INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Estratégias de Aprendizagem por Reforço para Configuração Dinâmica de Meta-heurísticas

VÍTOR JOSÉ HENRIQUES OLIVEIRA
Setembro de 2024

P.PORTO

Estratégias de Aprendizagem por Reforço para Configuração Dinâmica de Meta-heurísticas

Vítor José Henriques Oliveira

Aluno nº: 1850041

Dissertação para obtenção do Grau de
Mestre em Engenharia de Inteligência Artificial

Orientador: Professor Tiago Manuel Campelos Ferreira Pinto
Co-Orientador: Professor Carlos Fernando da Silva Ramos

Júri:
Presidente:
Isabel Cecília Correia da Silva Praça Gomes Pereira, Professora Coordenadora do ISEP do IPP

Vogais:
Diogo Emanuel Pereira Martinho, Professor Adjunto do ISEP do IPP
Tiago Manuel Campelos Ferreira Pinto, Professor Associado da UTAD

Porto, 30 de setembro de 2024

Resumo

A eficácia da otimização de problemas complexos está intimamente ligada à configuração de parâmetros em algoritmos meta-heurísticos. Embora já tenham sido propostos métodos automatizados para a escolha dos parâmetros de algoritmos para reduzir a necessidade de ajuste manual, existe ainda um potencial significativo, não explorado, de ajuste dinâmico de parâmetros de algoritmos durante a execução, o que pode melhorar o seu desempenho. Este estudo visa aferir a eficácia da definição manual de parâmetros em comparação com uma abordagem dinâmica baseada em aprendizagem por reforço, reduzindo a necessidade de intervenção humana e aumentando a eficiência operacional dos algoritmos.

Para alcançar este objetivo, adaptaram-se os métodos SARSA (*State-Action-Reward-State-Action*) e Deep SARSA para regular os parâmetros de algoritmos meta-heurísticos, em especial, o algoritmo genético. O modelo adotado é independente do problema a ser otimizado ou do algoritmo meta-heurístico selecionado, por isso, oferece a flexibilidade necessária, sendo apenas crucial escolher os parâmetros a ajustar durante o decorrer do processo de otimização de qualquer problema estudado. Estas metodologias foram testadas em funções *benchmark*, amplamente reconhecidas na literatura, e aplicadas nesta investigação nos seguintes cenários práticos: a otimização de portfólios de investimentos, na qual um participante possui ou pretende adquirir energia elétrica num mercado de eletricidade e a melhoria relacionada com a alocação de pacientes em Unidades de Cirurgia (UC) e em Unidades de Cuidados Intensivos (UCI), com o intuito de melhorar a eficiência da utilização de recursos limitados.

Os resultados demonstram que o algoritmo Deep SARSA, baseado em aprendizagem por reforço e redes neuronais, obtém frequentemente um melhor desempenho em comparação com a configuração manual, de cariz completamente aleatório. Este facto pode ser comprovado pela análise dos resultados das médias do número de execuções, nomeadamente, no problema das UC, onde o valor do teste ANOVA apresentou um *p-value* significativo igual a 0.014. Este desfecho sugere que abordagens dinâmicas de ajuste de parâmetros podem ser mais eficazes e oferecer uma alternativa viável a métodos estáticos de configuração, que possam potenciar soluções propostas para enfrentar os desafios em ambientes dinâmicos e incertos.

Palavras-chave: Algoritmo Genético, Aprendizagem Máquina, Aprendizagem por reforço, Configuração Dinâmica de Algoritmos, DAC, Otimização por enxame de partículas, SARSA, Deep-SARSA

Abstract

The effectiveness of optimizing complex problems is closely tied to parameter configuration in meta-heuristic algorithms. Although automated methods for selecting algorithm parameters have already been proposed to reduce the need for manual tuning, there remains significant untapped potential for dynamic parameter adjustment during algorithm execution, which could improve performance. This study aims to assess the effectiveness of manual parameter setting compared to a dynamic approach based on reinforcement learning, reducing the need for human intervention and increasing the operational efficiency of algorithms.

To achieve this objective, the SARSA (*State-Action-Reward-State-Action*) and Deep SARSA methods were adapted to regulate the parameters of meta-heuristic algorithms, particularly the genetic algorithm. The adopted model is independent of the problem to be optimized or the selected meta-heuristic algorithm, thus offering the necessary flexibility, with the only crucial requirement being the choice of parameters to adjust during the optimization process of any studied problem. These methodologies were tested on benchmark functions widely recognized in the literature and applied in this investigation to the following practical scenarios: portfolio optimization, where a participant owns or intends to acquire electricity in an energy market, and improvements related to the allocation of patients in Surgical Units (SU) and Intensive Care Units (ICU), with the goal of improving the efficiency of limited resource utilization.

The results demonstrate that the Deep SARSA algorithm, based on reinforcement learning and neural networks, often achieves better performance compared to completely random manual configuration. This finding is supported by the analysis of the average results across multiple runs, particularly in the SU problem, where the ANOVA test yielded a significant p-value of 0.014. This outcome suggests that dynamic parameter adjustment approaches may be more effective and provide a viable alternative to static configuration methods, potentially enhancing proposed solutions to address challenges in dynamic and uncertain environments.

Keywords: Dynamic Algorithm Configuration, Genetic Algorithm, Machine Learning, Particle Swarm Optimization, Reinforcement Learning, SARSA, Deep-SARSA

Agradecimentos

Dedico este trabalho a todos aqueles que me apoiaram ao longo deste percurso.

Em primeiro lugar, queria expressar a minha gratidão aos meus orientadores, o Professor Tiago Pinto e o Professor Carlos Ramos.

Ao Professor Tiago, agradeço pela disponibilidade incondicional, sempre presente para me orientar com conselhos valiosos. A sua orientação técnica e científica, combinada com um apoio generoso na revisão de artigos e no processo de escrita, foram essenciais para o sucesso deste trabalho.

Ao Professor Carlos Ramos, uma referência no empreendedorismo académico e na divulgação desta fascinante área de conhecimento, o meu sincero agradecimento, foi um privilégio contar com a sua experiência enquanto Professor.

Nunca é tarde para se aprender, mas é preciso querer e ter alguém que nos apoie, porque o sacrifício não é só do autor, por isso rendo-me aos que tornaram esta missão possível:

À Marina Oliveira, e aos meus filhos, Rafaela e Alexandre, pelo carinho, paciência e apoio incondicional. Sem eles a meu lado, este trabalho teria sido muito mais difícil. A todos, o meu obrigado!

Vítor Oliveira

There are no impossible dreams, just our limited perception of what is possible.

Beth Mende Conny

Índice

Lista de Acrónimos **xix**

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Problema . . . 5
1.3 Objetivos . . . 6
1.4 Contribuições . . . 7
1.5 Estrutura do Documento . . . 8
1.6 Resumo . . . 8

2 Enquadramento Teórico **9**
2.1 Otimização . . . 9
2.2 Meta-heurísticas . . . 10
2.3 Configuração de Parâmetros . . . 11
2.4 Resumo . . . 12

3 Revisão Literária **13**
3.1 Metodologia . . . 13
3.1.1 Perguntas de Investigação . . . 13
3.1.2 Fonte de Dados . . . 14
3.1.3 Termos de Pesquisa . . . 14
3.1.4 Critérios de Inclusão e Exclusão . . . 15
3.2 Resultados . . . 16
3.3 Discussão . . . 17
3.3.1 RQ1 - Quais os artigos que mencionam os tipos ou os modelos empregues na configuração automática de algoritmos meta-heurísticos (AAC) e como têm evoluído ao longo do tempo? . . . 17
3.3.2 RQ2- Quais os artigos que discutem a prática de ajuste dinâmico/online de parâmetros em algoritmos meta-heurísticos? . . . 19
3.3.3 RQ3 - Que artigos explicam os critérios que estão subjacentes ao ajuste dinâmico de algoritmos meta-heurísticos (populacionais) e como contribuem para a eficiência da solução global? . . . 21
3.3.4 RQ4 - Que artigos referem o contributo de meta-heurísticas na atividade hospitalar, em especial, no escalonamento de blocos operatórios? . . . 24

3.4 Resumo 25

4 Configuração Dinâmica de Algoritmos **27**
4.1 Conceptualização 27
4.2 Heurísticas e Parâmetros 31
4.2.1 Genetic Algorithm 32
4.2.2 Particle Swarm Optimization 35
4.3 Configuração Dinâmica de Algoritmos por Aprendizagem 38
4.3.1 Método SARSA 38
4.3.2 Método Deep SARSA 43
4.4 Tecnologias 51
4.5 Resumo 52

5 Resultados - Casos de Estudo **55**
5.1 Abordagem Preliminar 55
5.1.1 Método Manual 56
5.1.2 Otimização de Funções *Benchmark* 56
5.2 Otimização de Portfólio em Mercado de Eletricidade 62
5.2.1 Definição do Problema 62
5.2.2 Formulação Matemática 63
5.2.3 Resultados 65
5.3 Alocação de Doentes a Salas de Cirurgias 70
5.3.1 Definição do Problema 71
5.3.2 Formulação Matemática 72
5.3.3 Resultados 73
5.4 Alocação de Pacientes em Unidades de Cuidados Intensivos 75
5.4.1 Definição do Problema 76
5.4.2 Formulação Matemática 78
5.4.3 Resultados 79
5.5 Robustez dos Resultados: Teste de Hipóteses e ANOVA 81
5.6 Segurança e Privacidade dos Dados 82
5.7 Resumo 82

6 Conclusões **85**
6.1 Conclusões 85
6.2 Limitações e Trabalhos Futuros 86
6.3 Questões Éticas e Proteção de Dados 87

Bibliografia **89**

A Algoritmos **95**

B Casos de Estudo **99**

B.1 Optimização de Portfolio em Mercado de Energia 99
B.2 Alocação de Pacientes em Salas de Cirurgia 101
B.3 Alocação de pacientes em UCI 104

Lista de Figuras

1.1 Fluxograma sobre a metodologia proposta por Tiago Pinto et al. [7] para otimização de portfólio no mercado de energia 3
1.2 Número de doentes inscritos para cirurgias (SIGIC) e Número de LIC (Lista de Inscritos em Cirurgia) com tempo de espera inferior ao Tempo Máximo de Resposta Garantida (TMRG) em Portugal 4
1.3 Um cenário de configuração que inclui um algoritmo a ser configurado e uma coleção de instâncias de problemas adaptado de H.Hutter et al. [19] 6

3.1 Diagrama de fluxo de resultados de acordo com as perguntas de investigação . . . 17
3.2 Classificação dos métodos de configuração automática de algoritmos definida por Yasemin Eryoldaş [40]. 18
3.3 Esquema de classificação dos métodos de configuração automática de algoritmos de instâncias específicas e alheias por Yasemin et al. [46] 20
3.4 Algoritmo DE utilizando a proposta OAC-DE [14] 21
3.5 Taxonomia global da definição de parâmetros por Eiben et al. [33]. 23

4.1 Visão geral de DAC *online* adaptado de Adriaesen et al. [6] 28
4.2 Em cada episódio usando RL a política de configuração ajusta o valor dos parâmetros do algoritmos baseado no estado, ação e recompensa 29
4.3 Fluxograma padrão do GA [64] 32
4.4 Movimento de partículas de PSO adaptado de April Wang [70] 36
4.5 Mapa de calor da tabela Q-Value após execução da função *benchmark schwefel* . 41
4.6 Mecanismo do *Colab* desenhado por Biyanka Ekanayake [84] 52

5.1 Evolução dos Parâmetros nas Funções de Benchmark: Rosenbrock e Schwefel com 32 episódios 60
5.2 Evolução dos Parâmetros nas funções *Benchmark*: Rosenbrock e Schwefel em 50 episódios 61
5.3 (a) Volume de Compra e Venda de (Mw) em cada Mercado; (b) Preço de eletricidade em cada Mercado 67
5.4 Otimização do algoritmo PSO através do método Deep SARSA no problema de Portfólio (Episódios = 32) 68
5.5 Otimização do algoritmo PSO através do método Deep SARSA no problema de Portfólio (Episódios = 50) 69
5.6 Cronograma de planeamento de pacientes por salas de cirurgia 74

5.7 Atribuição de doentes a camas de UCI ao longo dos dias 81

B.1 Cronograma completo da melhor solução (B) encontrada na DAC para o problema da alocação de cirurgias 103

Lista de Tabelas

3.1 Questões de pesquisa 14
3.2 Fontes de dados eletrónicas 14
3.3 Domínios e termos-chave usadas para selecionar os termos de pesquisa 15
3.4 Consultas de pesquisa 15
3.5 Critérios incluídos 16

4.1 Limites de intervalos de valores, dentro dos quais os parâmetros do algoritmo (GA) podem variar 34
4.2 Tabela de ações implementadas para ajuste de parâmetros do algoritmo GA . . . 35
4.3 Notação e significado 35
4.4 Limites de intervalos de valores, dentro dos quais os parâmetros do algoritmo (PSO) podem variar 37
4.5 Tabela de ações implementadas para ajuste de parâmetros do algoritmo PSO . . . 37
4.6 Lista de estados implementados 40

5.1 Lista de funções *Benchmark* 57
5.2 Resultados da otimização do GA através dos métodos manual e SARSA em funções *Benchmark* para 32 episódios 58
5.3 Resultados da otimização do GA através dos métodos SARSA e Deep SARSA em funções *Benchmark* para 5 execuções 59
5.4 Configuração de parâmetros apurados nas funções *Rosenbrock* e *Schwefel* através do método Deep SARSA 62
5.5 Resultados da otimização do GA através dos métodos manual SARSA e Deep SARSA no problema de Portfólio para 32 episódios 66
5.6 Conjunto de parâmetros apurados para o algoritmo GA no problema de Portfólio através do método SARSA 66
5.7 Resultados da otimização do GA e PSO através dos métodos manual SARSA e Deep SARSA no problema de Portfólio para 5 execuções 67
5.8 Disponibilidade das salas durante 3 dias 73
5.9 Resultados da otimização do GA e PSO através dos métodos manual e Deep SARSA no problema de unidades de cirurgias para 5 execuções 73
5.10 Conjunto de parâmetros apurados para o algoritmo GA no problema de Unidades de Cirurgia através do método Deep SARSA 74
5.11 Distribuição dos pacientes por salas e por dias 75

5.12 Lista pacientes indicados para UCI 77
5.13 Resumo de penalidades e pesos adotados 79
5.14 Resultados da otimização do GA através dos métodos manual e Deep SARSA no problema de UCI para 5 execuções 80
5.15 Análise de robustez em resultados de custo médio apresentados em diferentes tabelas 82

B.1 Tipos de mercado onde cliente pode participar na ótica de comprador ou vendedor e limites de energia (MW) definidos por mercado 99
B.2 Preços esperados por mercado e período de negociação 99
B.3 Disponibilidade de mercado - Tipos de balanceamento de mercado (ativo/inativo) 100
B.4 Consumo Esperado - Dados de consumo (MW) 100
B.5 Tabela de resultados pbservados nos métodos Manual, SMAC e SARSA usando a heurística *Genetic Algorithm* (GA, Algoritmo Genético) em 32 episódios 100
B.6 Configurações dos parâmetros para o algoritmo GA para os melhores resultados (*Best*) obtidos na tabela anterior 101
B.7 Informação de cirurgias para doentes elegíveis 101
B.8 Disponibilidade das salas de cirurgia durante 3 dias 101
B.9 Otimização do agendamento de cirurgias usando os algoritmos, GA e PSO em 5 execuções 102
B.10 Lista de soluções resultantes da configuração de parâmetros 102
B.11 Resultados da configuração dos parâmetros para os algoritmos GA e PSO 102
B.12 Dados dos pacientes 104
B.13 Otimização do alocação UCI usando o algoritmo GA em 5 execuções 104

Lista de Símbolos

m	distância	metro
w	potência	whatt
Mw	potência	mega-watt
s	tempo	segundo
min	tempo	minuto
P	problema de otimização	Otimização
S	espaço de soluções	Otimização
Ω	conjunto de restrições entre as variáveis de decisão	Otimização
f	função objetivo que precisa ser otimizada	Otimização
a	ação a tomar no conjunto de ações A definidas	SARSA e D-SARSA
s	estado no conjunto de estados S definidos	SARSA e D-SARSA
r	recompensa obtida	SARSA e D-SARSA
$Q(s,a)$	elemento da tabela Q, correspondente a uma determinada ação a e um estado s	SARSA e D-SARSA
α	taxa de aprendizagem	SARSA e D-SARSA
γ	fator de desconto	SARSA e D-SARSA
ϵ	probabilidade de uma determinada ação	SARSA e D-SARSA

Lista de Acrónimos

AAC *Automatic Algorithm Configuration* (AAC, Configuração Automática de Algoritmos).

ACO *Ant Colony Optimization* (ACO, Otimização por Colónia de Formigas).

AI *Artificial Intelligence* (AI, Inteligência Artificial).

ANOVA Analysis Of Variance (ANOVA, Análise da Variância).

AOS *Adaptive Operator Selection*.

ASP *Aerial Surveying Problem* (ASP, Problema de Levantamento Aéreo.

CMA-ES *Covariance Matrix Adaptation Evolution Strategy*.

CSP *Constraint Satisfaction Problem* (CSP, Problema de Satisfação por Restições).

DAC *Dynamic Algorithm Configuration* (DAC, Configuração Dinâmica de Algoritmos).

DDQN *Double Deep Q-Network*.

DE *Differential Evolution* (DE, Evolução Diferencial).

DEAP *Distributed Evolutionary Algorithms in Python*.

Deep Q-Network *Deep Q-Network*.

Deep SARSA *Deep State-Action-Reward-State-Action*.

DRL *Deep Reinforcement Learning* (DRL, Aprendizagem por Reforço Profunda).

EA *Evolution Algorithm* (EA, Algoritmo Evolutivo).

ES *Evolution Strategy* (EA, Evolução Estratégica).

GA *Genetic Algorithm* (GA, Algoritmo Genético).

ICA *Imperialist Competitive Algorithm* (ICA, Algoritmo de Competição Imperialista).

IPTS *Instance-Specific Parameter Tuning Strategies* (IPTS, Estratégias de Ajuste de Instâncias Especificas para Instâncias).

MDP *Markov Decision Process* (MDP, Processo de Decisão Markov).

MKP *Multiple Knapsack Problem* (MKP, Problema da Mochila Múltipla).

ML *Machine Learning* (ML, Aprendizagem Máquina).

MSE *Mean Squared Error* (MSE, Erro Quadrático Médio).

OAC *Online Algorithm Configuration* (OAC, Configuração *Online* de Algoritmos).

PRISMA *Preferred Reporting Items for Systematic Reviews and Meta-Analyses*.

PSO *Particle Swarm Optimization* (PSO, Otimização por Enxame de Partículas).

Q-Learning *Q-Learning*.

ReLU *Rectified Linear Unit*.

RL *Reinforcement Learning* (RL, Aprendizagem por Reforço).

SA *Simulated Annealing* (SA, Recozimento Simulado).

SARSA *State-Action-Reward-State-Action*.

SMAC *Sequential Model-based Algorithm Configuration*.

TD *Temporal Difference* (TD, Diferença Temporal).

TMRG Tempo máximo de resposta garantida.

UC Unidades de Cirurgia.

UCI Unidades de Cuidados Intensivos.

VM *Virtual Machine* (VM, Máquina Virtual).

Capítulo 1

Introdução

Este capítulo introdutório aborda os recentes avanços em *Artificial Intelligence* (AI, Inteligência Artificial) e *Machine Learning* (ML, Aprendizagem Máquina), nomeadamente o uso de algoritmos meta-heurísticos para resolver problemas complexos de otimização em diversos setores. A eficiência e a eficácia destes algoritmos dependem fortemente da escolha adequada de parâmetros, o que faz com que a *Dynamic Algorithm Configuration* (DAC, Configuração Dinâmica de Algoritmos), desempenhe um papel central no alcance de soluções ótimas. Discute-se a importância dos algoritmos meta-heurísticos neste contexto, além dos desafios e objetivos relacionados com a configuração automática e adaptativa de parâmetros, utilizando estratégias de reforço como ferramenta fundamental para aprimorar o desempenho destes algoritmos.

Ao longo deste capítulo, são explorados os conceitos-chave de otimização, a relevância dos algoritmos meta-heurísticos, bem como a necessidade de ajustá-los dinamicamente para enfrentar as mudanças e incertezas inerentes aos problemas reais. O objetivo é fornecer uma base sólida sobre como a configuração dinâmica pode melhorar a eficiência de soluções em áreas como alocação de recursos, otimização financeira e outras aplicações práticas.

1.1 Contexto

O avanço recente em áreas científicas e tecnológicas impulsionou um rápido crescimento na complexidade e escala dos desafios de otimização enfrentados na sociedade atual. O aumento na dificuldade deste tipo de problemas, em questões como a alocação de recursos, a localização de instalações [1], a programação e o planeamento, rotas de veículos [2] e a otimização de portfólios no âmbito financeiro, entre outros exemplos, tornaram-se mais difíceis e complexos, muitas vezes, classificados de *NP-hard*.

Com o intuito de se alcançar o melhor resultado neste contexto, o conceito de otimização envolve, inúmeras vezes, a maximização do lucro ou minimização de custos [3]. Este desafio é particularmente evidente num ambiente de recursos escassos, onde a procura, frequentemente, excede a oferta disponível. A complexidade dos problemas citados requer, muitas vezes, abordagens mais avançadas, como a utilização de algoritmos meta-heurísticos, que são estratégias flexíveis e

adaptativas para explorar espaços de soluções em situações onde não há uma descrição matemática clara do problema ou quando a exatidão obriga a um esforço computacional impraticável.

A eficácia de algoritmos meta-heurísticos está intrinsecamente ligada à configuração de parâmetros. Dado o número significativo de parâmetros ajustáveis nestes algoritmos, a escolha adequada de valores é crucial para otimizar o seu desempenho, pelo que a configuração de parâmetros é uma prática fundamental na otimização, em especial, de meta-heurísticas [4]. Neste contexto, surge a questão da adaptabilidade dinâmica (*online*) versus o ajuste estático (*offline*). O controle dinâmico possibilita a modificação contínua dos parâmetros durante a execução, sendo essencial para lidar com mudanças nas condições do problema em tempo real. Em contrapartida, o ajuste estático envolve a determinação de valores ideais antes da execução.

A DAC tem ganho destaque na configuração de algoritmos, proporcionando soluções práticas e eficazes. Esta abordagem é amplamente utilizada em contextos onde os parâmetros têm um grande impacto no desempenho do algoritmo e apresenta várias vantagens, como a eficiência, uma vez que permite explorar melhor o espaço de parâmetros, encontrando combinações que levam a um aumento do desempenho. Além disso, a automação elimina a necessidade de tentativa e erro manual, facilitando o processo de ajuste [5]. Em alguns casos, também oferece adaptabilidade, permitindo que os parâmetros sejam ajustados dinamicamente durante a execução do algoritmo, de acordo com as condições observadas, o que resulta num comportamento mais robusto e flexível. Adriaensen et al. referem o DAC como uma abordagem dinâmica, sugerindo um cronograma de taxas de aprendizagem que pode depender de fatores como o tempo, alinhamento de gradientes anteriores e erros de treino/validação [6].

Este trabalho aborda a aplicação de ajuste dinâmico de parâmetros em algoritmos de otimização, tendo em vista responder a vários desafios da vida quotidiana em diferentes áreas, demonstrando a sua aplicabilidade em problemas diversos e de alta relevância prática.

Um exemplo significativo desta aplicação é a otimização de carteiras de compra e venda de energia, como o mercado ibérico. A necessidade de uma metodologia eficiente de otimização para fornecer o melhor perfil de investimento é destacada por Tiago Pinto et al., que exploram a aplicação dos algoritmos *Genetic Algorithm* (GA, Algoritmo Genético), *Particle Swarm Optimization* (PSO, Otimização por Enxame de Partículas) e métodos híbridos [6]. Estas abordagens visam melhorar a eficácia das estratégias de negociação de energia, considerando as especificidades e complexidades do mercado.

A otimização de uma carteira de compra e venda em mercados de energia, é primordial para maximizar lucros e minimizar riscos, garantindo que as decisões de investimento sejam as mais vantajosas possível dentro de um cenário competitivo e volátil. Tiago Pinto et al. propõem uma metodologia de otimização de portfólio, que fornece o melhor perfil de investimento para um *player* de mercado, considerando diferentes oportunidades de mercado [7].

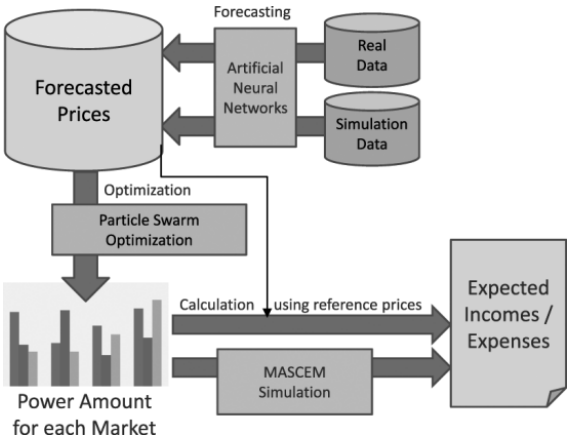


FIGURA 1.1: Fluxograma sobre a metodologia proposta por Tiago Pinto et al. [7] para otimização de portfólio no mercado de energia

Na ilustração 1.1 pode observar-se a utilização do algoritmo meta-heurístico PSO no processo de otimização para gestão de ativos energéticos.

Assim como na gestão de ativos energéticos, onde a otimização é crucial para maximizar lucros, no setor da saúde a eficácia na alocação de recursos também é fundamental.

Nesta conformidade, outro exemplo pertinente é a otimização dos blocos operatórios em hospitais e a alocação de doentes em unidades de cuidados intensivos. Com o envelhecimento da população, o aumento das intervenções cirúrgicas, o surgimento de novas doenças e restrições orçamentais mais rigorosas, os hospitais enfrentam uma crescente pressão para proporcionar cirurgias de alta qualidade a custos mínimos. A eficiência na gestão do bloco operatório, unidade nevrálgica do ponto de vista financeiro de um hospital, é crucial para reduzir custos, melhorar a satisfação dos pacientes e elevar a moral da equipa médica [8]. Portanto, otimizar o número de intervenções e os custos operacionais de unidades de cirurgia ou de cuidados intensivos torna-se uma prioridade para os gestores hospitalares.

Em todo o mundo, milhões de candidatos a cirurgias não urgentes aguardam atendimento. Os médicos e as equipas das unidades de cirurgia (UC), estão exaustos de tantas horas extras para atender às necessidades e, consequentemente, os hospitais sofrem dificuldades financeiras. Nada é mais perturbador que desmarcar cirurgias no próprio dia devido a problemas de agendamento e nada é mais caro para os hospitais do que uma sala cirúrgica subutilizada. As salas cirúrgicas estão sobrecarregadas com custos fixos. Um estudo de 2014, abrangendo mais de 300 hospitais da Califórnia, demonstrou que, em média, administrar uma sala de cirurgia custa US$ 37 a cada 60 segundos, o que equivale a cerca de US$ 2.220 por hora [9].

Na Europa, mais concretamente em cinco hospitais holandeses Sejal Patel et al. [10] estimou um custo de 9,45€ (8,60€-10,23€) e 19,88€ (16,10€-23,07€) para o bloco operatório convencional e híbrido, respetivamente. Os principais fatores que influenciam os custos do bloco operatório são: custos totais de inventário, custos totais de construção, taxa de utilização e custos totais de pessoal.

Em Portugal, a situação das listas de espera para cirurgia é uma preocupação crescente. Na figura 1.2, podemos ver que em Novembro de 2023 existiam segundo o portal do Estado *(transparencia.sns.gov.pt)* um total de 2.186.680 doentes inscritos. No entanto, destes, apenas 1.567.118 estão dentro dos limites estabelecidos pelo Estado, que determina um Tempo máximo de resposta garantida (TMRG) de 180 dias. Este dado revela uma realidade preocupante, já que aproximadamente 28% dos pacientes enfrentam tempos de espera que ultrapassam o prazo previsto pelo Estado. A necessidade de abordar eficazmente este problema torna-se evidente, sendo necessárias soluções que assegurem uma gestão mais eficiente das listas de espera e garantam o acesso oportuno e equitativo aos serviços de saúde.

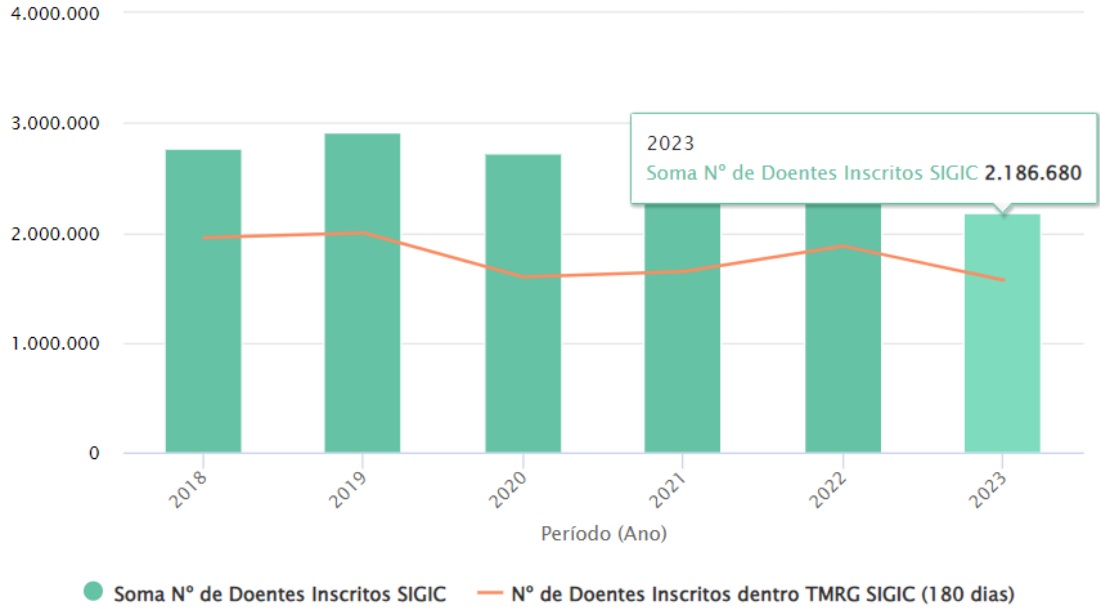


FIGURA 1.2: Número de doentes inscritos para cirurgias (SIGIC) e Número de LIC (Lista de Inscritos em Cirurgia) com tempo de espera inferior ao Tempo Máximo de Resposta Garantida (TMRG) em Portugal

O problema da gestão eficiente de blocos operatórios, envolve desafios como a atribuição de salas de cirurgia, agendamento de cirurgias, otimização de tempo, gestão de equipas médicas, minimização de custos e adaptação a mudanças.

Para David McMahon, a gestão de blocos operatórios e os tempos dos procedimentos cirúrgicos são difíceis de prever [11]. Os tempos "super-utilizado"e "subutilizado"são fatores que afetam o fluxo de trabalho da sala de cirurgia, pois ambos geram tempos mortos ou não produtivos. O tempo "subutilizado"significa que a sala de operações está livre, possivelmente uma cirurgia que terminou antes do tempo e o próximo caso não está pronto para entrar naquela sala. O tempo "super-utilizado"envolve a realização de horas extras na sala de cirurgia, ou seja, o tempo real das cirurgias excede o tempo atribuído àquela sala. Ambos os tempos têm um custo para o hospital, porém o custo da super utilização é 2,5 vezes maior que o custo da subutilização.

As práticas mais avançadas e as tendências atuais destacam o uso inovador de técnicas de aprendizagem máquina (ML) para a configuração automática e otimizada de algoritmos. O termo "configuração dinâmica de algoritmos"(DAC) foi introduzido recentemente por Biedenkapp et al. [12]. Esta questão tem recebido uma atenção considerável por parte da comunidade cientifica nos últimos anos, dada a capacidade de aprender automaticamente políticas capazes de controlar o ajuste inteligente de parâmetros das heurísticas, baseado em dados [13]. Essa abordagem proporciona uma otimização adaptativa, demonstrando que alguns parâmetros do algoritmo são melhor ajustados dinamicamente durante a execução [6].

Outra prática emergente é a integração de abordagens híbridas, combinando heurísticas: configuração automática e técnicas de otimização clássicas. Por exemplo, a abordagem por *Online Algorithm Configuration* (OAC, Configuração *Online* de Algoritmos) para o algoritmo *Differential Evolution* (DE, Evolução Diferencial) para adaptar a sua configuração de forma orientada por dados [14]. Esta sinergia entre métodos tradicionais e inovadores visa alcançar resultados mais robustos, explorando as vantagens de cada abordagem para enfrentar a complexidade crescente dos desafios na gestão de blocos operatórios. Essa flexibilidade e adaptabilidade refletem a procura por soluções mais eficazes e abrangentes.

A técnica de combinar heurísticas e aprendizagem por reforço pode também ser uma estratégia avançada e adaptativa para otimizar a gestão de blocos operatórios. Em particular, os métodos contemporâneos de *Reinforcement Learning* (RL, Aprendizagem por Reforço) que fazem uso de redes neuronais profundas, têm a capacidade de aprender representações significativas de forma eficiente. Essa abordagem permite que tais métodos lidem com espaços complexos de estado e ação. Um exemplo disso é o conceito de *Double Deep Q-Network* (DDQN), que foi introduzido por Hansen [15].

1.2 Problema

O problema da configuração de parâmetros de um algoritmo abordado neste trabalho pode ser descrito da seguinte maneira: dado um algoritmo, um conjunto de parâmetros associados e um conjunto de dados de entrada, em que o objetivo é identificar os valores desses parâmetros de forma *automática* e em *tempo de execução do algoritmo* que lhe permitam alcançar o melhor desempenho possível.

A parametrização de algoritmos meta-heurísticos é o desafio central da otimização, pois esses parâmetros influenciam significativamente o desempenho e a eficácia do algoritmo na pesquisa de soluções ótimas ou quase ótimas em problemas complexos. Parâmetros de algoritmos meta-heurísticos como a taxa de mutação e cruzamento do GA ou o fator de inércia e coeficientes de aceleração do PSO, precisam ser ajustados cuidadosamente para equilibrar a *exploration* e *exploitation* [16]-[17] no espaço de pesquisa. Um ajuste inadequado pode levar a soluções subótimas, convergência prematura ou mesmo à incapacidade de encontrar uma solução viável.

Tradicionalmente, a configuração desses parâmetros tem sido feita de forma *ad hoc*, baseada na intuição e experiência do investigador, seguida por uma série de experiências para identificar as configurações que proporcionam os melhores resultados. Esta metodologia pode levar a bons resultados em certos casos, mas é extremamente dependente do conhecimento humano, consome tempo, e geralmente explora um espaço de parâmetros bastante limitado, o que pode resultar na omissão de combinações potencialmente mais eficazes.

Vários autores, entre eles Eiben e Selmer Smit [18], oferecem uma revisão sobre a prática do ajuste manual de parâmetros em algoritmos evolutivos, discutindo as suas limitações e a necessidade de métodos automatizados para melhorar a eficiência do processo de ajuste. Consequentemente, a pesquisa por abordagens sistemáticas de configuração automática de parâmetros tem ganho destaque na literatura, procurando superar as limitações temporais e explorar de forma mais abrangente o espaço de parâmetros.

Por exemplo, a figura 1.3 ilustra um cenário de configuração automática de algoritmos, adaptado de Holger Hutter et al., onde o objetivo é otimizar os parâmetros de um algoritmo-alvo para resolver uma coleção de instâncias de problemas.

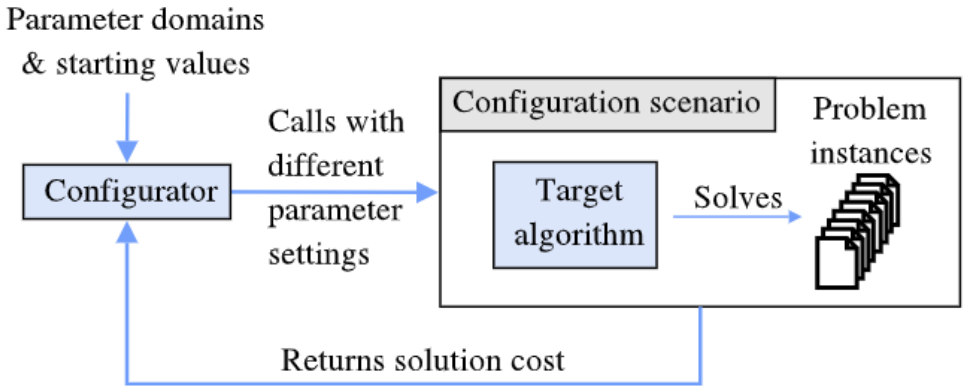


FIGURA 1.3: Um cenário de configuração que inclui um algoritmo a ser configurado e uma coleção de instâncias de problemas adaptado de H.Hutter et al. [19]

Esse ciclo de otimização iterativo envolve a repetição de chamadas do algoritmo com diferentes parâmetros até encontrar a configuração ideal, mas o ajuste ocorre antes do algoritmo ser utilizado para resolver novos problemas em tempo real. Isso caracteriza uma abordagem *offline*, pois a otimização dos parâmetros ocorre previamente.

1.3 Objetivos

Este trabalho tem, de uma forma geral, como objetivo o desenvolvimento do impacto do controle ou ajuste dinâmico de parâmetros em otimização de algoritmos, isto é, os parâmetros do algoritmo são ajustados durante a execução, em tempo real, com base nas condições do ambiente ou nas instâncias de problemas, a serem resolvidos no momento.

A importância da configuração dinâmica de parâmetros reside na sua capacidade de adaptar o comportamento em tempo real do algoritmo às características específicas do problema, potencialmente superando as limitações das abordagens estáticas tradicionais. Ao ajustar os parâmetros dinamicamente, é possível melhorar a exploração do espaço de pesquisa, evitar convergências prematuras e adaptar o algoritmo às diferentes fases da pesquisa.

Em concreto, pretende-se projetar o trabalho de forma flexível, permitindo a adaptação a alguns algoritmos e tipos de problemas, enquanto são incorporadas diferentes estratégias de ajuste de parâmetros. Além disso, é crucial considerar a robustez de uma ferramenta em ambientes dinâmicos, onde as características do problema podem mudar ao longo do tempo. Esses objetivos combinados visam criar uma solução adaptativa e abrangente, capaz de lidar com os desafios encontrados em diferentes contextos e cenários de otimização.

Em relação ao tipo de problema, este trabalho, com o auxilio de algoritmos meta-heurísticos, tem como objetivo, criar soluções eficazes e eficientes em relação à tomada de decisão no agendamento ou escalonamento de cirurgias em blocos operatórios ou na maximização do investimento de uma carteira de compra e venda de energia.

1.4 Contribuições

O principal contributo deste projeto é o desenvolvimento de uma abordagem inovadora para a configuração dinâmica online de algoritmos genéticos utilizando métodos de aprendizagem por reforço. Este trabalho destaca a importância da configuração automática de algoritmos, facilitando a adaptação e a eficiência dos modelos, e permitindo que os utilizadores se concentrem mais na implementação e na otimização dos resultados em vez de na configuração técnica dos algoritmos.

Alguns pontos específicos deste trabalho foram formalmente apresentados num artigo científico intitulado **Dynamic Online Parameter Configuration of Genetic Algorithms Using Reinforcement Learning**, escrito por Vítor Oliveira, Tiago Pinto e Carlos Ramos. Este artigo explora a forma como a integração de técnicas de aprendizagem por reforço pode aprimorar a configuração de algoritmos genéticos, oferecendo uma solução adaptativa e eficiente para a otimização de parâmetros em tempo real. Este artigo recebeu financiamento dos fundos nacionais através da FCT - Fundação para a Ciência e a Tecnologia, no âmbito do projeto LA/P/0063/2020. O artigo foi aceite para apresentação na conferência EPIA 2024 (*European Conference on Artificial Intelligence*), um evento de destaque no campo da inteligência artificial. A EPIA é reconhecida pelo seu rigor académico e pela qualidade dos trabalhos apresentados, refletindo a relevância e o impacto do nosso trabalho na comunidade científica.

Além disso, está em preparação um outro artigo para submissão num *journal* internacional, centrado no desenvolvimento e aplicação das abordagens *SARSA* e *D-SARSA* no contexto da saúde. Pretende-se que essas abordagens de aprendizagem por reforço possam ser aplicadas a tomadas de decisões em ambientes críticos da área da saúde, contribuindo para a melhoria da gestão de recursos.

O desenvolvimento desta pesquisa contribui igualmente para o avanço contínuo no campo da otimização adaptativa e da aplicação de técnicas de inteligência artificial em problemas complexos, promovendo uma melhor compreensão e aplicação prática dessas abordagens em diferentes domínios.

1.5 Estrutura do Documento

Esta secção indica a estrutura da dissertação. Assim, neste capítulo introdutório, enquadram-se todos os aspetos relevantes da definição do problema, motivação e objetivos. No capitulo 2 são introduzidos conceitos e modelos que suportam a área de conhecimento de enquadramento da tese, bem como para uma melhor compreensão do capítulo seguinte (3), composto pela revisão da literatura. No capítulo 4, é apresentada a conceptualização da solução, os modelos de referência e as técnicas de aprendizagem por reforço empregues, nomeadamente o SARSA e o Deep SARSA. O capítulo seguinte (5) explora a aplicação destes métodos em problemas de otimização, abordando diferentes casos de estudo, como funções *benchmark*, otimização de portfólios, alocação de doentes a blocos cirúrgicos e a enfermarias de cuidados intensivos. No final deste capítulo, é apresentada uma análise detalhada dos dados recolhidos, discutidas as implicações dos resultados e indicadas possíveis direções para futuras pesquisas. O ultimo capitulo (6) é dedicado à conclusão da dissertação, resumindo os principais pontos e refletindo sobre a contribuição desta pesquisa para o campo da configuração dinâmica de algoritmos.

1.6 Resumo

Neste capítulo introdutório, destacou-se a crescente complexidade dos desafios de otimização em problemas quotidianos. Referiu-se a questão de práticas inovadoras, como a DAC e a integração de abordagens híbridas. Definiram-se os objetivos do trabalho, tendo como foco o impacto do controle dinâmico de parâmetros em algoritmos de otimização, adaptáveis a diferentes contextos, nomeadamente, na gestão de blocos operatórios e unidades de cuidados intensivos em ambientes hospitalares, e gestão de ativos em mercados de energia, na área financeira.

Capítulo 2

Enquadramento Teórico

Neste capítulo, abordam-se os conceitos fundamentais relacionados com a temática da otimização, bem como alguns dos modelos mais relevantes desenvolvidos pela comunidade científica. Referimos-nos a algoritmos exatos, heurísticos e meta-heurísticos, no intuito de explorar a sua eficiência e eficácia na pesquisa pela solução ótima ou quase ótima num espaço de soluções.

Para atingir este objetivo, dedica-se atenção à escolha criteriosa, aos valores atribuídos e à importância dos diversos parâmetros dos algoritmos, pois desempenham um papel crucial na obtenção de soluções próximas do ótimo desejado. Discutem-se os conceitos associados ao ajuste manual, *offline* e *online* ou controle de parâmetros.

Este capítulo visa estabelecer as bases para uma compreensão mais aprofundada dos elementos essenciais que abrangem o campo da otimização, preparando o terreno para a análise dos modelos e técnicas que serão explorados mais à frente neste trabalho, sobre o controle ou ajuste online de parâmetros.

2.1 Otimização

Por otimização, entende-se a função de maximizar o resultado com recursos limitados, selecionando a melhor solução e minimizando os riscos, a partir de um conjunto de soluções disponíveis. Em geral, os problemas de otimização, segundo Basseur Jourdan et al. podem ser classificados por diferentes abordagens, por métodos exatos, heurísticos e meta-heurísticos [20].

Os métodos exatos, como por exemplo, o *Branch and Bound Algorithm* [21], garantem a solução mais precisa, se for executada até ao final. No entanto, quando as instâncias se tornam demasiado grandes, a otimização pode ser computacionalmente intensiva e demorada e, por isso, recorre-se frequentemente a heurísticas, em particular, a meta-heurísticas, heurísticas de nível superior. Estas são abordagens mais simples e rápidas para encontrar soluções aproximadas para problemas de otimização. Não garantem a solução ótima, mas são hábeis no que toca a encontrar boas soluções em tempo razoável.

Todos os problemas de otimização têm algumas variáveis de decisão, uma determinada função objetivo e algumas restrições. As técnicas de otimização são utilizadas para obter os valores das variáveis de decisão que otimizam uma função objetivo sujeita a determinadas restrições [22].

Formalmente, o problema de otimização P é definido como:

$$P = (S, \Omega, f) \tag{2.1}$$

onde S é o espaço de pesquisa definido sobre um conjunto finito de variáveis de decisão $X_i, i = 1, \ldots, n$; Ω é um conjunto de restrições entre as variáveis de decisão; f é uma função objetivo que precisa ser otimizada.

Para resolver P, uma solução s^* de P deve ser encontrada em :

- $f(s^*) \leq f(s), \forall s \in S$, no caso valor mínimo da função objetivo ou,
- $f(s^*) \geq f(s), \forall s \in S$, no caso valor máximo da função objetivo

Os problemas de otimização podem variar de acordo com a sua estrutura: problemas de otimização com um ou vários objetivos, com ou sem restrições, ou combinatórios. Os problemas de otimização que se baseiam num objetivo específico são considerados problemas de objetivo único ou problemas que podem ter vários objetivos combinados para formar um único objetivo principal. Os problemas de natureza multi-objetivo que envolvem múltiplos critérios ou objetivos contraditórios, devem ser satisfeitos com base em determinadas restrições [22].

2.2 Meta-heurísticas

Uma meta-heurística é uma estratégia de otimização de algoritmo de pesquisa para encontrar soluções aproximadas em problemas de otimização difíceis. Estas técnicas são geralmente aplicadas a problemas complexos, nos quais os métodos tradicionais de otimização podem não ser eficientes ou podem levar muito tempo até encontrar uma solução [23]. As meta-heurísticas são projetadas para explorar eficientemente o espaço de soluções na procura de melhores respostas, sem garantir a obtenção do ótimo global. Elas são frequentemente utilizadas em problemas *NP-hard*, conhecidos na teoria da complexidade computacional por descreverem tarefas que são consideradas desafiadoras, sem soluções competentes conhecidas até o momento.

Yang et. al. referem que muitos problemas de difícil complexidade, em atividades de gestão, economia e engenharia precisam de otimização e que estas meta-heurísticas podem fazer "*magia*"[24].

Existem várias meta-heurísticas populares, incluindo *Genetic Algorithm* (GA, Algoritmo Genético) [25], *Ant Colony Optimization* (ACO, Otimização por Colónia de Formigas) [26], *Particle Swarm Optimization* (PSO, Otimização por Enxame de Partículas) [27], *Simulated Annealing* (SA, Recozimento Simulado) [28], *tabu search* [29], entre outras. Essas abordagens são inspiradas em

processos naturais, comportamentos sociais ou outros fenómenos complexos, procurando capturar princípios gerais que possam ser aplicados a uma variedade de problemas de otimização.

2.3 Configuração de Parâmetros

Os algoritmos meta-heurísticos, como GA, PSO, SA, entre outros, possuem parâmetros que precisam ser configurados antes do início do processo de otimização. O desempenho de uma meta-heurística deriva da configuração de parâmetros, porque estes controlam o comportamento heurístico do algoritmo [30]. Por exemplo, no algoritmo SA, as soluções produzidas resultam da escolha dos seus parâmetros, como a temperatura inicial, o fator de resfriamento ou o número de interações.

A otimização do desempenho de algoritmos por meio do ajuste de parâmetros é uma etapa fundamental no desenvolvimento e aplicação de meta-heurísticas. A automação desse processo, envolvendo a criação de procedimentos algorítmicos para gerir o ajuste de parâmetros, tem recebido significativa atenção por parte de investigadores e profissionais ao longo das últimas duas décadas. Diversas abordagens automáticas de ajuste de parâmetros foram propostas nesse período, visando aprimorar a eficiência e eficácia desses algoritmos.

A literatura propõe duas abordagens principais para o ajuste de parâmetros em algoritmos de otimização: o ajuste de parâmetros *offline* e o ajuste *online* (também conhecido como controle de parâmetros). No ajuste *online* os valores iniciais dos parâmetros são fixados e modificados durante o processo de afinação, sendo classificados em determinístico, adaptativo ou auto-adaptativo [31],

A adaptação de parâmetros envolve a alteração direta dos valores dos parâmetros conforme estratégias específicas durante a execução do algoritmo, apesar de enfrentar o desafio da não universalidade [32]. Projetar estratégias de controle requer compreensão detalhada das mudanças ideais nos parâmetros durante a execução, muitas vezes baseadas em informações históricas.

No ajuste de parâmetros *offline*, os valores dos parâmetros são definidos antes da execução do algoritmo e permanecem constantes durante o processo [33]. Este ajuste, embora universal (aplicado a muitas meta-heurísticas), é frequentemente demorado devido à necessidade de várias execuções do algoritmo em diferentes configurações.

Para Eiben et. al. no seu trabalho sobre configuração de *Evolution Algorithm* (EA, Algoritmo Evolutivo), são consideradas duas abordagens principais para estabelecer os valores dos parâmetros: afinação de parâmetros e controle de parâmetros [31]. A afinação de parâmetros refere-se à prática comum de encontrar valores adequados para os parâmetros antes da execução do algoritmo. Nesse processo, procura-se identificar valores eficazes que permanecerão fixos ao longo da execução do algoritmo. Por seu lado, o controle de parâmetros implica ajustar dinamicamente os seus valores durante a execução do algoritmo, permitindo a adaptação contínua com base no progresso e nas condições do problema.

No artigo de Changwu Huang et al. é apresentada uma análise abrangente dessas abordagens, introduzindo uma nova classificação (ou taxonomia) de métodos automáticos de ajuste de parâmetros de acordo com a estrutura desses métodos [14]. As abordagens existentes são classificadas em três categorias:

- métodos simples de geração-avaliação;
- métodos iterativos de geração-avaliação;
- métodos de geração-avaliação em alto nível.

Essas categorias são revistas sequencialmente, destacando as principais vantagens e desvantagens de cada método. Essa abordagem é útil para que novos pesquisadores ou profissionais escolham métodos de ajuste apropriados. Além disso, são apontados desafios e direções para pesquisas futuras neste campo.

Alguns parâmetros também são denominados hiperparâmetros porque não são ajustados automaticamente durante o treino do modelo, mas sim configurados externamente. Pelo contrário, os parâmetros do modelo são os valores internos que o modelo aprende durante o treino. Esses parâmetros são ajustados internamente pelo modelo para melhor se ajustarem aos dados de treino. Por exemplo, nos pesos de uma rede neuronal ou nos coeficientes de uma regressão linear.

Existem dois tipos de parâmetros em modelos de aprendizagem máquina: um que pode ser inicializado e atualizado através do processo de aprendizagem de dados (por exemplo, os pesos dos neurónios em redes neuronais), denominado parâmetros de modelo; enquanto os outros denominados hiperparâmetros, não podem ser estimados diretamente a partir da aprendizagem de dados e devem ser definidos antes do treino de um modelo de ML porque definem a arquitetura do modelo [34].

2.4 Resumo

No capítulo 2, introduziu-se o enquadramento teórico relacionado com a otimização, explorando conceitos fundamentais e modelos. Referiu-se a diversidade de abordagens, incluindo algoritmos exatos, heurísticos e meta-heurísticos, ressaltando a importância dos parâmetros nesse contexto. Em seguida, discutiu-se o papel das meta-heurísticas como abordagens proficientes para problemas de otimização difíceis. Foram apresentadas algumas meta-heurísticas populares, como GA, PSO e SA e enfatizou-se a capacidade dessas técnicas de encontrar soluções aproximadas em cenários desafiadores. Foi esclarecida a distinção entre ajuste de parâmetros *offline* e *online* (controle de parâmetros) e explorou-se a importância de escolher valores adequados para otimizar o desempenho das meta-heurísticas.

Capítulo 3

Revisão Literária

À medida que o conhecimento cientifico vai avançando nas diferentes áreas, a importância de investigação em pesquisa de artigos torna-se fundamental. Esta abordagem é essencial para responder a questões críticas e facilitar a disseminação competente do conhecimento.

A revisão sistemática da literatura deve ser, então, realizada com o propósito de examinar de forma abrangente e imparcial os estudos científicos existentes sobre um tema específico. Esse processo inclui a extração, avaliação crítica e interpretação da literatura disponível, geralmente com o objetivo de responder a uma pergunta de pesquisa claramente formulada. Conforme definido por Kitchenham et al., uma revisão da literatura é considerada uma síntese de estudos científicos da mais alta qualidade sobre um determinado tema ou questão de pesquisa [35].

3.1 Metodologia

Neste estudo, optou-se por uma abordagem metodológica inspirada na filosofia PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*), embora sem a intenção de seguir rigorosamente todas as suas etapas. O processo começa abordando as Questões de Pesquisa (RQs) e o problema que impulsiona o projeto. Em seguida, são detalhadas as fontes de dados escolhidas, o processo de seleção e a escolha criteriosa dos termos de pesquisa. São estabelecidos critérios claros para a inclusão e exclusão de estudos. Por fim, é desenvolvida e refinada uma descrição dos artigos mais relevantes que abordam as questões de pesquisa.

3.1.1 Perguntas de Investigação

Para esta revisão, formulamos a questão principal de pesquisa da seguinte forma: “Quais os modelos de adaptação *online* ou controle de parâmetros de algoritmos meta heurísticos e em que se diferenciam de métodos *offline*, num processo de pesquisa, aplicado a problemas de planeamento na área hospitalar?", dividimos a questão principal em quatro questões secundárias específicas.

A tabela 3.1 mostra a lista de perguntas de pesquisa.

TABELA 3.1: Questões de pesquisa

	Questões Pesquisa
RQ1	Quais os artigos que mencionam tipos ou modelos empregues na *Automatic Algorithm Configuration* (AAC, Configuração Automática de Algoritmos) meta-heurísticos e como têm evoluído ao longo do tempo?
RQ2	Quais os artigos que discutem a prática de ajuste dinâmico/online de parâmetros em algoritmos meta-heurísticos?
RQ3	Que artigos explicam os critérios que estão subjacentes ao ajuste dinâmico de algoritmos meta-heurísticos (populacionais) e como contribuem para a eficiência da solução global?
RQ4	Que artigos referem o contributo de meta-heurísticas na atividade hospitalar, em especial, no escalonamento de blocos operatórios?

3.1.2 Fonte de Dados

A escolha cuidadosa de fontes eletrónicas de artigos, juntamente com a aplicação de operadores lógicos, tem como objetivo assegurar uma pesquisa abrangente e profícua para identificar artigos relevantes.

Nas pesquisas, as fontes de dados desempenham um papel crucial ao fornecerem o material necessário para responder às questões da pesquisa, sendo a seleção adequada essencial para a sua qualidade e abrangência.

Em alguns casos, outros estudos podem adotar uma abordagem mais abrangente, recolhendo dados por meio de três métodos distintos - entrevistas, observações e recolha de documentos científicos - a fim de ampliar o espectro de evidências e perspetivas [36].

No âmbito desta investigação, a base de dados eletrónicos *B-on* foi utilizada como fonte principal, conforme a tabela 3.2.

TABELA 3.2: Fontes de dados eletrónicas

Identificador	**Base Dados**	**URL**
DS1	B-on	https://b-on.pt

3.1.3 Termos de Pesquisa

A pesquisa de termos-chave é uma etapa fundamental no processo de revisão sistemática, seguindo a metodologia PRISMA. Para filtrar os artigos eletrónicos mais importantes, foram adotados critérios de seleção de termos de pesquisa em cada um dos domínios relevantes para esta revisão.

Os termos de pesquisa foram elaborados de forma abrangente e incluíram sinónimos e variações relevantes para garantir a recolha capaz de artigos pertinentes. Foram definidos 6 domínios, que vão desde a configuração automática de algoritmos, relacionada com algoritmos meta-heurísticos

até à aplicabilidade destes em problemas da área hospitalar. A tabela 3.3 indica estes grupos e os respetivos termos:

TABELA 3.3: Domínios e termos-chave usadas para selecionar os termos de pesquisa

Domínio	**Termos-Chave**
AAC	*Automatic Algorithm Configuration or Algorithm Parameter Adaptation or Tuning Parameters or Evolution AAC Models*
Metaheuristics	*Metaheuristics Algorithms or Population-Based Metaheuristics*
Search	*Dynamic Parameter Tuning or Exploration an Exploitation or Criteria Search*
Artificial Intelligence	*Machine Learning or Deep Learning or Artificial Neural Network or AI Models*
Global Solution	*Optimal Solution Research or Global Solution Discovery*
Operating Theatre	*Operating Room or Surgery Scheduling*

De seguida, os termos de pesquisa foram combinados usando operadores lógicos como *"AND"* e *"OR"* para criar consultas de pesquisa específicas e eficazes.

Estas consultas, referenciadas na tabela 3.4, foram aplicadas às fontes de dados eletrónicas selecionadas, permitindo a identificação de artigos que abordam os temas de interesse nos domínios da configuração automática de parâmetros de algoritmos meta-heurísticos. As pesquisas por termos-chave foram aplicadas ao *resumo* de cada artigo.

TABELA 3.4: Consultas de pesquisa

Pesquisa	***String***
QS1	*"automatic algorithm configuration methods OR automatic algorithm configuration models"AND "optimization OR metaheuristic algorithms"*
QS2	*"algorithm configuration"AND "metaheuristic algorithms"AND "online OR dynamic OR realtime OR parameters control OR tune"*
QS3	*"Automated algorithm configuration AND parameter tuning"AND "criteria OR solution OR strategy"*
QS4	*"algorithm configuration"AND "metaheuristics algorithms"AND "operating theatre OR elective surgeries"AND "planning OR scheduling"*

3.1.4 Critérios de Inclusão e Exclusão

Os critérios estabelecidos para inclusão e exclusão na pesquisa servem para garantir a qualidade e relevância das fontes selecionadas. Por exemplo, a inclusão de fontes revistas por pares (IC1) e a exigência de que as fontes descrevam metodologias específicas relacionadas com a configuração

automática de algoritmos meta-heurísticos (IC3), direciona a seleção para aquelas mais alinhadas com os objetivos da pesquisa.

Por outro lado, critérios de exclusão, como a restrição a fontes com mais de 10 anos (EC1) e aquelas que não estão em inglês (EC2), procuram manter a pesquisa atualizada e com uma linguagem uniforme. A tabela 3.5 mostra todos os critérios estabelecidos.

TABELA 3.5: Critérios incluídos

	Include Criteria
IC1	*The source is peer-reviewed*
IC2	*The source describes a significant contribution to the domains of study*
IC3	*The source describes a methodology or a tool for the evolution or evaluation considering online adjust the automatic algorithm configuration of metaheutisc algorithms*
	Exclude Criteria
EC1	*The source is more than 10 years old*
EC2	*The source is not written in English*
EC3	*The source is a book chapter, a dissertation, a systematic review, or a thesis*
EC4	*The contribution to the field of control parameters of the metaheuristics algorithms is unclear or is not the aim of the source*

3.2 Resultados

O processo de pesquisa deste trabalho envolve várias etapas. Inicialmente, identificaram-se um total de 947 registos através de pesquisas em base de dados, aos quais adicionamos mais 23 registos de outras fontes. Após a remoção de duplicados, ficamos com 238 registos.

Em seguida, realizámos a triagem, analisando os 732 registos identificados, com recurso aos critérios de inclusão e exclusão identificados na tabela 3.5, nomeadamente aqueles de caráter mais objetivo, por exemplo, não estarem publicados em inglês ou terem mais de 10 anos. Durante esta fase, excluímos 219 registos que não atendiam aos critérios estabelecidos.

Na etapa de elegibilidade, avaliámos resumos e palavras-chave de 513 artigos. Dessas avaliações, 467 artigos foram excluídos por razões de insignificância para a pesquisa.

Finalmente, foram incluídos 46 estudos na síntese qualitativa, significa que esses estudos cumpriram todos os critérios de elegibilidade e foram considerados adequados para a análise final (avaliação do conteúdo e qualidade dos estudos).

A figura 3.1 mostra as etapas e os números do processo de pesquisa, desde a identificação inicial até a inclusão dos estudos para a análise qualitativa.

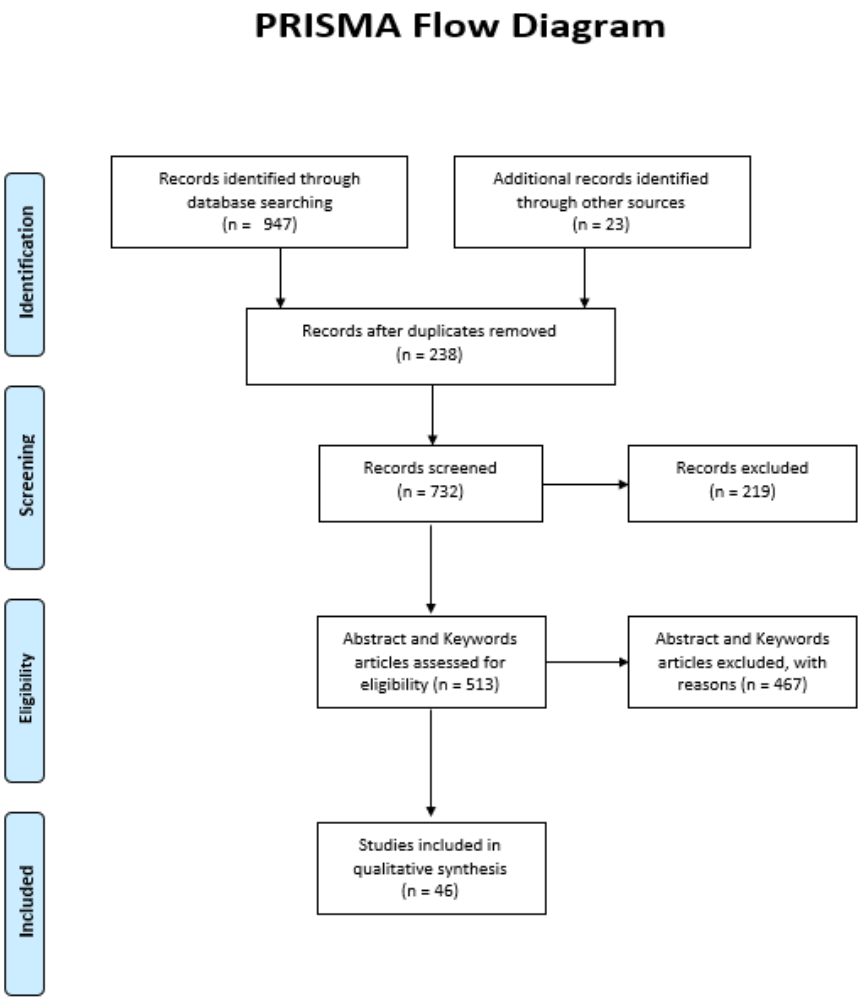


FIGURA 3.1: Diagrama de fluxo de resultados de acordo com as perguntas de investigação

3.3 Discussão

3.3.1 RQ1 - Quais os artigos que mencionam os tipos ou os modelos empregues na configuração automática de algoritmos meta-heurísticos (AAC) e como têm evoluído ao longo do tempo?

Em resposta às questões de pesquisa importa a forma como a configuração automática de algoritmos se processa.

A configuração automática de algoritmos é fundamental para otimizar o seu desempenho e emprega diversos métodos, dos quais se destacam: a pesquisa aleatória, que explora o espaço de parâmetros de forma aleatória; a otimização *bayesiana*, que utiliza modelos probabilísticos para encontrar configurações de maneira eficiente em funções objetivo; o *Sequential Model-based Algorithm Configuration* (SMAC), que emprega modelos preditivos de forma sequencial [37]; o ParamILS

(*Parameterized Algorithm Configuration Iterated Local Search*), que combina pesquisa local iterada com otimização de parâmetros [19]; o Irace (*Iterated Racing for Automatic Configuration of Algorithms*), que seleciona configurações eficazes de forma iterativa [38] e o MAC (*Many-objective Algorithm Configuration*), que lida com a otimização multi-objetivo na configuração de algoritmos [39]. Cada uma com suas particularidades, as abordagens mencionadas visam automatizar a escolha eficiente de parâmetros, maximizando o desempenho dos algoritmos em diferentes contextos.

Na figura 3.2 vemos uma representação de modelos ou configuradores de algoritmos *offline*.

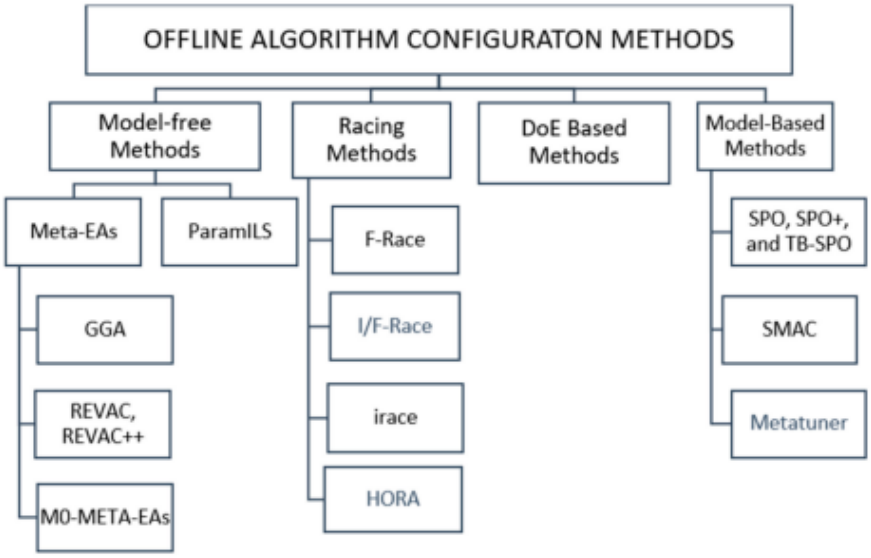


FIGURA 3.2: Classificação dos métodos de configuração automática de algoritmos definida por Yasemin Eryoldaş [40].

A configuração manual desses parâmetros é uma tarefa demorada e, muitas vezes, leva a resultados insuficientes. As diversas *frameworks* de configuração automática foram propostas para ajustar os parâmetros de algoritmos em diferentes instâncias de problemas.

Sabe-se, pela literatura, que nenhuma métrica caracteriza completamente a otimização dos algoritmos multi-objetivo (*MOO - Multiple-Objective Optimization*). Nesta medida, é mais apropriado avaliar esses algoritmos utilizando múltiplas métricas de desempenho complementares. No entanto, a AAC tradicional trata de otimizar uma única métrica de desempenho, seja para um algoritmo de objetivo único ou de multi-objetivo.

Blot et. al. introduzem uma abordagem multi-objetivo para AAC, comparando três indicadores ou métricas: otimização do hipervolume, otimização da soma ponderada de hipervolume e *spread*, e uma abordagem multi-objetivo genuína que considera ambos simultaneamente. Os resultados mostram que a AAC multi-objetivo supera a AAC de objetivo único. Para isso, foi utilizado o *MO-ParamILS* como configurador multi-objetivo, demonstrando resultados robustos em diferentes problemas MOO [41].

Um dos inconvenientes das técnicas de configuração *offline* surge quando não há um número significativo de instâncias de treino ou quando essas instâncias não são representativas, especialmente em novos problemas. Além disso, é necessário um período de treino demorado, após o qual a configuração deixa de evoluir, independentemente do número de instâncias encontradas [42].

3.3.2 RQ2- Quais os artigos que discutem a prática de ajuste dinâmico/online de parâmetros em algoritmos meta-heurísticos?

Fitzgerald et al. referem que existem duas abordagens para a configuração automática de algoritmos, *online* e *offline*, cada uma com vantagens e desvantagens [43]. Como exemplo, este autor cita alguns trabalhos sobre aprendizagem online no domínio estritamente relacionado com os portfólios de algoritmos.

Roberto Amadini et al, constrói uma agenda de solucionadores *Constraint Satisfaction Problem* (CSP, Problema de Satisfação por Restições) num portfólio online, sem qualquer treino prévio. Este autor encontra instâncias semelhantes à instância atual utilizando o método *k-Nearest Neighbours*. Em seguida, seleciona um subconjunto mínimo de soluções que podem resolver o maior número de instâncias vizinhas e regista-as conforme o número de instâncias resolvidas [44].

Por seu lado, o artigo intitulado *Evolving Instance-Specific Algorithm Configuration* de Yuri Malitsky et al. é uma abordagem de portfólio que realiza alguma aprendizagem *offline*, mas é capaz de o fazer evoluir, à medida que processa um fluxo de instâncias adaptando-se às instâncias em mudança. Isso é feito reagrupando dinamicamente as instâncias de entrada e atribuindo um conjunto de soluções a cada *cluster* [45].

Na área da configuração de algoritmos, Fitzgerald et al. demonstrou que as configurações podem ser melhoradas utilizando o tempo de inatividade. O configurador *online ReACTR* não requer instâncias anteriores nem um período de treino *offline*. As instâncias são resolvidas apenas uma vez, o que gera uma pequena sobrecarga em comparação com configuradores *offline*, que resolvem repetidamente as mesmas instâncias durante o treino. As configurações estão em constante evolução, de modo que, se o tipo de instâncias mudar, o configurador será capaz de se adaptar e encontrar configurações mais adequadas [42].

As *Instance-Specific Parameter Tuning Strategies* (IPTS, Estratégias de Ajuste de Instâncias Especificas para Instâncias) consideram a interação entre características mensuráveis do problema e valores específicos do algoritmo para desenvolver uma estratégia de ajuste. Yasemin Eryolda et al. apresenta uma definição do problema de configuração de algoritmo, uma visão geral de estudos de acesso aberto em IPTS conforme exposto na figura 3.3 e uma revisão dos métodos propostos nesta área [46]. No final, são discutidas as vantagens e desvantagens desses métodos, proporcionando o primeiro levantamento abrangente sobre as IPTS.

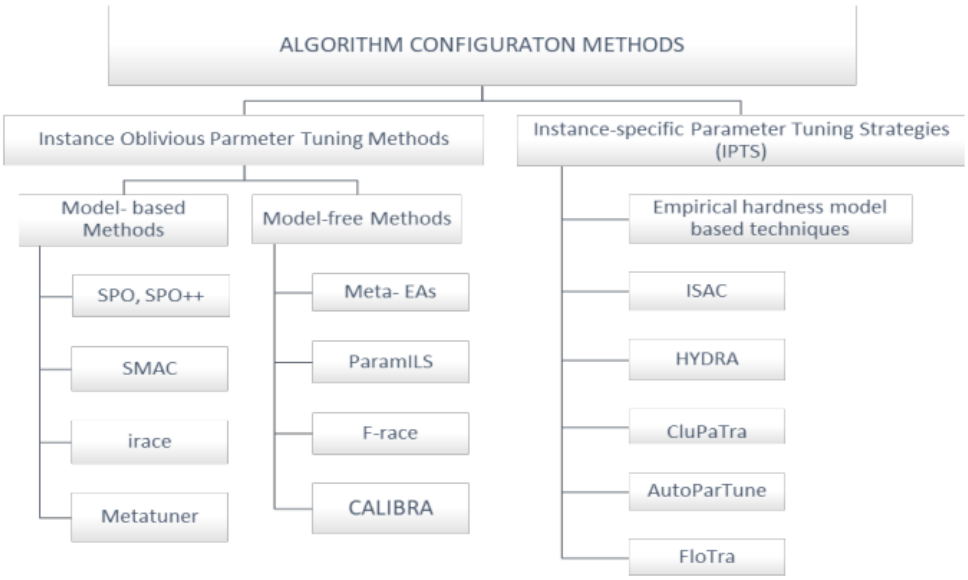


FIGURA 3.3: Esquema de classificação dos métodos de configuração automática de algoritmos de instâncias específicas e alheias por Yasemin et al. [46]

Por outro lado, a configuração *online* exige estratégias de controle diferenciadas e adequadas a cada algoritmo. Por exemplo, Mayer et al. considera que a estratégia de auto-adaptação da força da mutação em (μ / ρ, λ) - a *Evolution Strategy* (EA, Evolução Estratégica) - não é adequada para controlar o peso de inércia do algoritmo meta-heurístico PSO. Para desenvolver estratégias eficazes de controle de parâmetros para um algoritmo, é importante ter uma abordagem sobre como ajustar os parâmetros durante a execução para alcançar um desempenho otimizado. Isso implica compreender os valores apropriados dos parâmetros em diferentes fases da execução e utilizar informações históricas geradas durante as iterações do algoritmo [47].

O artigo de Bai Huang et al. propõe uma abordagem inovadora de configuração de algoritmo *online* para algoritmos evolutivos, especificamente o algoritmo de evolução diferencial [14]. A abordagem *Online Algorithm Configuration* (OAC, Configuração *Online* de Algoritmos) combina técnicas de aprendizagem por reforço e aprendizagem não supervisionada para adaptar dinamicamente a estratégia e os parâmetros de controle durante a pesquisa (figura 3.4). Os resultados experimentais mostram a superioridade do OAC-DE em relação a outros métodos adaptativos, destacando a sua capacidade e estabilidade. A análise de sensibilidade também evidência a robustez do OAC-DE em relação aos seus hiperparâmetros. Como trabalho futuro, pretendem estender essa abordagem para outros EAs e aplicá-la a diferentes tipos de problemas de otimização.

A fim de superar o desafio do ajuste *online* de parâmetros, foram propostos por Dzalbs et al., o *eTuner* e o *eTunerAlgo* [48]. O eTunerAlgo destaca-se por realizar tanto a seleção de algoritmos quanto o ajuste de parâmetros de forma automática. Os algoritmos propostos são avaliados em três meta-heurísticas - *Ant Colony Optimization* (ACO, Otimização por Colónia de Formigas), ES e *Imperialist Competitive Algorithm* (ICA, Algoritmo de Competição Imperialista) - e dois

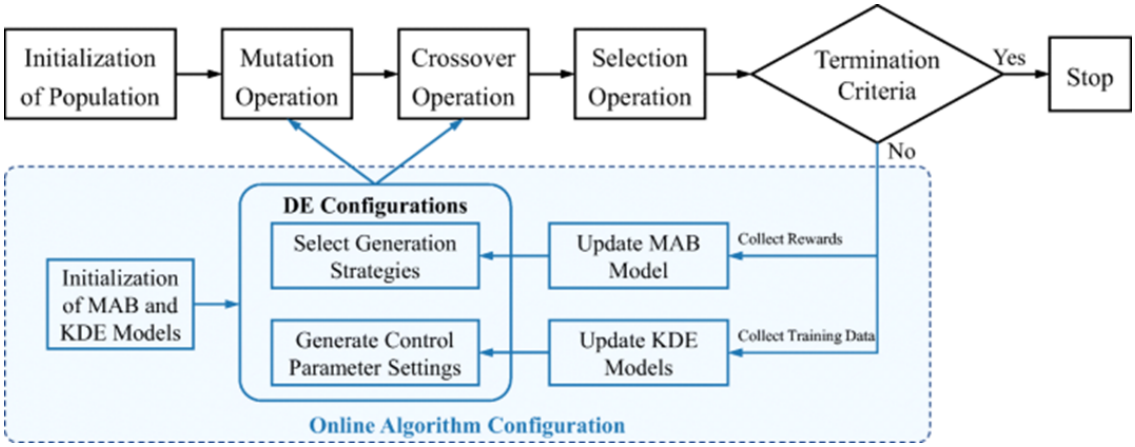


FIGURA 3.4: Algoritmo DE utilizando a proposta OAC-DE [14]

problemas *NP-hard*: o *Aerial Surveying Problem* (ASP, Problema de Levantamento Aéreo e o *Multiple Knapsack Problem* (MKP, Problema da Mochila Múltipla). Além disso, foi testada uma função *benchmark* de ajuste de meta-heurística contendo 18.760 configurações para uma avaliação conhecedora dos métodos. Os resultados experimentais indicam que apenas um pequeno conjunto de configurações pode ser avaliado dentro de um determinado tempo e que, nestes casos, a abordagem proposta é mais adequada.

3.3.3 RQ3 - Que artigos explicam os critérios que estão subjacentes ao ajuste dinâmico de algoritmos meta-heurísticos (populacionais) e como contribuem para a eficiência da solução global?

No que se refere ao controle de parâmetros auto-adaptativo, destaca-se uma especificação da classificação do ajuste dinâmico de parâmetros, *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) pela sua capacidade de ajustar automaticamente a matriz de covariância por estratégia evolutiva, o que permite que o algoritmo aprenda e adapte as direções mais promissoras para encontrar a solução ideal. A principal inovação do CMA-ES é a sua capacidade de ajustar a matriz de covariância no sentido de controlar o tamanho e a direção dos passos de pesquisa, tornando-o competente numa ampla variedade de problemas de otimização, sendo o método mais moderno para controle de parâmetros numéricos [49]. Como conclusão foram identificados alguns pontos-chave:

- Princípio de estimação: O CMA estima que a partir de um conjunto de passos selecionados, o seu uso é menos propenso à convergência prematura e suporta um comportamento exploratório no espaço de pesquisa.
- Controle de tamanho de passo: No CMA-ES, a adaptação da matriz de covariância é complementada com o controle de tamanho do passo. Este ajuste é baseado num princípio de adaptação diferente, o controle cumulativo do comprimento do caminho adapta-se

frequentemente a tamanhos de passo quase ótimos, geralmente conduzindo a comprimentos de passo consideravelmente maiores. Isso melhora a velocidade de convergência e as capacidades de pesquisa global em simultâneo.

- Tamanho da população, adaptação e taxas de mudança: Escolher o tamanho da população λ é sempre um compromisso. Pequenos λ levam a uma convergência mais rápida, e grandes λ ajudam a evitar ótimos locais.

O método *Adaptive Operator Selection* (AOS) é utilizado para identificar dinamicamente quais os operadores que devem ser aplicados durante a execução de um algoritmo de otimização (isto é, controle de parâmetros categóricos), com base no histórico de desempenho dos operadores disponíveis [50], [51]. Esse processo é geralmente dividido em dois componentes: o mecanismo de atribuição de créditos (CA), responsável por avaliar o desempenho dos operadores e a regra de seleção de operadores (OS), que determina o operador mais adequado a ser aplicada.

Ajustar dinamicamente os parâmetros, equivale a modificar os valores dos mesmos durante a execução de um algoritmo meta-heurístico, tendo em atenção o próprio processo de pesquisa. Para o caso de algoritmos evolutivos, a literatura revela duas formas principais de realizar esse ajuste: através de uma regra heurística que recebe *feedback* do estado atual da pesquisa e modifica os valores dos parâmetros ou incorporando os parâmetros nos cromossomas, tornando-os assim sujeitos a evolução ao longo das gerações [33].

Sobre a questão dos critérios de otimização, Soheila Ghambari et al. propuseram uma abordagem que utiliza uma distribuição desequilibrada do orçamento/previsão para as configurações geradas em vez de alocar um orçamento fixo para todas as configurações. Mais precisamente, eles demonstram que é necessário otimizar a avaliação da função de forma que o orçamento para cada configuração seja ajustado automaticamente durante o procedimento de otimização [52]. Contrariamente a outros métodos disponíveis na literatura (*Random Search, Bayesian, Irace, etc.*), alocam um orçamento pré-determinado para cada configuração gerada, esta abordagem permite uma maior flexibilidade e eficiência.

Em relação à gestão do tempo, este estudo propõe uma nova abordagem para a configuração automática de algoritmos evolutivos multi-objetivo, a fim de encontrar a melhor configuração utilizando o mínimo de tempo computacional. O desempenho da distribuição desequilibrada do orçamento foi investigado para diferentes configurações de um algoritmo-alvo, inspirado no algoritmo *Hyperband* [53].

Diversas técnicas de controle descritas na literatura podem ser classificadas com base em alguns critérios. A classificação para algoritmos evolutivos (EA) proposta por Angeline et el. baseia-se em dois principais critérios: os níveis de adaptação e os tipos de regras de atualização. São considerados três níveis de adaptação: populacional, individual e nível de componente. Ademais, identificam-se dois tipos de mecanismos de atualização: regras absolutas, que são pré-determinadas e indicam como as modificações devem ser realizadas e regras empíricas, que

envolvem competição entre valores de parâmetros, num processo conhecido como auto-adaptação [54].

Robert Hinterding et al. expandiram a classificação proposta por Angeline, adicionando um nível de adaptação extra, o nível do ambiente, subdividindo os mecanismos de atualização de forma mais detalhada, classificando-os em três categorias: determinísticos, adaptativos e auto-adaptativos [55]. Já a classificação de Smith e Fogarty é considerada a mais abrangente. Esta baseia-se em três critérios principais: o que está a ser adaptado, o âmbito da adaptação e a base para a mudança [56], [57]. O último critério divide-se em duas categorias: as evidências para a mudança e a regra ou algoritmo que efetua a mudança, sendo esta última subdividida em desacoplado/absoluto e fortemente acoplado/empírico, correspondendo esta última à auto-adaptação. Esta terminologia, no contexto de EA, conduz à taxonomia representada na figura 3.5.

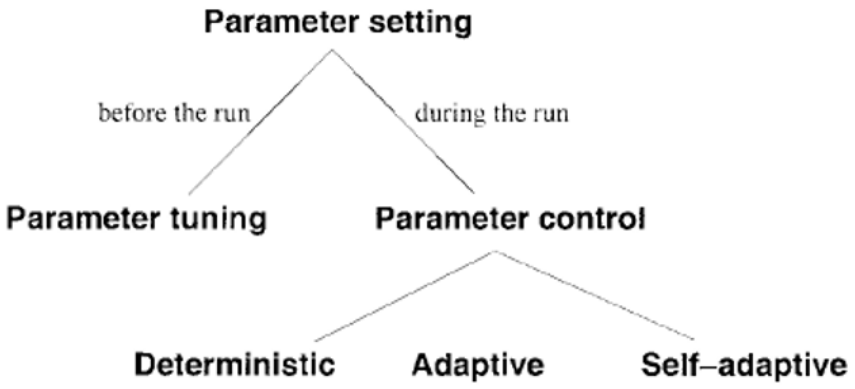


FIGURA 3.5: Taxonomia global da definição de parâmetros por Eiben et al. [33].

Assim, para Eiben et al, [33] os métodos para alterar um parâmetro podem ser classificados em três categorias principais, com base na forma como os parâmetros são modificados ao longo da execução do algoritmo:

- **Controle determinístico de parâmetros**: Usam regras fixas e pré-definidas para alterar os parâmetros, sem depender do desempenho ou da evolução do processo.
- **Controle adaptativo dos parâmetros**: Ajustam os parâmetros com base no desempenho dos indivíduos ou da população ao longo do tempo, reagindo às condições do algoritmo, ou seja, a algum tipo de *feedback*.
- **Controle de parâmetros auto-adaptativo**: Ajustam os parâmetros com base no desempenho dos indivíduos ou da população ao longo do tempo, reagindo às condições do algoritmo. Aqui, os parâmetros a serem adaptados são codificados nos cromossomas e sofrem mutação e recombinação.

Ainda nesse trabalho, este autor procura classificar as técnicas de controlo dos parâmetros em EA, respondendo aos aspetos seguintes:

- O que é alterado? (por exemplo, representação, função de avaliação, operadores, processo de seleção, taxa de mutação, etc.);
- Como é que a alteração é efetuada? (i.e., heurística determinística, heurística baseada em *feedback*, ou auto-adaptativa);
- O âmbito/nível da mudança (por exemplo, ao nível da população, nível individual, etc.);
- Os dados com base nos quais a mudança é efetuada (por exemplo acompanhamento do desempenho dos operadores, diversidade da população, etc.).

Resumindo, este autor reforça a ideia de uma abordagem bidimensional em EAs para classificar essas mudanças, com duas dimensões principais independentes entre si: tipos (determinísticos, adaptativos e auto-adaptativos) e componentes de alteração de parâmetros (representação, função de avaliação, operadores (mutação e recombinação), seleção, substituição e população).

Sobre a *Dynamic Algorithm Configuration* (DAC, Configuração Dinâmica de Algoritmos)), Adriaensen et al, destacam na sua pesquisa uma série de avanços recentes e estabelecem uma base sólida para investigações futuras neste domínio, considerado emergente [6]. De forma mais específica :

- Contextualizaram o DAC dentro do histórico mais amplo da pesquisa em inteligência artificial;
- Formalizaram o DAC como um problema computacional;
- Examinaram métodos previamente utilizados na literatura para abordar esse problema;
- Realizaram estudos de casos empíricos que exploram a aplicação do DAC em áreas como otimização evolutiva, planeamento de AI e ML

O texto também discute a importância de combinar *Reinforcement Learning* (RL, Aprendizagem por Reforço) e otimização de problemas de DAC que envolve a escolha de uma sequência de configurações em vez de uma única configuração, conforme o ajuste estático. Combinar RL e otimização pode explorar o melhor de ambas as abordagens.

Guo et al., discutem o uso de algoritmos evolutivos, como a evolução diferencial, para resolver problemas de otimização de parâmetros reais, mas reconhecem que a eficácia de um único algoritmo pode variar conforme o problema. Para superar essa limitação, propõem uma estrutura baseada em aprendizagem por reforço profundo (DRL - *Deep Reinforcement Learning* (DRL, Aprendizagem por Reforço Profunda)) para seleção dinâmica de algoritmos [58].

3.3.4 RQ4 - Que artigos referem o contributo de meta-heurísticas na atividade hospitalar, em especial, no escalonamento de blocos operatórios?

Como resposta à última questão de pesquisa H. Fei propõe projetar um cronograma semanal de cirurgias numa entidade hospitalar, onde blocos de tempo são reservados para cirurgiões em vez de especialidades. O foco é maximizar a utilização das salas de cirurgia, minimizar os custos

de horas extras e reduzir os tempos mortos inesperados entre casos cirúrgicos. O problema é abordado em duas fases:

- A resolução do problema de escalonamento para determinar a data de cirurgia para cada paciente.
- A elaboração do cronograma diário, considerando a disponibilidade de camas de recuperação.

O método proposto utiliza um modelo de programação inteira, um procedimento heurístico baseado em geração de colunas para o planeamento e um algoritmo genético híbrido para o agendamento diário.

Resumindo, os resultados indicam uma melhoria teórica sobre cronogramas reais em termos de tempos mortos, sobre a utilização das salas de cirurgia e sobre os custos de horas extras [59].

O Plano Nacional de Saúde de Portugal estabelece duas diretrizes principais para unidades hospitalares: aperfeiçoar a utilização eficiente dos recursos disponíveis e reduzir a lista de espera para cirurgias.

Inês Marques procura contribuir para a obtenção desses objetivos no campo da pesquisa operacional, com foco num estudo de caso de um problema de agendamento de cirurgias elegíveis num hospital público português [60]. São considerados dois critérios de otimização antagónicos: maximizar a ocupação da sala cirúrgica e maximizar o número de cirurgias agendadas. São também desenvolvidas duas versões de uma heurística genética de objetivo único e aplicadas a dados reais do hospital estudado.

3.4 Resumo

No capítulo 3 foi adotada uma abordagem metodológica inspirada na filosofia PRISMA, focada em questões de pesquisa bem definidas. A metodologia detalhada incluiu a formulação de perguntas de pesquisa específicas, a escolha cuidadosa de fontes de dados eletrónicos, a definição de termos de pesquisa abrangentes e a aplicação de critérios de inclusão e exclusão para garantir a qualidade e relevância dos estudos selecionados.

De seguida, o texto abordou a configuração automática de algoritmos meta-heurísticos, dividindo a discussão em três questões de pesquisa (RQ1, RQ2, RQ3) e uma quarta questão (RQ4) relacionada com o uso destes modelos de ML aplicados ao sector hospitalar.

Em resposta a estas questões, fez-se referência a diversos métodos estáticos, como pesquisa aleatória, otimização *Bayesiana*, *SMAC*, *ParamILS*, *Irace* e *MAC*, que desempenham um papel importante na otimização do seu funcionamento. Também foi abordada a eficácia da configuração multi-objetivo em comparação com a configuração de objetivo único. A pesquisa revelou que a abordagem multi-objetivo supera o equivalente a objetivo único, utilizando o *MO-ParamILS* como configurador multi-objetivo.

Sobre os critérios de ajuste dinâmico foram mencionados exemplos como: o controle do tamanho de passo na CMA-ES, uma técnica moderna para o controle de parâmetros numéricos; a AOS, que identifica dinamicamente quais os operadores a aplicar durante a execução do algoritmo, com base no histórico de desempenho e os algoritmos evolutivos, que apresentam diferentes abordagens para o ajuste de parâmetros, incluindo estratégias de auto-adaptação.

Por fim, este capítulo abordou a configuração automática de algoritmos, desde métodos específicos até aplicações práticas em ambientes hospitalares.

Capítulo 4

Configuração Dinâmica de Algoritmos

Neste capítulo, é apresentada a conceptualização da solução, os modelos de referência e as técnicas de *Artificial Intelligence (AI, Inteligência Artificial)* empregues, nomeadamente o SARSA e o Deep SARSA.

Na fase experimental, é investigada a aplicação e implementação de um modelo para a configuração automática de parâmetros de um algoritmo genético, integrando conceitos de aprendizagem por reforço. A abordagem inicial é centrada no uso de funções *benchmarks* para validar o desempenho dos modelos propostos, com o objetivo de estabelecer uma base sólida que permita, posteriormente, a aplicação dessa solução em diversos casos reais.

Finalmente, são referidos aspetos sobre as tecnologias empregues no desenvolvimento e implementação do método de configuração dinâmica de parâmetros dos algoritmos.

4.1 Conceptualização

Na configuração de algoritmos, os otimizadores de "*black-box*"enfrentam frequentemente desafios consideráveis, como a falta de acesso a informações detalhadas sobre o estado do processo e a necessidade de definir logo de início uma única configuração de parâmetros [12]. Essas limitações restringem a capacidade desses otimizadores de aprender e ajustar as sequências ótimas de parâmetros ao longo do tempo. Em contrapartida, agentes que utilizam informações dinâmicas e mais ricas sobre o estado do sistema podem otimizar e ajustar continuamente as suas políticas, aprimorando significativamente as estratégias de *Dynamic Algorithm Configuration (DAC, Configuração Dinâmica de Algoritmos)*.

No contexto do DAC, o objetivo principal é desenvolver políticas que possam ajustar os parâmetros de algoritmos de forma dinâmica, tanto em termos globais, como em termos específicos para cada instância de problema. A meta é criar políticas capazes de generalizar e resolver uma ampla variedade de problemas [61].

Este estudo explora um espaço de ação discreto e propõe uma política DAC que selecione, de forma dinâmica, as configurações de parâmetros para uma heurística específica. Algumas heurísticas mostram-se mais produtivas em determinados domínios e instâncias de problemas específicos

do que outras. Para enfrentar este desafio os algoritmos, genético e de otimização por enxame, foram adotados como heurísticas para a configuração automática de parâmetros, aplicados a uma variedade de problemas teóricos.

Os termos *online* (durante o uso) e *offline* (antes do uso) são, por vezes, utilizados como sinónimos para dinâmico e estático, respetivamente. Neste estudo, a figura 4.1 ilustra o fluxo de informação adotado entre a instância do problema e a solução proposta num ambiente *online*, onde a *Reinforcement Learning (RL, Aprendizagem por Reforço)* ocorre durante o tempo de uso. No entanto, para Adriaensen et al. também é possível utilizar uma política dinâmica *offline*, utilizando um conjunto de treino pré-definido e, no momento de uso, simplesmente executar essa política dinâmica em novas instâncias do problema [6].

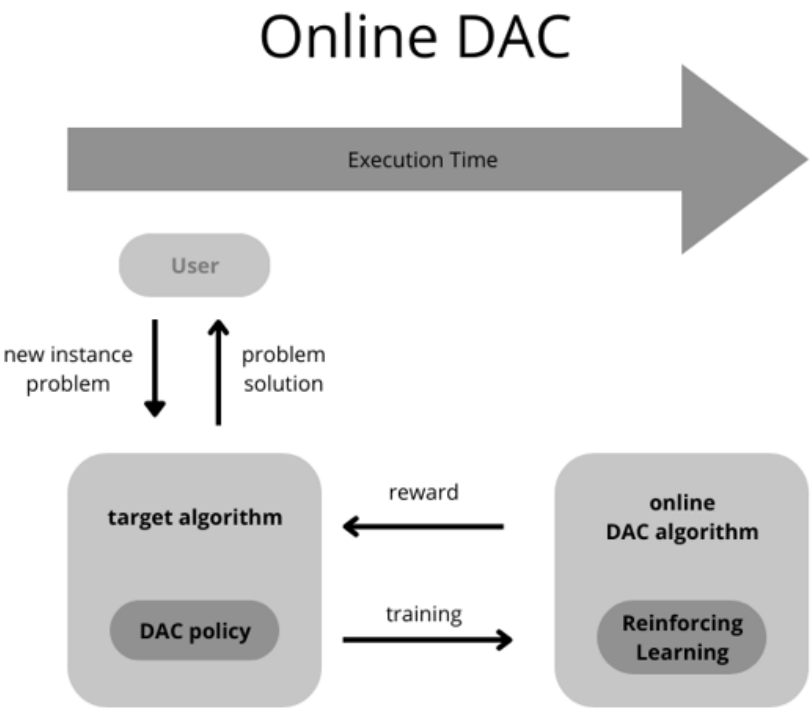


FIGURA 4.1: Visão geral de DAC *online* adaptado de Adriaesen et al. [6]

O processo começa com o utilizador fornecendo uma nova instância do problema (*new instance problem*) ao sistema, que pode ser qualquer desafio a ser resolvido, como a otimização de um processo. Em seguida, o algoritmo-alvo (*target algorithm*), que é configurado dinamicamente para lidar com essa instância, recebe o problema e aplica a política (*DAC Policy*) para gerar uma solução.

A política atua como a estratégia central, definindo a forma como o algoritmo-alvo deve ser ajustado durante a execução para otimizar o seu desempenho, considerando as condições específicas do problema. Neste caso, o algoritmo DAC *online* designado na figura 4.1 de *Online DAC Algorithm* utiliza técnicas de RL para aperfeiçoar continuamente a política e interage em tempo real com o

ambiente, recebendo *feedbacks* ou recompensas baseados na qualidade das soluções geradas pelo algoritmo-alvo. Este ciclo de treino e recompensa ocorre de forma contínua, permitindo que a política se torne cada vez mais capaz na resolução dos problemas.

Ao longo do tempo de execução, o sistema realiza um ciclo contínuo de ajuste e otimização. O algoritmo DAC *online* aperfeiçoa a política, que, por sua vez, configura o algoritmo-alvo para resolver a nova instância do problema de maneira mais proficiente. A aprendizagem da *DAC Policy* é efetuada através de um algoritmo que procura maximizar o valor esperado da recompensa ao longo das múltiplas interações com o ambiente (figura 4.2). Por fim, a solução do problema é entregue ao utilizador, completando o ciclo.

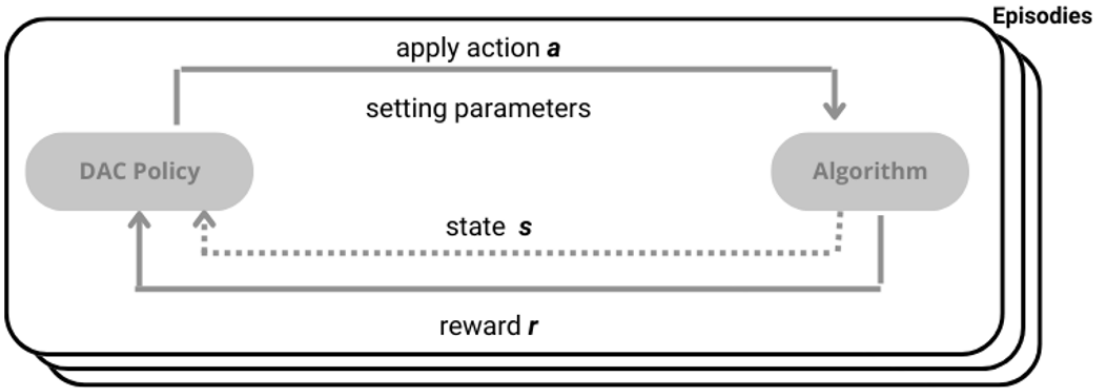


FIGURA 4.2: Em cada episódio usando RL a política de configuração ajusta o valor dos parâmetros do algoritmos baseado no estado, ação e recompensa

Neste trabalho, para testar o modelo, foram definidos 32 e 50 episódios. A escolha destas quantidades foi baseada numa análise prévia de desempenho com funções *benchmark*, onde 32 episódios se mostraram adequados para garantir um equilíbrio entre eficiência de recursos e robustez de resultados na aprendizagem dessa política. Em cenários mais complexos foram utilizados 50 episódios para refinar o modelo.

O conhecimento de políticas DAC orientado por dados, através de RL surge como uma orientação promissora, com duas abordagens principais a ser consideradas.

A primeira é o *Markov Decision Process (MDP, Processo de Decisão Markov)* contextual, onde os agentes de RL podem ser implementados usando consultas de uma tabela para espaços pequenos de ações e estados. Trata-se de uma estrutura matemática usada para descrever um ambiente em que um agente toma decisões. Esta técnica é definida por um conjunto de estados, um conjunto de ações, uma função de transição de estados e uma função de recompensa. Quando o MDP é contextual, tem em conta informações adicionais que podem alterar a dinâmica do ambiente, como características do estado ou parâmetros do problema. Algoritmos como o Q-Learning ou SARSA são comummente usados em RL para encontrar políticas ótimas num espaço MDP, sem precisar conhecer o modelo do ambiente. Estes utilizam uma tabela Q, onde cada entrada $Q(s,a)$

representa a qualidade da ação a no estado s. O objetivo é atualizar iterativamente esta tabela para refletir a recompensa esperada a longo prazo para cada ação.

A segunda abordagem envolve o uso de métodos de aproximação de funções, especialmente quando o espaço de estados e ações é muito grande para ser representado por uma tabela Q. Nesses casos, técnicas como redes neuronais são utilizadas para aproximar a função Q através de uma função parametrizada, permitindo generalizações com base num número limitado de exemplos. Esse processo, no qual são usadas redes neuronais que se aproximam da função Q, é conhecido como Deep Q-Network ou Deep SARSA. Em vez de armazenar uma tabela Q completa, a rede neuronal é treinada para estimar a função Q, oferecendo uma solução mais eficiente e escalável para ambientes complexos e de grande dimensão.

Os métodos denominados de *Temporal Difference* (TD, Diferença Temporal) são amplamente utilizados em algoritmos de aprendizagem por reforço, pois permitem que os agentes aprendam diretamente com a experiência, sem a necessidade de um modelo do ambiente. Uma das suas principais vantagens é a capacidade de oferecer garantias de convergência para um desempenho ótimo, além de serem relativamente simples de implementar [62].

Os métodos mencionados combinam conceitos de aprendizagem supervisionada e não supervisionada, atualizando a função valor-ação $Q(s,a)$ a cada passo com base na experiência imediata. Isso possibilita ao agente aprender de forma incremental, sem precisar armazenar todas as interações passadas, tornando os métodos TD ideais para ambientes dinâmicos e em constante mudança. No entanto, embora dispensem um modelo de ambiente, podem enfrentar dificuldades em lidar com a complexidade de certos tipos de ambiente.

Neste trabalho, o ambiente considerado é o espaço de configuração dos parâmetros ao longo dos diferentes estados do processo de otimização. As ações tomadas pelo agente correspondem a ajustes incrementais desses parâmetros, como, por exemplo, aumentar ou diminuir a taxa de mutação no GA ou alterar o fator de inércia no PSO.

Tanto SARSA como *Q-Learning* são exemplos de algoritmos TD com garantias teóricas de convergência para políticas ótimas, desde que haja um equilíbrio adequado entre *exploration* e *exploitation* de ações. SARSA é um método *on-policy*, o que significa que atualiza a função Q com base na política atual, incluindo ações exploratórias. Em contrapartida, Q-Learning, um método *off-policy*, otimiza a função Q independentemente da política seguida pelo agente. Essas diferenças influenciam diretamente a escolha do algoritmo, dependendo das características do problema e do ambiente.

Ao aplicar-se o SARSA para ajuste de parâmetros, o método considera a ação que foi efetivamente escolhida e executada pela política atual, incluindo ações exploratórias, durante a execução do algoritmo heurístico. Por exemplo, se a política atual sugere aumentar a taxa de mutação do GA num determinado estado e esta ação é executada, o método atualizará a função valor-ação com base no resultado observado. A equação 4.1 reflete a atualização da função Q:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[y - Q(s,a) \right] \tag{4.1}$$

onde y é o alvo de atualização (*target*) e α é a taxa de aprendizagem.

Pelo contrário, o Q-Learning atualiza os valores Q com base na melhor ação possível do próximo estado, independentemente da ação realmente selecionada. Nesse caso, o Q-Learning tende a explorar com maior agressividade o espaço de configuração para encontrar rapidamente os parâmetros ideais, mesmo que isso envolva riscos de ajustes abruptos. Definimos a atualização do valor Q da seguinte forma:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{4.2}$$

onde $\max_{a'} Q(s',a')$ é a estimativa do valor-ação ótimo no próximo estado s'.

Assim, os métodos SARSA e Deep SARSA foram selecionados para serem avaliados ao longo deste trabalho, pela sua capacidade de realizar ajustes de parâmetros de maneira gradual e controlada, o que pode ser particularmente útil em cenários onde mudanças abruptas poderiam comprometer a estabilidade e a convergência do algoritmo. A principal vantagem desses métodos é que eles ajustam os parâmetros com base nas ações realmente tomadas, o que proporciona um controle mais cuidado e seguro durante o processo de otimização, minimizando o risco de ajustes drásticos que poderiam afetar negativamente o desempenho global do sistema.

Quando se usa uma heurística como um GA ou PSO para resolver problemas de otimização, o desempenho do algoritmo pode depender fortemente dos parâmetros de controle, como taxa de mutação, taxa de cruzamento, fator de inercia, fator cognitivo, entre outros. A escolha por métodos de RL permite ajustar os parâmetros do algoritmo de otimização em tempo real, adaptando-se às mudanças no problema ou às condições do ambiente.

Além disso, esses métodos possibilitam um equilíbrio entre *exploration* e *exploitation* utilizando políticas como ϵ-greedy. Os métodos TD equilibram a exploração de novas configurações de parâmetros e a exploração de configurações já conhecidas por serem melhores. Através de iterações e *feedback* contínuo, o agente de RL pode aprender políticas que maximizam (ou minimizam) a função objetivo da heurística, levando a soluções mais competentes enquanto o algoritmo de otimização está em execução.

4.2 Heurísticas e Parâmetros

Nesta secção, abordaremos o ajuste online de parâmetros de algumas heurísticas, com foco no GA e no PSO, discutindo como esses algoritmos funcionam, quais são os principais parâmetros a ajustar e respetivos motivos.

4.2.1 Genetic Algorithm

O *Genetic Algorithm* (GA, Algoritmo Genético), concebido por Holland na década de 1960, é uma abordagem computacional inspirada na teoria da evolução de Darwin e enquadra-se na estrutura dos algoritmos populacionais [25]. A sua metodologia de pesquisa baseia-se em probabilidades, visando não necessariamente a obtenção de uma solução ótima, mas sim a exploração de soluções próximas do ótimo. Além do princípio da evolução natural, o GA incorpora conceitos biológicos, como o *crossover* (troca de informações entre indivíduos) e a mutação (alteração na composição genética de um indivíduo), refletindo os mecanismos da seleção natural. As características dos indivíduos, chamadas genes, são transmitidas, através de uma sequência de gerações, aos descendentes por intermédio da herança dos pais (cruzamento) e por novos genes criados (mutação) durante o processo de reprodução. Nesse sentido, os algoritmos genéticos assemelham-se à reprodução seletiva [63]. Uma descrição completa do GA é fornecida na figura 4.3 .

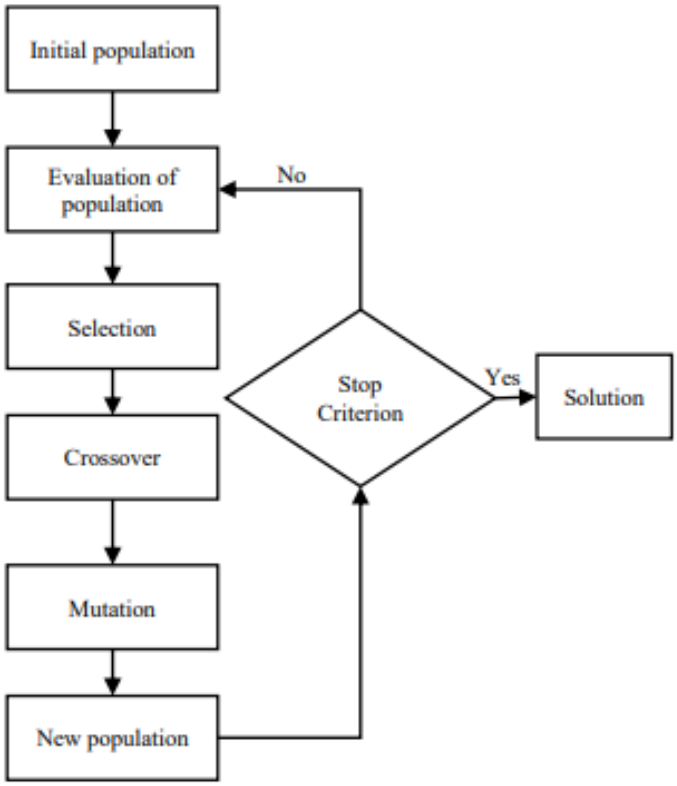


FIGURA 4.3: Fluxograma padrão do GA [64]

A ideia básica de um GA é utilizar uma população com soluções individuais obtidas aleatoriamente para cobrir toda a gama de soluções possíveis no espaço de soluções [65]. Em seguida, a função objetivo é acedida pelos indivíduos da população para avaliação, considerando o risco de cada variável e o índice de melhoria que se deseja alcançar.

A maioria das funções objetivo utilizadas em GAs são métodos estocásticos, aplicáveis em processos de seleção como a roleta, o torneio ou o *ranking*, que garantem a manutenção de uma

certa proporção de indivíduos na população [66]. Esses indivíduos selecionados avançam para a próxima etapa: o processo de cruzamento e mutação.

O cruzamento envolve a combinação do material genético de dois indivíduos progenitores para criar descendentes que herdam características de ambos os pais. A mutação consiste em pequenas alterações aleatórias no material genético de um indivíduo da população.

A nova geração de indivíduos é então formada. Em seguida, aplica-se o critério de parada, que pode ser de diversas naturezas: número de gerações, limite de tempo, ou estabilização da população por um número seguido de gerações. O algoritmo termina quando o critério de parada é atendido; caso contrário, retorna à fase de avaliação da população para iniciar um novo ciclo.

Neste trabalho, para a aplicação do algoritmo genético, utilizou-se a biblioteca *Distributed Evolutionary Algorithms in Python* (DEAP) em Python, considerando as seguintes características básicas:

- **Cruzamento**: Utilizou-se o método de cruzamento de dois pontos (*two-point crossover*) ou o cruzamento de ordem (*order crossover*).
- **Mutação**: Foi aplicada a mutação por alteração de bits (*mutFlipBit*) ou por troca de dois elementos (*mutSwap*).
- **Seleção**: Adotou-se o método de seleção por torneio, com tamanho de torneio igual a 3.

O operador de mutação *mutFlipBit* da biblioteca DEAP é utilizado para alterar bits individuais em representações binárias de cromossomas. No caso de cromossomas representados como permutações (ou listas de inteiros), o operador *mutSwap* troca dois elementos aleatórios.

O cruzamento de dois pontos (*two-point crossover*) mistura elementos dos dois pais, enquanto o cruzamento de ordem (*order crossover* - OX) é um operador específico para permutações que mantém a ordem relativa dos elementos dos cromossomas pais, gerando dois filhos válidos.

Depois de observadas as características principais, como cruzamento e mutação, importa referir que foram escolhidos os parâmetros como tamanho da população, taxa de cruzamento, taxa de mutação e número de gerações para ajuste dinâmico do GA por se tratarem de elementos críticos para o desempenho e eficácia do algoritmo. A escolha de intervalos razoáveis para cada parâmetro é crítica para garantir que o GA seja capaz de adaptar-se ao problema específico, maximizando a qualidade das soluções geradas. Estes intervalos são definidos antes da execução de um método de configuração, como o SMAC ou SARSA, e determinam o espaço de pesquisa dentro do qual o algoritmo tentará encontrar a configuração ideal para otimizar uma função objetivo.

Na tabela 4.1 são identificados os parâmetros sujeitos a configuração e os respetivos intervalos:

TABELA 4.1: Limites de intervalos de valores, dentro dos quais os parâmetros do algoritmo (GA) podem variar

Parâmetro	**Intervalo**
Population Size	[50, 500]
Crossover Rate	[0.6, 0.9]
Mutation Rate	[0.01, 0.1]
Generations	[30, 90]

Esses intervalos de parâmetros foram selecionados com base em recomendações amplamente aceites na literatura científica, considerando a sua eficácia comprovada face a uma ampla variedade de problemas práticos. O tamanho da população entre 50 e 500 indivíduos, por exemplo, foi escolhido para equilibrar a diversidade genética e a capacidade de exploração com o tempo de execução do algoritmo. Da mesma forma, as taxas de cruzamento entre 0,6 e 0,9 foram adotadas devido à sua capacidade de equilibrar a exploração de novas soluções com a preservação de combinações promissoras. A taxa de mutação, situada entre 0,01 e 0,1, foi selecionada para garantir um nível adequado de exploração sem desestabilizar o processo evolutivo.

Por fim, o número de gerações, definido entre 30 e 90, foi escolhido para equilibrar a qualidade da solução com o tempo de execução, garantindo que o algoritmo tenha tempo suficiente para convergir para soluções satisfatórias sem consumir recursos excessivos. Essas escolhas foram feitas para maximizar a performance do GA dentro dos limites práticos do problema em questão. Sobre este tema, Kenneth Alan De Jong e John J Grefenstett conduziram experiências para identificar valores recomendados de parâmetros, sugerindo valores que, praticamente em todos os casos, se enquadram dentro dos intervalos definidos [67], [68].

Ao avançar para o ajuste *online* de parâmetros usando RL, definimos os estados e ações a serem utilizados. Dentro dos valores de intervalo previamente estabelecidos, a DAC será projetada para encontrar a melhor configuração possível. Para avaliar a capacidade da utilização de RL na configuração dinâmica de parâmetros, são inicialmente definidos como entrada o número de episódios de treino, a quantidade de genes do indivíduo e a função objetivo utilizada.

A aprendizagem por reforço envolve um agente, um conjunto de estados $\mathcal{S}$ e um conjunto $\mathcal{A}$ de ações por estado. Ao realizar uma ação $a \in \mathcal{A}$, o agente faz a transição de estado para estado, interagindo com o ambiente. Executar uma ação num estado específico fornece ao agente uma recompensa (uma pontuação numérica). O SARSA, um método de RL, assimila uma política ideal que maximiza essas recompensas.

A tabela 4.2 mostra no caso concreto o conjunto de ações $\mathcal{A}$ possíveis em cada estado, no conjunto de estados $\mathcal{S}$ observáveis pelo agente apresentados neste método (SARSA).

A seleção dos valores escolhidos para as taxas de ajuste enquadram-se na designada pratica comum. Pequenos ajustes nas taxas de cruzamento e na mutação são preferíveis para manter a estabilidade do GA e são eficazes na pesquisa, sem comprometer a convergência [69].

TABELA 4.2: Tabela de ações implementadas para ajuste de parâmetros do algoritmo GA

Ação	Descrição	Taxa
No changes	Unchanged parameters	0%
More or Less Population	Increase or Decrease Population Size	5%
More or Less Crossover Rate	Increase or Decrease Crossover Rate	5%
More or Less Mutation Rate	Increase or Decrease Mutation Rate	1%
More or Less Generations	Increase or Decrease Generations	5%

4.2.2 Particle Swarm Optimization

Trata-se de um algoritmo criado por R. Eberhart e J. Kennedy e é inspirado no comportamento de um bando de aves ou de um cardume de peixes à procura do seu alvo, mediante um esforço conjunto [27]. A procura do melhor local é a procura do lugar ótimo, aparece computacionalmente como uma abstração da teoria evolutiva da biologia. É um algoritmo em que o enxame é a população e os indivíduos são as partículas. É um algoritmo que tem aplicações em vários campos da ciência ou engenharia como *Machine Learning* (ML) ou *Data Mining* (DM). Podemos designá-lo como um algoritmo simples, mas também bastante poderoso.

TABELA 4.3: Notação e significado

Notação	Significado
p_i	p_{besti} (personal best): a melhor posição encontrada pela partícula i
g	g_{best} (global best): a melhor posição encontrada por todas partículas
c_1, c_2	parâmetros cognitivo e social
w	ponderação da inércia
r_{1j}, r_{2j}	números aleatórios entre 0 e 1

O comportamento de cada partícula é baseado na sua própria experiência (*Pbest*), na experiência dos outros membros do enxame (*Gbest*) e no movimento das partículas. O movimento de cada partícula na pesquisa da localização ideal é definido pelo custo das várias soluções ou caminhos. Estas soluções aleatórias formam uma população composta por partículas, daí designar-se *Particle Swarm Optimization* (PSO, Otimização por Enxame de Partículas). Assim, o processo determina as melhores soluções candidatas, onde é avaliado o custo da solução e é determinado o mínimo local. Com a partilha desta informação sobre a partícula e sobre as outras partículas na vizinhança é possível registar, não só o mínimo local (*Pbest*), mas também o mínimo global (*Gbest*).

A localização seguinte, consequente do movimento de cada partícula, é baseada em três parâmetros: o fator de sociabilidade, o fator de individualidade e a velocidade máxima. O PSO combina estes parâmetros com um valor gerado aleatoriamente (entre 0 e 1) e calcula a posição seguinte da partícula. Os parâmetros designam-se por: fator de Atração Global (C2) - determina a atração (convergência) das partículas para a melhor solução descoberta por um membro do grupo (Termo

Social); fator de Atração Local (C1) - determina a atração da partícula com a sua melhor posição (Termo Cognitivo); fator de Velocidade (ω) - delimita o movimento, uma vez que este é direcional e determinado (Termo Inércia).

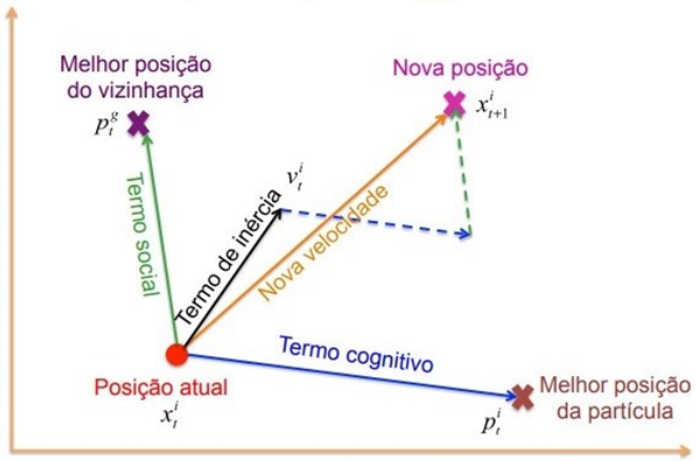


FIGURA 4.4: Movimento de partículas de PSO adaptado de April Wang [70]

Os novos valores da velocidade e posição representados da imagem anterior figura 4.4 resultam das equações seguintes: atualização da velocidade, equação (4.3) e atualização da posição, equação (4.4):

$$v_{i,t+1} = w \cdot v_{i,t} + c_1 \cdot \text{rand} \cdot (p_{i,t} - x_{i,t}) + c_2 \cdot \text{rand} \cdot (p_{g,t} - x_{i,t}) \tag{4.3}$$

O cálculo de nova velocidade $v\_i, t + 1$ combina a inércia, a influência da melhor posição pessoal e a influência da posição global. São aplicados limites para garantir que $v\_i, t + 1$ esteja dentro do intervalo aceitável. A amplitude destes limites é ajustada com o decorrer das iterações (entre 0.5 e 0.1).

$$x_{i,t+1} = x_{i,t} + v_{i,t+1} \tag{4.4}$$

Para a atualização da posição, adota-se a técnica de Efeito Espelho (*Mirror Effect*), quando se verifica a nova posição fora dos limites, multiplicado por -1. O efeito de espelho é uma técnica de contorno que ajusta a direção da velocidade quando uma partícula sai dos limites. A ideia é simular a um "espelho"nos limites do espaço de pesquisa para que a partícula "retorne"para dentro dos limites, em vez de simplesmente ser ignorada ou colocada de volta manualmente.

É importante que as partículas estejam dentro da região de pesquisa, evitando que se movam para regiões inviáveis e mantendo o processo de otimização dentro dos limites estabelecidos.

Como o trabalho se dedica ao ajuste online de parâmetros, importa indicar quais foram aqueles que podem afetar os resultados do PSO. Pela descrição teórica do algoritmo de otimização podemos apreciar os parâmetros representados na tabela seguinte:

TABELA 4.4: Limites de intervalos de valores, dentro dos quais os parâmetros do algoritmo (PSO) podem variar

Parâmetro	Intervalo
Fator inércia mínima	[0.1, 0.5]
Fator inércia máxima	[0.5, 0.9]
Fator cognitivo mínimo	[0.1, 1.0]
Fator cognitivo máximo	[1.0, 2.0]
Fator social mínimo	[0.1, 1.0]
Fator social máximo	[1.0, 2.0]
Taxa Reparação Direta	[0.1, 0.3]
Iterações	[10, 100]
Partículas	[5, 100]

Diga-se que, atendendo a alguns artigos, os valores estudados estão dentro dos intervalos indicados na tabela 4.4. No seu artigo de Y. Shi et al. concluiu que, para funções *benchmark* com populações de tamanhos 20, 40, 80 e 160, usadas com um peso de inércia decrescente que começa em 0,9 e termina em 0,4 e com cl=2 e c2=2 conseguem melhores resultados [71].

Para alem das taxas de ajuste, apresentadas na tabela 4.5, que funcionam como aumento ou redução consoante a ação pretendida pela política de recompensas da RL, foi aplicada uma percentagem de atualização fixa a estas taxas, na mudança de um estado para outro. Esta circunstância tem como objetivo controlar melhor a dicotomia entre a *exploration* e *exploitation* do espaço de soluções. Depois de alguns testes empíricos foi definido o valor de 10% sempre que se ultrapassar um dos estados pré-definidos nas 3 etapas: inicial, intermédia e final.

Pretende-se que o algoritmo no processo de controlo dinâmico de parâmetros possa, numa fase inicial, aumentar a exploração do espaço de soluções, garantir uma pesquisa ampla e evitar que fique preso em mínimos locais prematuros. De seguida, procura-se refinar a pesquisa e equilibrar entre *exploration* e *exploitation* para encontrar boas soluções que, na fase final, possam promover uma convergência mais precisa e evitar que o algoritmo se desvie da solução ótima encontrada.

Também, por ação da adoção de políticas ϵ-greedy em RL, estão incluídos mecanismos para que a pesquisa não se concentre em ótimos locais.

TABELA 4.5: Tabela de ações implementadas para ajuste de parâmetros do algoritmo PSO

Ação	Descrição	Taxa
No changes	Unchanged parameters	0%
Inertia Factor (Min and Max)	Increase or Decrease Inertia Rate	10%
Cognitive Factor (Min and Max)	Increase or Decrease Cognitive Rate	10%
Social Factor (Min and Max)	Increase or Decrease Social Rate	10%
Direct Repair Factor	Increase or Decrease Direct Repair Rate	5%
Iteractions	Increase or Decrease Interactions Size	20%
Particles	Increase or Decrease Particles Size	20%

Ao ajustar os parâmetros do PSO de forma diferente para fases distintas do processo de otimização, pode-se melhorar tanto a exploração inicial quanto a convergência final. O ajuste adaptativo baseado em fases permite uma pesquisa mais eficiente e permite obter melhores resultados para problemas complexos. A abordagem de atualização percentual, por aumento ou diminuição dos parâmetros, em cada interação é uma forma prática e flexível de responder à adaptação online de parâmetros. De acordo com Zhan et al. as estratégias de controle de parâmetros adaptativos podem ser desenvolvidas com base no estado evolutivo identificado [72].

4.3 Configuração Dinâmica de Algoritmos por Aprendizagem

Nesta secção, apresentamos uma abordagem para a DAC utilizando o SARSA, na escolha de parâmetros ao longo do tempo, adaptando-se melhor às condições específicas do problema e aperfeiçoando a eficiência e eficácia do processo de otimização.

4.3.1 Método SARSA

O algoritmo SARSA foi formalmente introduzido no livro *"Reinforcement Learning: An Introduction"* de Richard S. Sutton e Andrew G. Barto, uma obra de referência na área de aprendizagem por reforço. A partir destes conceitos teóricos, surgiram diversas aplicações práticas que adaptaram o algoritmo para enfrentar desafios reais em ambientes dinâmicos e incertos [73].

Por exemplo, mais recentemente, Qinglin Meng et al. propuseram um modelo de otimização com aprendizagem por reforço online para lidar com a imprevisibilidade das fontes renováveis, como a energia eólica e solar, procurando minimizar os custos operacionais dos sistemas de armazenamento de energia e otimizar a gestão de sistemas eólico-fotovoltaicos (WPESS) [74]. Utilizando o algoritmo SARSA, o método ajusta-se às incertezas e restrições do sistema.

Da mesma forma, Aadhith Sanjay Prasanna et al. aplicaram os algoritmos SARSA, Expected SARSA e Q-Learning para automatizar decisões no controle de tráfego aéreo (ATC), adaptando-se às mudanças no espaço aéreo e rastreando aviões com base em direção, velocidade e altitude [75].

A designação SARSA resulta dos cinco valores envolvidos na regra de atualização da matriz Q-Table: s, a, r, s´, a´, ou seja, estado, ação e recompensa atuais, estado e ação próximas. O SARSA segue uma estratégia de aprendizagem *on-policy*, ou seja, aprende e executa a política diretamente enquanto explora o ambiente. Além disso, é considerado um método de aprendizagem por diferença temporal (TD), pois atualiza as suas estimativas com base nas diferenças entre previsões consecutivas [76].

O Algorithm 4.1 representa o pseudo-código do método DAC para calcular a Q-Table considerando $\mathcal{S}$ (espaço) e $\mathcal{A}$ (ação).

Algorithm 4.1 Pseudo código adaptado de Richard Sutton [73]

Input: α **learning rate.** ϵ **random action probability,** γ **discount factor**
initialize $Q(s, a)$ **arbitrarily**
$\pi =$ **e-greedy policy** $Q(s, a)$
repeat (for each episode):
  initialize state s
  while s **is not a terminal state**
    choose a **from** s **using policy derived from** $Q(s, a)$
    take action a**, observe** r**,** s'
    $Q(s, a) \mathrel{+}= \alpha \cdot (r + \gamma \cdot Q(s', a') - Q(s, a))$
    $s = s'$
    $a = a'$

Onde:

- s: Estado atual
- a: Ação atual
- $Q()$: Matriz Q-Table
- s‘: Próximo estado
- a‘: Próxima ação
- α: Taxa de aprendizagem, normalmente definido entre 0 e 1. O valor 0 significa que os valores Q nunca são atualizados, portanto, nada é assimilado; definir esta taxa com um valor elevado, como 0,9, significa que a aprendizagem pode ocorrer rapidamente
- γ: Fator de desconto, também definido entre 0 e 1. Este fator faz com que as recompensas futuras valham menos do que as recompensas imediatas
- ϵ: Taxa de escolha de uma ação aleatória a partir de um determinado estado

Neste trabalho, os parâmetros do método SARSA foram definidos com os seguintes valores:

- a taxa de aprendizagem (α) é de 0.1
- o fator de desconto (γ) é de 0.9
- a política ϵ-greedy tem uma probabilidade ϵ de 0.1 para selecionar uma ação aleatória.

A função de valor $Q(s, a)$ é inicializada com valores arbitrários para todos os pares de estado-ação. Esses parâmetros são essenciais para determinar o comportamento do agente durante o processo de aprendizagem e afetam a forma como o agente equilibra a exploração de novas ações e a exploração das ações conhecidas para maximizar a recompensa.

Uma política ϵ-greedy é usada para equilibrar a *exploration* e *exploitation*. Com probabilidade ϵ, o agente escolhe uma ação aleatória (exploração); caso contrário, ele escolhe a ação que maximiza

o valor de $Q(s, a)$ para o estado atual (exploração). Isto significa que há uma chance (ϵ) de escolher uma ação aleatória e uma chance (1-ϵ) de escolher a ação com o maior valor Q.

ϵ-greedy policy:

Probability ϵ: $a = A(s)$ *random*

Probability 1-ϵ: $a = \arg\max Q(s, a)$

O método é repetido por um número definido de episódios, que correspondem às tentativas do agente de completar uma tarefa no ambiente. Para cada episódio existe um estado s definido. Enquanto o estado s não terminar, o agente escolhe uma ação a com base na política ϵ-greedy registada na tabela $Q(s, a)$, executa a ação a, observa a recompensa imediata r e o novo estado s', e atualiza a tabela Q com base na equação definida de seguida:

Atualização Q-table:

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \dot{Q}(s', a') - Q(s, a) \right]$$

Em SARSA, a atualização dos valores Q é feita usando a recompensa real recebida e a ação selecionada no próximo estado. O estado atual s é então atualizado para o próximo estado s', e a ação a é atualizada para a próxima ação a'. Esta atualização do valor Q reflete a recompensa imediata recebida e a recompensa futura com fator de desconto a partir do próximo estado e ação. O erro entre a recompensa esperada e a recompensa atual é usado para atualizar o valor Q para o par estado-ação atual.

State	Description
Start	Beginning phase
Half	Middle phase
End	End phase

Tabela 4.6: Lista de estados implementados

São definidos três estados distintos para guiar o comportamento do agente: início, meio e fim. Cada um desses estados representa um terço do número total de episódios. A tabela 4.6 mostra o conjunto de estados $\mathcal{S}$ observáveis pelo agente em cada episódio. O conjunto de ações $\mathcal{A}$ possíveis em cada episódio correspondem às alterações indicadas para cada parâmetro.

A divisão do processo de otimização nestes diferentes estados foi escolhida porque reflete bem a natureza dinâmica dos problemas de otimização. Nos estágios iniciais, é dada prioridade à exploração, pois o agente tem pouca ou nenhuma informação sobre a função objetivo ou sobre possíveis soluções ótimas. Por isso, aumentar a exploração neste ponto ajuda a alargar as possibilidades de descobrir áreas promissoras no espaço de pesquisa.

De seguida, para melhorar o desempenho da pesquisa à medida que o algoritmo progride, adotamos uma estratégia de transição entre fases onde a melhor solução do estado anterior é levada para o estado seguinte. Esse mecanismo de *warm-start* evita que o processo recomece do zero a

cada transição, aproveitando o conhecimento já adquirido para acelerar a convergência nas fases seguintes.

Para além disto, é adotada uma estratégia de redução progressiva do valor de ϵ em momentos de transição de estados. No inicio do processo de otimização é dada prioridade a uma alta taxa de *exploration*, ajustando a *exploitation* através na alteração de 1% do valor de ϵ, com o intuito de pesquisar as soluções já identificadas como promissoras.

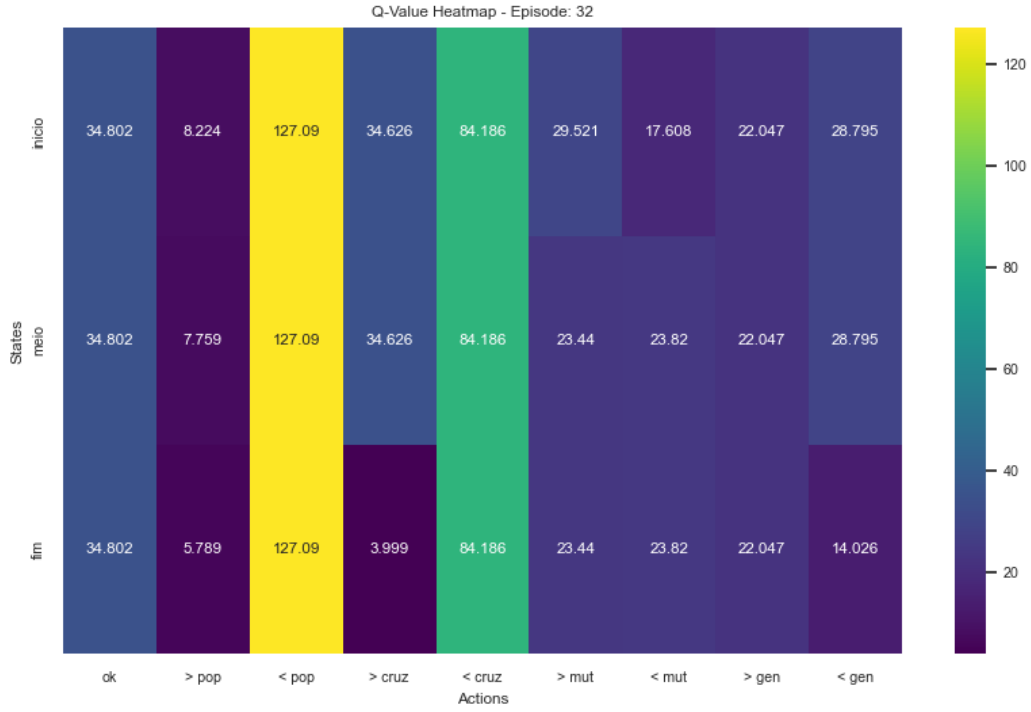


FIGURA 4.5: Mapa de calor da tabela Q-Value após execução da função *benchmark schwefel*

Durante a interação do agente com o ambiente, a matriz de valores é apreendida e atualizada em cada estado específico. A imagem da 4.5 representa um mapa de calor (*heatmap*) dos valores de Q ao longo dos diferentes estados e ações, após 32 episódios. No eixo vertical, observam-se os diferentes estados ("início", "meio", "fim"), enquanto no eixo horizontal estão representadas as diferentes ações possíveis.

As células que apresentam as cores roxa e azul indicam os valores mais baixos de Q, o que significa que essas combinações de estado-ação são as mais promissoras em termos de minimização do objetivo. No contexto do problema, esses valores mais baixos são desejáveis, já que o objetivo é minimizar uma determinada função de custo. Por exemplo, a ação $> pop$ no estado "início"apresenta um valor de Q baixo, sugerindo que aumentar a população nesse estágio é eficaz para alcançar uma melhor performance do algoritmo.

O *heatmap* demonstra claramente as preferências do agente em termos de ações para minimizar a função objetivo em diferentes estados. As combinações de estado-ação que resultam em menores valores de Q são as mais promissoras no contexto da minimização e o agente ajusta os seus parâmetros com base nessas informações para melhorar continuamente a performance do algoritmo de otimização.

Implementação do SARSA no ajuste dinâmico de parâmetros no GA Após a revisão da base teórica, o próximo passo é a implementação de uma política para ajustar dinamicamente os parâmetros do algoritmo meta-heurístico durante o processo de otimização. Esta abordagem permite a configuração adaptativa dos parâmetros do GA com base no seu desempenho ao longo do tempo, o que potencialmente aumenta a competência na procura de soluções ótimas.

O propósito é encontrar a configuração ideal que minimize o custo da função objetivo. O código detalhado no algoritmo 4.2 demonstra a aplicação do método SARSA para ajustar dinamicamente os parâmetros do GA num ambiente de RL. A cada episódio, o agente modifica os parâmetros com base no estado atual e na política memorizada, procurando melhorar continuamente o desempenho do GA.

Algorithm 4.2 Função *DAC Agent*

```
1:  def DAC_Agent(episodes,parameters,bench):
2:      env = RLEnvironment(parameters,bench)
3:      ql = sarsa()
4:      ql.create_table(states, actions)
5:      action = ql.policy(state)
6:      for episode in range(episodies):
7:          cfg = ql.param[state][action]
8:          next_state, reward, done, sol, fv, cfg = env.step(action, cfg)
9:          next_action = ql.policy(next_state)
10:         ql.learning(state, action, reward, next_state, next_action, cfg)
11:         action = next_action
12:         state = next_state
13:     end for
```

Em apêndice A.3 são apresentadas as dependências necessárias na função 4.2. Uma delas, o objeto *env* resulta da classe *RLEnvironment*, simula um ambiente de RL que interage com um GA. Nesse ambiente, o agente de RL ajusta parâmetros como: tamanho da população, taxa de cruzamento, taxa de mutação e número de gerações, visando encontrar a configuração ótima que maximize o valor da função *fitness*, uma métrica que avalia a qualidade das soluções geradas pelo GA.

O método *step*, que integra esta classe, representa um passo nesse ambiente, onde o agente seleciona uma ação para ajustar os parâmetros do GA. A atualização dinâmica dos parâmetros

ocorre em resposta às ações do agente conforme a política adotada. Por exemplo, a configuração dos parâmetros como tamanho da população ou número de gerações pode ser ajustada em 5% para cima ou para baixo, dependendo da ação selecionada. Esta adaptação contínua permite que a heurística procure as melhores soluções ao longo do tempo.

Por seu lado, o método *run_heuristic* executa o GA com a configuração atualizada e retorna os resultados, incluindo o melhor indivíduo e o seu valor de *fitness*, enquanto o método *compute_reward* calcula e retorna a recompensa baseada no valor de *fitness* obtido. No código atual, a recompensa é simplesmente o valor de *fitness*.

Para gerir a política de aprendizagem e decidir as ações a serem tomadas, o código cria um objeto ql do tipo $Sarsa$, indicado em apêndice (A.2). Esse objeto implementa o algoritmo SARSA, para aprender uma política ótima que maximize a recompensa total ao longo do tempo. O objeto ql é inicializado com três parâmetros principais: (ϵ=0.1), que define a taxa de exploração e a probabilidade de o agente escolher uma ação aleatória; (α=0.1), que controla a taxa de aprendizagem e o impacto dos novos valores de recompensa nos valores Q atuais; e (γ=0.9), que determina a importância das recompensas futuras em relação às recompensas imediatas.

O método *create_table* cria a tabela $Q(qsa)$, que armazena os valores Q para cada combinação de estados e ações, e também uma tabela $param$, que regista as configurações associadas a cada par estado-ação, como: tamanho da população, taxas de cruzamento e mutação, número de gerações e valor de *fitness*.

Já o método *policy* define a estratégia de escolha de ações com base nos valores Q atuais, permitindo que o agente explore novas ações ou siga a ação com o menor valor Q, dependendo de ϵ. O método *learning_initial* atualiza o valor Q da ação escolhida durante a fase inicial da aprendizagem, utilizando o valor de *fitness* como recompensa. Durante a aprendizagem contínua, o método *learning* ajusta os valores Q com base na recompensa recebida e nas estimativas futuras. Se uma nova configuração de parâmetros resultar num valor de *fitness* melhor, é substituída a configuração anterior na tabela $param$.

4.3.2 Método Deep SARSA

Deep SARSA é um método de aprendizagem por reforço profundo que combina SARSA com redes neuronais para lidar com espaços de estado e ação de alta dimensão. Exemplos recentes comprovam avanços significativos na aplicação destes métodos de RL.

No estudo de Jayarama Pradeep et al, foi introduzido o DeepFore como um sistema de previsão de energia que melhora a precisão na geração de energia eólica em sistemas híbridos. A integração de modelos meteorológicos detalhados com técnicas de *clustering*, otimização e Deep SARSA proporciona uma abordagem robusta para melhorar a previsão da produção de energia. [77].

Andreas Metzger et al., na sua pesquisa indicam uma evolução significativa na forma como os sistemas auto-adaptativos podem lidar com ambientes dinâmicos e complexos. Ao incorporar modelos de RL profundo, como *Deep Q-Network* ou Deep SARSA, estas novas estratégias podem

aumentar a capacidade do sistema de explorar e de adaptar-se a uma gama ainda maior de ações e estados [78].

O Deep Q-Network e o Deep SARSA são semelhantes na forma como aprendem estimativas de valor, mas diferem na política de atualização. O Deep SARSA é *on-policy*, o que significa que a atualização dos valores Q leva em consideração o equilibrio entre explorar a melhor ação atual ou outra ação exploratória com alguma probabilidade (tipicamente pequena).

Tal como referido anteriormente, pretende-se que o o agente de aprendizagem procure configurar os parâmetros de uma meta-heurística, por exemplo, o algoritmo genético. A estrutura do código consiste em definir o agente, definir o ambiente, criar a rede neuronal, treinar o agente e, finalmente, apresentar os resultados.

As ações representam os possíveis ajustes que o agente escolheu fazer, por exemplo, aumentar ou diminuir o tamanho da população, a taxa de *crossover*, a taxa de mutação e o número de gerações, cujos valores foram indicados para cada meta-heurística nas tabelas 4.2 e 4.5.

Os estados, por sua vez, refletem a configuração atual dos parâmetros em diferentes momentos do processo de otimização, seja no início, meio ou fim da execução do algoritmo. Na transição de estados foi introduzida uma variável designada de *threshold-based reset* particularmente útil em ambientes dinâmicos e no delicado equilíbrio das estratégias de ajuste adaptativo de parâmetros. Esta técnica visa uma abordagem renovada de *exploration* utilizando um mecanismo de reinicialização (*reset*) de parâmetros sempre que o desempenho da otimização seja inferior a um valor aceitável. Caso o desempenho ultrapasse esse limite sem apresentar ganhos significativos, o *reset* dos parâmetros impede a convergência prematura, renovando a capacidade de exploração do algoritmo.

Além da variável *threshold-based reset*, foi ainda considerada uma taxa de atualização de 10% nos valores de ajuste dos parâmetros das meta-heurísticas. O objetivo foi otimizar o processo de ajuste dinâmico, evitando a estagnação e permitindo uma adaptação mais dinâmico e progressiva.

Definir corretamente os parâmetros em cada estado do processo é crucial porque eles são a base sobre a qual a rede neuronal Q faz as suas previsões. Se em cada estado não se absorvem todas as informações relevantes, o agente não será capaz de aprender uma política capaz e não será capaz de tomar uma decisão informada sobre qual a ação a seguir. Em Deep SARSA, essa política é definida por uma rede neuronal, enquanto os estados representam a configuração atual do ambiente em que o agente está atuando.

De seguida, detalha-se o método Deep SARSA e os conceitos associados. O código referido em 4.3 mostra o processo de desenvolvimento do método usando *Replay Memory* (memória de repetição), o que não sendo obrigatória, serve para armazenar experiências anteriores numa memória de repetição. Cada transição contém o estado atual, a ação tomada, a recompensa obtida e o próximo estado e ação. A utilização de amostras de transições passadas ou *minibatchs* de treino permite maior qualidade na aprendizagem e evita correlações entre as atualizações consecutivas.

Algorithm 4.3 *Deep SARSA* utilizando *Experience Replay* adaptado de Volodymyr Mnih [79]

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1 to M **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    for $t = 1$ to T **do**
        With probability ϵ, select a random action a_t
        otherwise select $a_t = Q(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Um dos conceitos importantes é a função Q definida por uma rede neuronal que toma um estado e uma ação como entrada e retorna um valor de Q, que representa a expectativa de recompensa nessa ação-estado. Assim, a atualização desses valores segue a regra SARSA, que usa a transição completa (estado atual, ação atual, recompensa, próximo estado e próxima ação) para atualizar a função Q.

No código 4.4 é apresentada a estrutura de uma rede neuronal projetada para otimizar os parâmetros de um algoritmo heurístico usando RL.

Algorithm 4.4 Definição da Rede Neuronal Q

```
s_network = nn.Sequential()
  nn.Linear(states, 128) ,
  nn.ReLU() ,
  nn.Linear(128, 64) ,
  nn.ReLU() ,
  nn.Linear(64, actions)
```

onde

- **states**: Representa o número de elementos no estado, ou seja, a dimensão da entrada da rede neuronal.
- **nn.Sequential**: É a estrutura sequencial da rede neuronal *s_network*, que permite juntar camadas de forma organizada, definindo a ordem em que os dados passam pelas camadas.

- **nn.Linear(states, 128)**: Esta é a primeira camada linear (totalmente conectada), que recebe como entrada a dimensão dos estados e gera 128 saídas. Cada estado é conectado a cada um dos 128 neurónios da camada oculta.
- **nn.ReLU**(): Função de ativação ReLU que introduz não-linearidade, essencial para que a rede possa registar padrões ou relações complexas nos dados.
- **nn.Linear(128, 64)**: Segunda camada linear que recebe as 128 saídas da camada anterior e gera 64 saídas.
- **nn.ReLU**(): Outra função de ativação ReLU aplicada após a segunda camada para manter a não-linearidade.
- **nn.Linear(64, actions)**: Camada de saída, com número de saídas igual ao número de ações possíveis. Esta gera os valores Q, representando a estimativa de recompensa para cada ação.

Esta rede neuronal é composta por uma sequência de três camadas totalmente conectadas (lineares) e duas funções de ativação *Rectified Linear Unit* (ReLU). As camadas ocultas (com 128 e 64 unidades) utilizam ReLU para garantir que a rede seja capaz de aprender relações não-lineares entre os dados de entrada e as saídas. A última camada não possui uma função de ativação, pois a sua função é estimar diretamente os valores Q para cada uma das ações possíveis, o que é comum em redes neuronais usadas na aprendizagem por reforço.

A arquitetura sequencial da rede é simples e funcional: os dados fluem de forma unidirecional, camada a camada, desde a entrada (representada pelos estados) até a saída (as ações). Durante a fase de inferência, o estado atual é passado através das camadas e a rede calcula um valor Q para cada ação, permitindo que o agente escolha a melhor ação com base nesses valores. Durante o treino, a rede ajusta os seus pesos por meio de *backpropagation*, em que o erro na estimativa do valor Q é reavaliado na rede, ajustando os pesos de cada camada de acordo com os gradientes calculados.

Cada camada é definida com os seus parâmetros específicos, como o número de neurónios e a função de ativação. A primeira camada, camada de entrada (*states*) recebe o estado atual do ambiente, que representa a configuração dos parâmetros do algoritmo heurístico naquele momento. Esses parâmetros podem incluir, por exemplo, o tamanho da população, a taxa de mutação e outros fatores dependentes do problema. As duas camadas ocultas aplicam transformações, uma primeira (128 neurónios) cria uma representação densa do estado, para que a rede capte padrões complexos nos dados, e posteriormente, uma segunda camada oculta (64 neurónios) permite compactação progressiva das informações relevantes para a decisão. Nestas duas, é aplicada a função de ativação ReLU para não-linearidade à saída da camada.

Finalmente, a camada de saída produz N valores de Q, onde N é o número de ações possíveis que o agente pode escolher. Cada valor Q reflete a expectativa de recompensa para uma determinada ação. Uma camada linear na saída é geralmente usada para calcular os valores Q (Q-values), que correspondem à qualidade de cada ação possível num determinado estado. A ação escolhida é

aquela com o maior Q-value, o que dispensa a necessidade de funções de ativação como *softmax*. Nesse caso, os valores Q indicam diretamente a expectativa de recompensa para cada ação e o agente escolhe a ação que maximiza esse valor.

Resumindo, as dimensões de entrada da sua rede neuronal representam os parâmetros do algoritmo heurístico, e as saídas representam as ações possíveis para manipular esses parâmetros. Isso é apropriado, já que a rede neuronal é usada para aprender uma política que determina o ajuste dos parâmetros de uma heurística, com base em diferentes estados do ambiente.

Outro conceito importante, também considerado, no método SARSA prende-se com a noção de equilíbrio entre *exploration* (experimentar novas ações para descobrir o seu valor) e *exploitation* (seguir a política atual para maximizar as recompensas conhecidas). Quando a estratégia ϵ-greedy é configurada de maneira adequada facilita esse equilíbrio.

A política ϵ-greedy usada no código 4.5 opta por uma ação com base nos valores Q-value estimados pela rede neuronal. Dentro da probabilidade ϵ, escolhe-se uma **ação aleatória** para explorar, e dentro da probabilidade 1-ϵ, escolhe-se a ação com o maior Q-value para explorar.

Algorithm 4.5 Método *policy* da classe Deepsarsa

```
1: class Deepsarsa:
2:     def policy(self, state):
3:         av = self.s_network(state).detach()
4:         if torch.rand(1) < self.epsilon
5:             next_action = torch.randint(av.size()[1], (1, 1))
6:         else
7:             next_action = torch.argmin(av, dim=1, keepdim=True)
8:         endif
9:         return next_action
```

O método *policy* usa a rede neuronal *s_network* para calcular os valores Q-value para todas as ações possíveis num dado estado e escolhe a melhor ação com base na política ϵ-greedy.

Antes de apresentar o algoritmo 4.7 que representa o *core* do método Deep SARSA importa mencionar a técnica de *Experience Replay* [80], evocada por Dongbin Zhao et al. A inclusão da memória de repetição é uma extensão importante do método Deep SARSA.

Nos métodos tradicionais por reforço, o processo de aprendizagem e atualização ocorre de forma sequencial, onde cada nova amostra tem uma atualização imediata. Esse método pode ser bastante lento, pois cada atualização depende de uma nova experiência. Para melhorar a escalabilidade da aprendizagem, os dados históricos são armazenados em memória (*Experience Replay*) e reutilizados continuamente para novo treino.

O agente armazena as suas experiências (transições entre estados) numa memória de repetição D e cada transição contém o estado atual, a ação escolhida, a recompensa obtida e o próximo estado. Em cada iteração do treino, um *minibatch* aleatório dessas experiências, é registado e utilizado

Algorithm 4.6 Classe ReplayMemory

```
1:  class ReplayMemory:
2:      def __init__(self, capacity=1000000):
3:          self.capacity = capacity
4:          self.memory = []
5:          self.position = 0
6:
7:      def insert(self, transition):
8:          if len(self.memory) < self.capacity
9:              self.memory.append(None)
10:         end if
11:         self.memory[self.position] = transition
12:         self.position = (self.position + 1) % self.capacity
```

para treinar a rede neuronal, o que ajuda a reduzir a correlação entre as amostras e a evitar o *overfitting*.

A memória de repetição permite que o agente registe as experiências anteriores para treinar a rede neuronal, enquanto a rede-alvo (a ser referida mais adiante) estabiliza o treino ao fornecer valores-alvo mais consistentes e a política ϵ-greedy equilibra *exploration* e *exploitation* durante o treino.

Algorithm 4.7 Método *learning* da classe Deepsarsa

```
1:  class Deepsarsa:
2:      def learning(state, action, reward, done, next_state):
3:          self.memory.insert([state, action, reward, done, next_state])
4:          if (self.memory.can_sample(self.batch_size):
5:              state, action, reward, done, next_state = self.memory.sample(self.batch_size)
6:              qsa = self.s_network(state).gather(1, action)
7:              next_action = self.policy(next_state)
8:              next_qsa = self.target_s_network(next_state).gather(1, next_action)
9:              target = reward + self.gamma * next_qsa
10:             loss = F.mse_loss(qsa, target)
11:             self.s_network.zero_grad()
12:             loss.backward()
13:             self.optim.step()
14:             self.stats['MSE Loss'].append(loss.item())
```

O método *learning* atualiza os valores Q-value com base nas transições de estado observadas. Aproveita o *Mean Squared Error* (MSE, Erro Quadrático Médio) para ajustar os pesos da rede neuronal, minimizando a diferença entre o valor Q-value atual e o valor Q-value alvo, estimado pela recompensa recebida mais o valor descontado (fator de desconto) do próximo estado, sendo representado no código por $target$.

A rede neuronal é ensaiada para melhorar a política com base nas interações com o ambiente, que no caso do Deep SARSA pode ser definida pela equação 4.5:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{4.5}$$

onde:

- α é a taxa de aprendizagem.
- γ é o fator de desconto
- $Q(s_{t+1}, a_{t+1})$ é o valor da função Q para a próxima ação a_{t+1} escolhida para o próximo estado s_{t+1}, seguindo a política em uso

O valor Q-value é, assim, essencial para determinar quais ações que o agente deve tomar para maximizar as recompensas futuras. Este valor é continuamente ajustado durante o treino para melhorar a política do agente, permitindo uma tomada de decisão mais favorável.

A função de recompensa é outro componente essencial na aprendizagem por reforço, sendo importante garantir que esta forneça *feedback* útil sobre o desempenho da heurística escolhida, incentivando a aprendizagem de políticas eficazes. Deste modo, a função de recompensa deve ser desenvolvida para orientar o algoritmo em direção a soluções ótimas.

A equação seguinte representa o valor esperado da recompensa acumulada quando começamos no estado s, escolhemos a ação a, e, a partir daí, seguimos a política ótima. Por outras palavras, é a estimativa da importância da ação a no estado s, no contexto de futuras recompensas. A equação de *Bellman*, central em muitos destes algoritmos, é frequentemente usada para atualizar os valores Q(s, a) com base nas experiências acumuladas no ambiente e pode ser definida assim:

$$Q(s, a) = \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \mid s_t = s, a_t = a\right] \tag{4.6}$$

onde:

- R_t é a recompensa recebida no tempo t
- γ é o fator de desconto (entre 0 e 1), que determina a importância das recompensas futuras.

A importância desta função também sucede quando o valor da recompensa influencia indiretamente os pesos da rede neuronal. Isto, ocorre porque a recompensa é usada no cálculo da função de erro, a qual determina os gradientes dos pesos da rede durante o treino. Estes gradientes orientam a atualização dos pesos através de algoritmos de otimização, como o gradiente descendente. Deste modo, as recompensas altas para determinadas ações ou sequências de ações levam a ajustes nos pesos, de forma a probabilidade dessas ações serem escolhidas no futuro.

Além disso, o valor da recompensa influencia a política de *exploratiom* e *explotation* do agente. As ações que resultam em recompensas altas podem ser exploradas mais frequentemente, o que modifica os pesos para favorecer essas ações. Em algumas configurações, a recompensa

também pode afetar a aprendizagem de representações internas na rede neuronal, ajudando a rede a representar estados específicos ou características das observações de forma mais capaz.

Outro conceito importante em métodos de aprendizagem por reforço, como é caso, do *Deep SARSA* ou *Deep Q-Network* prende-se com a utilização de duas redes neuronais: rede principal (*s_network*) e a rede de destino (*target network*) [81].

A rede principal é responsável por gerar as estimativas dos valores Q (valores de ação), sendo atualizada regularmente durante o treino. Por outro lado, a rede de destino é utilizada para calcular os valores-alvo (*targets*) usados no treino da rede principal e é atualizada com menor frequência.

A atualização periódica da rede de destino ajuda a estabilizar o processo de treino. Caso esta atualização ocorra de forma muito frequente, a rede de destino poderá acompanhar as mudanças rápidas da rede principal, resultando em instabilidade no processo de aprendizagem. Ao atualizar a rede de destino a cada N episódios, é criado um suporte mais estável para o cálculo dos valores-alvo, o que contribui para melhorar a estabilidade do treino.

Pelo código apresentado em 4.8 são copiados os pesos da rede principal para a rede de destino, isto permite que a rede de destino passe a ter os mesmos pesos que a rede principal no momento da atualização. Esta atualização ocorre a cada 10 episódios (if episode % 10 == 0). Esse intervalo pode ser ajustado conforme a necessidade, com o objetivo de equilibrar a estabilidade e a relevância dos valores calculados pela rede de destino.

Algorithm 4.8 Método load_dict da classe Deepsarsa

```
1: class deepsarsa:
2:     def load_dict(self, episode, ep_return):
3:         self.stats['Returns'].append(ep_return)
4:         if episode %10 == 0:
5:             self.target_s_network.load_state_dict(self.s_network.state_dict())
6:         return self.stats
```

Após a possível atualização da rede de destino, o método retorna as estatísticas (*self.stats*), que podem incluir métricas como erro médio, usadas para monitorizar o desempenho do processo de aprendizagem.

A função F.mse_loss é utilizada para calcular o erro médio quadrático (*loss*) entre dois tensores: qsa, que representa os valores Q previstos pela rede neuronal para as ações selecionadas, e $target$ representa os valores Q alvo, obtidos pela rede destino.

O MSE é uma métrica comum para avaliar a diferença entre os valores previstos e os valores alvos, sendo calculado como a média dos quadrados das diferenças entre esses valores. A fórmula do MSE é:

$$\text{MSE Loss} = \frac{1}{n}\sum_{i=1}^{n}(qsa_{b,i} - target_{b,i})^2 \tag{4.7}$$

Onde:

- n é o número total de elementos nos tensores qsa e $target$
- $qsa_{b,i}$ é o valor Q previsto pela rede neuronal para a ação i no batch b
- $target_{b,i}$ é o valor Q alvo para a ação i no batch b

A função F.mse_loss do PyTorch calcula automaticamente o erro médio quadrático entre $qsa_{b,i}$ e $target_{b,i}$ e devolve o valor do MSE sobre todos os elementos.

4.4 Tecnologias

Nesta secção são descritas as tecnologias que foram empregues no desenvolvimento, começando pela linguagem de programação escolhida e as bibliotecas mais relevantes.

Como ambiente de programação escolheu-se o *Colab*, uma ferramenta de código aberto. O Colab permite aos utilizadores criar e compartilhar documentos que incluem código interativo, visualizações de dados e descrições detalhadas em *Markdown*. O Google Colaboratory, mais comummente referido como "Google Colab"ou simplesmente "Colab", é um projeto de pesquisa para protótipos de modelos de aprendizagem máquina em hardware poderoso, como GPUs e TPUs [82].

Por outro lado, permite a colaboração efetiva em ciência de dados para capitalizar a experiência de cada membro da equipa, contribuindo para aprimorar a qualidade do trabalho. Os *notebooks* computacionais representam uma solução interativa e conveniente para cientistas de dados compartilharem e acompanharem o processo de exploração de dados. Isso é alcançado através de uma combinação de código, texto narrativo, visualizações e outras formas avançadas de media [70].

O mecanismo de funcionamento do *Colab* é ilustrado na Figura 4.6. Este é um *notebook Jupyter* baseado na *web*, acedido gratuitamente através de uma conta do Google para executar código *Python*. Os *notebooks Colab* são armazenados no Google Drive, podendo ser acedida por qualquer navegador da web [83]. O ambiente *Colab* também permite configurar uma *Virtual Machine* (VM, Máquina Virtual) como tempo de execução conectando-se a Graphics Processing Units (GPUs, unidades de processamento gráfico) hospedadas pelo Google ou por meio dos serviços da plataforma Google *Cloud* [84].

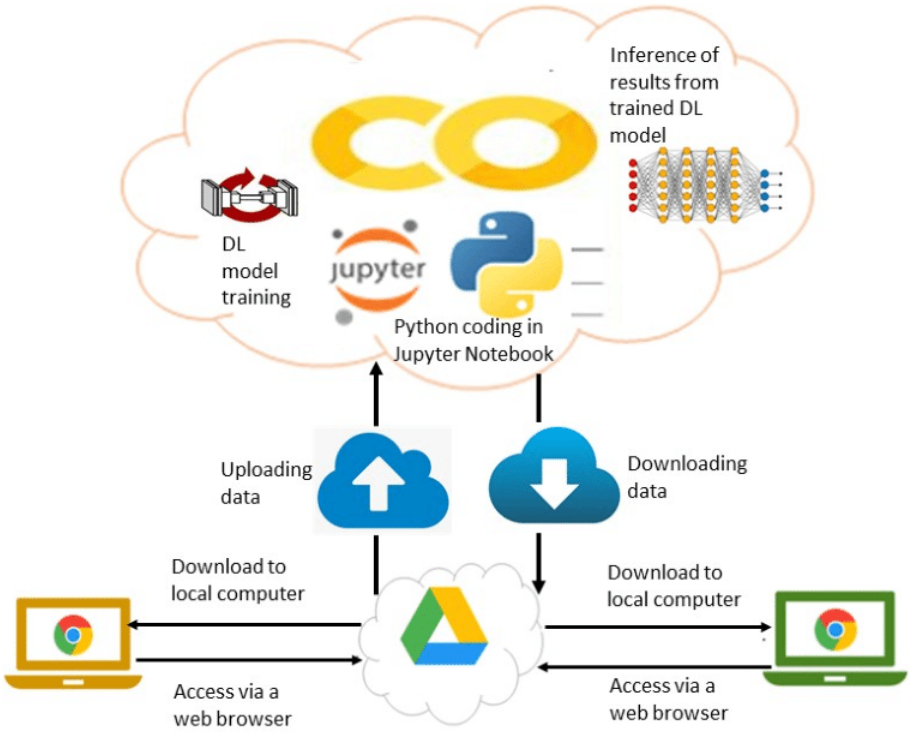


FIGURA 4.6: Mecanismo do *Colab* desenhado por Biyanka Ekanayake [84]

Para a implementação dos casos de uso nesta dissertação, optou-se pelo *Python* como linguagem de programação. Esta escolha deve-se à sua ampla utilização no campo de *Machine Learning* (ML, Aprendizagem Máquina), proporcionada pela simplicidade e facilidade de uso. Para mais, *Python* conta com uma comunidade grande e ativa que desenvolveu várias bibliotecas e *frameworks* para ML, tais como *TensorFlow*, *PyTorch* e *Scikit-learn*.

A *API Sequential* da biblioteca *PyTorch* (ou Sequential do *Keras*, em *TensorFlow*) permite criar esse tipo de rede de maneira conveniente e intuitiva.

Essas bibliotecas *Python* fornecem funções e modelos que podem ser empregues em tarefas como pré-processamento de dados, treino de modelos e avaliação. Essa abordagem facilita o trabalho dos programadores, permitindo que se concentrem no problema em questão, em vez de lidar com detalhes de implementação em baixo nível. Ademais, *Python* oferece um conjunto diversificado de bibliotecas para visualização, manipulação e análise de dados, como pandas, *numpy, matplotlib e seaborn*, todos essenciais para ML.

4.5 Resumo

Neste capitulo abordou-se a conceptualização da solução da configuração dinâmica de algoritmos, usando políticas de DAC.

Entende-se que os métodos de diferença temporal (TD), incluindo SARSA e Q-Learning, são fundamentais para a aprendizagem por reforço, já que permitem que os agentes aprendam diretamente através da sua experiência sem precisar de um modelo explícito do ambiente. Ambos os métodos têm suas próprias características e aplicações adequadas, dependendo se o cenário exige um método *on-policy* ou *off-policy*. A escolha entre SARSA e Q-Learning pode influenciar a forma como o agente aprende e se adapta ao ambiente, especialmente em termos de exploração e convergência para uma política ótima.

Com o método de Deep SARSA, combinando esta técnica de RL com uma rede neuronal pretende-se aprender uma política de controle que otimiza os parâmetros de um algoritmo heurístico com base em diferentes estados do ambiente. As dimensões de entrada correspondem ao estado atual (os parâmetros), e as saídas correspondem às possíveis ações que o agente pode tomar para ajustar esses parâmetros. A simplicidade da estrutura sequencial da rede torna o processo eficiente, enquanto as camadas de ativação *ReLU* garantem que a rede possa capturar a complexidade necessária para modelar o problema de otimização de forma eficaz.

A implementação prática destes métodos foi efetuada com recurso à linguagem de programação *Python* no ambiente *Colab*.

Capítulo 5

Resultados - Casos de Estudo

O objetivo deste capítulo é apresentar os resultados experimentais e validar os métodos SARSA e Deep SARSA em contexto de otimização e planeamento online de parâmetros de algoritmos heurísticos. O capítulo explora a aplicação e a capacidade destes métodos em problemas de otimização, abordando diferentes casos de estudo, em funções *benchmark*, otimização de portfólios, alocação de doentes a Unidades de Cirurgia (UC) e a Unidades de Cuidados Intensivos (UCI).

No final deste capítulo, é apresentada uma análise detalhada dos dados recolhidos e são discutidas as implicações dos resultados. Esta análise experimental é crucial para validar as abordagens propostas e identificar oportunidades de melhoria, quer no campo do aprendizagem por reforço, quer na aplicação prática de algoritmos populacionais. Também, é feita uma breve referência à importância das medidas de segurança e privacidade para garantir a integridade e confidencialidade dos dados recolhidos.

5.1 Abordagem Preliminar

Em relação aos casos de estudo, o capítulo começa por abordar o uso de funções *benchmark* para testar e validar os métodos de configuração, proporcionando uma base sólida para a comparação dos algoritmos meta-heurísticos em cenários controlados. Esta análise preliminar permite observar o desempenho dos algoritmos em problemas padronizados, facilitando a avaliação das suas capacidades em ambientes diferentes.

Neste contexto, comparam-se métodos baseados em algoritmos de reforço, como SARSA e Deep SARSA, com um *método manual*, que utiliza parâmetros gerados aleatoriamente, permitindo uma avaliação abrangente da eficiência e eficácia dos algoritmos em ambientes práticos e de alta variabilidade.

A seguir, detalha-se o método Manual, que será utilizado para avaliar o desempenho dos algoritmos em comparação com as abordagens automáticas mencionadas.

5.1.1 Método Manual

O método manual adotado para a avaliação dos algoritmos heurísticos consiste num procedimento em que os seus parâmetros são definidos aleatoriamente em cada execução. O processo é implementado através de um código que executa o *Genetic Algorithm* (GA, Algoritmo Genético) com uma população inicial gerada com base em configurações aleatórias de parâmetros dentro de intervalos definidos neste estudo e avaliado ao longo de uma série de episódios.

A abordagem manual tem como propósito avaliar a performance do algoritmo heurístico em condições distintas, registando o melhor resultado e a configuração dos parâmetros a cada execução. Nesta análise de desempenho, também são recolhidos dados como a média do valor *fitness*/função objetivo da solução encontrada, o desvio padrão e o tempo médio. Estas métricas permitem comparar o comportamento com os métodos, SARSA e Deep SARSA, fornecendo uma base de referência para analisar as vantagens e limitações destas abordagens. O código 5.1 descreve este processo,

Algorithm 5.1 Código para método manual de configuração de parâmetros

```
1: Input: Número de episódios, parâmetros iniciais
2: Output: Resultados, soluções e configurações
3: resultados ← lista vazia
4: solucoes ← lista vazia
5: configuration ← lista vazia
6: for cada episódio em range(episodes) do
7:     alg ← ALG(genes, bench, chromosome_binarie)
8:     config_parameters ← alg.initialparameters(parameters)
9:     (solution, fitness_value) ← alg.run()
10:    append(fitness_value, resultados)
11:    append(solution, solucoes)
12:    append(config_parameters, configuration)
13: end for
14: end_time ← time.time()
```

Esse processo de ajuste manual pode ser visto como uma abordagem exploratória, onde os parâmetros são definidos sem uma estratégia sistemática, baseando-se principalmente em tentativas e erros, pelo que a falta de uma estratégia de otimização adaptativa para os parâmetros pode resultar em desempenhos menos consistentes e potencialmente menos capazes.

5.1.2 Otimização de Funções *Benchmark*

Com base na análise teórica discutida no capítulo 4, foi utilizado um conjunto de *benchmarks* para validar o desempenho das políticas de *Dynamic Algorithm Configuration* (DAC, Configuração Dinâmica de Algoritmos). O objetivo principal foi determinar a configuração ideal de parâmetros

para algoritmos selecionados, como GA ou PSO, visando alcançar soluções próximas do valor ótimo num conjunto de funções *benchmark*. Para avaliar a capacidade do modelo em lidar com diferentes tipos de problemas, foram considerados vários problemas de otimização, incluindo a análise de funções *benchmark* amplamente reconhecidas na literatura [85].

A seguir, na tabela 5.1, são apresentadas as funções *benchmark* referidas no estudo:

Tabela 5.1: Lista de funções *Benchmark*

Benchmark	**Descrição**
Hypersphere	A hypersphere centered in the origin. $$f(\mathbf{x}) = \sum_{i=0}^{N-1} x_i^2$$
hyperellipsoid	A rotated hyperellipsoid centered in the origin. $$f(\mathbf{x}) = \sum_{i=0}^{N-1} \sum_{j=0}^{n} x_j^2$$
schwefel	Non-convex and (highly) multimodal. Location of the minima are geometrical distant. $$f(\mathbf{x}) = 418.9829 \cdot n - \sum_{i=1}^{n} x_i \cdot \sin\left(\sqrt{\lvert x_i \rvert}\right)$$
rosenbrock	Non-convex and unimodal. Global minimum difficult to approximate. $$f(\mathbf{x}) = \sum_{i=0}^{N-2} (100(x_i - x_i^2)^2 + (x_1 - 1)^2)$$
rastrigin	Non-convex and (highly) multimodal. Location of the minima are regularly distributed. $$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$$
griewank	Non-convex and (highly) multimodal $$f(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Esta abordagem permite uma avaliação abrangente do comportamento do modelo em diversas

situações, proporcionando uma compreensão mais profunda das capacidades, limitações e do desempenho das políticas de DAC.

As funções descritas na tabela 5.1, apresentadas por meio de fórmulas matemáticas, são utilizadas como funções objetivo ou *fitness* em problemas de otimização. Elas desempenham um papel crucial na definição do valor da recompensa em problemas de *Reinforcement Learning* (RL, Aprendizagem por Reforço) e na formulação de políticas estado-ação. Cada função *benchmark* oferece uma perspetiva sobre o comportamento de algoritmos de otimização em diferentes contextos, permitindo uma avaliação das políticas propostas. Em RL, a função de recompensa é essencial para orientar o agente em direção a comportamentos desejáveis, sendo esta a escolha adequada de funções objetivo fundamental para o sucesso da aprendizagem.

Nesta primeira experiência é observada a evolução dos parâmetros do GA, escolhidos de forma aleatória, dentro dos limites indicados na tabela 4.1. Foram conduzidos testes utilizando dois métodos: SARSA e Manual. Ambos os métodos foram aplicados sob as mesmas condições e pressupostos, isto é, cada configuração compreendeu a execução única de 32 episódios para cada função de otimização ou *benchmark*: *Rosenbrock*, *Hipersphere*, *Binary*, *Hyperellipsoid*, *Schwefel*, *Rastrigin* e *Griewank*.

Além do ajuste dinâmico ou *online* dos parâmetros do GA, foram utilizados operadores específicos de mutação e cruzamento. Foi aplicado o operador de mutação *mutFlipBit*, o operador de cruzamento *cxTwoPoint* e o método de seleção por torneio (tamanho=3) da biblioteca DEAP. Esses operadores foram escolhidos para garantir a diversidade e a exploração eficiente do espaço de pesquisa, permitindo uma avaliação robusta do desempenho dos algoritmos sob diferentes configurações e condições experimentais.

Os resultados apresentados na tabela 5.2, resumem o desempenho dos dois métodos de otimização para problemas de minimização. A tabela inclui métricas de melhor resultado, média, desvio padrão e tempo de execução para cada função *benchmark*.

TABELA 5.2: Resultados da otimização do GA através dos métodos manual e SARSA em funções *Benchmark* para 32 episódios

Benchmark	Best		Mean All Best		Standard Deviation		Time	
	Manual	SARSA	Manual	SARSA	Manual	SARSA	Manual	SARSA
Rosenbrock	8,786	9,000	57,532	62,279	0,573	47,650	5,668	4,190
Hipersphere	0,000	0,000	0,016	0,002	0,040	0,002	5,542	4,453
Binary	0,000	0,000	0,000	0,000	0,000	0,000	1,331	1,417
Hyperellipsoid	0,000	0,000	0,717	0,034	0,104	0,034	10,382	8,467
Schwefel	42,569	24,180	354,346	113,902	198,252	135,677	8,121	11,102
Rastrigin	0,000	0,000	0,103	0,505	0,079	0,505	10,315	9,316
Griewank	0,000	0,000	0,447	0,179	10,101	0,179	18,220	19,286

Os resultados indicam que, em muitos dos problemas de minimização considerados, tanto o método SARSA quanto o método Manual alcançaram resultados próximos do valor ótimo, que é 0. O que

parece indicar que ambos os métodos são eficazes na resolução desses problemas. No entanto, observa-se que o método SARSA tende a apresentar resultados mais consistentes, com valores médios e desvios-padrão menores, o que indica uma maior estabilidade na sua performance.

Outra das técnica inovadoras que podem ajudar no refinamento dos parâmetros que afetam a obtenção de soluções eficientes e ótimas, refere-se ao *Deep Q-Network* ou mais especificamente o Deep SARSA. Importa, por isso, avaliar e comparar o desempenho dos métodos SARSA e Deep SARSA para o GA, testando-os em cenários de otimização através das funções *Rosenbrock* e *Schwefel*. Nesta abordagem, os testes foram realizados com 5 execuções para cada método, utilizando 32 e 50 episódios em cada. Os resultados apresentados na tabela 5.3, mostram métricas para o melhor resultado, a média dos melhores resultados de cada execução, desvio-padrão e tempos médios

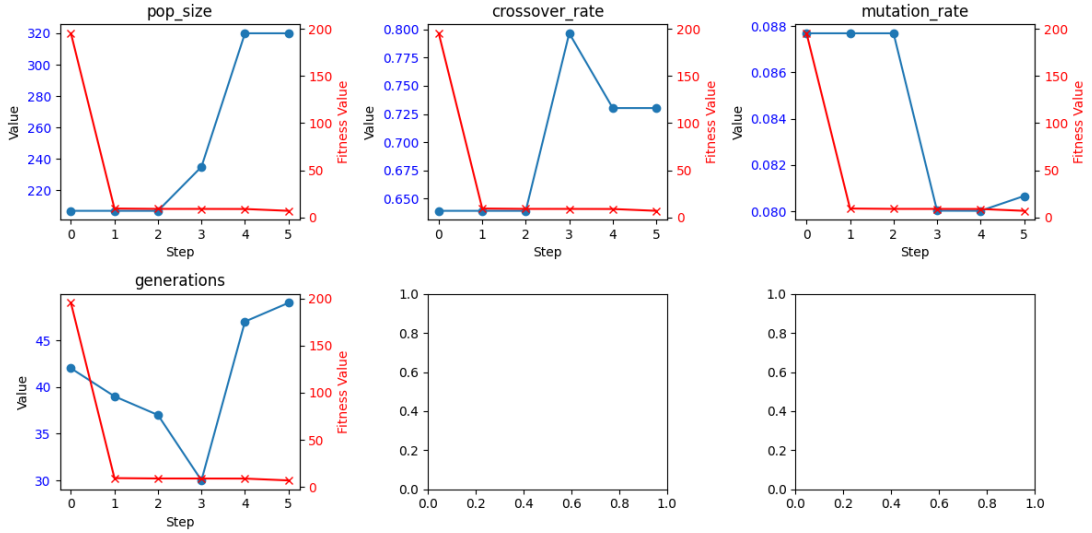
TABELA 5.3: Resultados da otimização do GA através dos métodos SARSA e Deep SARSA em funções *Benchmark* para 5 execuções

Benchmark	Ep	Best		Mean all Best		Standard Deviation		Time	
		D-SARSA	SARSA	D-SARSA	SARSA	D-SARSA	SARSA	D-SARSA	SARSA
Rosenbrock	32	6,87	8.13	8.05	8.34	0.72	0.27	14.06	6.84
Rosenbrock	50	5.51	7.76	6.94	8.14	0.74	0.25	32.92	23.01
Schwefel	32	22.28	13,76	28.87	31.49	5.76	14.64	46.47	13.74
Schwefel	50	11.03	22.94	18.47	39.29	4.66	19.32	68.10	20.06

Os resultados referidos na tabela 5.3 mostram que o Deep SARSA superou o SARSA na maioria dos valores obtidos das funções. Na função *Rosenbrock*, o Deep SARSA obteve melhores resultados mínimos e médias com 32 e 50 episódios, apesar de exigir mais tempo de execução. Na benchmark *Schwefel*, o Deep-SARSA também se destacou, apresentando resultados significativamente melhores e médias mais baixas, embora com um tempo de execução mais prolongado.

A análise gráfica da figura 5.1 representa a evolução dos parâmetros do GA para a evolução dos melhores resultados da função objetivo usando o método Deep SARSA.

(A) Função Benchmark - Rosenbrock (Epi=32)



(B) Função Benchmark - Schwefel (Epi=32)

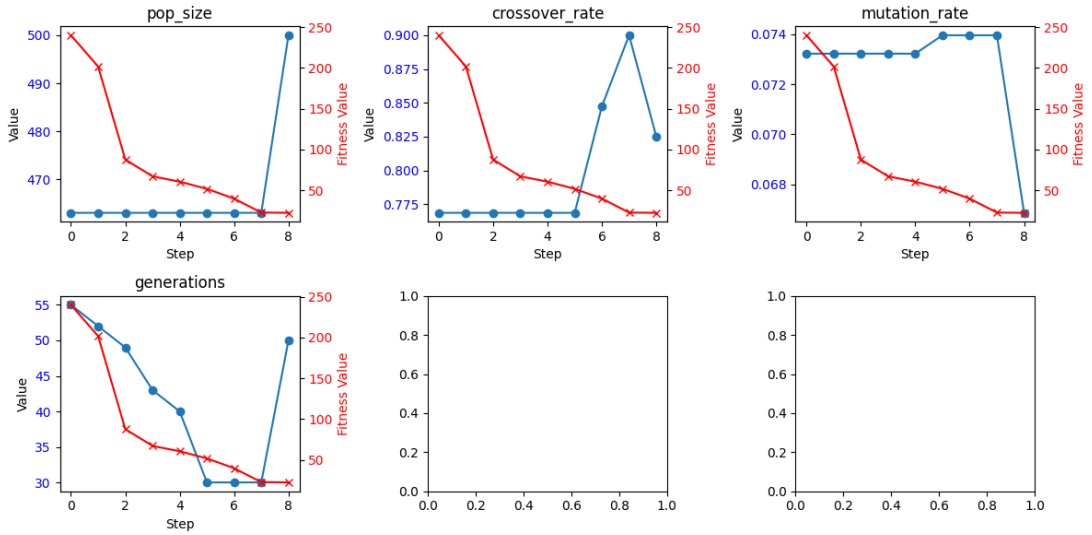
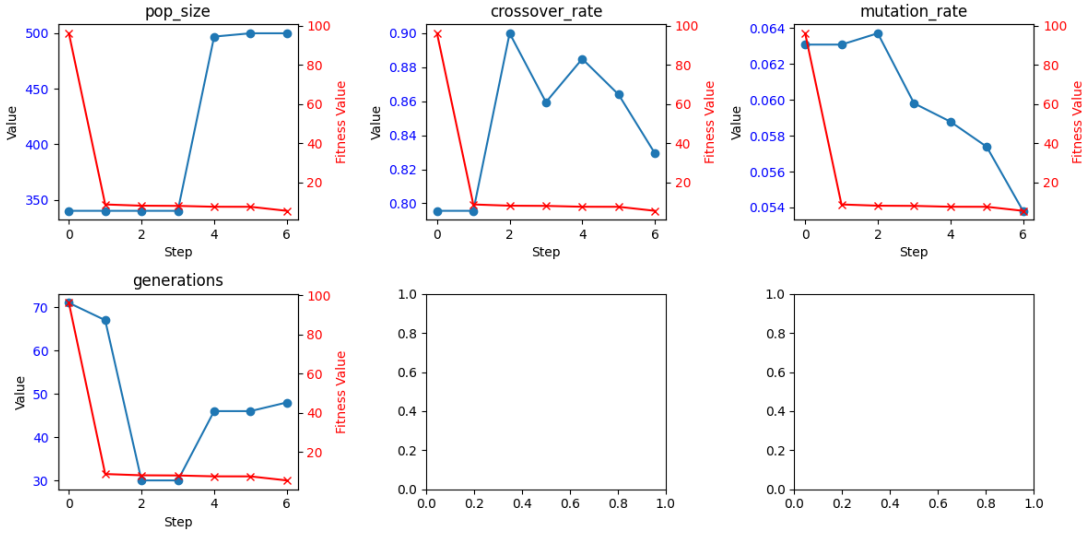


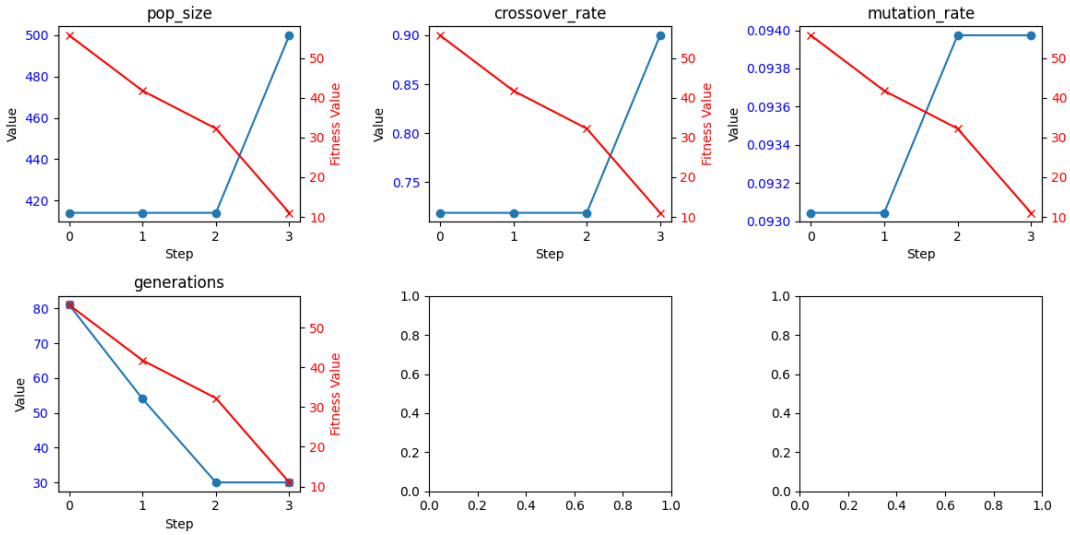
FIGURA 5.1: Evolução dos Parâmetros nas Funções de Benchmark: Rosenbrock e Schwefel com 32 episódios

Na otimização, por minimização, da função *Rosenbrock* com uma execução com 32 episódios, pode verificar-se que os parâmetros, depois de um ajuste inicial, oscilaram entre si, mas mantendo a tendência de estabilização próxima da melhor solução. O melhor desempenho (6.87) na figura 5.1a foi conseguido próximo do fim com o aumento da população para 320 indivíduos. Algo interessante, porque também acontece com a função *Schwefel* que encontrou perto do fim a solução ótima com o aumento claro da população para 500 indivíduos e o aumento do número de gerações para 50. De salientar que as taxas de cruzamento e mutação estabilizaram nos valores entre [0.7-0.9] e [0.07-0.09], facto que é corroborado pela literatura sobre esta matéria. A figura

5.2 mostra a evolução dos parâmetros para 50 episódios nas duas funções anteriores, usando o Deep SARSA.



(A) Função *Benchmark Rosenbrock* (Episódios = 50)



(B) Função *Benchmark Schwefel* (Episódios = 50)

FIGURA 5.2: Evolução dos Parâmetros nas funções *Benchmark*: Rosenbrock e Schwefel em 50 episódios

Com 50 episódios os resultados melhoraram ligeiramente, embora a evolução dos parâmetros se enquadre nos valores anteriores, isto é, com o tamanho de populações e taxas de cruzamento altas, e taxas de mutação e gerações mais reduzidas. Na ultima figura 5.2b, a melhor solução estabilizou com uma taxa de mutação mais alta (0.094) e o número de gerações mais baixo (30).

A tabela 5.4 mostra os resultados finais da melhores configurações obtidas com as funções testadas.

A *letra* da primeira coluna da tabela anterior à designação da função na tabela representa o resultado apresentado no gráfico da figura anterior.

TABELA 5.4: Configuração de parâmetros apurados nas funções *Rosenbrock* e *Schwefel* através do método Deep SARSA

Function	Episode	Pop_Size	Crossover_Rate	Mutation_Rate	Generations
Initial Values		[50-500]	[0,6-0,9]	[0,01-0,1]	[30-90]
(A) Rosenbrock	32	320	0.730	0.081	49
(B) Schwefel	32	500	0.825	0.067	50
(A) Rosenbrock	50	500	0.829	0.054	48
(B) Schwefel	50	500	0.900	0.094	30

Após a análise da abordagem aplicada às funções *benchmark*, a técnica de ajuste de parâmetros por RL, especificamente o método SARSA, foi também utilizada e analisada no contexto de um problema de otimização de portfólio para compra e venda de energia num mercado aberto. Esta aplicação, detalhada na secção seguinte, resultou na publicação de um artigo científico, conforme mencionado anteriormente.

5.2 Otimização de Portfólio em Mercado de Eletricidade

Além das funções anteriores, considerou-se um problema de otimização de portfólio, no mercado de eletricidade que visa maximizar o lucro resultante do investimento de um participante, em ambiente de mercado dinâmico. Cada oportunidade é caracterizada por um preço de transação previsto, que pode variar ao longo do tempo. O desafio reside em definir um modelo de otimização capaz de encontrar uma combinação ótima de volume de negociação (compra e venda) que deve ser realizada de acordo com os preços esperados e as restrições específicas do mercado, exigindo uma abordagem flexível e adaptativa [7].

5.2.1 Definição do Problema

O problema reflete a realidade de mercados de eletricidade liberalizados, onde tanto participantes, como fornecedores ou consumidores, precisam de tomar decisões estratégicas para compra e venda de energia, aproveitando oportunidades em diferentes períodos de negociação. Esse tipo de problema é particularmente desafiador devido à volatilidade dos preços e à necessidade de reagir rapidamente às mudanças nas condições de mercado. Adicionalmente, o problema inclui a incerteza inerente aos mercados de eletricidade, onde as variações de preços podem ser influenciadas por fatores externos, como mudanças na oferta e procura, condições climáticas ou políticas regulamentares. Portanto, o modelo também precisa ser capaz de incluir essas incertezas, ajustando dinamicamente a estratégia de negociação para otimizar os resultados.

Assim, em concreto, um problema de otimização de portfólio no mercado de eletricidade envolve considerar diferentes tipos de informação, descrito em [86], como:

1. **Tipos de Mercado:** Define os mercados nos quais cada cliente pode participar, especificando se o cliente pode atuar como comprador ou vendedor. Os limites de energia dados em (Mw) para compra (*Lim_buy*) e venda (*Lim_Sell*) são também especificados. Por exemplo, a Tabela B.1 apresenta os tipos de mercado e os respetivos limites de energia:
   - Tipos de mercado: Spot, Iday1, Iday2, Iday3, Iday4, Iday5, Iday6, Local.
   - Lim_buy (Mw): Valores variando de 0.01 a 0.17, dependendo do mercado.
   - Lim_Sell (Mw): Valores variando de 0.01 a 0.17, dependendo do mercado.<
2. **Preços esperados:** Contém os preços esperados por megawatt-hora (€/MWh) para cada mercado e período de negociação. A Tabela B.2 fornece uma visão dos preços esperados, permitindo a estimativa dos custos e a valorização do objetivo de minimizar o custo:
   - Exemplo para o período 1: Spot - 175.28, Iday1 - 182.31, Iday2 - 190.22, etc.
   - Exemplo para o período 24: Spot - 153.92, Iday1 - 168.89, Iday2 - 192.65, etc.
3. **Disponibilidade de mercado:** Indica a disponibilidade para compra ou venda em cada mercado e período de negociação. A Tabela B.3 mostra a autorização para intervenção em cada mercado:
   - Exemplo para o período 1: Spot - 1 (ativo), Iday1 - 1 (ativo), Iday2 - 1 (ativo), etc.
   - Exemplo para o período 24: Spot - 1 (ativo), Iday1 - 1 (ativo), Iday2 - 1 (ativo), etc.
4. **Consumo esperado:** Fornece o consumo esperado em cada período de negociação, essencial para garantir que a solução esteja equilibrada. A Tabela B.4 apresenta o consumo esperado em MW:
   - Exemplo para o período 1: 0.469 MW.
   - Exemplo para o período 24: 0.515 MW.

5.2.2 Formulação Matemática

A Equação 5.1 representa a função objetivo, que modela a otimização do portfólio de participação no mercado dos jogadores. Esta função considera a produção esperada de um jogador de mercado para cada período de cada dia, e a quantidade de energia a ser negociada em cada mercado é otimizada para obter a máxima renda que pode ser alcançada [7]. O objetivo é alcançar a menor diferença possível entre os preços de compra e venda nos mercados disponíveis.

$$
\begin{aligned}
&Spow_{(M1\ldots\text{NumM})}, Bpow_{(S1\ldots\text{NumS})} \\
&= \max \left[\begin{array}{l} \sum_{M=M1}^{NumM} \left(Spow_{(M,d,p)} \times ps_{(M,d,p)} \times Asell_M \right) - \\ \sum_{S=S1}^{NumS} \left(Bpow_S \times ps_{(S,d,p)} \times Abuy_S \right) \end{array} \right]
\end{aligned} \tag{5.1}
$$

$$\forall d \in \mathbb{N}_{\text{day}}, \forall p \in \mathbb{N}_{\text{per}}, \text{Asell}_M \in \{0,1\}, \text{Abuy} \in \{0,1\}$$

Na equação (5.1), d representa o dia da semana, $Nday$ representa o número de dias, p representa o período de negociação, $Nper$ representa o número de períodos de negociação, $Asell_M$ e $Abuy_S$ são variáveis booleanas, indicando se o jogador pode entrar nas negociações em cada tipo de mercado, M representa o mercado referido, $NumM$ o número de mercados, S representa uma sessão do mercado de equilíbrio, e $NumS$ representa o número de sessões. As variáveis $ps(M,d,p)$ e $ps(S,d,p)$ representam os preços esperados (previstos) de venda e compra de eletricidade em cada sessão de cada tipo de mercado, em cada período de cada dia. As saídas são $Spow_M$ representando a quantidade de energia a ser vendida no mercado M, e $Bpow_S$ representando a quantidade de energia a ser comprada na sessão S.

Na equação (5.2) é expressa a forma como os preços de negociação são obtidos, designadamente os preços de venda $ps(M,d,p)$ e os preços de compra $ps(S,d,p)$.

$$\begin{aligned} ps_{M,d,p} &= Value(d,p,Spow_M,M) \\ ps_{S,d,p} &= Value(d,p,Spow_S,S) \end{aligned} \tag{5.2}$$

$$\text{Value(day, per, Pow, Market)} = \text{Data(fuzzy(pow), day, per, Market)} \tag{5.3}$$

O $Value$ é obtido pela equação (5.3), e é calculado a partir da aplicação dos métodos de clustering e fuzzy. Com a implementação deste modelo, é possível obter os preços de mercado com base na quantidade negociada. Para isso, a modelagem dos preços considera a produção esperada de um jogador de mercado para cada período de cada dia. Os resultados da aplicação desta metodologia podem ser observados em [87]. A equação (5.3) define esta condição, onde $Data$ se refere aos dados históricos que correlacionam a quantidade de energia transacionada, o dia, o período do dia e a sessão de mercado particular. A equação (5.4) representa a principal restrição deste problema. A restrição impõe que a quantidade total de energia que pode ser vendida no conjunto de todos os mercados nunca seja superior à produção total esperada (TEP) do jogador, mais o total de energia comprada.

$$\sum_{M=M1}^{NumM} Spow_M \leq TEP + \sum_{S=S1}^{NumS} Bpow_S \tag{5.4}$$

As equações (5.5) a (5.7) introduzem restrições adicionais personalizadas para as nuances do problema, incluindo tipos de mercado, quantidades de negociação e tipos de jogadores suportados (por exemplo, geração baseada em fontes renováveis, cogeração). Os produtores de energia renovável (*Renew_prod*) estão sujeitos a uma restrição de capacidade máxima de produção. Por outro lado, para os produtores termolétricos, a produção (*Therm_prod*) deve atender a um limite mínimo devido a restrições operacionais. Se o jogador for um produtor de energia termoelétrica, a produção deve ser estabelecida num mínimo, pois não é viável desligar completamente a planta de produção, como pode ser observado pela equação (5.7). Se o produtor for baseado em energia

renovável, a única restrição é a capacidade máxima de produção, conforme indicado na equação (5.6).

$$TEP = \sum Energy_{prod}, \text{Energy}_{prod} \in Renew_{prod}, Therm_{prod}\} \tag{5.5}$$

$$0 \leq Renew_{prod} \leq Max_{prod} \tag{5.6}$$

$$Min_{prod} \leq Therm_{prod} \leq Max_{prod}, \text{if} Therm_{prod} > 0 \tag{5.7}$$

5.2.3 Resultados

Para avaliar as abordagens de reforço no problema de otimização do portfólio de investimentos, foram utilizados três métodos: o método manual (seleção aleatória de parâmetros), o SARSA e o SMAC. Este último foi incluído na comparação de resultados dada a relevância de se tratar de um modelo *offline*, num contexto de métodos *online* e também, por ter sido anteriormente analisado num ambiente de configuração automática de algoritmos que resultou num artigo publicado sobre este mesmo problema de otimização [86].

O SMAC (*Sequential Model-based Algorithm Configuration*) é uma ferramenta versátil para otimização de parâmetros de algoritmos, incluindo a otimização de hiperparâmetros em algoritmos de ML. A ideia central da otimização *Bayesian*, nesta abordagem, é construir um modelo probabilístico da função objetivo. A finalidade deste modelo é dar-nos uma estimativa da função objetivo, que no início tem um grau de incerteza grande. A cada observação da função, aprende-se mais sobre o próprio modelo, tornando-o mais apto. Na sua formulação, o SMAC utiliza quatro componentes principais para executar um processo de otimização: *configuration space*, *target function*, *scenario* e *facade*.

- **Configuration space** - É definido como espaço de pesquisa dos hiperparâmetros, ou seja, os intervalos e valores dos parâmetros ajustáveis (p.e. pop_size, crossover_rate, mutation_rate, generations)
- **Target Function** - Esta função tendo como entrada uma configuração da *Configuration Space*, retorna um valor, neste caso, o desempenho do problema de portfólio de otimização usando o GA
- **Scenario** - É usado para fornecer variáveis de ambiente, como um limite de tempo ou de número de evoluções, para o processo de otimização.
- **Facade** - É o ponto de entrada para o SMAC, que constrói um pipeline de otimização padrão. Foi definida a *BBFacade*, sendo considerada adequada para a escolha da melhor configuração de híper-parâmetros para um processo gaussiano de máxima probabilidade encontrada.

A tabela 5.5 apresenta os resultados obtidos através da heurística do GA para todos os métodos, tendo sempre como requisitos, o conjunto de parâmetros e os seus intervalos pré-definidos. Com esses dados, é possível comparar os métodos em termos de qualidade das soluções encontradas e do tempo necessário para as suas execuções.

TABELA 5.5: Resultados da otimização do GA através dos métodos manual SARSA e Deep SARSA no problema de Portfólio para 32 episódios

Method	Best	Mean All Best	Standard Deviation	Mean Time
MANUAL	78.43	79.21	0.28	16.81
SMAC	78.33	78.97	0.43	42.63
SARSA	78.16	78.66	0.55	18.83

Com base nos resultados apresentados na tabela 5.5, o método Manual, indica que, ao longo de 32 episódios (ou iterações), a melhor solução encontrada teve um valor de 78,434, com uma média de 79,211 e um desvio padrão de 0,281. O tempo médio de execução para cada episódio foi de aproximadamente 16,809s. Por outro lado, o método SARSA encontrou uma solução muito próxima, com um valor de 78,157, enquanto a média foi de 78,661 e o desvio padrão de 0,554. O tempo médio de execução para cada episódio foi de cerca de 18,831 s. Por sua vez, o método SMAC, alcançou resultados intermédios entre as abordagens anteriores.

Em conclusão, o método SARSA apresentou um desempenho ligeiramente superior em termos de minimização dos valores da função objetivo. O SARSA conseguiu, em média, encontrar soluções mais próximas do ótimo (78.66), embora tenha mostrado maior variabilidade nos resultados e um custo de tempo médio razoável.

TABELA 5.6: Conjunto de parâmetros apurados para o algoritmo GA no problema de Portfólio através do método SARSA

	Pop_Size	Crossover_Rate	Mutation_Rate	Generations
Initial Values	[50-500]	[0,6-0,9]	[0,01-0,1]	[30-90]
Portfolio Optimization	448	0,789	0,069	36

O melhor desempenho obtido de 78.16 resultou na configuração final dos parâmetros do GA apresentada na tabela 5.6, dentro dos intervalos iniciais pré-definidos. Após a otimização, os valores ajustados foram os seguintes: População (Pop_Size) de 448, Taxa de Crossover (Crossover_Rate) de 0,789, Taxa de Mutação (Mutation_Rate) de 0,069 e Número de Gerações (Generarions) de 36.

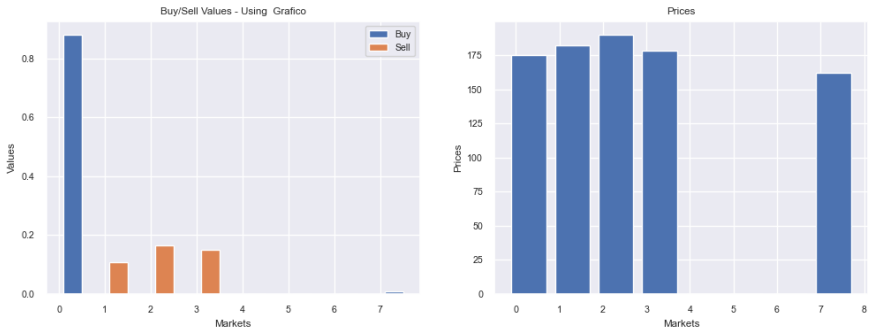


FIGURA 5.3: (a) Volume de Compra e Venda de (Mw) em cada Mercado; (b) Preço de eletricidade em cada Mercado

Para fins ilustrativos, a figura 5.3 exibe os resultados da otimização do portfólio para um período-alvo (período 1). Na figura em (a) é mencionado o volume (MW) alocado para compra e venda em cada mercado disponível para o período considerado, enquanto em (b) são apresentados os preços de mercado esperados para o período em questão, em todos os mercados considerados.

A adaptabilidade do ajuste de parâmetros de heurísticas por meio de RL foi também investigada acrescentando o método Deep SARSA aplicado não só ao GA mas também a outras heurísticas, especificamente, o PSO. Com este estudo pretende-se também avaliar a relevância do ajuste dinâmico de parâmetros, comparando diferentes algoritmos meta-heurísticos de otimização.

No caso específico do PSO foram realizados ajustes aos parâmetros como os limites, mínimos e máximos, de inércia (w), os coeficientes de aceleração cognitiva ($c1$) e social ($c2$), além do número de partículas e iterações, conforme os intervalos estabelecidos na tabela 4.4 do capítulo anterior.

Os resultados da otimização de portfólio, indicados na tabela 5.7, mostram uma análise comparativa entre os métodos de ajuste de parâmetros manual, SARSA e Deep SARSA aplicados aos algoritmos meta-heurísticos, GA e PSO. Para um conjunto de 5 execuções, cada método foi testado ao longo de 32 e 50 episódios, e os resultados apresentados em termos de melhor solução (menor valor), média, desvio padrão e tempo médios por execução.

TABELA 5.7: Resultados da otimização do GA e PSO através dos métodos manual SARSA e Deep SARSA no problema de Portfólio para 5 execuções

Heuristic	Method	Episodes	Best	Mean All Best	Standard Deviation	Mean Time
GA	Manual	32	78,36	78,51	0,104	17,307
GA	SARSA	32	78,28	78,47	0,144	20,213
GA	D-SARSA	32	78,11	78,31	0,153	49,474
GA	D-SARSA	50	78,03	78,14	0,078	127,741
PSO	Manual	32	78,06	78,15	0,090	10,135
PSO	Manual	50	78,07	78,16	0,084	15,207
PSO	D-SARSA	32	78,01	78,09	0,044	14,985
PSO	D-SARSA	50	78,00	78,07	0,053	34,468

Na análise de resultados da tabela 5.7, observa-se que o PSO ajustado com Deep SARSA obteve os melhores resultados em termos de mínimo e média dos melhores valores, além de apresentar a menor variabilidade. Esta solução revelou menor valor mínimo de 78,01 e uma média de 78,09 para 32 episódios, e 78,07 para 50 episódios, respetivamente. Além disso, o desvio padrão foi significativamente baixo, com 0,044 para 32 episódios e 0,053 para 50 episódios, indicando uma alta estabilidade nas soluções.

Ao comparar as heurísticas, o PSO ajustado com Deep SARSA obteve o melhor desempenho geral na otimização por minimização, com menores valores e menor variabilidade em comparação com o GA, mesmo com ajustes avançados como o Deep SARSA.

O ajuste Manual geralmente oferece um tempo de execução menor, no entanto, o Deep SARSA aplicado ao PSO, embora leve mais tempo, proporciona soluções de maior qualidade, o que pode justificar o custo computacional adicional.

No caso da otimização de portfólio, os resultados mostram diferenças claras entre os métodos Manual, SARSA e Deep SARSA, aplicados aos algoritmos GA e PSO. Os dados destacaram a superioridade do Deep SARSA em termos de estabilidade e eficiência, oferecendo flexibilidade significativa e aperfeiçoando o desempenho das heurísticas, PSO e GA, o que demonstra grande potencial para aplicações futuras na otimização adaptativa.

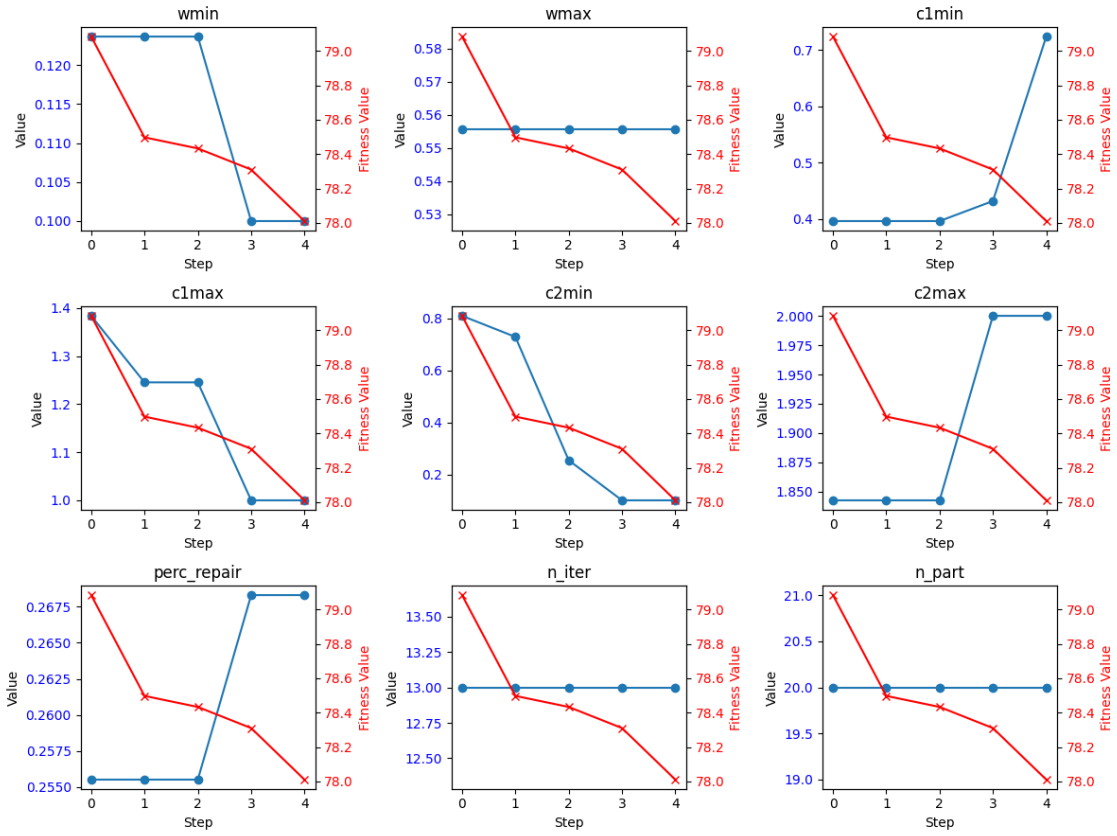


FIGURA 5.4: Otimização do algoritmo PSO através do método Deep SARSA no problema de Portfólio (Episódios = 32)

A imagem 5.4 mostra a evolução de diferentes parâmetros no ajuste do PSO utilizando o Deep SARSA para resolver o problema de otimização de portfólio ao longo de 32 episódios. A análise dos gráficos pode ser dividida em dois eixos: a evolução dos parâmetros do PSO (linhas azuis) e o valor de *fitness* obtido (linhas vermelhas).

Da evolução dos parâmetros ao longo das iterações, verifica-se que a diminuição de $c1$ significa um menor peso dado à experiência individual das partículas do PSO. O comportamento dos coeficientes sociais $c2$ indica que, à medida que o valor aumenta, a *fitness* melhora, especialmente no caso de $c2max$. Esta verificação sugere que a este problema de otimização, é mais vantajoso atribuir uma importância maior à colaboração entre as partículas (componentes sociais), permitindo um comportamento coletivo mais apto.

Parâmetros como w e $c2$ têm um impacto importante na convergência do PSO, enquanto parâmetros como n_iter e n_part permanecem relativamente estáveis após um certo ponto. O ajuste dinâmico mostra como os parâmetros do PSO se adaptam ao problema de otimização ao longo dos episódios, melhorando gradualmente a qualidade das soluções encontradas.

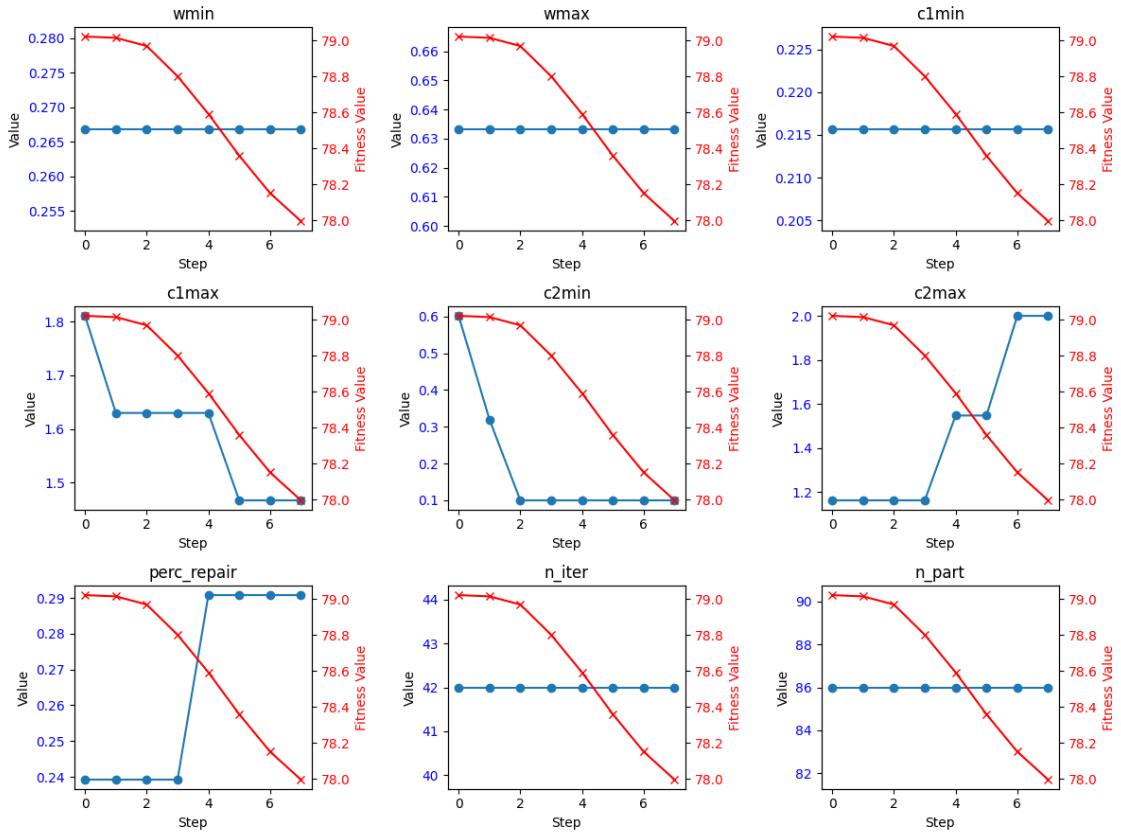


FIGURA 5.5: Otimização do algoritmo PSO através do método Deep SARSA no problema de Portfólio (Episódios = 50)

Na função de uma população de partículas diferente e de um número de episódios superior, na figura 5.5 pode observar-se mais uma vez que os parâmetros, $c1$ e $c2$ têm um impacto significativo

na função objetivo. A queda de um e subida de outro, respetivamente, afetam a performance do algoritmo. De maneira similar, $perc_repair$ (percentagem de reparação) também oferece uma correlação forte com a queda da função objetivo,

Conclui-se que a combinação de uma heurística adequada com um ajuste preciso de parâmetros é fundamental para otimizar a solução de problemas complexos. A seleção da heurística pode ser baseada na natureza específica do problema, enquanto o ajuste de parâmetros visa maximizar o desempenho da heurística escolhida. A prática de testar diferentes heurísticas e ajustar os seus parâmetros conforme necessário garante uma abordagem mais robusta e capaz para encontrar soluções ótimas.

Finalmente, estes resultados indicam que o Deep SARSA é mais eficaz na obtenção de soluções de alta qualidade e oferece maior consistência em diferentes cenários de otimização. Dado o seu desempenho superior, o Deep SARSA é recomendado para casos de estudo mais complexos, como a alocação de pacientes e a otimização de recursos, devido à sua capacidade de alcançar soluções melhores e mais consistentes.

5.3 Alocação de Doentes a Salas de Cirurgias

O planeamento competente de cirurgias em hospitais é um problema crítico devido à alta procura e aos recursos limitados. A manutenção das salas de cirurgia é dispendiosa, tornando essencial otimizar a sua utilização por forma a minimizar o valor da função objetivo e maximizar a eficiência. Um dos principais desafios é a alocação de cirurgias elegíveis, que precisam ser programadas dentro de um horizonte de planeamento, considerando restrições como prazos, duração das cirurgias e disponibilidade das salas.

A aplicação de algoritmos heurísticos é amplamente reconhecida na literatura como uma abordagem capaz para lidar com problemas complexos de otimização, entre os quais a gestão de recursos hospitalares e a alocação de pacientes. Estes métodos oferecem soluções práticas e funcionais, principalmente em situações em que os métodos exatos são impraticáveis devido à complexidade e à escala dos problemas.

Vários autores destacam a importância da otimização por algoritmos heurísticos e meta-heurísticos. Por exemplo, A. Rais et al. aprofundam a discussão deste algoritmos, nomeadamente, o GA e o *Simulated Annealing* (SA, Recozimento Simulado) no setor hospitalar, incluindo a alocação de pacientes e a gestão de recursos hospitalares [88].

Entre essas técnicas, os GAs têm-se mostrado promissores devido à sua capacidade de encontrar soluções próximas do ótimo. Inspirados pelos processos de seleção natural, evolução e hereditariedade, os GAs são amplamente utilizados em problemas de agendamento de máquinas paralelas, que compartilham semelhanças com o agendamento de salas de cirurgia.

O caso de estudo aqui abordado, sobre a alocação de doentes a salas de cirurgia, foi adaptado do trabalho de Y.K. Lin et al., cujo artigo *"A Hybrid Genetic Algorithm for Operating Room Scheduling"* serve de base a seguir, para a definição do problema e sua análise [89].

5.3.1 Definição do Problema

O problema em questão envolve a alocação de um conjunto de cirurgias N em salas cirúrgicas M_d dentro de um horizonte de planeamento H com o objetivo de minimizar o custo operacional total. Cada cirurgia i tem uma duração operacional t_i e um prazo D_i.

Uma vez iniciadas, as cirurgias, devem ser concluídas sem interrupção. Cada sala de cirurgia k num dia d tem um horário de abertura regular RT_{kd} e um máximo de horas extras permitidas OT_{kd}.

Definição do Individuo - O individuo ou cromossoma na representação genética para este problema pode ser definido como uma sequência de genes onde cada gene representa um paciente e a sua alocação a uma cirurgia, sala e dia específicos. A estrutura do individuo (cromossoma) é então uma lista que mapeia cada doente para a cirurgia, a sala e o dia dentro do horizonte de planeamento. Especificamente, o cromossoma pode ser representado como:

Sequência de Alocação - Cada posição no cromossoma representa uma cirurgia e contém informação relacionada com a cirurgia, sala e dia, indicando qual, onde e quando a cirurgia deve ser realizada. Para 24 pacientes pode ser representado da seguinte forma:

$$[2, 6, 4, 13, 16, 5, 14, 7, 12, 18, 15, 17, 1, 22, 11, 21, 9, 24, 8, 3, 10, 23, 19, 20]$$

Pressupostos e Restrições:

- Número de Dias no Horizonte de Planeamento: $H = 3$
- Penalidade para horas extras: $\alpha = 1.5$
- Penalidade por Alocação no Dia Seguinte: É permitido que uma cirurgia seja alocada se não houver espaço suficiente no dia corrente, respeitando as horas extras permitidas. Se não puder ser alocada, a penalidade $P1 = 500$ será aplicada para cirurgias não alocadas.
- Penalidade por Tempo Excedido: Se o tempo total ocupado por dia excede o tempo disponível mais as horas extras permitidas OC_{kd}. Se exceder, a penalidade é aplicada para esse dia $P2 = 500$
- Informações das cirurgias: Cada cirurgia possui um identificador, uma duração específica e um prazo. As informações são fornecidas numa lista contendo 24 cirurgias. Os dados do paciente e cirurgia são representados por uma estrutura de dados (dicionário) da seguinte forma:

$$'id' : 15, 's\_id' : 3, 'duration' : 90, 'deadline' : 3$$

- Tempo de Abertura Regular e Disponibilidade das Salas: A disponibilidade das salas em minutos para cada dia é fornecida um dicionário, com diferentes salas disponíveis em diferentes dias. Pode ser representado da seguinte forma para a Sala 1 disponível por 8h no Dia 1, 8h no Dia 2, 0h no Dia 3

$$1 : [480, 480, 0]$$

- Máximo de Horas Extras : O limite máximo de horas extras permitidas por sala é de 120 minutos (2 horas).
- Parâmetros do Algoritmo Genético: O vetor de parâmetros do GA está definido entre os limites estabelecidos

5.3.2 Formulação Matemática

O custo operacional da sala da k no dia d é calculado considerando dois componentes:

1. **Custo de Tempo Não Utilizado (UC_{kd})** : Se a soma dos tempos de operação das cirurgias agendadas f_{kd} na sala k no dia d, for menor do que o tempo de abertura regular RT_{kd}, ocorre um custo desperdiçado pelo tempo não utilizado $UC_{kd} = RT_{kd} - f_{kd}$
2. **Custo Operacional de Horas Extras (OC_{kd})** : Se a soma dos tempos de operação for maior do que RT_{kd}, ocorre um custo operacional de horas extras $OC_{kd} = f_{kd} - RT_{kd}$.
3. **Custo da Sala (C_{kd})** : O custo operacional da sala k no dia d é $C_{kd} = UC_{kd} + \alpha OC_{kd}$, onde α é o custo de penalidade das horas extras.

O objetivo é minimizar o custo operacional total ao longo do horizonte de planeamento:

$$\min \sum_{d=1}^{H} \sum_{k=1}^{M_d} C_{kd}. \tag{5.8}$$

A função objetivo do problema, derivada da equação 5.8 reflete os custos operacionais totais associados à alocação das cirurgias. A função pode ser expressa como:

$$\text{Custo Total} = \sum_{d=1}^{H} \sum_{k=1}^{M_d} [(RT_{kd} - f_{kd}) + \alpha \cdot \max(0, f_{kd} - RT_{kd})\text{P1} + \text{P2}] \tag{5.9}$$

A abordagem proposta utiliza os algoritmos populacionais meta-heurísticos (GA e PSO) para explorar possíveis alocações de cirurgias e minimizar o custo operacional total. Cada individuo representa uma alocação específica de cirurgias, enquanto a função objetivo avalia a qualidade desta alocação em termos de custos operacionais. O valor da penalidade para horas extras α é crucial para refletir a importância relativa das horas extras em comparação com o tempo não utilizado.

A eficiência do algoritmo depende da representação do indivíduo, da função objetivo e da qualidade dos parâmetros. A avaliação e ajuste desses parâmetros são essenciais para melhorar a performance do algoritmo, além de validar a abordagem em diferentes instâncias.

5.3.3 Resultados

Para analise dos resultados foi considerado o cenário de salas e horas disponíveis, com 2 horas por dia de disponibilidade de horas extras, apresentado na tabela 5.8

Tabela 5.8: Disponibilidade das salas durante 3 dias

Dia	**Sala 1**	**Sala 2**	**Sala 3**
1	480	480	240
2	240	0	0
3	0	240	480

O apêndice B.2 contem informação detalhada sobre os dados que serviram esta experiência, nomeadamente a lista de pacientes contendo as cirurgias a efetuar. Para além disso, foram considerados os valores de $\alpha = 1.5$ e $OC_{kd} = 60min$.

Os resultados são apresentados na tabela 5.9 em termos da melhor solução, média das melhores soluções, desvio-padrão médio e tempo médio de execução por execução. A otimização foi testada utilizando técnicas manuais e os algoritmos GA e PSO ao longo de 32 e 50 episódios para 5 execuções.

Tabela 5.9: Resultados da otimização do GA e PSO através dos métodos manual e Deep SARSA no problema de unidades de cirurgias para 5 execuções

Heuristic	Method	Episodes	Best	Mean All Best	Standard Deviation	Mean Time(s)
GA	Manual	32	102,5	114,5	7,31	22,08
GA	D-SARSA	32	52,5	87,0	28,91	78,31
GA	D-SARSA	50	52,5	77,5	25,05	111,46
PSO	Manual	32	80,0	111,5	16,32	43,91
PSO	D-SARSA	32	57,5	74,0	21,13	86,20
PSO	D-SARSA	50	72,5	100,0	18,03	60,31

Os resultados obtidos na otimização do agendamento de cirurgias mostram que o uso do método Deep SARSA no ajuste de parâmetros ofereceu ganhos interessantes em comparação com os métodos manuais, tanto para o GA como no PSO. A média dos melhores resultados indicou, para 32 episódios, que o Deep SARSA com o valor de 87,0 foi superior ao o ajuste pelo processo manual (114,5). O mesmo aconteceu com o PSO ajustado com o Deep SARSA, em que a performance foi superior, com um custo mínimo de 72,5 e uma média de 100,0, em comparação com o PSO ajustado manualmente, que teve um custo mínimo de 80,0 e uma média de 111,5.

Os resultados demonstram que o ajuste dinâmico de parâmetros usando Deep SARSA proporcionou uma melhoria significativa na qualidade das soluções para o problema de agendamento de cirurgias, tanto para GA quanto para PSO. Embora o tempo de execução tenha aumentado com o ajuste dinâmico, a redução nos custos e na variabilidade das soluções sugere que o Deep SARSA é eficaz na obtenção de melhores resultados em problemas complexos de otimização.

TABELA 5.10: Conjunto de parâmetros apurados para o algoritmo GA no problema de Unidades de Cirurgia através do método Deep SARSA

	Pop_Size	Crossover_Rate	Mutation_Rate	Generations
Initial Values	[50-500]	[0,6-0,9]	[0,01-0,1]	[30-90]
Surgeries Problem	380	0,8916	0,0969	30

A **Sequência de Alocação** para a melhor solução obtida contém informação relacionada com a cirurgia, sala e dia, indicando qual, onde e quando a cirurgia deve ser realizada. Para 24 pacientes o resultado foi a seguinte:

$$[4, 20, 30, 1, 29, 16, 27, 9, 11, 18, 23, 19, 25, 5, 8, 2, 14, 7, 3, 28, 21, 17, 26, 15]$$

Para fins ilustrativos a figura 5.6 representa a alocação da solução final sob o aspeto gráfico. Esta figura mostra parte da distribuição de pacientes pelas diferentes salas de cirurgia e dias, considerando a duração das cirurgias e o tempo disponível em cada sala. Algumas salas excedem o tempo disponível, embora dentro das horas extras previstas, enquanto outras utilizam exatamente o tempo permitido. A imagem completa pode ser observada em apêndice B.1.

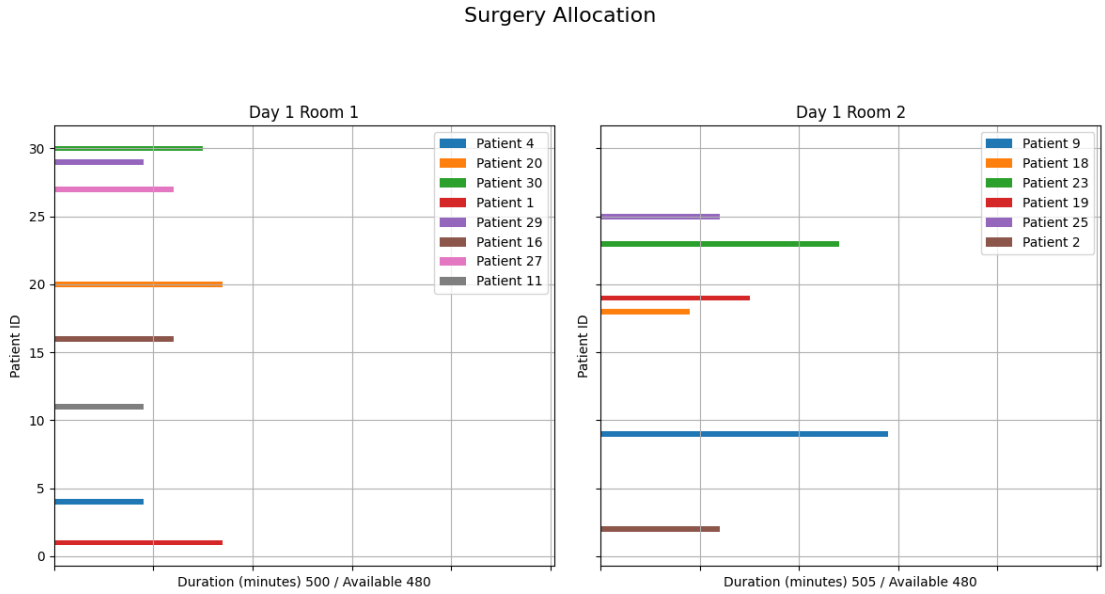


FIGURA 5.6: Cronograma de planeamento de pacientes por salas de cirurgia

A tabela 5.11 ilustra com maior detalhe a alocação de cirurgias em três salas de cirurgia ao longo de três dias, destacando a distribuição dos pacientes e a duração das cirurgias em relação ao tempo disponível. No Dia 1, as três salas estão ocupadas, com a Sala 1 e a Sala 2 ultrapassando ligeiramente os 480 minutos disponíveis, alocando 500 e 505 minutos, respetivamente. A Sala 1 alocou pacientes como os Pacientes 4, 20, 30, 1, 29, 16, 27 e 11, enquanto a Sala 2 incluiu os pacientes 9, 18, 23, 19, 25 e 2. Já a Sala 3, dentro do limite de 240 minutos, acomodou as cirurgias dos Pacientes 5, 14 e 7.

No Dia 2, apenas a Sala 1 foi utilizada, alocando o Paciente 8 por 240 minutos, enquanto as Salas 2 e 3 permaneceram vazias. No Dia 3, a Sala 1 não teve cirurgias programadas, e a Sala 2 alocou o Paciente 3 por 240 minutos. A Sala 3 acomodou vários pacientes, utilizando 450 dos 480 minutos disponíveis.

TABELA 5.11: Distribuição dos pacientes por salas e por dias

Dia	**ID**	**ID / Tempos**								**Total**
D1	ID	4	20	30	1	29	16	27	11	
	Sala 1	45m	85m	75m	85m	45m	60m	60m	45m	500m
D1	ID	9	18	23	19	25	2	-	-	
	Sala 2	145m	45m	120m	75m	60m	60m	-	-	505m
D1	ID	5	14	7	-	-	-	-	-	
	Sala 3	120m	60m	60m	-	-	-	-	-	240m
D2	ID	8	-	-	-	-	-	-	-	
	Sala 1	240m	-	-	-	-	-	-	-	240m
D2	ID	-	-	-	-	-	-	-	-	
	Sala 2	-	-	-	-	-	-	-	-	0
D2	ID	-	-	-	-	-	-	-	-	
	Sala 3	-	-	-	-	-	-	-	-	0
D3	ID	-	-	-	-	-	-	-	-	
	Sala 1	-	-	-	-	-	-	-	-	0
D2	ID	8	-	-	-	-	-	-	-	
	Sala 2	240m	-	-	-	-	-	-	-	240m
D3	ID	28	21	17	26	15	-	-	-	
	Sala 3	90m	90m	90m	90m	90m	-	-	-	450m

5.4 Alocação de Pacientes em Unidades de Cuidados Intensivos

O aumento constante do número e da complexidade das cirurgias em todo o mundo requer medidas preventivas capazes e cuidados pós-operatórios adequados para prevenir complicações e disfunções orgânicas. O equilíbrio, entre a gravidade do estado de saúde do paciente, disponibilidade de recursos (como camas e pessoal de enfermagem) e as políticas hospitalares, torna-se uma tarefa complexa. Importa, por isso, racionalizar as solicitações de camas em UCI, reduzir o tempo de

espera dos pacientes e minimizar os custos financeiros associados aos cuidados pós-operatórios. O objetivo principal é garantir que cada paciente receba o nível de cuidado apropriado enquanto se maximiza a qualidade do uso dos recursos hospitalares.

Ao ajustarem os parâmetros dinamicamente, evitam-se decisões abruptas e ineficazes no fluxo de pacientes. Isso é especialmente importante nesta área, onde a estabilidade das operações é crucial para garantir a segurança e o bem-estar dos pacientes. Ao aplicar essas técnicas, os hospitais podem não só melhorar a alocação de recursos em tempo real, mas também adaptar-se melhor a cenários imprevistos, como o aumento repentino da procura por camas em UCI.

5.4.1 Definição do Problema

A utilização de dados provenientes do período pré-cirúrgico, intra-operatório e pós-operatório imediato, combinada com a aplicação de *scores* de risco, revela-se uma abordagem possível para a otimização da alocação dos pacientes em unidades de cuidados intensivos (UCI). Eveline HJ Mestrom et al. [90] destacam a importância de integrar diferentes parâmetros clínicos e combinação destes fatores para melhorar a capacidade de internamento. Esta interligação de dados e de pessoas revela diversas fontes de incerteza. Os pacientes podem ser admitidos diretamente na UCI, chegar após cirurgias programadas, ou serem transferidos do Unidade de urgência, com ou sem passagem pela sala de cirurgia.

A UCI interage com outros departamentos cruciais como a sala de recuperação pós-anestésico (SRPA), unidade de urgência, o centro cirúrgico, e as unidades a jusante, como a unidade de cuidados intermediários e as enfermarias gerais. A coordenação competente entre os referidos departamentos é essencial para otimizar o fluxo de pacientes e garantir a qualidade do atendimento em toda a cadeia hospitalar.

A este propósito, um dos tópicos importantes para o fluxo de doentes nestes departamentos envolve a alocação eficiente de camas para atender às necessidades de cuidados intensivos dos pacientes, a par com vários outros aspetos relevantes neste problema, como: a avaliação de doentes (*score*) ou a programação de recursos materiais e pessoais.

Neste caso de estudo, procurou aplicar-se mais uma vez a DAC aos parâmetros do GA num contexto de alocação de camas, sob incertezas quanto ao número de pacientes elegíveis, de urgência ou já aceites em UCI e o tempo previsto de internamento. O artigo de Fang Wan [91] serviu de inspiração para definir uma solução para este problema com o intuito de minimizar o número de camas de UCI necessárias e maximizar a utilização de recursos, garantindo a admissão do número máximo de pacientes.

Definição do Individuo - Os pacientes que compõe o individuo (cromossoma) possuem características relevantes para a solução do problema, sendo cada um composto por 12 genes, selecionados aleatoriamente. O cromossoma ou individuo pode ter a seguinte composição:

$$[1, 2, 11, 5, 7, 15, 12, 10, 16, 14, 18, 17]$$

Em que os genes podem ser explicados da seguinte forma:

- Cada elemento da lista representa *ID* do paciente (ex: 1, 2, 11, 5);
- A ordem dos *IDs* na lista indica a sequência na qual os pacientes serão alocados nas camas da UCI ao longo dos dias;
- Valor nulo 0 representa camas vazia ou seja sem qualquer alocação.

O método utilizado para gerar o cromossoma começa por inicializar os genes com o valor 0, representando este camas vazias. Em seguida, os genes são preenchidos com os pacientes já internados, cujos dias de admissão são negativos, e que, por isso, ocupam obrigatoriamente as primeiras posições do cromossoma. Estes pacientes são considerados parte da população inicial e são alocados automaticamente. A distribuição dos *IDs* dos futuros pacientes é realizada em camas disponíveis, respeitando uma ordem de chegada ou prioridade baseada em parâmetros clínicos, como o *score* de urgência ASA (American Society of Anesthesiologists) e o tipo de cirurgia realizada. A sequência de *IDs* no cromossoma é crucial para manter a conformidade com as restrições clínicas, como a manutenção dos pacientes na mesma cama e a correta alocação dos pacientes prioritários nos dias estabelecidos.

Os dados do paciente estão representados de forma simplificada na tabela 5.12:

Tabela 5.12: Lista pacientes indicados para UCI

ID	**Score**	**TYPE**	**SA**	**LOS**
1	3	Major	-2	3
...	...	...	....	....
18	1	Minor	0	0

A presente tabela resume os dados relevantes dos pacientes que necessitam de cuidados na UCI, incluindo *ID*, nível de urgência (*ASA*), tipo de intervenção (TYPE), status de admissão (*SA*) e os dias estimados na UCI (*LOS*). A estrutura do paciente foi adaptada para incluir informações ou pressupostos que suportam a dinâmica de entrada e saída na UCI:

- **ID** : Identificador único do paciente
- **Score** : *Score* de Nível de Urgência (*ASA* - American Society of Anesthesiologists)
- **Tipo** : Classificação do tipo cirurgia (Major/Minor), o que pode influenciar a prioridade na alocação.
- **Status de Admissão** :
  - **< 0** Indica há quantos dias o paciente foi admitido na UCI.
  - **0** Indica que a entrada é imediata na UCI.
  - **> 0** Indica em quantos dias o paciente será admitido na UCI.

  - A soma entre o *LOS* e o *SA* indica quando o paciente terá alta. Por exemplo, se *SA* = -2 e *LOS* = 3, a alta será no dia seguinte (*SA* + *LOS* = 1).

- **LOS** (*Length of Stay*) : Dias estimados que o paciente permanecerá na UCI.

5.4.2 Formulação Matemática

Para cumprir o objetivo em UCI de minimizar o número de camas em doentes não prioritários, maximizar a utilização de recursos e garantir a admissão do número máximo de pacientes, importa ao concretizar-se a função objetivo minimizar uma serie de penalidades associadas à alocação das camas. Os seguintes aspetos foram tidos em consideração:

- **Ocupação** - Penaliza a ocupação inadequada ou a subutilização de camas
- **Tempo Espera** - Penaliza pacientes que esperam mais que o previsto para serem alocados a camas
- **Prioridade** - Penaliza a alocação tardia de pacientes com *score* / prioridade elevada
- **Sem Alocação** - Penalização por não alocação de doentes prioritários

Tanto a estrutura do individuo como a função objetivo consideram, não apenas a situação atual dos camas, mas também as futuras admissões e saídas, otimizando a utilização dos recursos ao longo de um o período planeado. Assim sendo, a função objetivo de minimização pode ser expressa como:

$$\text{Minimizar } Z = \sum_{d=1}^{days} \sum_{l=1}^{beds} \left(P_{\text{pr}} \times f_{\text{pr}} + P_{\text{es}} \times f_{\text{es}} + P_{\text{sa}} \times f_{\text{sea}} + P_{\text{oc}} \times f_{\text{oc}} \right) \tag{5.10}$$

Onde:

- Z é o valor total da função objetivo.
- d representa os dias do planeamento (1 a N).
- l representa as camas disponíveis (1 a N).
- P_{pr}, P_{es}, P_{sa}, P_{oc} são os pesos atribuídos a cada critério (prioridade, tempo de espera, sem_alocação, ocupação).
- f_{pr}, f_{es}, f_{sa}, f_{oc} são as funções de custo ou penalidade associadas a cada critério.

A função objetivo foi desenhada para maximizar o atendimento dos pacientes com maior gravidade (*score* ASA mais alto), enquanto minimiza a subutilização de camas e o tempo de espera para os pacientes elegíveis. Por isso, foram aplicadas penalidades quando pacientes com alta prioridade não conseguem uma alocação rápida, ou quando há ociosidade excessiva das camas disponíveis. O quadro 5.13 apresenta um resumo sobre os valores de penalidades e peso adotados:

TABELA 5.13: Resumo de penalidades e pesos adotados

Objetivo	Valor	Peso	Justificação
Não Alocados	100 or 50	0.5	Garantir que todos os pacientes críticos sejam alocados em camas é da máxima importância. A não alocação de um paciente, especialmente em UCI, pode ter consequências graves, por isso a penalidade é alta. O peso é o maior entre os critérios, refletindo a prioridade de um hospital.
Tempo Espera	5	0.2	Pacientes que permanecem na UCI além do necessário ocupam camas que poderiam ser usadas por outros. A penalidade moderada incentiva a alta médica indicada das camas, promovendo a eficiência. O peso reflete a importância de um uso eficiente, sem ser tão prioritário quanto a alocação inicial.
Prioridade	20	0.4	Pacientes com alta prioridade devem ser atendidos rapidamente. A penalidade e o peso refletem a necessidade de priorizar pacientes com maior gravidade (como aqueles com alta classificação ASA), sem a sobrepor à necessidade absoluta de garantir que todos sejam atendidos.
Ocupação	1	0.1	Camas vazias num cenário onde há pacientes esperando são indesejáveis, mas o impacto é menos crítico que a não alocação direta ou a permanência excessiva. Portanto, a penalidade e o peso são relativamente baixos, incentivando o uso equilibrado dos recursos sem que seja um fator determinante.

Embora a literatura mencione frequentemente a resolução desse tipo de problema utilizando modelos de programação linear ou programação estocástica, conforme o trabalho de [92]. Estes estudos também reconhecem a existência de restrições, tais como penalidades por exceder a capacidade da UCI ou por ociosidade no centro cirúrgico.

5.4.3 Resultados

A forma alocação utilizada no GA prioriza pacientes com base em critérios clínicos e de planeamento, distribuindo-os de forma otimizada numa UCI composta por 4 camas durante 7 dias para diferentes tipos de categorias de doentes. Assim, um *paciente elegível* é alguém agendado para cirurgia, ou seja, um paciente que se enquadra no caso de estudo referido na secção 5.3, enquanto

que um *paciente de emergência* requer atendimento médico imediato e o *paciente UCI* é alguém que se encontra sob observação na UCI.

Além disso, os resultados da aplicação do GA em diferentes execuções estão descritos na tabela 5.14. A otimização de alocação foi avaliada usando dois métodos: *Manual* e *D-SARSA*, variando o número de episódios entre 32 e 50.

TABELA 5.14: Resultados da otimização do GA através dos métodos manual e Deep SARSA no problema de UCI para 5 execuções

Heuristic	Method	Episodes	Best	Mean All Best	Standard Deviation	Mean Time(s)
GA	Manual	32	297.5	299.1	1.05	19.51
GA	D-SARSA	32	297.5	298.8	1.70	60.70
GA	Manual	50	297.5	298.1	1.15	29.78
GA	D-SARSA	50	297.5	297.5	0.00	111.37

Da análise dos resultados, observa-se que, independentemente do método ou do número de episódios, o valor mínimo alcançado pelo GA foi sempre o mesmo: 297,5. Isso indica que este algoritmo consegue consistentemente encontrar a melhor solução possível (ou uma solução próxima do ótimo) dentro das limitações impostas pelo problema, especialmente, a insuficiência e variabilidade dos dados de teste.

Contudo, ao observarmos a variação da média dos melhores resultados (*Mean All Best*), é possível notar que, com 32 episódios, o método Deep SARSA tem um desempenho ligeiramente melhor (298,8) em comparação com o método manual (299,1), o mesmo acontece quando o número de episódios é aumentado para 50, com uma média de 297,5 e desvio padrão zero, o que significa que todas as execuções alcançaram o melhor valor. Estes resultados parecem indicar que a integração com Deep SARSA podem oferecer uma vantagem em termos de qualidade média das soluções.

A figura 5.7 que se segue, mostra a alocação de pacientes em camas de UCI ao longo de 7 dias. Cada célula na matriz representa uma cama num determinado dia, com o número de *ID* do paciente alocado a essa cama. Pode verificar-se, que os pacientes 1, 2, e 11 permanecem nas mesmas camas (camas 1, 2 e 3, respetivamente) do dia 1 ao dia 4, o que é um comportamento esperado, indicando que esses pacientes foram alocados por mais de um dia consecutivo e a solução preservou a continuidade.

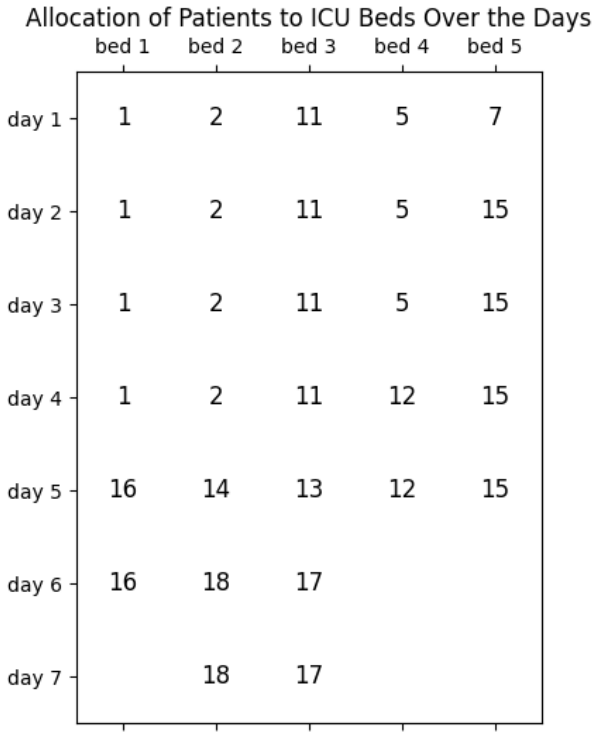


FIGURA 5.7: Atribuição de doentes a camas de UCI ao longo dos dias

5.5 Robustez dos Resultados: Teste de Hipóteses e ANOVA

O objetivo desta secção é avaliar a robustez dos resultados obtidos pela aplicação de estratégias de reforço na configuração dinâmica de algoritmos meta-heurísticos. Através da aplicação de testes estatísticos, como o teste de hipóteses e Analysis Of Variance (ANOVA, Análise da Variância), procura-se determinar se as diferenças observadas entre os métodos aplicados, são estatisticamente significativas, garantindo a validade das conclusões sobre o desempenho dos algoritmos.

A ANOVA é uma técnica estatística criada por Ronald A. Fisher em 1920 para comparar as médias de três ou mais grupos. O objetivo do ANOVA é verificar se há evidência significativa de que pelo menos uma das médias dos grupos seja diferente das outras. A técnica baseia-se na decomposição da variabilidade total dos dados em variabilidade entre grupos e variabilidade dentro dos grupos, sendo o teste ANOVA utilizado para testar a hipótese nula de que as médias dos grupos são iguais.

$$F = \frac{\text{Variabilidade entre os grupos}}{\text{Variabilidade dentro dos grupos}} \tag{5.11}$$

Antes da aplicação do teste de ANOVA, foram cumpridos os seus pressupostos, nomeadamente, o facto de as amostras serem independentes, não terem *outliers* significativos, seguirem uma distribuição normal (avaliadas pelo teste *Shapiro-Wilk*) ou obedecerem a uma homogeneidade

entre grupos de variâncias (avaliadas pelo teste de *Levene*). Esta análise de robustez sobre os dados das estratégias de reforço no contexto da DAC, indicaram que na maioria das situações não existem diferenças significativas. No entanto, vejamos em particular alguns resultados:

TABELA 5.15: Análise de robustez em resultados de custo médio apresentados em diferentes tabelas

Tabela	**Descrição**	**p-value**	H_0
5.2	Manual vs SARSA (Benchmark)	0.5339	False
5.3	Deep-SARSA vs SARSA (Benchmark)	0.5363	False
5.7	Manual vs SARSA vs D-SARSA (Portfolio)	0.2559	False
5.9	Manual vs D-SARSA (Cirurgias)	0.014	True

A informação apresentada na tabela 5.15 demonstra que apesar das variações observadas nos desempenhos das diferentes estratégias de reforço, essas variações não são estatisticamente significativas porque o p-value superior a 0.05. Este foi o valor do critério de decisão para aceitação da hipótese nula no teste de hipóteses (H_0). Aceitar esta hipótese, ou seja, para valores inferiores a p-value implica rejeitar a hipótese alternativa. No caso em concreto apenas na análise de resultados da otimização de salas de cirurgia é que foi validada a hipótese nula, ou seja, confirma-se a diferença observada nos resultados dos experiências entre o método Deep SARSA e Manual. Esta validação foi comprovada pelo teste *post-hoc* de Tukey HSD com o valor de 22.75. Em suma, podemos referir que pese embora os resultados apresentados forneçam uma análise inicial sobre o desempenho das diferentes estratégias de reforço, é importante salientar que as cinco execuções realizadas representam uma limitação do estudo. Para obter resultados estatisticamente mais robustos, seria necessária comprovar com milhares de execuções, para além do contraponto com outras alternativas DAC existentes.

5.6 Segurança e Privacidade dos Dados

No contexto de um ambiente hospitalar, destaca-se a importância crítica de adotar medidas rigorosas para garantir a privacidade dos dados dos pacientes. Inicialmente, é possível utilizar dados sintéticos ou fictícios, porém realistas, para treinar e testar o modelo sem as restrições associadas ao uso de dados reais. Na fase de validação com dados reais, é fundamental seguir estritamente as diretrizes do hospital que fornece esses dados. Esta abordagem facilita o desenvolvimento inicial do modelo de forma capaz, seguido por uma validação robusta, em conformidade com as orientações específicas do ambiente hospitalar.

5.7 Resumo

Este capitulo começou com a comparação entre métodos manuais e automáticos para a configuração de parâmetros em algoritmos genéticos, focando a utilização de técnicas de aprendizagem

por reforço e a sua aplicação inicial em funções *benchmark*, para avaliar e testar a capacidade de diferentes algoritmos em ambientes controlados.

De seguida, foi introduzido o problema da otimização de portfólio, utilizando o método SARSA para maximizar retornos de investimentos e minimizar riscos na seleção de ativos em mercados de energia elétrica. Posteriormente, foi implementado o Deep SARSA, uma versão que integra redes neuronais para melhorar a capacidade de tomada de decisão.

Finalmente, foram discutidos dois casos de estudo hospitalares. O primeiro caso envolveu a alocação de pacientes a cirurgias, utilizando técnicas de otimização para distribuir os pacientes por UC, considerando as restrições de tempo, número de salas e prioridade dos pacientes. O segundo caso abordou a alocação de pacientes em UCI, onde os pacientes foram alocados com base na sua necessidade clínica, tempo estimado de permanência e urgência. Esses exemplos demonstraram a aplicabilidade de estratégias de configuração, como o SARSA e, especialmente, o Deep SARSA, em algoritmos de otimização para enfrentar desafios reais do setor da saúde.

Capítulo 6

Conclusões

6.1 Conclusões

Nesta tese foi apresentada a aplicação de políticas de ajuste dinâmico de algoritmos (DAC) utilizando métodos de *Reinforcement Learning* (RL, Aprendizagem por Reforço) para a configuração de parâmetros em algoritmos meta-heurísticos. Os estudos envolveram desde funções *benchmark* até problemas práticos de otimização, nomeadamente, o problema de um portfólio de investimentos e a alocação de pacientes em blocos de cirurgia (UC) e unidades de cuidados intensivos (UCI). A abordagem apresentada de otimização de problemas com base em meta-heurísticas como *Genetic Algorithm* (GA, Algoritmo Genético), *Particle Swarm Optimization* (PSO, Otimização por Enxame de Partículas) e em técnicas de RL como, o SARSA e o Deep SARSA, mostra-se promissora em relação à complexidade e à variabilidade dos cenários estudados.

Os resultados obtidos demonstraram que as políticas de DAC baseadas em RL proporcionaram melhorias interessantes em termos de ajuste de parâmetros comparativamente aos métodos tradicionais. A principal contribuição para este avanço foi a capacidade de ajuste dinâmico dos parâmetros, ao longo do tempo, com base no *feedback* contínuo do ambiente, ao invés de uma configuração estática. Além disso, esta abordagem é agnóstica quer ao problema a ser resolvido quer ao algoritmo utilizado, o que significa que a metodologia usada pode ser aplicado a uma ampla gama de contextos sem depender das especificidades do problema ou da natureza do algoritmo. Esta independência permitiu otimizar a execução dos algoritmos meta-heurísticos em diferentes tipos de problemas, oferecendo uma maior robustez e adaptabilidade às variações nos dados de entrada e nas condições dos problemas. Para o utilizador final, a única exigência solicitada é a definição dos parâmetros, que serão ajustados dinamicamente.

Nos casos de estudo apresentados neste trabalho, o modelo proposto mostrou, de uma forma geral, ganhos na qualidade das soluções finais. Na otimização de portfólios, os algoritmos ajustados dinamicamente por Deep SARSA, conseguiram encontrar composições melhores, atendendo aos resultados para 50 episódios, das funções-objetivo (78.00 para melhor desempenho, 78.07 para média de todas as execuções e um desvio-padrão de 0.053). Os resultados com 32 episódios foram mais equilibrados (78.01 para melhor desempenho, 78.09 para média de todas as execuções, um desvio-padrão de 0.044 e melhor tempo médio 14.985 segundos). Confrontando estes resultados

com os outros métodos, esta análise mostra, face ao problema em concreto, um balanceamento mais fiável, denotando mais equilíbrio entre risco e retorno sobre o investimento (ROI), otimizando, assim, a alocação de ativos em cenários de incerteza.

A metodologia adotada, independentemente do problema a ser otimizado ou do algoritmo meta-heurístico selecionado, oferece a flexibilidade necessária para estudar problemas em outras setores de atividade. Com efeito, a aplicação desta solução ligada à área médica, especificamente à otimização de recursos em blocos cirúrgicos e unidades de cuidados intensivos, amplia o alcance da solução, tornando-a mais abrangente e versátil.

Um exemplo claro da importância desta metodologia é alocação eficiente de pacientes em Unidades de Cirurgia, um problema crítico que afeta diretamente a qualidade do planeamento e o uso adequado dos recursos hospitalares. Ao analisar os resultados obtidos pela implementação destas técnicas de DAC, neste caso particular, pode concluir-se que as suas políticas, usando o método Deep SARSA conseguem melhorar a alocação de recursos críticos. As poucas execuções efetuadas mostram ainda assim, que o Deep SARSA (52.5 para o melhor resultado, 87.0 para a média das execuções) quando comparado com o método Manual (102.5 para o melhor desempenho e 114.5 para as médias de execuções) obteve variações estatisticamente significativas pela análise do teste de ANOVA.

Em conclusão, a implementação da metodologia DAC, baseada em métodos SARSA e Deep SARSA, demonstra ser eficaz na otimização de parâmetros de algoritmos meta-heurísticos, permitindo um ajuste contínuo e adaptativo durante a execução. Pode dizer-se que esta abordagem superou as limitações das técnicas tradicionais de configuração estática (*offline*), que não se adaptam adequadamente a mudanças no ambiente nos problemas tratados. Deste modo, o processo adaptativo dos parâmetros do algoritmo meta-heurístico resulta num avanço importante no que toca ao desempenho de otimização de problemas. As métricas de avaliação comprovam uma maior eficiência dos algoritmos configurados dinamicamente, tanto em cenários de *benchmark* como em problemas do mundo real.

6.2 Limitações e Trabalhos Futuros

Apesar dos progressos alcançados nesta tese, foram identificadas algumas limitações, que implicam melhorias futuras.

Uma das principais limitações observadas foi o critério adotado para o ajuste dos parâmetros com base num valor percentual fixo para cada ação. Embora esta abordagem tenha evidenciado resultados satisfatórios, é possível que tenha limitado a exploração de soluções mais adequadas para determinados problemas. Pese embora, o ajuste de percentuais fixos se enquadre dentro dos valores aceites pela comunidade científica, este pode não ser suficientemente adaptativo para acompanhar a complexidade crescente de certos ambientes.

Outro aspeto a melhorar, prende-se com a avaliação da performance ao longo de cada estado do processo. No trabalho atual, a evolução dos resultados foi observada de forma agregada,

todavia uma análise mais informada sobre, quão próximo ou não, estamos da solução ótima em cada etapa, pode trazer benefícios adicionais. Assim, introduzir métricas que acompanhem o desempenho em tempo real relativamente à solução ótima esperada, não ajudaria somente a acelerar a convergência do algoritmo, mas também forneceria *insights* mais detalhados sobre o comportamento dos parâmetros em diferentes estágios do processo.

Para que esta metodologia possa reforçar a consolidação dos resultados obtidos será necessário um maior número de execuções que permitam soluções significativamente mais robustas. Tal não foi possível nesta investigação devido a limitações de tempo e recursos disponíveis.

Este aumento do número de execuções, pode e deve, ainda, englobar a comparação com outros modelos de *Automatic Algorithm Configuration* (AAC, Configuração Automática de Algoritmos), o que poderá trazer uma visão mais abrangente dos resultados. A escolha da heurística pode ser integrada como parte da política assimilada pelo agente de RL, em vez de fixar uma única heurística. O agente pode aprender que heurísticas são mais adequadas em diferentes estados do processo, ajustando a abordagem de otimização de forma dinâmica. Essa flexibilidade pode ser modelada como uma ação adicional no espaço de decisões, permitindo ao agente, não apenas o ajuste dos parâmetros de uma heurística, mas também a decisão sobre que heurística utilizar num determinado estado.

A aplicação prática dessas soluções em cenários reais na área da saúde, com mais informação e detalhe, é um ponto importante a ser explorado no futuro

6.3 Questões Éticas e Proteção de Dados

Apesar dos benefícios previstos no sector da Saúde, a utilização da IA envolve muitos riscos. Para além da questão da confiabilidade e aceitação das sistemas de IA pelos profissionais da saúde, existem questões como a proteção de dados ou a ética profissional que é vital garantir. A confidencialidade dos dados dos pacientes é fundamental. Isso inclui informações médicas, pessoais e qualquer dado relacionado à saúde, como, por exemplo, o direito do titular dos dados de não ficar sujeito a nenhuma decisão tomada exclusivamente com base no tratamento automatizado plasmado no artº 22 do REGULAMENTO (UE) 2016/679 DO PARLAMENTO EUROPEU E DO CONSELHO de 27 de abril de 2016. (RGPD - Regulamento Geral para a Proteção de Dados).

A garantia da ética na IA é fundamental para evitar que a tecnologia seja usada com finalidades prejudiciais ou discriminatórias, por isso, no planeamento e agendamento de intervenções cirúrgicas, conceitos como o consentimento informado ou os padrões éticos na tomada de decisão devem ser levadas em consideração. O consentimento informado deve ser obtido de maneira clara e compreensível, garantindo que os pacientes estejam cientes de como suas informações serão utilizadas.

As decisões relacionadas ao planeamento e agendamento de intervenções cirúrgicas devem ser baseadas em considerações como a gravidade da condição do paciente, a necessidade da cirurgia e a equidade no acesso aos procedimentos.

Bibliografia

[1] Mark Daskin. «Network and discrete location: models, algorithms and applications». Em: *Journal of the Operational Research Society* 48.7 (1997), pp. 763–764.

[2] Gilbert Laporte. «Fifty years of vehicle routing». Em: *Transportation science* 43.4 (2009), pp. 408–416.

[3] Frank Hutter, Lars Kotthoff e Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[4] Johann Dréo. *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media, 2006.

[5] Ke Xue et al. «Multi-agent dynamic algorithm configuration». Em: *Advances in Neural Information Processing Systems* 35 (2022), pp. 20147–20161.

[6] Steven Adriaensen et al. «Automated dynamic algorithm configuration». Em: *Journal of Artificial Intelligence Research* 75 (2022), pp. 1633–1699.

[7] Tiago Pinto et al. «Adaptive portfolio optimization for multiple electricity markets participation». Em: *IEEE Transactions on Neural Networks and Learning Systems* 27.8 (2015), pp. 1720–1733.

[8] David H. Rothstein e Mehul V. Raval. «Operating room efficiency». Em: *Seminars in Pediatric Surgery* 27.2 (2018). The Perioperative Experience, pp. 79–85. ISSN: 1055-8586.

[9] Maggard-Gibbons M. Childers CP. «Understanding Costs of Care in the Operating Room». Em: *JAMA surgery* (2018).

[10] Sejal Patel et al. «Understanding the costs of surgery: a bottom-up cost analysis of both a hybrid operating room and conventional operating room». Em: *International Journal of Health Policy and Management* 11.3 (2022), p. 299.

[11] David Mc Mahon e Joseph Walsh. «Operating Theatre Scheduling Using Optimisation Algorithms and Techniques: A Review». Em: *2019 30th Irish Signals and Systems Conference (ISSC)*. IEEE. 2019, pp. 1–7.

[12] André Biedenkapp et al. «Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework». Em: *ECAI 2020*. IOS Press, 2020, pp. 427–434.

[13] Deyao Chen et al. «Using Automated Algorithm Configuration for Parameter Control». Em: *arXiv preprint arXiv:2302.12334* (2023).

[14] Changwu Huang, Hao Bai e Xin Yao. «Online algorithm configuration for differential evolution algorithm». Em: *Applied Intelligence* 52.8 (2022), pp. 9193–9211.

[15] Samantha Hansen. «Using deep q-learning to control optimization hyperparameters». Em: *arXiv preprint arXiv:1602.04062* (2016).

[16] Olusegun Olorunda e Andries P. Engelbrecht. «Measuring exploration/exploitation in particle swarms using swarm diversity». Em: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. 2008, pp. 1128–1134. DOI: 10.1109/CEC.2008.4630938.

[17] Lin Lin e Mitsuo Gen. «Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation». Em: *Soft Computing* 13 (2008), pp. 157–168. URL: https://api.semanticscholar.org/CorpusID:40526902.

[18] Agoston E Eiben e Selmar K Smit. «Parameter tuning for configuring and analyzing evolutionary algorithms». Em: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 19–31.

[19] Frank Hutter et al. «ParamILS: an automatic algorithm configuration framework». Em: *Journal of artificial intelligence research* 36 (2009), pp. 267–306.

[20] L. Jourdan, M. Basseur e E.-G. Talbi. «Hybridizing exact methods and metaheuristics: A taxonomy». Em: *European Journal of Operational Research* 199.3 (2009), pp. 620–629. ISSN: 0377-2217.

[21] Eugene L Lawler e David E Wood. «Branch-and-bound methods: A survey». Em: *Operations research* 14.4 (1966), pp. 699–719.

[22] Naji Mohd Kashif Hussain, Shi Cheng Salleh Mohd e Yuhui Shi. «Metaheuristic research: a comprehensive survey». Em: *Metaheuristic research: a comprehensive survey* 52 (2019), pp. 2191–2233. ISSN: 1573-7462.

[23] Changwu Huang, Yuanxiang Li e Xin Yao. «A Survey of Automatic Parameter Tuning Methods for Metaheuristics». Em: *IEEE Transactions on Evolutionary Computation* 24.2 (2020), pp. 201–216.

[24] Xin-She Yang. «Nature-Inspired Mateheuristic Algorithms: Success and New Challenges». Em: *Journal of Computer Engineering and Information Technology* 01 (nov. de 2012).

[25] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[26] Marco Dorigo, Mauro Birattari e Thomas Stutzle. «Ant colony optimization». Em: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.

[27] James Kennedy e Russell Eberhart. «Particle swarm optimization». Em: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[28] Scott Kirkpatrick, C Daniel Gelatt Jr e Mario P Vecchi. «Optimization by simulated annealing». Em: *science* 220.4598 (1983), pp. 671–680.

[29] Fred Glover. «Future paths for integer programming and links to artificial intelligence». Em: *Computers & operations research* 13.5 (1986), pp. 533–549.

[30] Holger H Hoos. «Automated algorithm configuration and parameter tuning». Em: *Autonomous search*. Springer, 2012, pp. 37–71.

[31] Agoston E Eiben e Selmar K Smit. «Evolutionary algorithm parameters and methods to tune them». Em: *Autonomous search*. Springer, 2012, pp. 15–36.

[32] ES Skakov e VN Malysh. «Parameter meta-optimization of metaheuristics of solving specific NP-hard facility location problem». Em: *Journal of Physics: Conference Series*. Vol. 973. IOP Publishing. 2018, p. 012063.

[33] Agoston E Eiben, Robert Hinterding e Zbigniew Michalewicz. «Parameter control in evolutionary algorithms». Em: *IEEE Transactions on evolutionary computation* 3.2 (1999), pp. 124–141.

[34] Li Yang e Abdallah Shami. «On hyperparameter optimization of machine learning algorithms: Theory and practice». Em: *Neurocomputing* 415 (2020), pp. 295–316. ISSN: 0925-2312.

[35] Barbara Kitchenham et al. «Systematic literature reviews in software engineering–a systematic literature review». Em: *Information and software technology* 51.1 (2009), pp. 7–15.

[36] Saldana Johnny. *Fundamentals of Qualitative Research.* Understanding Qualitative Research. Oxford University Press, 2011. ISBN: 9780199737956.

[37] Frank Hutter, Holger H. Hoos e Kevin Leyton-Brown. «Sequential Model-Based Optimization for General Algorithm Configuration». Em: *Learning and Intelligent Optimization*. Ed. por Carlos A. Coello Coello. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523. ISBN: 978-3-642-25566-3.

[38] Manuel López-Ibáñez et al. «The irace package: Iterated racing for automatic algorithm configuration». Em: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[39] Hojjat Rakhshani et al. «MAC: Many-objective Automatic Algorithm Configuration: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings». Em: ResearchGate, fev. de 2019, pp. 241–253. ISBN: 978-3-030-12597-4.

[40] Yasemin Eryoldaş e Alptekin Durmuşoglu. «A literature survey on offline automatic algorithm configuration». Em: *Applied Sciences* 12.13 (2022), p. 6316.

[41] Aymeric Blot et al. «Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems». Em: *Evolutionary Computation* 27.1 (mar. de 2019), pp. 147–171. ISSN: 1063-6560.

[42] Tadhg Fitzgerald e Barry O'Sullivan. «Candidate selection and instance ordering for realtime algorithm configuration». Em: *Fundamenta Informaticae* 166.2 (2019), pp. 141–166.

[43] Tadhg Fitzgerald, Yuri Malitsky e Barry O'Sullivan. «Reactr: Realtime algorithm configuration through tournament rankings». Em: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. Citeseer. 2015.

[44] Roberto Amadini, Maurizio Gabbrielli e Jacopo Mauro. «SUNNY: a lazy portfolio approach for constraint solving». Em: *Theory and Practice of Logic Programming* 14.4-5 (2014), pp. 509–524.

[45] Yuri Malitsky e Yuri Malitsky. «Evolving instance-specific algorithm configuration». Em: *Instance-Specific Algorithm Configuration* (2014), pp. 93–105.

[46] Yasemin Eryoldaş e Alptekin Durmuşoğlu. «A Literature Survey on Instance Specific Algorithm Configuration Methods». Em: (2021).

[47] Silja Meyer-Nieberg e Hans-Georg Beyer. «Self-adaptation in evolutionary algorithms». Em: *Parameter setting in evolutionary algorithms*. Springer, 2007, pp. 47–75.

[48] Ivars Dzalbs e Tatiana Kalganova. «Simple generate-evaluate strategy for tight-budget parameter tuning problems». Em: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 783–790.

[49] Nikolaus Hansen. «The CMA evolution strategy: a comparing review». Em: *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms* (2006), pp. 75–102.

[50] Pietro A Consoli et al. «Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis». Em: *Soft Computing* 20 (2016), pp. 3889–3914.

[51] Xiuli Wu et al. «An evolutionary hyper-heuristic for the software project scheduling problem». Em: *Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14*. Springer. 2016, pp. 37–47.

[52] Soheila Ghambari et al. «Unbalanced budget distribution for automatic algorithm configuration». Em: *Soft Computing* 26.3 (2022), pp. 1315–1330.

[53] Stefan Falkner, Aaron Klein e Frank Hutter. «BOHB: Robust and efficient hyperparameter optimization at scale». Em: *International conference on machine learning*. PMLR. 2018, pp. 1437–1446.

[54] Peter J Angeline et al. «Adaptive and self-adaptive evolutionary computations». Em: *Computational intelligence: a dynamic systems perspective* (1995), pp. 152–163.

[55] Robert Hinterding, Zbigniew Michalewicz e Agoston E Eiben. «Adaptation in evolutionary computation: A survey». Em: *Proceedings of 1997 Ieee International Conference on Evolutionary Computation (Icec'97)*. IEEE. 1997, pp. 65–69.

[56] James E Smith. «Self-adaptation in evolutionary algorithms for combinatorial optimisation». Em: *Adaptive and multilevel metaheuristics* (2008), pp. 31–57.

[57] James Edward Smith e Terence C Fogarty. «Operator and parameter adaptation in genetic algorithms». Em: *Soft computing* 1 (1997), pp. 81–87.

[58] Hongshu Guo et al. «Deep Reinforcement Learning for Dynamic Algorithm Selection: A Proof-of-Principle Study on Differential Evolution». Em: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54.7 (2024), pp. 4247–4259. DOI: 10.1109/TSMC.2024.3374889.

[59] H. Fei, N. Meskens e C. Chu. «A planning and scheduling problem for an operating theatre using an open scheduling strategy». Em: *Computers & Industrial Engineering* 58.2 (2010). Scheduling in Healthcare and Industrial Systems, pp. 221–230. ISSN: 0360-8352.

[60] Inês Marques, M Eugénia Captivo e Margarida Vaz Pato. «Scheduling elective surgeries in a Portuguese hospital using a genetic heuristic». Em: *Operations Research for Health Care* 3.2 (2014), pp. 59–72.

[61] André Biedenkapp. «Dynamic Algorithm Configuration by Reinforcement Learning». Tese de doutoramento. University of Freiburg, Germany, 14 de out. de 2022. published.

[62] Matthew Hausknecht, Peter Stone e Off-policy Mc. «On-policy vs. off-policy updates for deep reinforcement learning». Em: *Deep reinforcement learning: frontiers and challenges, IJCAI 2016 Workshop*. AAAI Press New York, NY, USA. 2016.

[63] Pedro Domingos. *A revolução do algoritmo mestre*. Editorial Presença, 2017.

[64] Ricardo Faia, Tiago Pinto e Zita Vale. «GA optimization technique for portfolio optimization of electricity market participation». Em: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2016, pp. 1–7.

[65] Kaspar Höschel e Vasudevan Lakshminarayanan. «Genetic algorithms for lens design: a review». Em: *Journal of Optics* 48 (2019), pp. 134–144.

[66] Anupriya Shukla, Hari Mohan Pandey e Deepti Mehrotra. «Comparative review of selection techniques in genetic algorithm». Em: *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*. 2015, pp. 515–519.

[67] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan, 1975.

[68] John J Grefenstette. «Optimization of control parameters for genetic algorithms». Em: *IEEE Transactions on systems, man, and cybernetics* 16.1 (1986), pp. 122–128.

[69] Agoston E Eiben e James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[70] April Yi Wang et al. «How data scientists use computational notebooks for real-time collaboration». Em: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW (2019), pp. 1–30.

[71] Y. Shi e R.C. Eberhart. «Empirical study of particle swarm optimization». Em: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. 1999, 1945–1950 Vol. 3.

[72] Zhi-Hui Zhan et al. «Adaptive Particle Swarm Optimization». Em: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6 (2009), pp. 1362–1381.

[73] Richard S Sutton e Andrew G Barto. «Reinforcement learning: An introduction». Em: *Robotica* 17.2 (1999), pp. 229–235.

[74] Qinglin Meng et al. «An Online Reinforcement Learning-based Energy Management Strategy for Microgrids with Centralized Control». Em: *IEEE Transactions on Industry Applications* (2024), pp. 1–10.

[75] Aadhith Sanjay Prasanna et al. «Application of Reinforcement Learning Models for Optimization of Air Traffic Control Protocols». Em: *2024 10th International Conference on Applied System Innovation (ICASI)*. 2024, pp. 190–192.

[76] Justin A Boyan. «Technical update: Least-squares temporal difference learning». Em: *Machine learning* 49 (2002), pp. 233–246.

[77] Jayarama Pradeep et al. «DeepFore: A Deep Reinforcement Learning Approach for Power Forecasting in Renewable Energy Systems». Em: *Electric Power Components and Systems* (2024), pp. 1–17.

[78] Andreas Metzger et al. «Realizing self-adaptive systems via online reinforcement learning and feature-model-guided exploration». Em: *Computing* 106.4 (2024), pp. 1251–1272.

[79] Volodymyr Mnih et al. «Playing atari with deep reinforcement learning». Em: *arXiv preprint arXiv:1312.5602* (2013).

[80] Dongbin Zhao et al. «Deep reinforcement learning with experience replay based on SARSA». Em: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2016, pp. 1–6.

[81] Volodymyr Mnih et al. «Human-level control through deep reinforcement learning». Em: *nature* 518.7540 (2015), pp. 529–533.

[82] Ekaba Bisong. «Google Colaboratory». Em: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, pp. 59–64.

[83] Masaru Ohkawara, Hideo Saito e Issei Fujishiro. «Experiencing GPU path tracing in online courses». Em: *Graphics and Visual Computing* 4 (2021), p. 200022.

[84] Biyanka Ekanayake. «A deep learning-based building defects detection tool for sustainability monitoring». Em: jun. de 2022.

[85] Luca Baronti. «Benchmark Functions - A Python Library». Em: (2021). (Acedido em 09/05/2024).

[86] Vitor Oliveira et al. «Automatic Configuration of Genetic Algorithm for the Optimization of Electricity Market Participation Using Sequential Model Algorithm Configuration». Em: *EPIA Conference on Artificial Intelligence*. Springer. 2022, pp. 245–257.

[87] Ricardo Faia, Tiago Pinto e Zita Vale. «Dynamic fuzzy estimation of contracts historic information using an automatic clustering methodology». Em: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2015, pp. 270–282.

[88] Abdur Rais e Ana Viana. «Operations research in healthcare: a survey». Em: *International transactions in operational research* 18.1 (2011), pp. 1–31.

[89] Yang-Kuei Lin e Yin-Yi Chou. «A hybrid genetic algorithm for operating room scheduling». Em: *Health care management science* 23 (2020), pp. 249–263.

[90] Eveline HJ Mestrom et al. «Prediction of postoperative patient deterioration and unanticipated intensive care unit admission using perioperative factors». Em: *PLoS One* 18.8 (2023), e0286818.

[91] Fang Wan et al. «Two-stage multi-objective optimization for ICU bed allocation under multiple sources of uncertainty». Em: *Scientific Reports* 13.1 (2023), p. 18925.

[92] Aida Jebali e Ali Diabat. «A stochastic model for operating room planning under capacity constraints». Em: *International Journal of Production Research* 53.24 (2015), pp. 7252–7270.

Apêndice A

Algoritmos

Algorithm A.1 *DAC Agent*

```
1:  def DAC_Agent(episodes, parameters, bench):
2:      env = RLEnvironment(parameters, bench)
3:      ql = sarsa()
4:      ql.create_table(states, actions)
5:      action = ql.policy(state)
6:      for episode in range(episodies):
7:          cfg = ql.param[state][action]
8:          next_state, reward, done, sol, fv, cfg = env.step(action, cfg)
9:          next_action = ql.policy(next_state)
10:         ql.learning(state, action, reward, next_state, next_action, cfg)
11:         action = next_action
12:         state = next_state
13:     end for
```

O algoritmo *DAC Agent* é o corpo central da configuração de parâmetros meta-heurísticos. Nele é aplicado o método *SARSA* para ajustar os parâmetros de forma adaptativa durante a execução, usando recompensas para a escolha de ações em cada estado. A cada interação com o ambiente, o agente otimiza sua política, selecionando as melhores configurações ao longo dos episódios. .

O código implementa o algoritmo SARSA para aprendizado por reforço. Na inicialização, a classe define os principais parâmetros do agente, incluindo a taxa de exploração (ϵ), a taxa de aprendizagem (α) e a taxa de desconto (γ), além de inicializar as tabelas Q() e dos parâmetros ($param$). A classe permite que o agente aprenda e melhore suas decisões em um ambiente dinâmico, ajustando suas políticas de ação com base nas recompensas obtidas.

Algorithm A.2 Classe Sarsa

```
1:  class Sarsa:
2:      def Init(self, epsilon, alpha, gamma):
3:          self.epsilon = epsilon                                  ▹ Exploitation rate
4:          self.alpha = alpha                                      ▹ Learning rate
5:          self.gamma = gamma                                      ▹ Discount rate
6:          self.qsa = None
7:          self.param = None
8:
9:      def create_table(self, states, actions):
10:         rows = len(states)
11:         cols = len(actions)
12:         self.qsa = np.zeros((rows, cols))
13:         self.param = np.array([[(0, 0.0, 0.0, 0, np.inf, 0) for _ in range(cols)] for _ in range(rows)])
14:
15:     def copy_state(self, initial, final):
16:         self.qsa[final, :] = self.qsa[initial, :]
17:         self.param[final, :] = self.param[initial, :]
18:
19:     def policy(self, state):
20:         rvalue = random.uniform(0, 1)
21:         if (rvalue < self.epsilon):
22:             index = np.random.randint(len(self.qsa[state]))
23:         else:
24:             av = self.qsa[state]
25:             index = np.random.choice(np.flatnonzero(av = av.min()))
26:         return index
27:
28:     def learning_initial(self, state, action, configuration):
29:         self.qsa[state][action] = round(self.alpha * configuration[4], 3)
30:         self.param[state][action] = configuration
31:
32:     def learning(self, state, action, reward, next_state, next_action, configuration):
33:         current_qsa = self.qsa[state][action]
34:         next_qsa = reward + self.gamma * self.qsa[next_state][next_action]
35:         self.qsa[state][action] += round(self.qsa[state][action] + self.alpha * (next_qsa - current_qsa), 3)
36:         if (self.param[state][action][4] > configuration[4])
37:             self.param[state][action] = configuration
```

Algorithm A.3 Classe RLEnvironment

```
class RLEnvironment:
    def Init(self, genes, parameters, bench, states_max, choromose_binarie=True):
        self.state = 0
        self.one = 0
        self.states_max = states_max
        self.pop_size_range = parameters['pop_size_range']
        self.crossover_range = parameters['crossover_rate_range']
        self.mutation_range = parameters['mutation_rate_range']
        self.generations_range = parameters['generations_range']
        self.ga = GAHeuristic(genes, bench, choromose_binarie)

    def reset(self):
        self.pop_size = np.random.randint(self.pop_size_range[0], self.pop_size_range[1] + 1)
        self.crossover_rate = np.random.uniform(self.crossover_range[0], self.crossover-_range[1])
        self.mutation_rate = np.random.uniform(self.mutation_range[0], self. mutation_range[1])
        self.generations = np.random.randint(self.generations_range[0], self.generations_range[1] + 1)
        _, fitness = self.ga.run(self.pop_size, self.crossover_rate, self.mutation_rate, self.generations)
        Return (self.pop_size, self.crossover_rate, self.mutation_rate, self.generations, fitness, 1)

    def run_heuristic(self):
        Return self.ga.run(self.pop_size, self.crossover_rate, self.mutation_rate, self.generations)

    def step(self, action, configuration):
        self.pop_size = int(configuration[0])
        self.crossover_rate = configuration[1]
        self.mutation_rate = configuration[2]
        self.generations = int(configuration[3])
        individual, fitness_value = self.__get_actions(action)
        sequence = configuration[5] + 1
        reward = self.compute_reward(fitness_value, sequence)
        self.one = self.one + 1
        done = False
        if (self.one > self.states_max):
            done = True
            self.state = self.state + 1
            self.one = 0
        Return self.state, reward, done, individual, fitness_value, (self.pop_size, self.crossover_rate,
    self.mutation_rate, self.generations, fitness_value, sequence)

    def compute_reward(self, fitness_value, sequence):
        Return fitness_value

    def __get_action(self,action):
        if action==1: self.pop_size = int(min(self.pop_size * 1.05, self.pop_size_range[1]))
        elif action==2: self.pop_size = int(max(self.pop_size_range[0], self.pop_size * 0.95))
        elif action==3: self.crossover_rate = min(self.crossover_rate * 1.05, self.crossover_rate_range[1])
        elif action==4: self.crossover_rate = max(self.crossover_rate_range[0], self.crossover_rate * 0.95)
        elif action==5: self.mutation_rate = min(self.mutation_rate * 1.01, self.mutation_rate_range[1])
        elif action==6: self.mutation_rate = max(self.mutation_rate_range[0], self.mutation_rate * 0.99)
        elif action==7: self.generations = int(min(self.generations * 1.05, self.generations_range[1]))
        elif action==8: self.generations = int(max(self.generations_range[0], self.generations * 0.95))
        return self.ga.run(self.pop_size,self.crossover_rate,self.mutation_rate, self.generations)
```

Apêndice B

Casos de Estudo

Este anexo tem como objetivo complementar o corpo principal do trabalho, fornecendo informações detalhadas sobre os dados usados como requisitos e sobre os resultados das experiências verificadas nos três cenário específicos: otimização de portfólios, alocação de doentes em blocos operatórios e alocação de doentes em unidades de cuidados intensivos.

B.1 Optimização de Portfolio em Mercado de Energia

A Tabela B.1 representa os tipos de mercados (e.g. Spot, Iday1, ..., Iday6, Local), onde cada cliente ou utilizador, pode participar na ótica de comprador ou vendedor. São, ainda, indicados os limites de energia, em MW definidos por mercado (e.g. Lim_buy e Lim_Sell).

TABELA B.1: Tipos de mercado onde cliente pode participar na ótica de comprador ou vendedor e limites de energia (MW) definidos por mercado

Name	**Spot**	**Iday1**	**Iday2**	**Iday3**	**Iday4**	**Iday5**	**Iday6**	**Local**
Buy	1	1	1	1	1	1	1	1
Lim_buy (MW)	1	0.17	0.17	0.17	0.17	0.17	0.17	0.01
Sell	0	1	1	1	1	1	1	1
Lim_Sell (MW)	1	0.17	0.17	0.17	0.17	0.17	0.17	0.01
Buy_and_Sell	1	0	0	0	0	0	0	0

Na tabela seguinte conseguimos obter para cada período de negociação os preços esperados por mercado, são dados importantes para estimar os resultados e valorizar o objetivo, que é alcançar o menor custo para cada utilizador.

TABELA B.2: Preços esperados por mercado e período de negociação

Período	**Spot**	**Iday1**	**Iday2**	**Iday3**	**Iday4**	**Iday5**	**Iday6**	**Local**
1	175.28	182.31	190.22	178.58	0	0	0	162
2	172.8	178.84	178.08	168.08	0	0	0	172
...	...	...	...	...	...	...	...	...
24	153.92	168.89	192.65	188.45	192	191.5	189.97	195

A Tabela B.3 indica em que mercado e período de negociação um utilizador pode intervir, ou seja, se está autorizado a comprar ou vender em cada sessão. Esta informação é do tipo booleano, em que o algarismo (1) indica disponibilidade, enquanto o (0) o contrário.

TABELA B.3: Disponibilidade de mercado - Tipos de balanceamento de mercado (ativo/inativo)

Período	**Spot**	**Iday1**	**Iday2**	**Iday3**	**Iday4**	**Iday5**	**Iday6**	**Local**
1	1	1	1	1	0	0	0	0
2	1	1	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...
24	1	1	1	1	1	1	1	1

Finalmente, a última tabela carregada refere-se ao consumo esperado em cada período de negociação. Trata-se de uma informação necessária, porque pode condicionar a solução, através de uma penalização, caso esta não seja equilibrada.

TABELA B.4: Consumo Esperado - Dados de consumo (MW)

Período	**Consumo (MW)**
1	0.469
2	0.064
...	....
24	0.515

De seguida, apresentamos os resultados verificados na otimização do portfólio de investimentos em mercados de energia elétrica para os diferentes métodos analisados.

TABELA B.5: Tabela de resultados pbservados nos métodos Manual, SMAC e SARSA usando a heurística *Genetic Algorithm* (GA, Algoritmo Genético) em 32 episódios

Método	**Best**	**Mean All Best**	**Standard Deviation**	**Mean Time**
Manual	78.43	79.21	0.28	16.81
SMAC	78.33	78.97	0.43	42.63
SARSA	78.16	78.66	0.55	18.83

Em face destes resultados, foram também apuradas as configurações para os melhores resultados obtidos na rubrica *Best*.

Pela tabela podemos verificar que os métodos SMAC e SARSA utilizam populações maiores, taxas de *crossover* relativamente altas e taxas de mutação moderadas para explorar o espaço de soluções de forma eficaz. O SARSA, com uma taxa de mutação mais elevada, permitiu mais diversidade, embora com menos gerações, tendo tempos de execução mais rápidos. O SMAC focou em mais gerações, o que ajudou na estabilidade, mas com um custo de tempo maior.

TABELA B.6: Configurações dos parâmetros para o algoritmo GA para os melhores resultados (*Best*) obtidos na tabela anterior

Portfolio Optimization	Pop_Size	Crossover_Rate	Mutation_Rate	Generations
Initial Values	[50-500]	[0,6-0,9]	[0,01-0,1]	[30-90]
MANUAL	277	0,699	0,037	59
SMAC	443	0,829	0,040	90
SARSA	448	0,789	0,069	36

B.2 Alocação de Pacientes em Salas de Cirurgia

A próxima tabela B.7 representa uma lista de doentes a aguardar por cirurgia, onde consta a identificação do doente e da cirurgia, o tempo previsto de duração em minutos da cirurgia e o tempo em dias para que se faça a cirurgia.

TABELA B.7: Informação de cirurgias para doentes elegíveis

Patient ID	**Surgerie ID**	**Duration (min)**	**DeadLine (days)**
1	1	85	1
2	2	60	1
3	3	240	3
4	4	45	2
5	5	120	1
6	1	130	1
7	2	60	1
8	3	240	3
9	6	145	1
...	...	...	...
25	2	60	1
26	3	90	3
27	2	60	1
28	3	90	3
29	4	45	2
30	5	75	1

A tabela seguinte apresenta a informação sobre a disponibilidade das salas de cirurgia, para 3 dias. Em cada dia estão indicados o tempo por sala, expresso em *min*.

TABELA B.8: Disponibilidade das salas de cirurgia durante 3 dias

Dia	**Sala 1**	**Sala 2**	**Sala 3**
1	480	480	240
2	240	0	0
3	0	240	480

Depois de aplicados os métodos DAC este problema foram registados os seguintes resultados:

TABELA B.9: Otimização do agendamento de cirurgias usando os algoritmos, GA e PSO em 5 execuções

	Heuristic	**Method**	**Episodes**	**Best**	**Mean All Best**	**Standard Deviation**	**Mean Time (s)**
A	GA	Manual	32	102,5	114,5	7,31	22,08
B	GA	D-SARSA	32	52,5	87,0	28,91	78,31
C	GA	D-SARSA	50	52,5	77,5	25,05	111,46
D	PSO	Manual	32	80,0	111,5	16,32	43,91
E	PSO	D-SARSA	32	57,5	74,0	21,13	86,20
F	PSO	D-SARSA	50	72,5	100,0	18,03	60,31

Com estes resultados alcançados pela seleção de parâmetros nos algoritmos meta-heurísticos, as soluções encontradas foram as seguintes:

TABELA B.10: Lista de soluções resultantes da configuração de parâmetros

	Solução
A	{16, 1, 25, 10, 30, 6, 5, 29, 18, 14, 2, 4, 23, 19, 20, 24, 8, 3, 22, 15, 26, 17, 21, 28}
B	{4, 20, 30, 1, 29, 16, 27, 9, 11, 18, 23, 19, 25, 5, 8, 2, 14, 7, 3, 28, 21, 17, 26, 15}
C	{7, 6, 2, 27, 5, 10, 24, 14, 19, 11, 22, 18, 13, 16, 30, 8, 12, 3, 4, 28, 21, 15, 17, 26}
D	{16, 11, 13, 12, 14, 7, 24, 25, 19, 22, 4, 27, 29, 23, 6, 3, 18, 20, 8, 17, 26, 28, 21, 15}
E	{19, 10, 11, 24, 22, 27, 12, 14, 18, 25, 4, 9, 20, 23, 16, 3, 8, 13, 29, 17, 21, 26, 28, 15}
F	{10, 19, 25, 9, 27, 14, 23, 13, 24, 2, 22, 29, 12, 20, 3, 18, 8, 7, 16, 17, 28, 15, 21, 26}

Em face destes resultados foram obtidos as seguintes configurações para os algoritmos GA e PSO:

TABELA B.11: Resultados da configuração dos parâmetros para os algoritmos GA e PSO

	pop_size	**cross**	**mut**	**gen**					
A	238	0.809	0.013	81	-	-	-	-	-
B	380	0.892	0.097	30	-	-	-	-	-
C	500	0.821	0.066	31	-	-	-	-	-

	wmin	**wmax**	**c1min**	**c1max**	**c2min**	**c2max**	**repair**	**iter**	**part**
D	0.359	0.791	0.537	1.705	0.769	1.917	0.300	98	86
E	0.237	0.900	0.441	2.000	0.141	2.000	0.298	91	74
F	0.100	0.739	0.420	1.482	0.947	1.000	0.275	86	70

A figura representa o cronograma de alocação de pacientes para blocos de cirurgia nos próximos dias, para a solução [4, 20, 30, 1, 29, 16, 27, 9, 11, 18, 23, 19, 25, 5, 8, 2, 14, 7, 3, 28, 21, 17, 26, 15]

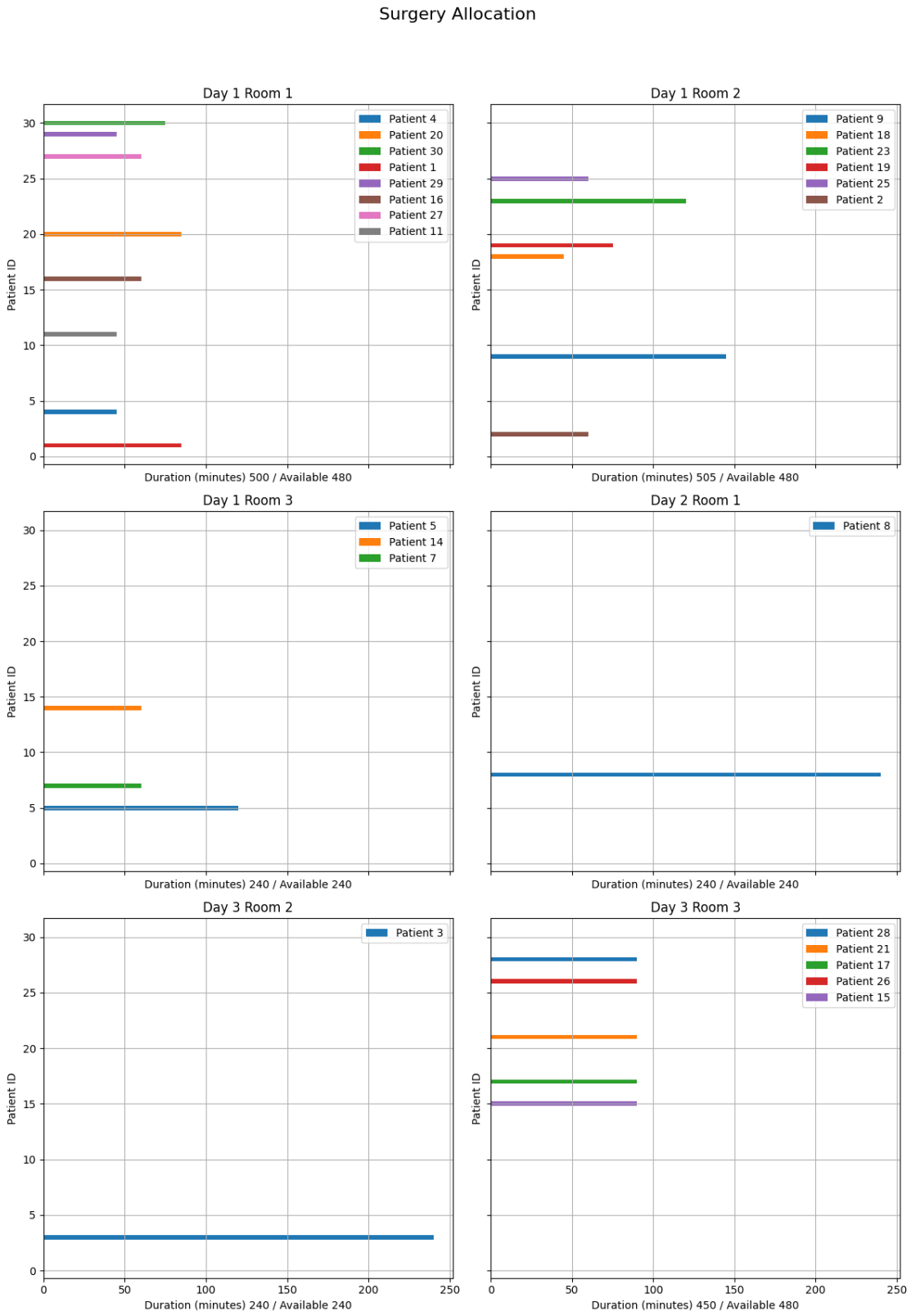


FIGURA B.1: Cronograma completo da melhor solução (B) encontrada na DAC para o problema da alocação de cirurgias

B.3 Alocação de pacientes em UCI

Como informação subjacente ao problema de UCI fazemos referência à tabela seguinte onde esta representada a identificação do paciente **ID**, o nível de risco do dente **ASA**, o tipo de cirurgia **Type**, o status sobre a admissão do doente na unidade **SA** e a previsão de permanência em UCI **LOS**.

TABELA B.12: Dados dos pacientes

ID	ASA	Type	SA	LOS
1	3	Major	-2	4
2	2	Major	-1	4
3	4	Major	0	5
4	1	Minor	0	0
5	5	Major	1	3
6	1	Minor	1	0
7	2	Major	2	1
8	1	Minor	2	0
9	3	Major	2	3
10	2	Minor	3	0
11	5	Major	3	4
12	4	Major	3	2
13	3	Minor	3	1
14	3	Major	4	1
15	4	Major	4	4
16	5	Major	4	2
17	3	Major	5	3
18	2	Major	5	3

Para este problema de alocação de pacientes a UCI foram alcançados os seguintes resultados:

TABELA B.13: Otimização do alocação UCI usando o algoritmo GA em 5 execuções

Heuristic	Method	Episodes	Best	Mean All Best	Standard Deviation	Mean Time (s)
GA	Manual	32	297.5	299.1	1.05	19.51
GA	D-SARSA	32	297.5	298.8	1.70	60.70
GA	Manual	50	297.5	298.1	1.15	29.78
GA	D-SARSA	50	297.5	297.5	0.00	111.37