.d.).

5.2.6 Relé Zigbee

Para servir de atuador sobre as luminárias, dada a impossibilidade de controlo de intensidade, foi escolhido um relé sem fios, modelo Sonoff BasicZBR3, demonstrado na Figura 5.7. Instalando 2 interruptores adjacentes ao quadro elétrico, a jusante dos medidores de consumo, é possibilitado o controlo ligar/desligar para cada grupo de lâmpadas da instalação. Este interruptor comunica também através do protocolo Zigbee com o Raspberry Pi.

Figura 5.7: Relé Zigbee Sonoff BasicZBR3. Retirado de ("Sonoff BasicZBR3 - Interruptor Zigbee", s.d.).

5.3 Implementação de *Software*

Expõe-se neste subcapítulo a implementação da dissertação ao nível computacional. Para este tópico é relevante a descrição do programa de automação residencial utilizado, bem como de integrações que este possibilita, de outras aplicações. É ainda discutido o *framework* da linguagem Python usado para desenvolver um algoritmo evolucionário, bem como a implementação dos algoritmos de regressão e ainda dos sistemas de controlo e monitorização. Por fim, é exposta a implementação da interface com o utilizador.

5.3.1 Sistema Operativo *Home Assistant*

O *Home Assistant* é um programa de automação residencial gratuito e de código aberto, que permite a integração das várias tecnologias usadas atualmente no contexto da Internet das Coisas. Como tal, é capaz de interagir com dispositivos utilizando vários protocolos, nomeadamente: Bluetooth, MQTT, Zigbee, Z-Wave e Wi-Fi. Este software pode ser instalado como sistema operativo e acedido noutro computador através de uma interface web, representada na Figura 5.8.

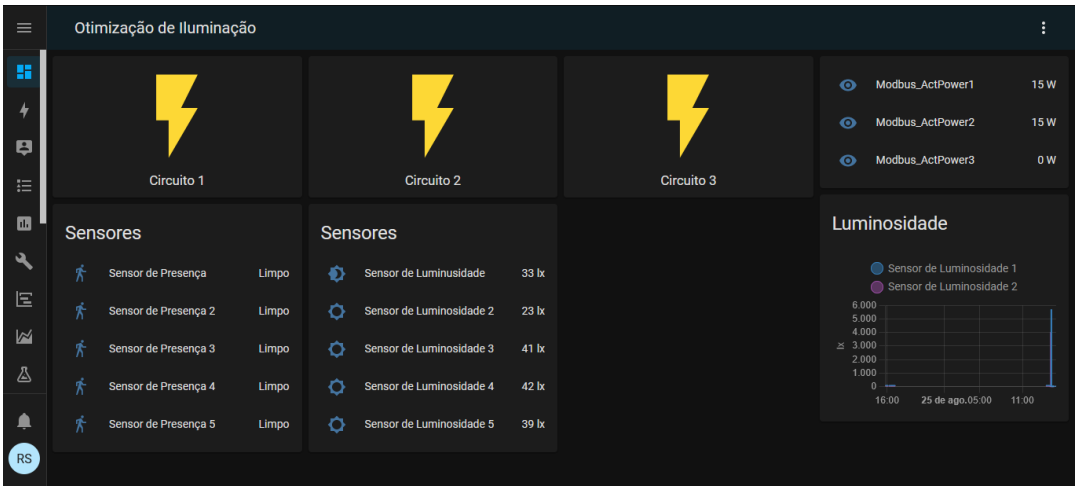


Figura 5.8: *Frontend* do Sistema *Home Assistant*.

O uso do sistema *Home Assistant* no contexto desta dissertação prende-se com a integração de produtos de marcas distintas, ou que usem diferentes tecnologias, aumentando as possibilidades na escolha de sensores e atuadores e a simplicidade na configuração dos equipamentos.

Para além disto, o programa possui também integrações de outras aplicações, como é o caso de *AppDaemon* e *JupyterLab*, que permitem o desenvolvimento de aplicações Python para o controlo dos equipamentos. Foram ainda usadas as integrações com o sistema de bases de dados InfluxDB, para guardar informação histórica dos sensores e o *addon* Grafana, que possibilita a construção de gráficos com esses dados, bem como a integração Node-RED, uma ferramenta de programação em fluxo, que foi utilizada para organizar a informação adquirida.

Integração *Zigbee2MQTT*

O *add-on* Zigbee2MQTT, instalado no sistema Home Assistant, permite a comunicação com alguns dispositivos Zigbee sem a necessidade de utilização do dispositivo coordenador de cada uma das marcas, permitindo assim a utilização de equipamentos de marcas diferentes. Para isto, a integração utiliza a antena USB Zigbee adquirida.

Assim, a comunicação com os sensores e atuadores é feita através do protocolo de rede MQTT. Este é um protocolo destinado ao uso em comunicações de máquina para

máquina, no contexto da internet das coisas, usando pouca banda larga. O protocolo MQTT funciona com um modelo de publicação-subscrição e tipicamente corre sobre redes TCP/IP, embora aqui seja adaptado para o uso numa rede Zigbee.

Integrações *AppDaemon* e *JupyterLab*

Os *addons AppDaemon* e *JupyterLab* foram usados essencialmente para executar *scripts* na linguagem Python, sendo que o primeiro foi usado para a execução do algoritmo genético, usando a biblioteca DEAP, descrita abaixo, bem como o painel de controlo para o utilizador, enquanto que o segundo *addon* foi utilizado para treinar e produzir os resultados dos algoritmos de regressão, utilizando a biblioteca *Scikit Learn*, colocando os resultados em ficheiros acessíveis aos *scripts* que correm no *AppDaemon*. A utilização de dois *addons* prende-se com a impossibilidade de importar certas bibliotecas Python ao usar a integração *AppDaemon*, enquanto que com *JupyterLab* não é possível controlar os circuitos de iluminação ou executar os *scripts* automaticamente e periodicamente.

5.3.2 Implementação dos Algoritmos de Regressão

Existem vários algoritmos de *machine learning* cujo objetivo é gerar árvores de decisão, estes procuram encontrar condições que relacionem determinados valores de variáveis de entrada com resultados nas variáveis de saída. ("Scikit Learn - Decision Trees", s.d.) São exemplos destes algoritmos o ID3 e C4.5 que, por usarem variáveis discretas, são considerados algoritmos de classificação. No caso da implementação, é usado o algoritmo CART (do inglês *Classification and Regression Trees*), que pode ser usado para esse efeito, mas também aplicado a variáveis contínuas, sendo assim considerado também um algoritmo de regressão.

Com as árvores de regressão, o objetivo passa por combinar o conceito de árvore de decisão com o de regressão linear, havendo um processo de indução através de partições recursivas das amostras. Este tipo de algoritmos é frequentemente utilizado na previsão de consumo de energia.(Rokach & Maimon, 2014)

A sua representação em árvore resulta em modelos facilmente interpretáveis, nomeadamente para árvores de dimensão pequena, para humanos e computadores. Além disto, por ser o algoritmo a escolher a importância que dá a cada uma das variáveis de entrada, este é capaz de excluir autonomamente variáveis de previsão que se mostrem irrelevantes.(Hastie, Tibshirani & Friedman, 2009)

Os algoritmos de regressão foram implementados utilizando a biblioteca Python *Scikit Learn* ("Scikit Learn", s.d.), mais especificamente a sua classe *DecisionTreeRegressor*. O modelo para previsão da ocupação é um modelo bi-dimensional, já que recebe informação acerca do dia da semana e o intervalo do dia para prever um valor de ocupação. Na Figura 5.9 está representado um exemplo de árvore de decisão gerada por este algoritmo. Já o modelo de previsão de luminosidade recebe informação recebe dados de 3 fontes: o mês

do ano, o intervalo do dia e o número de circuitos de iluminação ligados, para prever um valor de luminosidade.

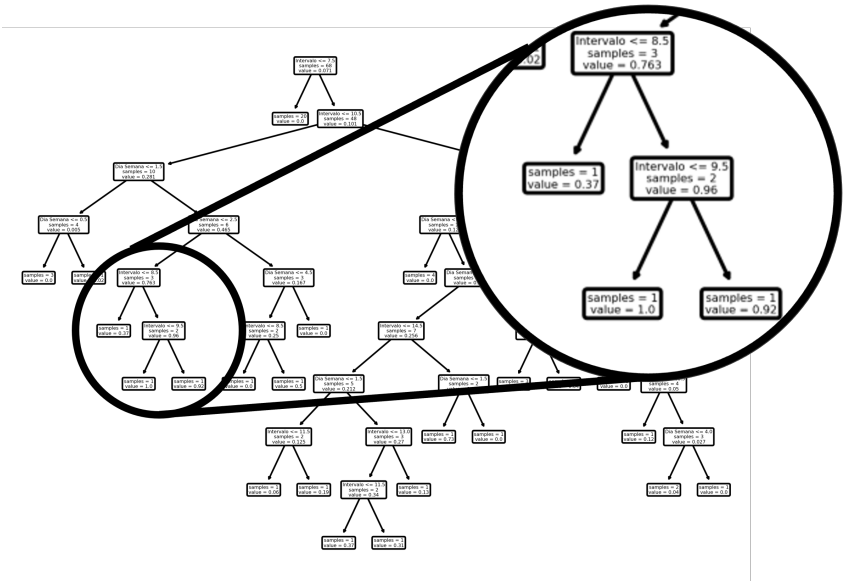


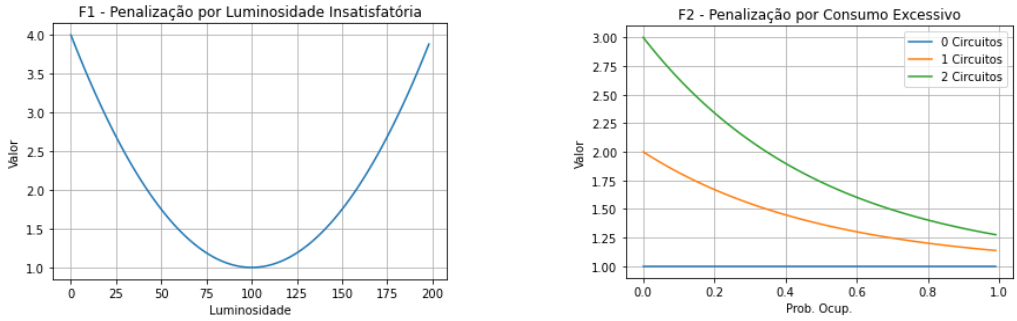
Figura 5.9: Exemplo de Árvore de Decisão gerada pelo algoritmo de previsão de ocupação.

5.3.3 Implementação do Algoritmo Genético

O algoritmo genético foi desenvolvido na linguagem Python, recorrendo à biblioteca DEAP (*Distributed Evolutionary Algorithms in Python*) ("Documentação DEAP", s.d.), que permite a prototipagem rápida com algoritmos evolucionários, oferecendo várias funções já definidas para as várias componentes de um algoritmo.

O *script* desenvolvido é baseado no exemplo de algoritmo genético binário disponível em ("Exemplo DEAP - Algoritmo Genético", s.d.), com algumas alterações de modo a adaptar ao problema a resolver, principalmente no que toca à função de avaliação e à estrutura que o código deve ter para ser utilizado como aplicação *AppDaemon*. O algoritmo genético é executado 5 minutos antes do início de cada hora, de modo a gerar um vetor de controlo que o sistema possa utilizar no início da hora seguinte.

A evolução dá-se ao longo de 200 gerações e para cada intervalo, é seleccionado um indivíduo da população, que corresponde a um vetor binário de 3 índices, através de um processo de avaliação cuja função é a exposta na Equação 5.3 e representada graficamente na Figura 5.11, que se pretende minimizar. A função, que resulta da multiplicação das equações 5.1 e 5.2, representadas graficamente na Figura 5.10, utiliza as variáveis $Lum_{Pretendida}$, que representa um valor definido pelo utilizador e $Lum_{Esperada}$, valor obtido do algoritmo de regressão, adaptado ao intervalo e número de circuitos ligados. O número de circuitos de iluminação ligados é representado por $N_{circuitos}$ e $P_{ocupação}$ é o valor obtido através do algoritmo de regressão, que prevê a taxa de ocupação num determinado intervalo. Esta função tem ainda uma componente de penalização, sendo que a



(a) Função F1

(b) Função F2

Figura 5.10: Gráficos das componentes da função de avaliação.

aptidão de um indivíduo é multiplicada por um fator de 100 no caso de a luminosidade que se espera desse vetor de controlo resulte num valor inferior a 90% do pretendido.

$$F_1 = 3 \times \left(\frac{|Lum_{Esperada} - Lum_{Pretendida}|}{Lum_{Pretendida}} \right)^2 + 1 \tag{5.1}$$

$$F_2 = \frac{N_{circuitos}}{e^{2Prob_{ocup}} + 1} \tag{5.2}$$

$$F = F_1 \times F_2 \times Penalizacao \tag{5.3}$$

O objetivo da função de avaliação é compatibilizar a poupança de energia e o conforto do utilizador.Para isto, com a componente F_2, são prejudicados os indivíduos cuja luminosidade esperada seja muito diferente da pretendida. Já a componente F_2, aumenta o seu valor quando existe utilização de mais circuitos face a taxas de ocupação mais baixas.

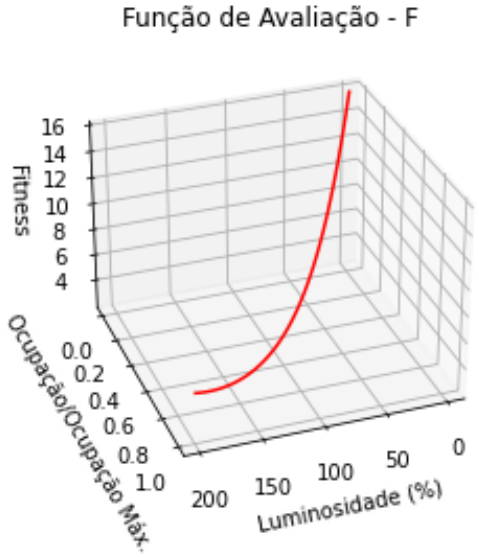


Figura 5.11: Gráfico em três dimensões da função de avaliação.

A recombinação é feita através do processo *One Point Crossover*, com uma probabilidade de 0.5, a mutação é feita por Troca de Bit com a mesma probabilidade e a seleção de descendentes processa-se através de torneios, com a participação de três indivíduos.

5.3.4 Implementação do Controlo do Sistema

O sistema que gere a iluminação para além dos algoritmos de inteligência artificial, está composto por uma parte de aquisição de dados e um *script* na linguagem Python.

O programa em Python, que controla os circuitos de iluminação, está encarregue, no princípio de cada hora, de ligar ou desligar os circuitos de acordo com a solução obtida pelo algoritmo genético, além de ligar um dos circuitos de iluminação no caso de deteção de presença num intervalo de tempo em que todos os circuitos estejam desligados.

A organização dos dados dos sensores é feita automaticamente por um fluxo criado com o *addon* Node-RED, como demonstra a Figura 5.12, este fluxo é composto por várias funções que fazem pedidos de informação à base de dados InfluxDB, processando-os posteriormente através de funções JavaScript. Este fluxo é corrido uma vez no final de cada hora e analisa as informações relativas a esse intervalo de tempo.

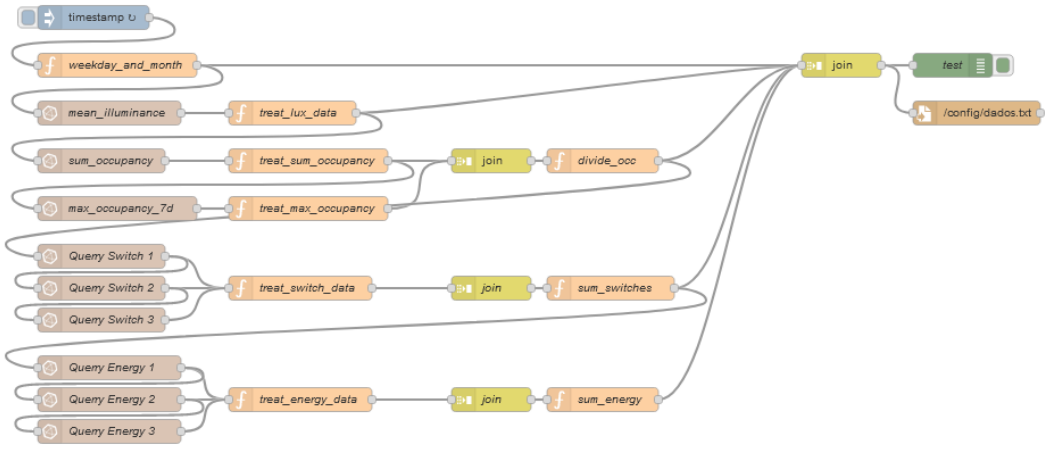


Figura 5.12: Fluxo desenvolvido no *addon* Node-RED para agregar os dados adquiridos.

Os dados reunidos são agregados em duas *strings*, que são posteriormente colocadas em dois ficheiros. Num destes, são contemplados os seguintes valores: dia da semana, mês, intervalo (hora do dia), luminosidade média, rácio de ocupação face à ocupação máxima, número de circuitos ligados e ainda potência média nesse intervalo de tempo, como representado na Figura 5.13. Esta é a informação que posteriormente é usada para treino dos algoritmos de regressão, que produzem informação para a etapa de avaliação do algoritmo genético. Outro ficheiro é preenchido com *strings* que serão usadas pelo sistema de monitorização, contendo informação sobre o estado dos circuitos de iluminação e a potência média de cada um, do modo explicitado na Figura 5.14

Variável	Dia da Semana (0 a 6)	Mês (0 a 11)	Hora do Dia (0 a 23)	Lumino-sidade	Taxa de Ocupação (0 a 1)	Circuitos Ligados (0 a 2)	Potência Média
Tipo de Dados	Inteiro	Inteiro	Inteiro	Real	Real	Inteiro	Real
Unidades	-	-	-	Lux	-	-	W
Exemplo	1,	7,	14,	120,	0.7,	2,	100

Figura 5.13: Exemplo da estrutura que agrega os dados recolhidos.

Variável	Circuito 1 (0 ou 1)	Circuito 2 (0 ou 1)	Potência Média 1	Potência Média 2
Tipo de Dados	Binário	Binário	Real	Real
Unidades	-	-	W	W
Exemplo	1,	1,	50,	50

Figura 5.14: Exemplo da estrutura que agrega os dados recolhidos relativos à potência média.

5.3.5 Implementação do Sistema de Monitorização

O sistema de monitorização permite a deteção de falhas no sistema de iluminação, através da comparação da potência esperada para cada circuito de iluminação, com a potência média medida.

Consiste na utilização de dados recolhidos sobre a potência média de cada circuito, quando está ligado. Com estes dados é construído um vetor que representa a potência média de cada circuito. Posteriormente, a cada hora de funcionamento, um *script* na linguagem *Python* compara os valores de potência de cada circuito no intervalo de tempo anterior aos valores médios definidos no vetor. No caso de os valores reais serem inferiores a 90% do esperado, ou caso os circuitos estejam ligados e o valor do consumo for nulo, o sistema de monitorização gera uma mensagem de acordo, que é apresentada no painel de interface com o utilizador.

5.3.6 Implementação da Interface com o Utilizador

Para interação entre o utilizador e o sistema de controlo, foi criada uma interface intuitiva, desenvolvida através do *addon AppDaemon*, que permite a utilização de código YAML para elaborar painéis de controlo e informação, que ficam expostos ao utilizador quando este acede a um determinado porto através do navegador.

Nesta página, cuja modelação está representada na Figura 5.15, é possível ao utilizador definir o nível de luminosidade pretendida para 2 períodos distintos, um diário e um nocturno, bem como definir as horas a que cada um destes períodos inicia. É-lhe ainda disponibilizada informação referente à luminosidade média no espaço e ocupação nas últimas horas, bem como um indicador da previsão de ocupação para a hora atual. Para além disto, permite ao utilizador o controlo manual dos circuitos de iluminação, para o

caso de ocorrer algum problema. Os avisos produzidos pelo sistema de monitorização são também apresentados nesta página.

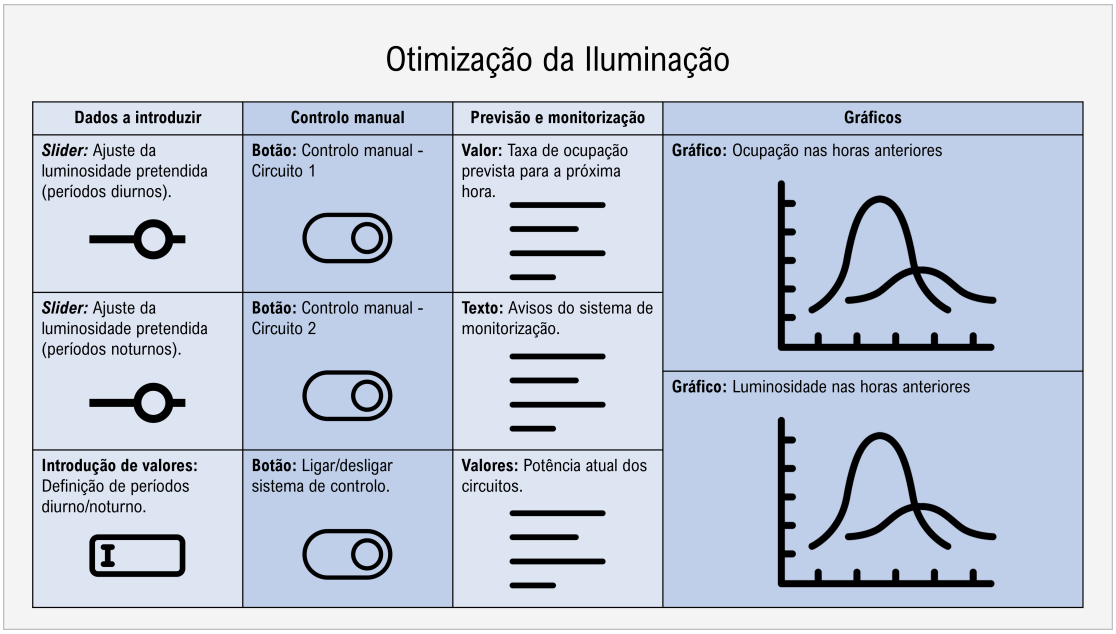


Figura 5.15: Diagrama da interface com o utilizador.

A interface implementada através de código YAML no *addon Appdaemon* está representada na Figura 5.16.

Figura 5.16: Interface com o utilizador.

6

Resultados

Neste capítulo são expostos os resultados da implementação do sistema. Para obter os resultados sobre o funcionamento do sistema, são necessárias duas etapas de execução: inicialmente deve ser recolhida informação acerca do nível de luminosidade, taxa de ocupação e também das restantes variáveis, para que os algoritmos de previsão sejam treinados de modo a reconhecerem a influência destas variáveis no nível de luminosidade e taxa de ocupação. De seguida, com os modelos produzidos, o sistema de controlo deve ser ligado, sendo que o algoritmo genético utiliza a informação dos modelos para definir o esquema de controlo a utilizar em cada intervalo.

6.1 Algoritmos de Previsão

Numa primeira etapa, cuja duração foi de 113 horas, o sistema recolheu informação sobre a luminosidade média do espaço, taxa de ocupação e potência média, sendo os circuitos manualmente ligados ou desligados. Com esta informação, é gerado um modelo relativamente à luminosidade média com base no intervalo, número de circuitos e mês do ano, bem como um modelo para a taxa de ocupação face ao intervalo e dia da semana.

A informação é agregada de hora a hora e diz respeito à potência média (em W), medida pelos medidores de consumo, e à luminosidade média (em lux), medida pelos sensores de luminosidade instalados.

Das 113 medições horárias, 26 foram efetuadas sem nenhum circuito de iluminação ligado, 37 com apenas um circuito e 50 com dois circuitos de iluminação ligados.

6.1.1 Previsão do Nível de Luminosidade

Nas figuras 6.1, 6.2 e 6.3, estão representadas (por pontos) as médias horárias dos valores obtidos pelos sensores de luminosidade instalados, bem como as curvas referentes ao modelo produzido pelos algoritmos de previsão, que geram árvores de decisão. Estes gráficos representam dados referentes ao mês de setembro.

Observando os gráficos, são percetíveis as razões para algumas áreas do modelo em que as alterações na luminosidade média não seriam as esperadas, percebendo-se que são

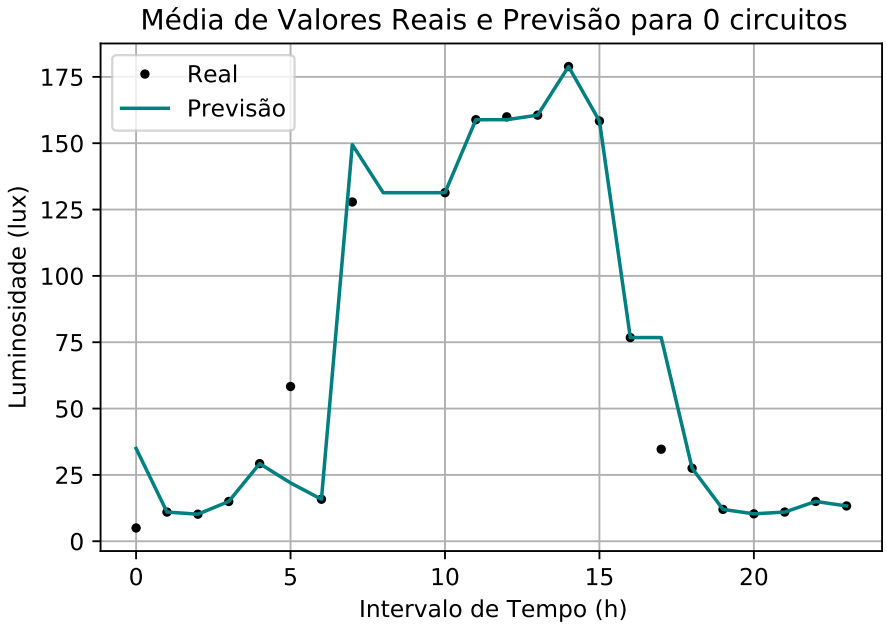


Figura 6.1: Comparação dos valores obtidos com a previsão do algoritmo para zero circuitos ligados.

Figura 6.2: Comparação dos valores obtidos com a previsão do algoritmo para um circuito ligado.

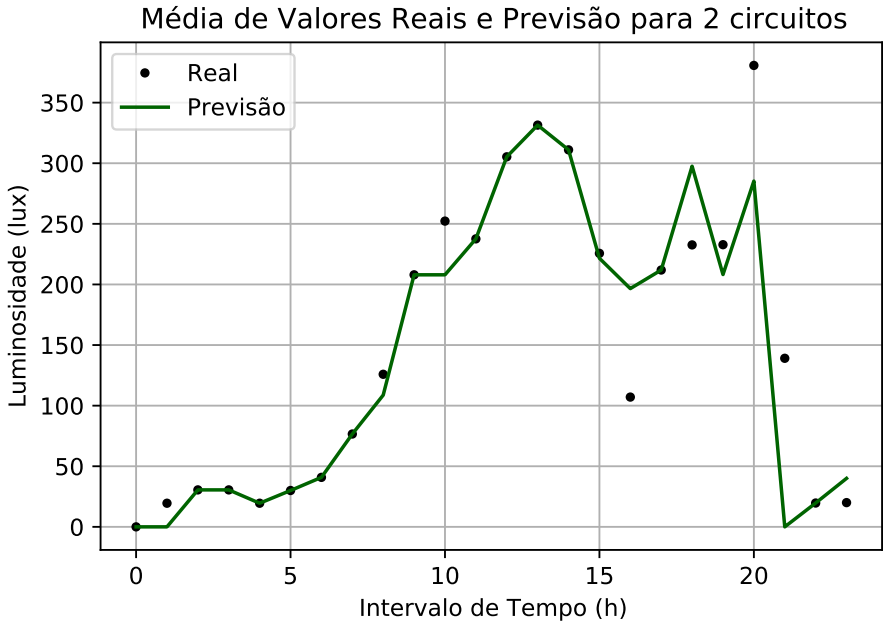


Figura 6.3: Comparação dos valores obtidos com a previsão do algoritmo para dois circuitos ligados.

provenientes dos valores recebidos dos sensores de luminosidade. A causa destes valores díspares está relacionada com fatores como o grau de incidência dos raios solares, ou sombreamento devido a fatores exteriores ao sistema.

Com as 3 curvas do gráfico anterior, obtém-se o modelo a partir do qual, para um determinado mês, e a partir do intervalo do dia e do número de circuitos ligados, é possível prever a luminosidade média no espaço. Este modelo está representado graficamente na figura 6.4

Com os dados recolhidos durante a execução do sistema de controlo, é possível validar o modelo produzido nesta etapa, comparando os valores do modelo com os valores registados. Esta comparação é feita na Figura 6.5.

Comparando as curvas, verifica-se que o modelo consegue obter resultados pouco precisos em relação aos valores recolhidos pelo sistema durante a fase de execução,isto é justificado principalmente, pela influência de fatores exteriores, como a iluminação natural que entra no espaço, que pode variar, ainda que em períodos análogos de dias diferentes. O erro relativo médio entre os valores previstos e os valores reais é de 78,2%. Observando o gráfico percebe-se que uma porção significante do erro está relacionada com um erro no modelo de previsão da luminosidade, possível de verificar no máximo relativo no intervalo das 20h, na Figura 6.3.

6.1.2 Previsão da Taxa de Ocupação

Na Figura 6.6 está representado graficamente o modelo gerado para previsão da ocupação, dependente do dia da semana e do intervalo do dia

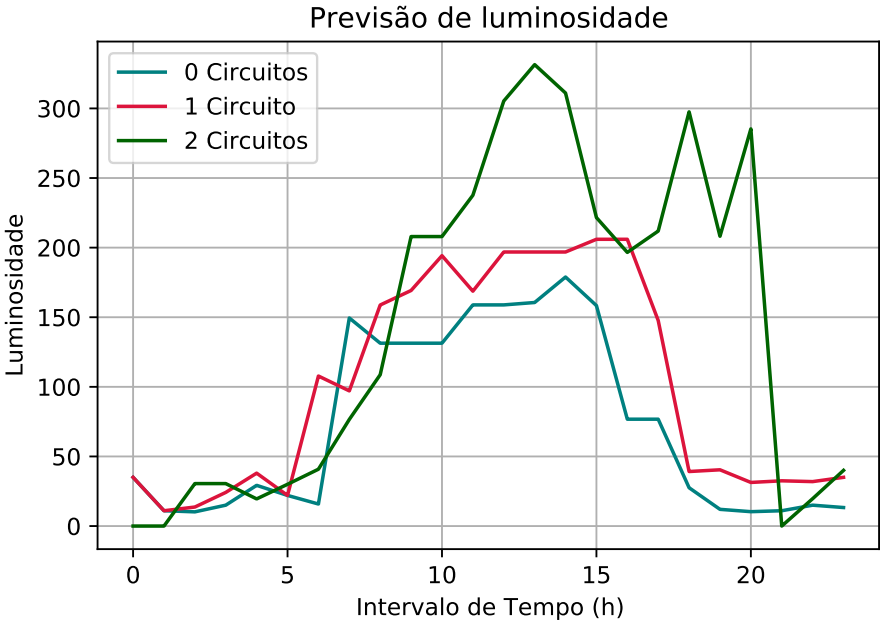


Figura 6.4: Modelo de previsão da luminosidade média.

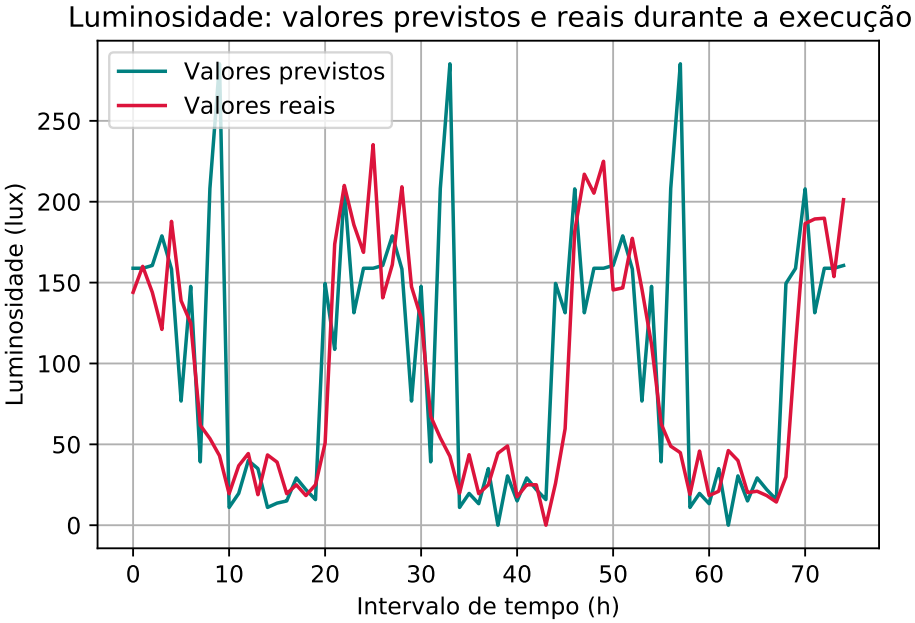


Figura 6.5: Valores previstos e registados do nível de luminosidade.

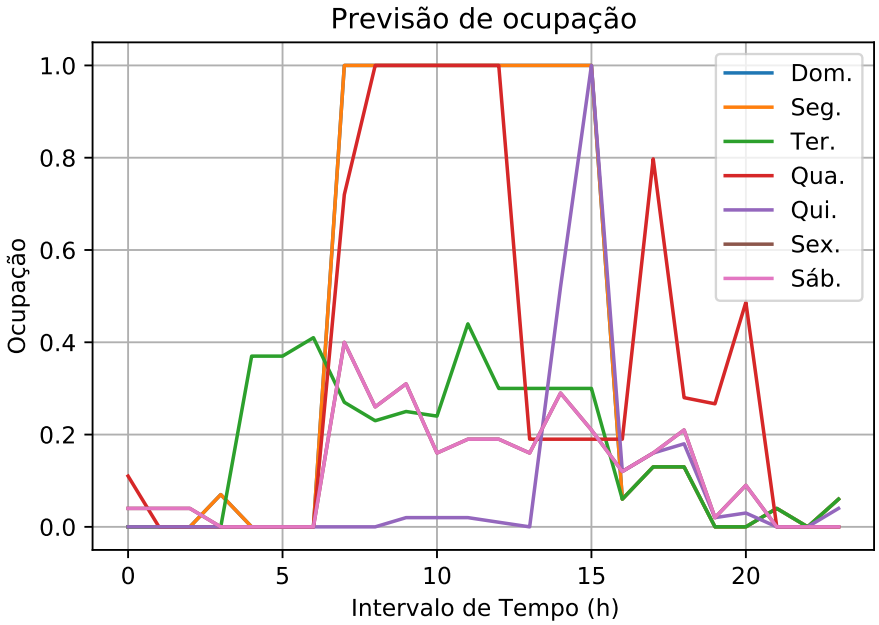


Figura 6.6: Modelo de previsão da taxa de ocupação.

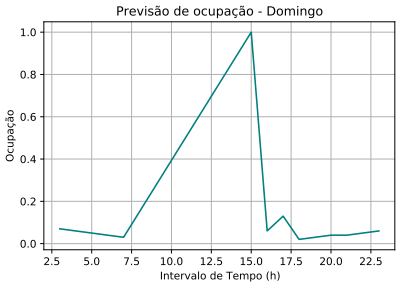
Na Figura 6.7, estão representados os valores previstos pelo modelo de previsão de ocupação, separando as curvas do gráfico anterior, bem como a média por hora dos valores reais, obtidos através dos sensores, com os quais o modelo de previsão foi treinado.

Analisando os gráficos, evidencia-se o facto de não haver valores recolhidos para os dias de Sábado e Domingo, o que tem uma influência negativa sobre o modelo, sendo que o algoritmo de previsão acaba por considerar estes dias como sendo similares a outros. Os restantes gráficos mostram que o modelo produziu resultados adequados, de acordo com os valores reais que recebeu.

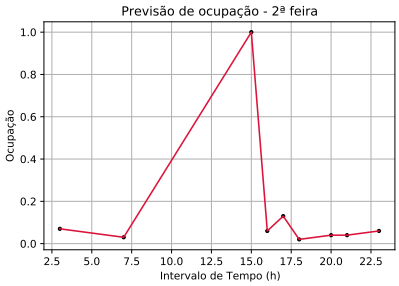
Durante o período de execução do sistema de controlo, os dados referentes à taxa de ocupação continuaram a ser recolhidos, o que permite fazer a comparação na Figura 6.8, entre os valores previstos para ocupação em cada intervalo e os valores efetivamente registados.

Comparando os dois gráficos, verifica-se que é possível ver padrões em que o modelo prevê uma maior afluência de pessoas no espaço e isso se acaba por verificar, embora sem precisão em relação à taxa de ocupação. Em relação aos valores reais, os valores previstos apresentam um erro médio bastante elevado, de 234,7%, o que se explica em parte pela dificuldade em prever precisamente a quantidade de pessoas no espaço, sendo esta uma variável bastante volátil. Contudo, sendo que o valor da taxa de ocupação é usado para que o algoritmo genético tenha em conta se a determinada altura haverá um fluxo alto de pessoas no espaço ou não, de modo a ajustar a avaliação de cada solução, não é necessário um elevado nível de precisão.

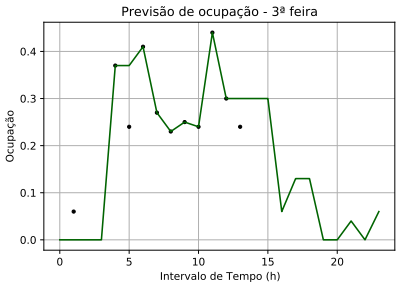
Qualquer um dos modelos teria beneficiado de um aumento da quantidade de informação recebida, o que não foi possível devido a restrições de tempo, bem como restrições



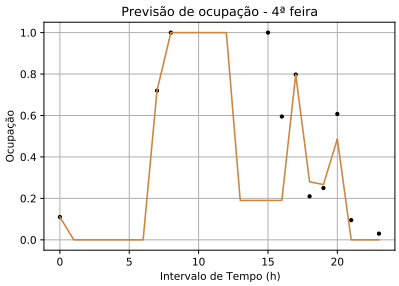
(a) Domingo.



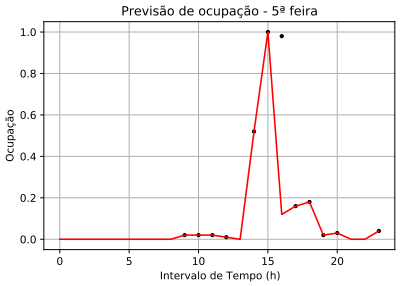
(b) 2ª feira.



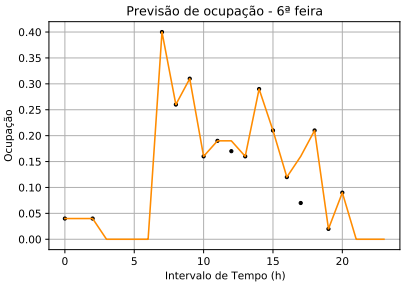
(c) 3ª feira.



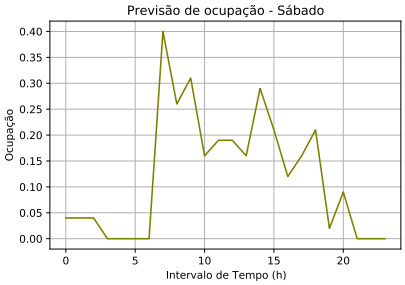
(d) 4ª feira.



(e) 5ª feira.



(f) 6ª feira.



(g) Sábado.

Figura 6.7: Gráficos do modelo de previsão de ocupação, conforme o dia da semana.

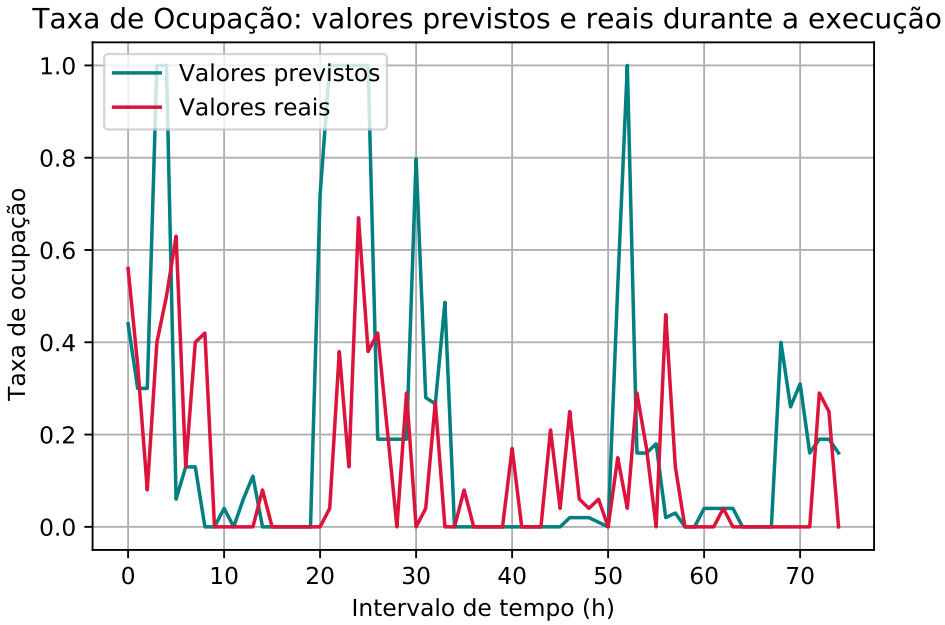


Figura 6.8: Valores previstos e registados da taxa de ocupação.

associadas à pandemia de COVID-19.

6.2 Execução do Sistema de Controlo

Numa segunda etapa, o sistema de controlo de iluminação foi ligado, sendo assim avaliado o sistema de decisão baseado no algoritmo genético aliado aos modelos descritos anteriormente. Esta etapa teve uma duração de 75 horas ao longo de 4 dias, recolhendo informação horária acerca de quais circuitos estiveram ligados, bem como a potência de cada um destes.

Ao testar o sistema, pretende-se avaliar as duas métricas que o sistema tenta otimizar: por um lado o consumo de energia deve ser menor que o consumo anterior, por outro, o nível de luminosidade do espaço deve manter-se igual ou superior àquele definido pelo utilizador para cada um dos dois períodos do dia.

6.2.1 Consumo de Energia

A potência dos circuitos do sistema de iluminação medida durante 75 horas. Foi verificado que a potência média do circuito 1 é de 116,3 W, e do circuito 2, 192,4 W. Assim, numa hora, os dois circuitos consomem, em média, 308,7 Wh. Isto significa que, num período de análogo ao da execução do sistema de controlo, caso os dois circuitos estivessem ligados ininterruptamente, resultaria um consumo de 23,152 kWh.

Com a utilização do algoritmo genético e do sistema de controlo, foi possível diminuir este valor em 70%, para um consumo de energia de 6,940 kWh.

Ligando os dois circuitos durante os perídos considerados diurnos (das 8h as 20h), seria expectável obter um consumo de 12,039 kWh. Sendo assim, o sistema de controlo implementado representaria ainda uma poupança de 42%. A tabela 6.1 esquematiza esta informação.

Tabela 6.1: Comparação de opções de controlo em período análogo

Tipo de controlo	**Consumo esperado (kWh)**	**Percentagem de poupança do Sistema Desenvolvido** (%)
Sempre ligado	23,152	70
Apenas períodos diurnos	12,039	42
Sistema de controlo desenvolvido	6,940	—

Na Figura 6.9 expõe-se aquela que foi a potência média do sistema de iluminação ao longo da execução do sistema de controlo.

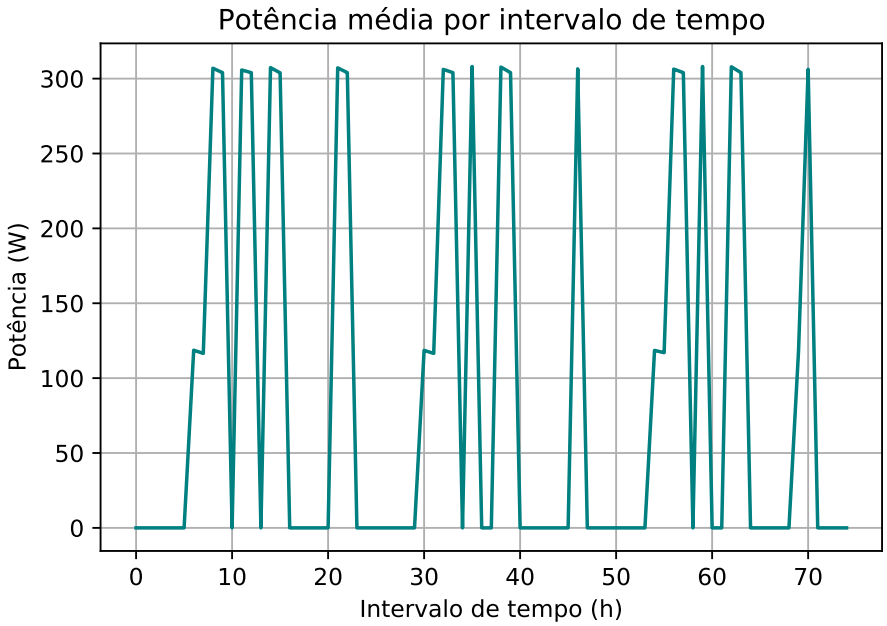


Figura 6.9: Potência média por intervalo de tempo.

Verifica-se que o sistema ligou sempre o mesmo circuito (de menor consumo) nas circunstâncias em que apenas um circuito de iluminação era necessário. É também perceptível, e mais facilmente verificado na Figura 6.10, em que o valor médio e o desvio padrão da potência por número de circuitos são apresentados, que a potência de cada um dos circuitos se manteve essencialmente constante, não se tendo detetado portanto nenhuma anomalia ou defeito nos circuitos de iluminação ou luminárias, pelo que não houve igualmente nenhum alerta gerado pelo sistema de monitorização.

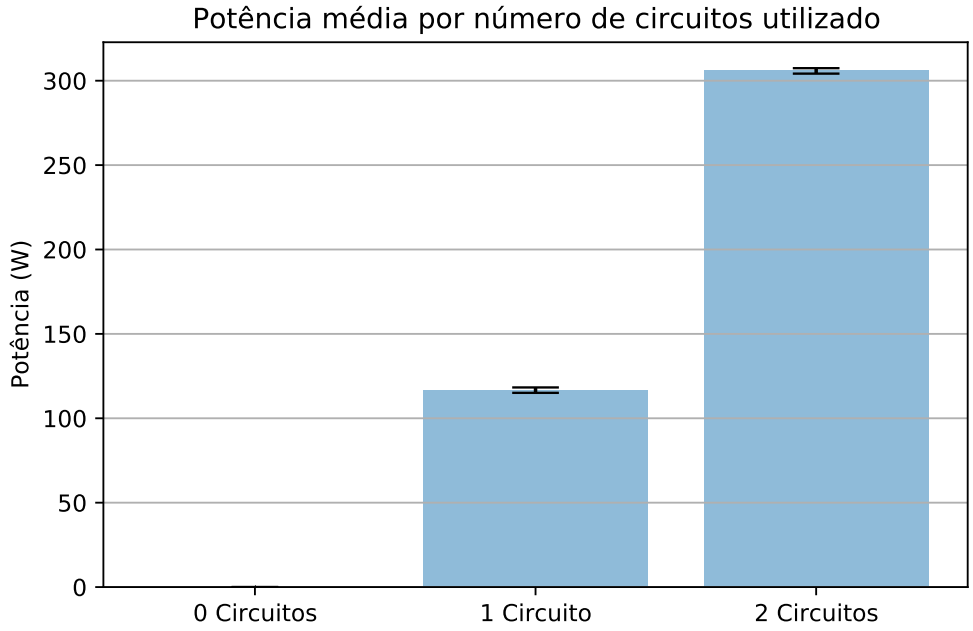


Figura 6.10: Potência média por número de circuitos ligados.

6.2.2 Nível de Luminosidade

Na interface com o utilizador foi definido um valor de 100 lux para a luminosidade pretendida durante os períodos diurnos e uma luminosidade de 0 lux para os períodos noturnos. Analisando os resultados, verifica-se que em 65 dos 75 intervalos analisados, a luminosidade foi satisfatória, obtendo-se um valor igual ou superior ao pretendido, isto representa 87% de intervalos de tempo com luminosidade pretendida.

No gráfico da Figura 6.11, podem observar-se os níveis de luminosidade no decorrer da execução, indicando a linha a vermelho o nível de luminosidade pretendido nos períodos diurnos, que estão sombreados.

É possível, através da figura, verificar que as exigências de luminosidade definidas foram atingidas, como anteriormente descrito.

Contudo, verifica-se que a luminosidade nos períodos diurnos é bastante superior aquela que foi definida pelo utilizador, o que para além de representar um possível gasto desnecessário de recursos (nos casos em que o sistema liga mais circuitos do que aqueles necessários para atingir a luminosidade exigida), pode significar uma diminuição no nível de conforto do utilizador, sendo que em algumas circunstâncias o valor registado foi o dobro daquele que foi solicitado. As causas destes níveis elevados de luminosidade prendem-se essencialmente com a influência dos circuitos que não são controlados pelo sistema de controlo, bem como com a entrada de luz natural no edifício, através de janelas para o exterior, sendo que neste caso não é possível reduzir a luminosidade.

O facto de o controlo da iluminação não ter a capacidade de regular a intensidade das lâmpadas, retira alguma granularidade à ação do sistema de controlo, pelo que a única

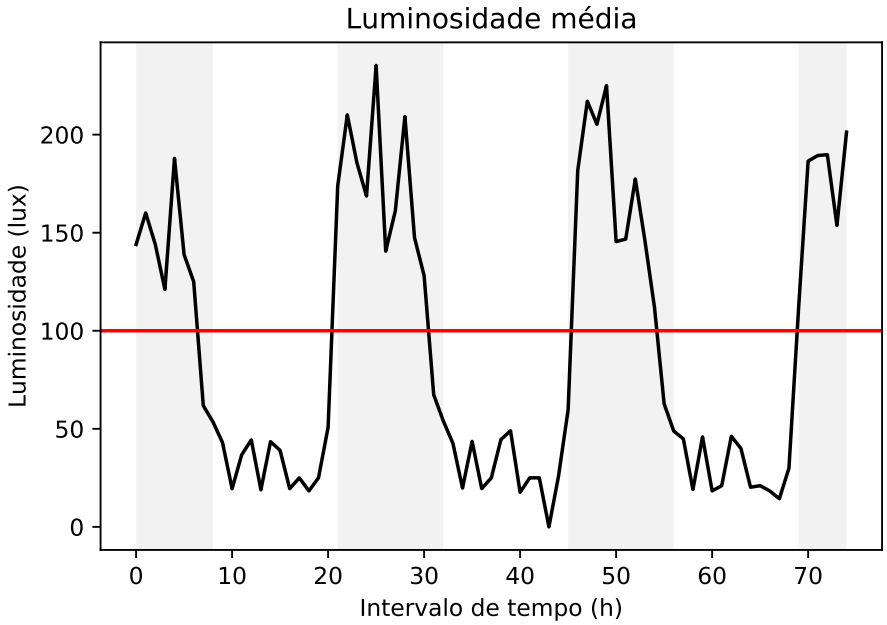


Figura 6.11: Luminosidade média e luminosidade pretendida.

solução para atingir o valor de luminosidade pretendido, pode acabar por excedê-lo.

A iluminação excedente no período noturno deve-se ao facto do sistema de controlo não atuar sobre todos os circuitos de iluminação que servem o espaço, sendo ainda que o edifício não pode ficar desprovido de iluminação durante o período noturno, por razões de segurança.

Na Figura 6.12 estão representadas as curvas de luminosidade média, bem como da taxa de ocupação durante o decorrer da execução do sistema de controlo.

Neste gráfico é possível observar a correlação entre o aumento do nível de luminosidade imposto pelo sistema de controlo e o aumento da taxa de ocupação do espaço. Este efeito deve-se à inclusão da taxa de ocupação esperada como variável de entrada do algoritmo genético utilizado, cuja função de avaliação valoriza soluções de controlo em que o rácio entre número de circuitos utilizados e a taxa de ocupação esperada seja mais baixo.

As partes do gráfico em que esta correlação não se verifica devem-se essencialmente a disparidades entre a taxa de ocupação prevista pelo algoritmo de previsão para certos intervalos e aquela que efetivamente se verificou ou às restantes componentes da função de avaliação, como a luminosidade pretendida, que faz com que o sistema tente manter um determinado nível de iluminação, ainda que a taxa de ocupação esperada seja baixa.

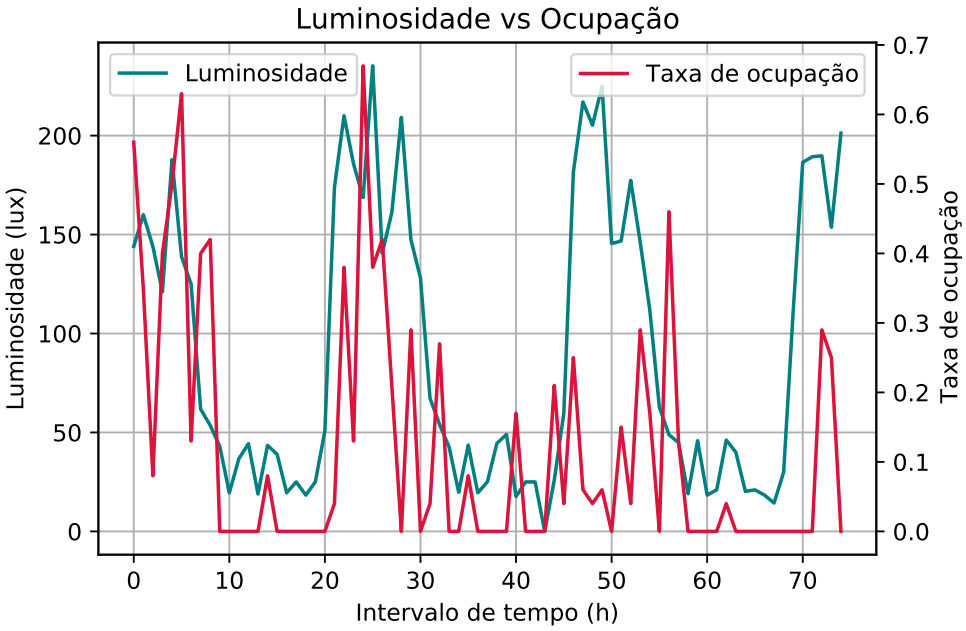


Figura 6.12: Luminosidade média e taxa de ocupação.

7

Conclusão

Neste último capítulo é feita uma síntese, bem como conclusões finais sobre o trabalho desenvolvido no âmbito desta dissertação, no que diz respeito à implementação e resultados obtidos. São ainda sugeridos possíveis desenvolvimentos futuros para o sistema implementado.

7.1 Observações Finais

Desenvolveu-se nesta dissertação um sistema de controlo otimizado para luminárias, através do uso de inteligência artificial. Este sistema tem a capacidade de, para um determinado espaço, decidir quais circuitos de iluminação ligar ao longo de um dia, dividido em vários períodos. Para além do controlo, foi implementado um sistema de monitorização, que permite a deteção de avarias nas luminárias.

O trabalho começou com uma pesquisa profunda sobre algoritmos evolucionários, de modo a entender os princípios gerais por detrás destes algoritmos, e as componentes pelas quais são tipicamente formados. Foi também importante clarificar as características dos vários tipos de algoritmos, bem como perceber a aplicabilidade de cada um em diferentes contextos. Nesta pesquisa foram também estudados trabalhos semelhantes ao que se pretendia fazer nesta dissertação, de modo a poder avaliar as escolhas feitas e os resultados produzidos.

Seguiu-se a definição da arquitetura a utilizar, em que ficou estipulado que o sistema devia ser composto por sensores como medidores do consumo de energia e sensores de luminosidade e presença e ainda atuadores com controlo ligar/desligar sobre o sistema de iluminação. Estes sensores e atuadores devem comunicar com um computador central, preferencialmente através de ligações sem fios, através de protocolos de comunicação como Zigbee, MQTT e Modbus.

Na escolha posterior dos componentes para a camada física do sistema procurou-se escolher equipamentos comerciais que pudessem ser prontamente adquiridos e a um baixo custo, de modo a permitir a replicação do sistema em contexto doméstico e de edifícios públicos de qualquer dimensão, optando-se assim por sensores de luminosidade

e presença da marca Xiaomi, medidores de consumo da marca Orno e ainda relés da marca Sonoff. Todos estes equipamentos comunicam com um computador Raspberry Pi através do protocolo sem-fios Zigbee, à excepção do medidor de consumo que o faz utilizando o protocolo Modbus Serial, através de ligações RS-485.

Para a camada de decisão, foram desenvolvidos modelos de previsão, utilizando a biblioteca Python *Scikit-Learn* e a sua implementação do algoritmo CART (do inglês *Classification and Regression Trees*), bem como um algoritmo genético, implementado através da biblioteca Python DEAP (do inglês *Distributed Evolutionary Algorithms in Python*). A estes algoritmos juntam-se sistemas de controlo e monitorização, implementados igualmente em *scripts* na linguagem Python, e cujas funções são, respetivamente, enviar para os relés sem fios controlos de ligar ou desligar, de acordo com o resultado do algoritmo genético e verificar a potência de cada um dos circuitos, permitindo verificar quando o estes apresentam um desvio da norma.

Foi ainda desenvolvida uma interface gráfica, que possibilita a interação do utilizador com o programa, através da definição de níveis de luminosidade pretendida distintos, para um período noturno e diurno, bem como um controlo manual do sistema, visualização de avisos gerados pelo sistema de monitorização e ainda dados recolhidos recentes.

Os resultados obtidos com a execução do sistema de controlo provam que o objetivo da dissertação foi cumprido, proporcionando uma poupança de 70% no consumo de energia face à realidade antes da tese, em que os circuitos de iluminação estavam constantemente ligados e ainda uma poupança de 42% face à solução anteriormente utilizada, em que apenas estariam ligados durante o período diurno. Aliada a uma redução significativa do consumo de energia, está a manutenção do nível de luminosidade pretendido, durante os períodos definidos como diurnos, o que contribui para um maior conforto do utilizador.

7.2 Trabalho Futuro

Como desenvolvimento futuro propõem-se algumas alterações ao sistema, que resultariam numa maior poupança de energia ou num maior conforto do utilizador, melhorando a experiência que este tem com o sistema. Estas alterações são as seguintes:

- Alteração das luminárias para umas com funcionalidade de regulação da intensidade, permitindo um controlo mais granular da luminosidade no espaço;
- Separação dos circuitos de iluminação por áreas da instalação e alteração do algoritmo, permitindo a definição de áreas e de iluminações pretendidas para cada área;
- Desenvolvimento de uma aplicação para telemóvel que permita uma interação mais fácil do utilizador com o sistema;
- Implementação do sistema noutra plataforma que permita executar os algoritmos de regressão periodicamente, resultando em previsões mais precisas.

Bibliografia

Ahmad, M. W., Mourshed, M., Mundow, D., Sisinni, M. & Rezgui, Y. (2016). Building energy metering and environmental monitoring – A state-of-the-art review and directions for future research. *Energy and Buildings*, *120*, 85–102. (Ver p. 30).

Alliance, Z. (s.d.). Zigbee Specification. 2015. (Ver p. 29).

Bäck, T. (2002). Adaptive business intelligence based on evolution strategies: some application examples of self-adaptive software. *Information Sciences*, *148*(1), 113–121. (Ver p. 16).

Bull, L. (2004). *Applications of Learning Classifier Systems*. (Ver pp. 13, 17).

Caldas, L. & Norford, L. (2003). Genetic Algorithms for Optimization of Building Envelopes and the Design and Control of HVAC Systems. *Journal of Solar Energy Engineering*, *125*(3), 343–351. (Ver p. 19).

Collette, Y. & Siarry, P. (2003). *Multiobjective Optimization: Principles and Case Studies* (1st). Springer. (Ver p. 4).

Čongradac, V., Milosavljevic, B., Velickovic, J. & Prebiracevic, B. (2012). Control of the lighting system using a genetic algorithm. *Thermal Science*, *16*(1), 237–250. (Ver pp. 24, 26).

Consumo de energia eléctrica: total e por tipo de consumo. (s.d.). PORDATA. Visitado: 09-01-2021. Obtido de https://www.pordata.pt/Portugal/Consumodeenergiaelectrica-totaleportipodeconsumo-1124. (Ver pp. 1, 2)

Dam, H., Abbass, H., Lokan, C. & Yao, X. (2007). Neural-Based Learning Classifier Systems. *IEEE Transactions on Knowledge and Data Engineering*, *20*(1), 26–39. (Ver p. 12).

Documentação DEAP. (s.d.). Visitado: 02-08-2021. Obtido de https://deap.readthedocs.io/en/master/. (Ver p. 43)

Dorigo, M., Birattari, M. & Stützle, T. (2006). Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique. *IEEE Computational Intelligence Magazine*, *1*, 28–39. (Ver p. 17).

Eiben, A. & Smith, J. (2007). *Introduction to Evolutionary Computing*. Springer. (Ver pp. 5, 6, 8–14).

Elbeltagi, E., Hegazy, T. & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, *19*(1), 43–53. (Ver pp. 11, 15).

Electricity Customers. (s.d.). EPA. Visitado: 09-01-2021. Obtido de https://www.epa.gov/energy/electricity-customers. (Ver p. 1)

Energy consumption in households. (s.d.). EUROSTAT. Visitado: 09-01-2021. Obtido de https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_consumption_in_households#Energy_consumption_in_households_by_type_of_end-use. (Ver p. 1)

Exemplo DEAP - Algoritmo Genético. (s.d.). Visitado: 02-08-2021. Obtido de https://github.com/DEAP/deap/blob/master/examples/ga/onemax_short.py. (Ver p. 43)

Ferreira, C. (2001). Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems*, *13*(2), 87–129. (Ver p. 12).

Fonseca, C. (1995). Multiobjective Genetic Algorithms with Application to Control Engineering Problems. (Ver pp. 5, 9).

Galasiu, A., Newsham, G., Suvagau, C. & Sander, D. (2013). Energy Saving Lighting Control Systems for Open-Plan Offices: A Field Study. *LEUKOS - Journal of Illuminating Engineering Society of North America*, *4*. (Ver p. 2).

Gómez-Lorente, D., Rabaza, O., Espín, A. & Peña-García, A. (2013). Optimization of efficiency and energy saving in public lighting with multi-objective evolutionary algorithms. *Renewable Energy and Power Quality Journal*, *10*(11), 62–65. (Ver p. 18).

Gómez-Lorente, D., Rabaza, O., Estrella, A. & Peña-García, A. (2013). A new methodology for calculating roadway lighting design based on a multi-objective evolutionary algorithm. *Expert Systems with Applications*, *40*(6), 2156–2164. (Ver p. 18).

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd). Springer, New York. (Ver p. 42).

Haynes, B. (2008). The impact of office comfort on productivity. *Journal of Facilities Management*, *6*, 37–51. (Ver p. 2).

Iacomussi, P., Radis, M., Rossi, G. & Rossi, L. (2015). Visual Comfort with LED Lighting. *Energy Procedia*, *78*, 729–734. 6th International Building Physics Conference, IBPC 2015. (Ver p. 2).

Kusiak, A., Tang, F. & Xu, G. (2011). Multi-objective optimization of HVAC system with an evolutionary computation algorithm. *Energy*, *36*(5), 2440–2449. (Ver p. 19).

Kusiak, A. & Zheng, H. (2010). Optimization of wind turbine energy and power factor with an evolutionary computation algorithm. *Energy*, *35*(3), 1324–1332. (Ver p. 19).

Lezama, F., Soares, J., Sucar, L., Cote, E. & Vale, Z. (2017). Differential evolution strategies for large-scale energy resource management in smart grids. Em *The Genetic and Evolutionary Computation Conference Companion*, Berlim. (Ver p. 19).

Lezama, F., Soares, J. & Vale, Z. (2019). Optimal Bidding in Local Energy Markets using Evolutionary Computation. Em *20th International Conference on Intelligent System Application to Power Systems (ISAP*, New Delhi, India. (Ver p. 19).

Lourenço, J. M. (2021). *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. Obtido de https://github.com/joaomlourenco/novathesis/raw/master/template.pdf. (Ver p. iii)

Modbus. (s.d.). Visitado: 24-07-2021. Obtido de https://modbus.org/faq.php. (Ver p. 30)

Ngo, M., Nguyen, X., Duong, Q. & Nguyen, H. (2019). Adaptive Smart Lighting Control based on Genetic Algorithm. Em *25th Asia-Pacific Conference on Communications, Ho Chi Minh City*, Vietnam. (Ver pp. 23, 24, 26).

OASIS. (s.d.). MQTT Specification. 7 de março de 2019. (Ver p. 30).

Orno: Medidor de Consumo. (s.d.). Visitado: 19-05-2021. Obtido de https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjgqK6fhNbwAhXvyYUKHUqZBrAQFjABegQIAxAD&url=https%5C%3A%5C%%5C%2Fwww.img.orno.pl%5C%2Fmanuals%5C%2FOR-WE-504_manual_EN.pdf&usg=AOvVaw0rTX10Yz53uInLN6brbUhW. (Ver p. 40)

Particle Swarm Optimization (PSO). (s.d.). Visitado: 12-02-2021. Obtido de %7Bhttps://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html%7D. (Ver p. 14)

PcDiga: DetetorLux. (s.d.). Visitado: 19-05-2021. Obtido de https://www.pcdiga.com/casa-e-ar-livre/smart-home-e-iluminacao/sensores-e-detetores/sensor-mi-light-detection-ytc4043gl. (Ver p. 39)

PcDiga: Sensor de Presença. (s.d.). Visitado: 19-05-2021. Obtido de https://www.pcdiga.com/casa-e-ar-livre/smart-home-e-iluminacao/sensores-e-detetores/sensor-de-movimento-xiaomi-mi-motion-sensor-ytc4041gl. (Ver p. 39)

Petrinska, I., Georgiev, V. & Ivanov, D. (2018). Lighting control system for public premises, based on evolutionary optimization algorithm. Em *20th International Symposium on Electrical Apparatus and Technologies*, Burgas. (Ver pp. 22, 26).

Poli, R., Langdon, W. & Mcphee, N. (2008). *A Field Guide to Genetic Programming*. (Ver p. 16).

Resolução do Conselho de Ministros n.º 53/2020. (2020). Diário da República. (Ver p. 1).

Rokach, L. & Maimon, O. (2014). *Data Mining with Decision Trees* (2nd). WORLD SCIENTIFIC. (Ver p. 42).

Scikit Learn. (s.d.). Visitado: 02-08-2021. Obtido de https://scikit-learn.org/stable/. (Ver p. 42)

Scikit Learn - Decision Trees. (s.d.). Visitado: 02-08-2021. Obtido de https://scikit-learn.org/stable/modules/tree.html. (Ver p. 42)

Selvi, V. & Umarani, R. (2010). Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications*, *5*(4), 1–6. (Ver pp. 13, 14, 17).

Si, W., Ogai, H., Li, T. & Hirai, K. (2013). A novel energy saving system for office lighting control by using RBFNN and PSO. Em *IEEE 2013 Tencon - Spring* (pp. 347–351). (Ver pp. 20, 21, 26).

Silva, A. & Abrao, P. (2002). Applications of Evolutionary Computation in Electric Power Systems. Em *Proceedings of the 2002 Congress on Evolutionary Computation*. (Ver p. 18).

Slowik, A. & Kwasnicka, H. (2018). Nature Inspired Methods and Their Industry Applications—Swarm Intelligence Algorithms. *IEEE Transactions on Industrial Informatics*, *14*(3), 1004–1015. (Ver p. 17).

Slowik, A. & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, *32*, 12363–12379. (Ver pp. 16, 18, 25).

Sonoff BasicZBR3 - Interruptor Zigbee. (s.d.). Visitado: 19-05-2021. Obtido de https://www.electrofun.pt/en/sonoff/sonoff-basiczbr3-zigbee-diy-smart-switch. (Ver p. 40)

Tamilselvi, S., Baskar, S., Anandapadmanaban, L., Karthikeyan, V. & Rajasekar, S. (2018). Multi objective evolutionary algorithm for designing energy efficient distribution transformers. *Swarm and Evolutionary Computation*, *42*, 109–124. (Ver p. 19).

Viani, F., Polo, A., Anselmi, N., Salucci, M. & Giarola, E. (2017). Evolutionary Optimization Applied to Wireless Smart Lighting in Energy-Efficient Museums. *IEEE Sensors Journal*, *17*(5), 1213–1214. (Ver pp. 19, 20, 26).

Wang, W., He, G. & Wan, J. (2011). Research on Zigbee wireless communication technology. Em *2011 International Conference on Electrical and Control Engineering* (pp. 1245–1249). doi:10.1109/ICECENG.2011.6057961. (Ver p. 29)

Wikimedia. (s.d.). Visitado: 12-02-2021. Obtido de https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Aco_branches.svg/2000px-Aco_branches.svg.png. (Ver p. 15)

Wikimedia: Raspberry Pi. (s.d.). Visitado: 19-05-2021. Obtido de https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_B%5C%2B_(26931245278).png. (Ver p. 38)

Wilson, D., Rodrigues, S., Segura, C., Loshchilov, I., Hutter, F., Buenfil, G., ... Hernández-Aguirre, A. (2018). Evolutionary computation for wind farm layout optimization. *Renewable Energy*, *126*, 681–691. (Ver p. 19).

Yu, X. & Gen, M. (2010). *Introduction to Evolutionary Algorithms*. Springer. (Ver pp. 5, 6, 12).

Yuen, S. Y., Chow, C. K., Zhang, X. & Lou, Y. (2016). Which algorithm should I choose: An evolutionary algorithm portfolio approach. *Applied Soft Computing*, *40*, 654–673. (Ver p. 16).