



CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em Modelagem
Matemática e Computacional

APRENDIZADO POR REFORÇO
MULTIAGENTE E MULTIOBJETIVO VIA
MULTI-ARMED BANDIT NA
ARQUITETURA MULTIAGENTE PARA
META-HEURÍSTICAS AMAM

Tese de Doutorado submetida ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como parte dos requisitos necessários para obtenção do título de Doutor em Modelagem Matemática e Computacional.

Aluno : Jardell Fillipe da Silva

Orientador : Prof. Dr. Sérgio Ricardo de Souza (CEFET-MG)

Co-Orientadora : Profa. Dra. Maria Amélia Lopes Silva (UFV)

Belo Horizonte - MG
Julho de 2026

S586a Silva, Jardell Fillipe da
Aprendizado por reforço multiagente e multiobjetivo via multi-armed bandit na arquitetura multiagente para meta-heurísticas AMAM / Jardell Fillipe da Silva. – 2026.
153 f.

Tese (doutorado) apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Sérgio Ricardo de Souza.
Coorientadora: Maria Amélia Lopes Silva.
Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Sistemas Multiagente – Teses. 2. Aprendizagem por reforço – Modelos Matemáticos – Teses. 3. Otimização Multiobjetivo – Teses. 4. Alocação de Recursos – Teses. 5. Meta-heurísticas – Teses. 6. Arquitetura de Software – Teses. I. Souza, Sérgio Ricardo de. II. Silva, Maria Amélia Lopes. III. Centro Federal de Educação Tecnológica de Minas Gerais. IV. Título.

CDD 519.6



SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

**“APRENDIZADO POR REFORÇO MULTIAGENTE E MULTIOBJETIVO
VIA MULTI-ARMED BANDIT NA ARQUITETURA MULTIAGENTE PARA
META-HEURÍSTICAS AMAM”.**

Tese de Doutorado apresentada por **Jardell Fillipe da Silva**, em 15 de julho de 2026,
ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do
CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Sérgio Ricardo de Souza (Orientador)
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br **SERGIO RICARDO DE SOUZA**
Data: 26/08/2026 20:40:42-0300
Verifique em <https://validar.iti.gov.br>

Profª. Drª. Maria Amélia Lopes Silva (Coorientadora)
Universidade Federal de Viçosa

Documento assinado digitalmente
gov.br **MARIA AMELIA LOPES SILVA**
Data: 17/09/2026 15:16:05-0300
Verifique em <https://validar.iti.gov.br>

Profª. Drª. Ana Lúcia Cetertich Bazzan
Universidade Federal do Rio Grande do Sul

Documento assinado digitalmente
gov.br **ANA LUCIA CETERTICH BAZZAN**
Data: 23/09/2026 09:15:17-0300
Verifique em <https://validar.iti.gov.br>

Profª. Drª. Myriam Regattieri De Biase da Silva Delga
Universidade Tecnológica Federal do Paraná

Documento assinado digitalmente
gov.br **MYRIAM REGATTIERI DE BIASE DA SILVA DELGA**
Data: 24/10/2026 16:48:46-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Alisson Marques da Silva
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br **ALISSON MARQUES DA SILVA**
Data: 27/08/2026 18:03:15-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Marcone Jamilson Freitas Souza
Universidade Federal de Ouro Preto

Documento assinado digitalmente
gov.br **MARCONE JAMILSON FREITAS SOUZA**
Data: 15/09/2026 12:35:41-0300
Verifique em <https://validar.iti.gov.br>

Visto e permitida à impressão,

Prof. Dr. Alisson Marques da Silva
Coordenador do Programa de Pós-Graduação Stricto Sensu em
Modelagem Matemática e Computacional



SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

DECLARAÇÃO

Declaro que **JARDELL FILLIPE DA SILVA** realizou todas as correções exigidas pela banca examinadora em sua Tese de Doutorado, intitulada *“APRENDIZADO POR REFORÇO MULTIAGENTE E MULTIOBJETIVO VIA MULTI-ARMED BANDIT NA ARQUITETURA MULTIAGENTE PARA META-HEURÍSTICAS AMAM”*, cuja defesa foi realizada em de 15 de julho de 2026, e que, portanto, o presente exemplar representa a versão final desse trabalho acadêmico, estando, assim, pronto para ser publicado.

Belo Horizonte, 23 / 10 / 2026

Documento assinado digitalmente
 **SERGIO RICARDO DE SOUZA**
Data: 26/08/2026 20:40:42-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr.Sérgio Ricardo de Souza
(Orientador)

Aos meus pais, Geraldo e Maria Zélia, pela vida e pelos valores que me formaram.
À minha esposa Juliana, pelo amor e companheirismo em cada etapa desta jornada.
Aos meus filhos Anne e Raul, por serem minha maior motivação.

Agradecimentos

Agradecimento: “reconhecimento e declaração de se estar grato por algo dado ou feito por outrem”. Sendo assim, toda conquista requer agradecimentos.

Primeiramente, agradeço à minha esposa, Juliana, pelo carinho, pela paciência e pela compreensão nos momentos em que não pude estar tão presente quanto gostaria. Por se tornar referência nesses momentos e minha maior incentivadora nesse processo, a primeira pessoa a acreditar em mim, mesmo quando eu não acreditava. Obrigado por ser essa pessoa maravilhosa e especial, que me deu o prazer de dividir o mundo.

Agradeço aos meus filhos, Anne e Raul, por serem a razão de eu batalhar todos os dias e por, mesmo de forma inconsciente, compreenderem que esse processo era em busca de um objetivo maior. Por serem meu refúgio nas horas de pressão e minha maior inspiração.

Aos meus pais, Geraldo e Maria Zélia, agradeço pela vida e pelos valores que me formaram, pelo apoio incondicional durante todo esse processo e por não medirem esforços para que isso acontecesse.

Sou grato aos meus professores orientadores, Sérgio Ricardo de Souza e Maria Amélia Lopes Silva, pela orientação, pela dedicação e pelo conhecimento transmitido. Se não fosse por vocês, nada disso seria possível. Agradeço ainda pelos conselhos, não apenas relacionados ao nosso trabalho, mas também à vida; pelos papos sobre filhos, esporte, política, religião e qualquer outro tema que nos “desviava” por algumas horas na semana.

Professor Sérgio, sempre pronto para ouvir e emitir suas opiniões precisas e cirúrgicas sobre qualquer tema. Pelas sugestões de leituras, livros e até mesmo pela sugestão de assistir a novelas para relaxar. Por ser enfático em afirmar que “onde” é advérbio de lugar e que, como eu sou “computeiro”, escrever é uma dificuldade. Afinal de contas, tudo isso que estamos fazendo era trivial, ou não. Obrigado.

Professora Maria Amélia, sempre pronta para nos ajudar, sendo cúmplice dos nossos problemas e dos nossos atrasos, às vezes programados. Pelos papos em nossas reuniões de uma hora, nas quais 50 minutos eram sobre problemas sérios não relacionados à tese e os outros dez eram dedicados a resolver, de fato, o problema. Também agradeço pelas conversas sobre as dificuldades e os prazeres de sermos pais.

Estendo meus agradecimentos aos professores do PPGMMC/CEFET-MG com os quais cruzei nesse processo, pelo conhecimento e pelo auxílio no decorrer das disciplinas. Em especial, aos professores Marcone Jamilson Freitas Souza e Elisângela Martins de Sá, que me acompanharam desde o início dessa jornada. À professora Elizabeth Fialho Wanner, que sempre se colocou à disposição para auxiliar no processo de definição dos testes estatísticos.

Agradeço também aos professores e profissionais com os quais tive o prazer de conviver durante esse tempo, seja em bancas de defesa, disciplinas isoladas ou visitas, dos quais cito alguns a seguir. À professora Ana Lúcia Cetertich Bazzan, pela oportunidade de cursar sua disciplina remotamente na UFRGS e de contribuir com discussões sobre o trabalho. Ao professor Antônio Carlos Fava de Barros, que foi meu orientador na graduação e sempre esteve disposto a ajudar. Aos professores Marlos Machado e Levi Lelis, que me receberam por uma semana na Universidade de Alberta e contribuíram com dicas e opiniões.

Registro minha gratidão aos membros da banca examinadora, pela disposição em avaliar esta tese. Agradeço também às instituições CEFET - MG, UFV - Campus Florestal e PUC Minas.

Aos meus familiares, agradeço pelo amor, pelo apoio e por serem uma válvula de escape nesse processo. À minha irmã, ao meu cunhado e à minha sobrinha, pela disponibilidade e pelo apoio. Aos meus tios e tias, pelo suporte. Aos meus primos, pelas resenhas, pelo apoio e pelas conversas fiadas. Aos meus sogros, por sempre me apoiarem. Em memória dos meus avós, dos quais não posso deixar de destacar a Vó Maria, que não pôde me ver formar, mas sempre carregava a “preocupação” ao dizer: “... coitado, ele estuda, estuda, mas não formal!”.

Não poderia esquecer dos amigos que carrego pela vida e daqueles que fiz durante esse processo, com quem sempre caminhei lado a lado. Agradeço ao Marcos, com quem estive junto desde o mestrado e sempre tenho algum tema para compartilhar, seja sobre futebol, política ou religião. Lembro ainda dos amigos que compartilham as aflições do mestrado/doutorado, Romário, Jeverson, Warley, Alexandre, Fernando, Fábio, entre outros, e dos colegas da sala de estudos do PPGMMC, com os quais sempre tínhamos conselhos, ideias e papos furados quando a sala ainda era habitada (Wesley, Glender, Alexandre, entre outros). E a todos aqueles que encontrei nesse processo e que, de certa forma, deixaram suas marcas.

Aos meus colegas de trabalho, seja nos Correios, na UFV ou na PUC Minas. Colegas com os quais dividimos os dias, as aflições e as glórias. Citar nomes poderia me fazer ser injusto e esquecer alguém, mas aqui gostaria de agradecer a uma pessoa com um dos maiores corações que encontrei, que me chamava de doutor e que também não pôde me ver chegar ao final dessa jornada. Obrigado por lutar por nós, Dani.

Agradeço a Deus, que se fez presente de diferentes formas ao longo desta caminhada: no amor, na fé, na esperança, na força interior e nas pessoas que estiveram ao meu lado. Por fim, deixo meu agradecimento a todos que, de certa forma, auxiliaram direta ou indiretamente nesta etapa. Obrigado a todos vocês.



Resumo

Esta tese propõe mecanismos de aprendizado por reforço integrados ao *framework* multiagente AMAM (Arquitetura MultiAgente para Metaheurísticas) para a solução de problemas de otimização combinatória usando meta-heurísticas. Todas as contribuições são validadas experimentalmente em instâncias de referência do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), problema combinatório clássico e amplamente estudado na literatura. Inicialmente, apresenta-se uma análise do *framework* AMAM, investigando-se seu comportamento sob diferentes configurações de escalabilidade, diversidade e cooperação entre agentes, estabelecendo, assim, as bases e parâmetros que orientam os desenvolvimentos subsequentes. Os experimentos confirmaram, via testes estatísticos não paramétricos, que o AMAM é escalável, com o aumento do número de agentes melhorando a qualidade das soluções sem comprometer significativamente o tempo de execução, e que suas ferramentas de cooperação e de diversidade melhoram os valores da função objetivo em relação à configuração de referência da literatura. Em seguida, propõe-se o algoritmo MaB *Pool*, que integra o método *Multi-armed Bandit* à estrutura de cooperação do AMAM: cada solução do *Pool* de Soluções compartilhado desse *framework* é tratada como um braço de um problema de *bandit stateless*, e a seleção é guiada por uma política de aprendizado que equilibra diversificação e intensificação, demonstrando ganhos de qualidade em relação às políticas de seleção sem aprendizado. Os testes computacionais mostraram que o MaB *Pool* superou tanto o algoritmo de referência quanto o cenário sem aprendizado, em valor da função objetivo e em tempo de execução. Em continuidade, propõe-se o algoritmo MAMaB-PSO (*Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation*), algoritmo multiagente que incorpora os componentes cognitivo e social da meta-heurística PSO (*Particle Swarm Optimization*) ao processo de seleção do *Pool* de Soluções, permitindo que cada agente combine sua melhor experiência individual com a melhor experiência coletiva do sistema, em um equilíbrio dinâmico entre autonomia e cooperação; o algoritmo é formulado de forma genérica, instanciável com diferentes políticas de seleção. Os experimentos mostraram que a incorporação dos mecanismos do algoritmo PSO preservou a qualidade das soluções, com equivalência estatística validada, e introduziu ganhos computacionais relevantes em duas variantes, identificadas como Pareto-dominantes do experimento. Por fim, estende-se o MaB *Pool* ao cenário multiobjetivo, substituindo o Q-valor escalar de cada solução por um vetor de recompensas avaliado por dominância de Pareto, de modo que critérios como qualidade da solução, diversidade estrutural e redução do número de rotas sejam aprendidos simultaneamente pela camada de cooperação, sem alterar a modelagem mono-objetivo do problema subjacente. Os experimentos indicaram que a formulação multiobjetivo preserva a qualidade final das soluções, com equivalência

estatística validada, e reduz o tempo de execução nas classes mais desafiadoras de instâncias do PRVJT.

Palavras-chave: Sistemas Multiagente, Aprendizado por Reforço Multiagente, Aprendizado por Reforço Multiobjetivo, *Multi-armed Bandit*, *Particle Swarm Optimization*, Meta-heurísticas, *Framework AMAM*.

Abstract

This thesis proposes reinforcement learning mechanisms integrated into the AMAM (Multi-Agent Architecture for Metaheuristics) multi-agent framework for solving combinatorial optimization problems using metaheuristics. All contributions are experimentally validated using benchmark instances of the Vehicle Routing Problem with Time Windows (VRPTW), a classic combinatorial problem widely studied in the literature. Initially, an analysis of the AMAM framework is presented, investigating its behavior under different configurations of scalability, diversity, and cooperation between agents, thus establishing the basis and parameters that guide subsequent developments. Experiments confirmed via non-parametric statistical tests that AMAM is scalable, with increasing the number of agents improving the quality of solutions without significantly compromising runtime, and that its cooperation and diversity tools improve objective function values compared to the reference configuration from the literature. Subsequently, the MaB Pool algorithm is proposed, which integrates the Multi-armed Bandit method into the AMAM cooperation framework: each solution in the shared Solution Pool of this framework is treated as an arm of a stateless bandit problem, and selection is guided by a learning policy that balances diversification and intensification, demonstrating quality gains compared to selection policies without learning. Computational tests demonstrated that MaB Pool outperformed both the baseline algorithm and the scenario without learning in terms of objective function values and runtime. Furthermore, the MAMaB-PSO algorithm (Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation) is proposed, a multi-agent algorithm that incorporates the cognitive and social components of the PSO (Particle Swarm Optimization) meta-heuristic into the Solution Pool selection process, enabling each agent to combine its best individual experience with the best collective experience of the system in a dynamic balance between autonomy and cooperation. The algorithm is formulated generically, allowing for instantiation with different selection policies. Experiments showed that incorporating the PSO algorithm mechanisms preserved the quality of the solutions, with validated statistical equivalence, and introduced relevant computational gains in two variants, identified as Pareto-dominant in the experiment. Finally, the MaB Pool is extended to the multi-objective scenario by replacing the scalar Q-value of each solution with a reward vector evaluated based on Pareto dominance, so that criteria such as solution quality, structural diversity, and reduction in the number of routes are learned simultaneously by the cooperation layer, without altering the single-objective modeling of the underlying problem. Experiments indicated that the multi-objective formulation preserves the final quality of the solutions, with validated statistical equivalence, and reduces execution time for the most challenging classes of VRPTW instances.

Keywords: Multi-Agent Systems, Multi-Agent Reinforcement Learning, Multi-Objective Reinforcement Learning, Multi-Armed Bandit, Particle Swarm Optimization, Meta-heuristics, AMAM Framework.

Sumário

Lista de Tabelas	xv
Lista de Figuras	xviii
Lista de Algoritmos	xix
Lista de Abreviaturas e Siglas	xx
1 Introdução	1
1.1 Apresentação Geral	1
1.2 Justificativa	4
1.3 Objetivos	5
1.3.1 Objetivo Geral	5
1.3.2 Objetivos Específicos	5
1.4 Metodologia de Pesquisa	6
1.4.1 Análise do <i>Framework</i> AMAM	6
1.4.2 Aprendizado por reforço no <i>Framework</i> AMAM	7
1.5 Contribuições da Tese	8
1.6 Estrutura da Tese	9
2 Fundamentação	11
2.1 <i>Frameworks</i> para Otimização	11
2.1.1 <i>Framework</i> AMAM	12
2.2 Sistemas Multiagente	17
2.2.1 Cooperação	18
2.2.2 Diversidade	19
2.2.3 Escalabilidade	20
2.3 Meta-heurísticas e Aprendizado de Máquina	24
2.3.1 Aprendizado por Reforço	26
2.3.2 Algoritmo <i>Multi-armed Bandit</i>	29
2.3.3 Aprendizado por Reforço Multiagente	34
2.3.4 Algoritmo <i>Multi-armed Bandit</i> Multiagente	35
2.3.5 Aprendizado por Reforço Multiobjetivo	37
2.3.6 Algoritmo <i>Multi-armed Bandit</i> Multiobjetivo	39
2.4 Considerações Parciais: <i>Q-learning</i> e <i>Multi-armed Bandit</i>	41
3 Descrição dos Experimentos	43
3.1 Problema de Roteamento de Veículo com Janela de Tempo	43
3.1.1 Estruturas de vizinhança do PRVJT	44
3.1.2 Perturbações para o PRVJT	45
3.2 Agentes implementados	46

3.3	Dados comuns dos Experimentos	50
4	Análise do <i>Framework</i> AMAM	53
4.1	Escalabilidade	53
4.1.1	Definição da Experimentação	54
4.1.2	Resultados Computacionais	55
4.1.3	Análise estatística	60
4.1.4	Ajuste Linear	62
4.1.5	Agente Único × Multiagente	64
4.1.6	Considerações Parciais	67
4.2	Cooperação	67
4.2.1	Definição da Experimentação	68
4.2.2	Resultados Computacionais	69
4.2.3	Considerações Parciais	72
4.3	Diversidade	75
4.3.1	Definição da Experimentação	76
4.3.2	Resultados Computacionais – Diversidade	77
4.3.3	Considerações Parciais	81
4.4	Modelagem de Recompensas	82
4.4.1	Definição da Experimentação	84
4.4.2	Resultados Computacionais	85
4.4.3	Considerações Parciais	87
5	Aprendizado por Reforço Multiagente no <i>Framework</i> AMAM	88
5.1	<i>Multi-armed Bandit Pool</i>	88
5.1.1	Algoritmo <i>Multi-armed Bandit Pool</i>	90
5.1.2	Alterações na coleta de soluções	90
5.1.3	Alterações na exclusão de soluções	91
5.1.4	Funcionamento do <i>Multi-armed Bandit Pool</i>	91
5.1.5	Definição da Experimentação	92
5.1.6	Resultados Computacionais	93
5.1.7	Considerações Parciais	101
5.2	<i>Multi-armed Bandit</i> Multiagente	102
5.2.1	Modelagem do sistema multiagente	103
5.2.2	Cenários de recompensa	103
5.2.3	Aprendizado multiagente em ambiente não estacionário	104
5.2.4	Aprendizado cooperativo via <i>Pool</i> de soluções	104
5.2.5	Algoritmo MAMaB-PSO	106
5.2.6	Variantes da Política de Seleção MAMaB-PSO	107
5.2.7	Definição da Experimentação	110
5.2.8	Comparação entre os Cenários das Famílias de Políticas	111
5.2.9	Análise das Variantes do MAMaB-PSO	118
5.2.10	Considerações Parciais	127

6	Aprendizado por Reforço Multiobjetivo no <i>Framework</i> AMAM	129
6.1	<i>Multi-armed Bandit</i> Multiobjetivo	129
6.1.1	Vetor de recompensa social	130
6.1.2	Estrutura vetorial associada às soluções	131
6.1.3	Variantes da Política de Seleção Multiobjetivo	132
6.1.4	Algoritmo <i>Multi-armed Bandit</i> Multiobjetivo	134
6.1.5	Definição da Experimentação	134
6.1.6	Resultados Computacionais	135
6.1.7	Considerações Parciais	143
6.2	<i>Pool</i> de Soluções Multiobjetivo	144
6.2.1	Descrição do <i>Pool</i> Multiobjetivo	144
6.2.2	Definição da Experimentação	147
6.2.3	Resultados Computacionais	149
6.2.4	Considerações Parciais	156
7	Considerações Finais e Direções Futuras	158
7.1	Considerações Finais	158
7.2	Publicações derivadas desta Pesquisa	162
7.2.1	Artigo Apresentado em Evento	162
7.2.2	Artigo Publicado em Revista	163
7.2.3	Artigo Submetido a Revista	163
7.3	Proposta de Continuidade	163
	Referências Bibliográficas	166

Lista de Tabelas

2.1	Algoritmos <i>Multi-armed Bandit</i>	31
3.1	Características gerais dos agentes implementados	49
4.1	Valor de função objetivo de cada instância avaliada do PRVJT $\times$ Número de Agentes. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. O destaque é feito por linha, entre os dez cenários de número de agentes; a coluna BKV traz o melhor valor conhecido na literatura e não entra na comparação.	56
4.2	Tempo de execução médio $\times$ Número de agentes	57
4.3	Análise dos resultados apresentados nas Tabelas 4.1 e 4.2	58
4.4	Função Objetivo e <i>gap</i> (%) ILSN $\times$ ILSBI	70
4.5	Tempo médio de execução (s) ILSN $\times$ ILSBI	71
4.6	Função Objetivo e <i>gap</i> (%) do cenário ILSN com o tempo de execução como critério de parada	74
4.7	Função Objetivo e <i>gap</i> (%)	78
4.8	Tempo médio de execução (s)	79
4.9	Teste 2 a 2 - Função Objetivo	82
4.10	Teste 2 a 2 - Tempo de Execução	83
4.11	<i>Gap</i> (%) médio para as instâncias testadas. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. O destaque é feito por linha, entre as quatro colunas de número de agentes. Valores negativos indicam resultados melhores que os de Silva et al. (2019), tomados como referência no cálculo do <i>gap</i> apresentado no início desta seção.	85
5.1	Função Objetivo e <i>gap</i> (%) - <i>Multi-armed Bandit</i>	94
5.2	Tempo de execução <i>Multi-armed Bandit Pool</i>	95
5.3	Ajuste de parâmetros <i>Multi-armed Bandit Pool</i> - Função objetivo	96
5.4	Ajuste de parâmetros <i>Multi-armed Bandit Pool</i> - Tempo de Execução	97
5.5	<i>Framework AMAM</i> com <i>Multi-armed Bandit Pool</i> - Função objetivo	99
5.6	<i>Framework AMAM</i> com <i>Multi-armed Bandit Pool</i> - Tempo de Execução	100
5.7	Melhores soluções por Cenários - <i>Multi-armed Bandit Pool</i>	102
5.8	Cenários experimentais avaliados, organizados por família de política. Todos os cenários utilizam a mesma composição de agentes, ou seja, ILS_MIX com 20 agentes, descrita na Seção 5.2.7.	110
5.9	Parâmetros das políticas MaB utilizados nos experimentos.	112

5.10	Média do <i>gap</i> (%) por instância: comparação entre os cenários representativos. Os valores após $\pm$ são o desvio-padrão das 30 execuções independentes de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	113
5.11	Média do <i>gap</i> (%) por bloco de instâncias e cenário. Os valores após $\pm$ são o desvio-padrão das 30 execuções independentes de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	113
5.12	Menor <i>gap</i> (%) por instância e cenário (melhor solução entre as 30 execuções). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. A linha <i>Vitórias</i> indica em quantas instâncias cada cenário atinge o menor <i>gap</i> (empates incluídos).	115
5.13	Média do tempo de execução (s) por instância: comparação entre os cenários representativos. Os melhores valores são os menores e estão destacados em negrito	117
5.14	Média do tempo de execução (s) por bloco de instâncias e cenário. . .	117
5.15	Média do <i>gap</i> (%) por instância e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	119
5.16	Média do <i>gap</i> (%) por bloco de instâncias e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	120
5.17	Menor <i>gap</i> (%) por instância e configuração (melhor solução entre as 30 execuções). A linha <i>Vitórias</i> indica em quantas instâncias cada configuração atinge o menor <i>gap</i> (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	121
5.18	Média do tempo de execução (s) por instância e configuração.	122
5.19	Média do tempo de execução (s) por bloco de instâncias e configuração.	123
6.1	Cenários experimentais do MaB Multiobjetivo. Todos os cenários utilizam a mesma composição de agentes, o ILS_MIX com 20 agentes, descrita na Seção 5.2.7.	135
6.2	Parâmetros das políticas MaB Multiobjetivo.	135
6.3	Média do <i>gap</i> (%) por instância e configuração. Os valores após $\pm$ são o desvio-padrão das 30 execuções de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	136
6.4	Média do <i>gap</i> (%) por bloco de instâncias e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. Os valores após $\pm$ são o desvio-padrão calculado sobre todas as execuções das instâncias do bloco e, portanto, refletem também a heterogeneidade entre instâncias, não apenas a variação entre execuções.	137

6.5	Menor <i>gap</i> (%) por instância e configuração (melhor solução entre as 30 execuções). A linha <i>Vitórias</i> indica em quantas instâncias cada configuração atinge o menor <i>gap</i> (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	139
6.6	Média do tempo de execução (s) por instância e configuração.	140
6.7	Média do tempo de execução (s) por bloco de instâncias e configuração.	140
6.8	Parâmetros de manutenção do <i>Pool</i> multiobjetivo.	148
6.9	Cenários experimentais do <i>Pool</i> Multiobjetivo. O prefixo PM indica execução sobre o <i>Pool</i> multiobjetivo (<code>pool_multi</code>); os demais cenários operam sobre o <i>Pool</i> de diversidade (<code>pool_diversity</code>).	148
6.10	Média do <i>gap</i> (%) por instância e cenário. Cenários PM operam sobre o <i>Pool</i> multiobjetivo. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	149
6.11	Média do <i>gap</i> (%) por bloco de instâncias e cenário. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	151
6.12	Menor <i>gap</i> (%) por instância e cenário (melhor solução entre as 30 execuções). A linha <i>Vitórias</i> indica em quantas instâncias cada cenário atinge o menor <i>gap</i> (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.	152
6.13	Média do tempo de execução (s) por instância e cenário. Cenários PM operam sobre o <i>Pool</i> multiobjetivo.	153
6.14	Média do tempo de execução (s) por bloco de instâncias e cenário.	154

Lista de Figuras

2.1	Dimensões do <i>Framework AMAM</i> . Adaptado de Silva (2019).	12
2.2	Relação entre estados (vizinhanças) e ações possíveis (Silva et al., 2019).	14
2.3	Exemplos de cálculo da distância entre as soluções para o PRVJT, conforme Silva et al. (2019)	17
2.4	Modelo de Aprendizado por Reforço (Sutton e Barto, 2018).	26
3.1	Uma solução s para o PRVJT (Silva et al., 2019).	44
3.2	Instanciação do <i>Framework AMAM</i> para o PRVJT com 4 agentes. Note que o agente ILS_ALS_NO não possui acesso ao <i>Pool</i> de soluções.	50
4.1	Comportamento do <i>gap</i> médio em relação às melhores soluções conhecidas, à média das soluções e à mediana das soluções para os 10 cenários nas 30 execuções.	58
4.2	Tempo de execução médio $\times$ número de agentes.	58
4.3	Comportamento do <i>Framework AMAM</i> ao resolver a instância R210 do PRVJT em relação à função objetivo e valores de tempo de execução $\times$ Número de agentes.	59
4.4	Comparação dos valores médios da função objetivo (<i>gap</i> (%)) e do tempo de execução em relação ao aumento no número de agentes.	60
4.5	Comportamento do <i>gap</i> em uma escala logarítmica.	62
4.6	Regressão Linear para Valores de Função Objetivo.	63
4.7	Regressão Linear para Valores de Tempo de Execução	64
4.8	Agente único $\times$ Multiagente.	66
4.9	Agente único $\times$ Multiagente tempo de execução (C - Cluster).	67
4.10	ILSN $\times$ ILSBI $\times$ ILSNMIX	72
4.11	Evolução ILSN $\times$ ILSBI $\times$ ILSNMIX - instância RC108 com 4 agentes	73
4.12	Cenário ILSN e critério de parada: iterações sem melhora $\times$ tempo de execução, <i>gap</i> (%)	74
4.13	Diversidade: <i>Gap</i> (%) e tempo de execução.	80
4.14	Evolução diversidade - instância RC108 com 4 agentes	81
4.15	<i>Gap</i> (%) médio por recompensa e por número de agentes.	86
4.16	Evolução das modelagens de recompensas - instância C208.	86
5.1	<i>Gap</i> (%) e Tempo de Execução - <i>Multi-armed Bandit Pool</i>	98
5.2	Evolução <i>Multi-armed Bandit Pool</i> - instância R112 com 4 agentes	99
5.3	Distribuição do <i>gap</i> (%) por cenário ao longo das 24 instâncias avaliadas.	114
5.4	Distribuição do <i>gap</i> (%) por cenário e bloco de instâncias (C, R e RC).	114

5.5	Número de instâncias em que cada cenário encontra a melhor solução (menor <i>gap</i>), separando vitórias estritas (único mínimo) de empates. .	116
5.6	Distribuição do tempo de execução (s) por cenário ao longo das 24 instâncias avaliadas.	116
5.7	Distribuição do tempo de execução (s) por cenário e bloco de instâncias (C, R e RC).	118
5.8	Distribuição do <i>gap</i> (%) por configuração ao longo das 24 instâncias avaliadas.	119
5.9	Distribuição do <i>gap</i> (%) por configuração e bloco de instâncias (C, R e RC).	120
5.10	Número de instâncias em que cada configuração encontra a melhor solução (menor <i>gap</i>), separando vitórias estritas (único mínimo) de empates.	121
5.11	Distribuição do tempo de execução por configuração ao longo das 24 instâncias avaliadas.	123
5.12	Distribuição do tempo de execução (s) por configuração e bloco de instâncias (C, R e RC).	124
5.13	Evolução temporal do <i>gap</i> em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem ao MULTI-EG (vermelho) e ao PSO-DUCB (laranja); as demais variantes PSO são mostradas em cinza. As faixas sombreadas indicam o intervalo interquartilico (IQR) entre as 30 execuções independentes. O eixo <i>x</i> está em escala logarítmica.	125
5.14	Perfil de desempenho (Dolan e Moré, 2002) dos sete métodos avaliados nas 24 instâncias de Solomon. O eixo <i>x</i> representa a razão de desempenho τ em relação ao melhor método em cada instância; o eixo <i>y</i> indica a fração das instâncias em que o método possui razão $\leq \tau$. .	127
6.1	Distribuição do <i>gap</i> (%) por configuração ao longo das 24 instâncias avaliadas.	137
6.2	Distribuição do <i>gap</i> (%) por configuração e bloco de instâncias (C, R e RC).	138
6.3	Número de instâncias em que cada configuração encontra a melhor solução (menor <i>gap</i>), separando vitórias estritas (único mínimo) de empates.	139
6.5	Distribuição do tempo de execução (s) por configuração e bloco de instâncias (C, R e RC).	140
6.4	Distribuição do tempo de execução (s) por configuração ao longo das 24 instâncias avaliadas.	141
6.6	Evolução temporal do <i>gap</i> em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem ao PSO-DUCB (laranja, melhor multiagente) e ao MO-PSO-DPUCB (roxo, melhor multiobjeto); as demais variantes MO são mostradas em cinza. As faixas sombreadas indicam o IQR entre as 30 execuções independentes. O eixo <i>x</i> está em escala logarítmica. . .	142

6.7	Perfil de desempenho (Dolan e Moré, 2002) dos seis métodos avaliados nas 24 instâncias de Solomon. O eixo x representa a razão de desempenho τ ; o eixo y indica a fração das instâncias em que o método possui razão $\leq \tau$	143
6.8	Distribuição do <i>gap</i> (%) por cenário ao longo das 24 instâncias avaliadas. Cenários PM operam sobre o <i>Pool</i> multiobjetivo.	150
6.9	Distribuição do <i>gap</i> (%) por cenário e bloco de instâncias (C, R e RC).	151
6.10	Número de instâncias em que cada cenário encontra a melhor solução (menor <i>gap</i>), separando vitórias estritas (único mínimo) de empates.	152
6.11	Distribuição do tempo de execução (s) por cenário ao longo das 24 instâncias avaliadas.	154
6.12	Distribuição do tempo de execução (s) por cenário e bloco de instâncias (C, R e RC).	154
6.13	Evolução temporal do <i>gap</i> em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem aos pares PSO-DUCB/PM-PSO-DUCB e MO-PSO-DPUCB/PM-MO-PSO-DPUCB; as demais variantes são mostradas em cinza. As faixas sombreadas indicam o IQR entre as 30 execuções independentes. O eixo x está em escala logarítmica.	155
6.14	Perfil de desempenho (Dolan e Moré, 2002) dos nove cenários avaliados nas 24 instâncias de Solomon. O eixo x representa a razão de desempenho τ ; o eixo y indica a fração das instâncias em que o cenário possui razão $\leq \tau$	156

Lista de Algoritmos

1	Busca Local Adaptativa - <i>ALS-QLearning</i> (Silva, 2019)	15
2	<i>Um algoritmo bandit simples</i>	30
3	Busca Local Iterada (<i>Iterated Local Search – ILS</i>)	47
4	Descida em Vizinhança Variável (<i>Variable Neighborhood Descent – VND</i>)	47
5	Descida pela Melhor Melhoria (<i>Best Improvement Descent</i>)	48
6	Descida pela Primeira Melhoria (<i>First Improvement Descent</i>)	48
7	Descida pela Melhoria Aleatória (<i>Random Improvement Descent</i>) . .	48
8	<i>Multi-armed Bandit Pool</i>	92
9	MAMaB-PSO - <i>Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation</i>	106
10	<i>Multi-armed Bandit</i> Multiobjetivo no <i>Pool</i> da AMAM	134
11	Pool Multiobjetivo	145
12	Interação Agente – <i>Pool</i> de soluções	146

Lista de Abreviaturas e Siglas

ACO	<i>Ant Colony Optimization</i> , Otimização por Colônia de Formigas
AMAM	Arquitetura MultiAgente para Metaheurística
BKV	<i>Best Known Value</i> , melhor valor conhecido na literatura para a instância
C, R, RC	grupos de instâncias de Solomon: <i>Cluster</i> , <i>Random</i> e <i>Random Cluster</i>
DUCB	<i>Discounted Upper Confidence Bound</i> , UCB com desconto
EG	ϵ -greedy, política ϵ -gulosa de seleção
FO	Função Objetivo, valor reportado nas tabelas de resultados
FRAMaB	<i>Fitness-Rate-Average based Multi-armed Bandit</i>
FRRMaB	<i>Fitness-Rate-Rank based Multi-armed Bandit</i>
GA	<i>Genetic Algorithm</i> , Algoritmo Genético
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
ILS	<i>Iterated Local Search</i> , Busca Local Iterada
ILS _ ALS	<i>Iterated Local Search com Adaptive Local Search</i> , agente base utilizado nos experimentos
IQR	<i>Interquartile Range</i> , intervalo interquartilico
MaB	<i>Multi-armed Bandit</i>
MaB-OS	<i>Multi-armed Bandit Operator Selection</i> , seleção de operador por MaB
MAMaB-PSO	<i>Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation</i>
MARL	<i>Multi-Agent Reinforcement Learning</i> , Aprendizado por Reforço Multiagente
MAS	<i>Multiagent System</i> , Sistema Multiagente
MH	Meta-heurística
ML	<i>Machine Learning</i> , Aprendizado de Máquina
MO	Multiobjetivo, prefixo das variantes multiobjetivo
MOF	<i>Metaheuristic Optimization Framework</i>
MOMaB	<i>Multi-Objective Multi-armed Bandit</i>
MOMDP	<i>Multi-Objective Markov Decision Process</i>
MORL	<i>Multi-Objective Reinforcement Learning</i>
PM	<i>Pool</i> Multiobjetivo, prefixo dos cenários executados sobre o <i>Pool</i> multiobjetivo

PQL	<i>Pareto Q-learning</i>
PRVJT	Problema de Roteamento de Veículos com Janela de Tempo
PSO	<i>Particle Swarm Optimization</i> , Otimização por Enxame de Partículas
RL	<i>Reinforcement Learning</i> , Aprendizado por Reforço
SA	<i>Simulated Annealing</i> , Recozimento Simulado
SHI	critério de exclusão de solução do <i>Pool</i> pela menor diversidade $g(\phi_i)$, função de nicho definida na Expressão (2.1)
SL	<i>Supervised Learning</i> , Aprendizado Supervisionado
SWUCB	<i>Sliding-Window Upper Confidence Bound</i> , UCB com janela deslizando
TS	<i>Tabu Search</i> , Busca Tabu
UCB	<i>Upper Confidence Bound</i> , limite superior de confiança
UL	<i>Unsupervised Learning</i> , Aprendizado não Supervisionado
VND	<i>Variable Neighborhood Descent</i>
VNS	<i>Variable Neighborhood Search</i>
VRPTW	<i>Vehicle Routing Problem with Time Windows</i> , denominação em inglês do PRVJT

Convenção de nomenclatura dos cenários e variantes experimentais. Os nomes dos cenários são compostos a partir das siglas acima e, por isso, não constam desta lista. O prefixo **MO-** indica a versão multiobjetivo de uma variante e o prefixo **PM-** indica execução sobre o *Pool* multiobjetivo. Os sufixos **-EG**, **-UCB**, **-DUCB**, **-SWUCB**, **-FRR** e **-FRA** indicam a política de seleção empregada (respectivamente *ϵ -greedy*, UCB, DUCB, SWUCB, FRRMaB e FRAMaB); **-DPUCB** designa o DUCB adaptado ao caso multiobjetivo e **-mPQL** a versão *stateless* do *Pareto Q-learning*. As famílias **MONO**, **MULTI** e **HYBRID** são as configurações de referência sem o componente PSO. Nos cenários do *MaB Pool*, **MAX** e **SOFT** designam a forma de coleta da solução (*qmax* e *softmax*) e **SHI** e **Q** o critério de exclusão (menor diversidade e menor valor q), como em **MAXSHI**, **MAXQ**, **SOFTSHI** e **SOFTQ**. Os cenários **ILSN**, **ILSBI**, **ILSFI**, **ILSRI**, **ILSMI** e **ILSNMIX**, assim como os agentes **ILS_ALS_NO**, **ILS_ALS_BI**, **ILS_ALS_FI** e **ILS_ALS_RI**, referem-se às configurações de agentes **ILS_ALS** empregadas em cada experimento (N/NO: sem acesso ao *Pool*; BI, FI e RI: busca local *Best*, *First* e *Random Improvement*; MI: mistura de agentes; NMIX: configuração parcialmente cooperativa, em que metade dos agentes acessa o *Pool* e a outra metade não).

Capítulo 1

Introdução

1.1 Apresentação Geral

Um sistema multiagente pode ser definido como um sistema composto por vários agentes individuais introduzidos em um ambiente de busca (Wooldridge, 2009). Esses agentes possuem características distintas e agem de maneiras diversas. Na concepção desse sistema, é possível incluir habilidades como cooperação, coordenação, competição e negociação entre os agentes. Os benefícios da utilização desses sistemas incluem capacidade de solução de problemas complexos, rapidez na solução de problemas, flexibilidade de implementação, escalabilidade e o aumento da capacidade de resposta. Um agente é uma unidade computacional situada em um ambiente específico, cada qual com um objetivo próprio e a capacidade de agir de forma autônoma em relação a esse objetivo. Embora esses agentes possuam comportamentos individuais, o sistema pode ser projetado para que atuem de forma articulada em torno de um objetivo comum, seja cooperando, seja competindo entre si (Balaaji e Srinivasan, 2010). A modelagem baseada em agentes é uma forma de simular sistemas complexos, criando agentes individuais que seguem regras e, em grande parte, trabalham de forma cooperativa. De acordo com Cuevas et al. (2024), há um crescente interesse no estudo da utilização da modelagem baseada em agentes para solução de problemas de otimização. Os autores destacam que essa abordagem pode ser aplicada a uma ampla gama de problemas complexos, permitindo a obtenção de boas respostas nesses ambientes de busca e gerando soluções eficazes para problemas difíceis.

Atualmente, a comunidade científica voltada para a resolução de problemas de otimização combinatória, especialmente aquelas que se valem de meta-heurísticas, busca integrar sistemas multiagente através de *frameworks* específicos para essa área de pesquisa. Esses *frameworks* são instrumentos que estruturam e simplificam a implementação de códigos, reunindo conjuntos genéricos e reutilizáveis, tornando-se, assim, eficazes na aplicação de métodos de solução. A adoção de tais *frameworks* para resolver problemas de otimização tem crescido consideravelmente entre os pesquisadores. Trabalhos como os de Parejo et al. (2012) e Silva et al. (2018) destacam essa demanda crescente. Esses sistemas têm sido escolhidos devido à facilidade de programação e à possibilidade de reutilização de métodos já disponíveis nessas estruturas, agilizando a implementação e facilitando a criação de métodos de solução híbridos. Em sua revisão, Silva et al. (2018) exploram essas ferramentas, detalhando suas características e capacidades, com foco especial em *frameworks* baseados em

sistemas multiagente.

A partir da análise de [Silva et al. \(2018\)](#), destaca-se a utilização do *Framework* AMAM. Esse *Framework*, cujo acrônimo significa Arquitetura MultiAgente para Metaheurística, foi proposto inicialmente por [Silva \(2007\)](#), e é um *framework* fundamentado no conceito de sistemas multiagente, direcionado à resolução de problemas de otimização combinatória por meio da implementação de meta-heurísticas na forma de agentes. Essa estrutura permite a implementação de meta-heurísticas híbridas através da disponibilização de códigos prontos na estrutura. O *framework* AMAM também possui um mecanismo de cooperação entre os agentes, no qual soluções são compartilhadas durante o processo de busca e assegura a escalabilidade da estrutura mediante à facilidade de inclusão de novos agentes. Estratégias de aprendizado por reforço foram incorporadas ao *framework*, conforme descrito por [Silva et al. \(2019\)](#); [Silva \(2019\)](#).

Assim como em todo sistema computacional, os sistemas multiagente enfrentam desafios de implementação e gerenciamento. Em muitos casos, esses sistemas precisam aprimorar a autonomia e o aprendizado dos agentes. Além disso, é crucial o gerenciamento e a troca de conhecimento entre os agentes, e técnicas de aprendizado de máquina podem ser aplicadas para facilitar e aprimorar a implementação desses resultados. No *Framework* AMAM, a autonomia é efetivamente explorada: cada agente executa sua própria meta-heurística em uma linha de execução independente e decide localmente o curso de sua busca, sem coordenação centralizada, cabendo ao *Pool* de soluções ser o único ponto de contato entre eles. Essa autonomia é ampliada pelo aprendizado por reforço embarcado no agente, que auxilia na melhor definição da ordem das estruturas de vizinhança a explorar, sendo esse um dos aspectos centrais investigados nesta tese, em particular no tocante à modelagem das recompensas.

Na literatura atual, diversos estudos exploram a integração entre meta-heurísticas e aprendizado de máquina. [Karimi-Mamaghan et al. \(2022\)](#) apresentam uma análise sobre essa união, focando na resolução de problemas de otimização. Eles demonstram a versatilidade desses algoritmos em vários cenários, como na seleção de operadores em algoritmos genéticos, ajuste de parâmetros em meta-heurísticas e seleção de etapas em hiper-heurísticas, dentre outros. Em um contexto de modelagem baseada em agentes, cada meta-heurística é capaz de implementar seu próprio algoritmo de aprendizado. Nesse sentido, as técnicas de aprendizado por reforço destacam-se, pois um agente aprende seu comportamento através de tentativas e erros, interagindo em um ambiente dinâmico. Ao contrário do aprendizado supervisionado, que é altamente específico para o problema tratado, um algoritmo de aprendizado por reforço requer apenas um modelo de recompensas pré-estabelecido.

A cooperação entre agentes em um sistema multiagente permite o compartilhamento, a coordenação e a tomada de decisões conjuntas. De acordo com [Wooldridge \(2009\)](#), a maioria dos sistemas multiagente que operam com cooperação parte do pressuposto da benevolência entre os agentes, ou seja, compartilham um objetivo comum, evitando potenciais conflitos. No contexto de um sistema multiagente cooperativo, o aprendizado pode ocorrer de várias maneiras, e diversos aspectos da cooperação devem ser analisados. Uma das formas é o compartilhamento de soluções ([Wooldridge, 2009](#)), que envolve a troca de soluções ou partes delas durante o processo de busca.

Um desafio relevante em estruturas de cooperação é a gestão das soluções a serem compartilhadas entre os agentes. Nesse contexto, os métodos de aprendizado por reforço podem se revelar ferramentas fundamentais. Algoritmos amplamente difundi-

dos na literatura, como o *Q-learning* e o *Multi-armed Bandit* (MaB), podem atuar de maneira eficaz nessas estruturas, otimizando e agilizando seu funcionamento. Ainda há questões em aberto sobre como lidar com esses problemas, especialmente quando consideramos o aprendizado por reforço em múltiplos agentes. Dentre estas questões podemos suscitar:

- (i) Será suficiente cooperar nas soluções, compartilhar informações e aprendizado em um sistema multiagente para resolver problemas de otimização?
- (ii) Qual a influência destas técnicas na qualidade das soluções? E, finalmente:
- (iii) Qual a influência destas técnicas no tempo de execução?

Embora o algoritmo *Multi-armed Bandit* tenha sido amplamente estudado no contexto de um único agente tomador de decisão, a extensão para cenários multiagente introduz desafios adicionais relevantes. Quando múltiplos agentes interagem simultaneamente com o mesmo conjunto de alternativas, como ocorre no compartilhamento de soluções em um *Pool* cooperativo, as estimativas individuais tendem a ser redundantes e o processo de exploração passa a ser descoordenado entre os agentes (Liu e Zhao, 2010). Formulações cooperativas do MaB, como o *Cooperative UCB* (*Upper Confidence Bound*) proposto por Landgren et al. (2021), buscam reduzir essa redundância por meio do compartilhamento de informações, de modo a acelerar a convergência do sistema. Entretanto, um desafio central nessas formulações é definir como cada agente deve ponderar sua própria experiência acumulada em relação ao conhecimento coletivo produzido pelo grupo (Landgren et al., 2021). Priorizar exclusivamente o aprendizado individual pode gerar comportamentos especializados, porém redundantes entre agentes; ao contrário, privilegiar apenas a informação coletiva pode suprimir a diversidade e induzir convergência prematura. A meta-heurística *Particle Swarm Optimization* (PSO) (Kennedy e Eberhart, 1995) oferece uma perspectiva relevante para esse dilema: no PSO, cada partícula ajusta sua trajetória a partir de sua melhor experiência histórica individual e da melhor experiência observada pelo grupo, promovendo um equilíbrio dinâmico entre comportamento individual e cooperação coletiva. A incorporação desses componentes cognitivo e social ao processo decisório do MaB multiagente configura, portanto, uma direção promissora para sistemas multiagente de otimização combinatória. No entanto é importante salientar que as partículas não possuem autonomia decisória.

Na estrutura de cooperação de um *framework* multiagente, o *Pool* de soluções é tipicamente gerenciado por critérios escalares de qualidade, nos quais cada solução armazenada recebe um único valor de recompensa que reflete sua utilidade histórica para os agentes. Entretanto, a gestão eficiente de um *Pool* cooperativo envolve múltiplos aspectos simultaneamente, tais como a qualidade da solução e sua diversidade em relação às demais soluções armazenadas, que podem ser conflitantes entre si. Um único valor escalar pode não ser suficiente para capturar essa multiplicidade de critérios relevantes para a cooperação (Roijers et al., 2013). Nesse contexto, extensões multiobjetivo do aprendizado por reforço têm sido propostas na literatura para tratar cenários em que a recompensa associada a cada alternativa é descrita por um vetor de valores (Drugan e Nowé, 2013; Van Moffaert e Nowé, 2014; Huanca-Anquise et al., 2023). Nesses modelos, critérios de dominância de Pareto substituem as comparações

escalares, permitindo que o agente avalie e selecione alternativas considerando múltiplos critérios de forma simultânea. A extensão do *Pool* de soluções e do algoritmo MaB para o contexto multiobjetivo representa, portanto, uma contribuição relevante para o aprimoramento da dimensão de cooperação em *frameworks* multiagente voltados à resolução de problemas de otimização. Cabe destacar que essa formulação multiobjetivo é também multiagente, pois opera sobre o mesmo *Pool* de soluções compartilhado por todos os agentes do *framework*, de modo que o aprendizado vetorial ocorre na dimensão social da estrutura e não isoladamente em cada agente.

Considerando o que foi apresentado, este trabalho propõe estender o aprendizado por reforço em um *framework* multiagente para a resolução de problemas de otimização utilizando meta-heurísticas, abrangendo as dimensões mono-agente, multiagente e multiobjetivo. O *framework* escolhido para o desenvolvimento desta tese é o AMAM. Para alcançar tal objetivo, se faz necessário analisar as principais características do *framework* escolhido, como escalabilidade, diversidade, cooperação e aprendizado, com o intuito de investigar as possibilidades de implementação de técnicas de aprendizado individual e social nessa estrutura. Como problema-instância, esta tese utiliza o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), um problema NP-difícil amplamente discutido na literatura.

1.2 Justificativa

A utilização de *frameworks* multiagente para resolver problemas de otimização combinatória se torna uma opção de grande interesse devido à facilidade de implementação e organização das estruturas de códigos de programação. Cada agente possui visões distintas do espaço de busca do problema, resultando em diferentes formas e tempos de aprendizado e evolução durante o processo de busca. A criação de mecanismos de aprendizado individual e social para os agentes representa uma ferramenta valiosa, considerando a complexidade na resolução desses problemas. Permitir que os agentes aprendam, seja de forma individual ou social, por meio do compartilhamento de soluções, comunicação ou técnicas de aprendizado, pode otimizar o desempenho do sistema e reduzir o tempo de processamento. Além disso, em cenários com múltiplos agentes interagindo sobre um mesmo *Pool* de soluções, a seleção aleatória das soluções compartilhadas pode gerar exploração redundante e prejudicar a eficiência cooperativa do sistema. O uso de algoritmos de aprendizado por reforço, como o *Multi-armed Bandit*, na gestão desse *Pool* permite que o *framework* identifique, de forma adaptativa, quais soluções contribuem efetivamente para o desempenho coletivo. Por fim, a adoção de critérios multiobjetivo na avaliação das soluções do *Pool*, considerando simultaneamente qualidade e diversidade, amplia a capacidade de cooperação do *framework*, evitando a convergência prematura para regiões restritas do espaço de busca.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho consiste em estudar, implementar e adicionar, na estrutura de um *framework* multiagente para solução de problemas de otimização combinatória, técnicas de aprendizado por reforço, investigando as dimensões mono-agente, multiagente e multiobjetivo.

1.3.2 Objetivos Específicos

Com o intuito de alcançar o objetivo principal deste trabalho, se faz necessário a definição dos seguintes objetivos específicos.

- (i) compreender a estrutura e o funcionamento do *framework* AMAM, escolhido como teste para este trabalho, para melhor implementação destas técnicas;
- (ii) avaliar o comportamento do *Framework* AMAM quanto à modelagem de recompensas do seu algoritmo de aprendizagem por reforço (*Q-learning*) contido nas buscas locais dos agentes implementados;
- (iii) analisar a escalabilidade do *Framework* AMAM, avaliando sua capacidade de melhora no desempenho do sistema multiagente à medida que novos agentes são adicionados, sem interferência do desempenho dos agentes individualmente e no processo de cooperação;
- (iv) avaliar o comportamento do *Framework* AMAM quanto à sua cooperação entre agentes. Desta forma, este trabalho busca analisar a dimensão social deste *framework* com o intuito de validá-la;
- (v) verificar a influência da diversificação dos agentes durante o processo de busca, implementando agentes com comportamentos diferentes entre si na estrutura do *framework*;
- (vi) desenvolver e aplicar um algoritmo de aprendizado por reforço baseado no *Multi-armed Bandit* para gerenciar o *Pool* utilizado no compartilhamento de soluções, na estrutura social do *framework*, com o intuito de reduzir o número de passos indesejados que essa cooperação pode acarretar;
- (vii) estudar, implementar e adicionar técnicas de aprendizado por reforço multiagente ao *Framework* AMAM, propondo um mecanismo de cooperação entre agentes que aprendem simultaneamente sobre o mesmo *Pool* de soluções;
- (viii) propor um algoritmo *Multi-armed Bandit* multiobjetivo que opera sobre o *Pool* de cooperação com recompensas vetoriais, substituindo o aprendizado escalar por estimativas vetoriais sensíveis a múltiplos critérios de qualidade e utilizando dominância de Pareto como critério de seleção;

- (ix) estender o *Pool* de soluções para uma formulação multiobjetivo, baseada em dominância de Pareto e *crowding distance*, com vistas a equilibrar simultaneamente qualidade e diversidade entre as soluções mantidas na estrutura de cooperação, oferecendo suporte estrutural ao aprendizado vetorial proposto no objetivo anterior.

1.4 Metodologia de Pesquisa

A metodologia de desenvolvimento desta tese organizou-se em duas etapas complementares. Na primeira etapa, foi realizado um estudo aprofundado dos conceitos fundamentais abordados neste trabalho, seguido de uma análise do estado da arte e de experimentos computacionais voltados à análise do *Framework* AMAM quanto à sua escalabilidade, cooperação, diversidade e modelagem de recompensas no algoritmo de aprendizado. Na segunda etapa, foram implementadas e avaliadas técnicas de aprendizado por reforço na dimensão social do *Framework* AMAM, com foco na gestão do *Pool* de compartilhamento de soluções por meio do algoritmo *Multi-armed Bandit*. Ambas as etapas foram validadas sobre o PRVJT. As subseções seguintes descrevem detalhadamente esses processos.

1.4.1 Análise do *Framework* AMAM

Para estudo do estado da arte e análise do *Framework* AMAM, foram seguidas as seguintes etapas:

- (i) Revisão Bibliográfica: foi realizada uma revisão ampla dos temas relacionados à proposta, com o intuito de levantar todo o referencial necessário ao desenvolvimento do trabalho. Os pontos-chave desta pesquisa incluíram o estudo de técnicas de aprendizado por reforço mono e multiagente, bem como mono e multiobjetivo, e a aplicação dessas técnicas em *frameworks* para solução de problemas de otimização. Além disso, buscou-se compreender o estado da arte dos trabalhos correlatos, abrangendo ferramentas de pesquisa para sistemas multiagente, tais como modelos de recompensas, escalabilidade, cooperação, diversidade de agentes, autonomia, compartilhamento de soluções e aprendizado por reforço;
- (ii) Modelagem de Recompensas: foram propostas e avaliadas seis diferentes formas de modelar a recompensa no algoritmo de aprendizado por reforço do agente, conforme implementado originalmente por [Silva et al. \(2019\)](#), no qual a recompensa utilizada era determinística. Cada modelagem foi baseada no valor da função objetivo obtido pelos agentes após a busca local. A experimentação computacional foi realizada em um *cluster*, analisando valores de função objetivo e tempo de execução. Testes estatísticos foram empregados para identificar diferenças significativas entre os resultados, com o objetivo de aumentar a autonomia da estrutura multiagente por meio da modelagem da recompensa;
- (iii) Análise da Escalabilidade: esta etapa consistiu na criação de uma metodologia para analisar a escalabilidade do *Framework* AMAM. A avaliação considerou a

inclusão de agentes na estrutura do *framework* em dez cenários com até sessenta agentes idênticos. Recursos computacionais distribuídos foram utilizados para garantir a equiparação entre os cenários. As medidas de avaliação foram os valores de função objetivo e tempo de execução. Experimentos computacionais foram realizados e os resultados foram analisados por meio de tabelas, gráficos e testes estatísticos, comparando sistemas mono e multiagente com diferentes números de agentes.

- (iv) *Cooperação*: esta implementação visou avaliar a dimensão de cooperação do *Framework AMAM*, comparando sua utilização com e sem o *Pool* de soluções disponível na estrutura. Testes computacionais foram realizados com diferentes números de agentes, avaliando as métricas de valores de função objetivo e tempo de execução. Os resultados foram submetidos a testes estatísticos para identificar diferenças significativas na utilização da estrutura, além de gráficos que demonstraram a evolução do processo de busca em diferentes cenários.
- (v) *Diversidade*: esta etapa teve como propósito avaliar o comportamento do *Framework AMAM* em relação à diversidade dos agentes em sua execução. Foram desenvolvidos quatro cenários com três agentes distintos para verificar como a heterogeneidade dos agentes afeta o processo de busca do *framework*. Os experimentos computacionais foram realizados em um *cluster*, considerando as demandas de recursos, e os valores de função objetivo e tempo de execução foram avaliados, analisando os resultados por meio de testes estatísticos.

Todos os cenários de experimentação das etapas realizadas foram executados em um *cluster*, com distribuição equitativa dos recursos computacionais, visando à máxima justiça em relação à disponibilidade de *hardware*. Para isso, levaram-se em consideração os seguintes fatores: (i) utilização de um *cluster* para a execução dos testes computacionais; (ii) execução das *threads* JAVA nas *threads* físicas disponíveis no computador utilizado; (iii) garantia de memória suficiente para a execução do número de agentes definidos.

1.4.2 Aprendizado por reforço no *Framework AMAM*

A segunda fase da pesquisa consistiu em propor, implementar e avaliar técnicas de aprendizado por reforço na estrutura do *Framework AMAM*, com ênfase no gerenciamento inteligente do *Pool* de cooperação. Para isso, foram seguidas as etapas descritas a seguir:

- (i) *Multi-armed Bandit Pool*: foi proposto e implementado um algoritmo baseado no método *Multi-armed Bandit* para o gerenciamento do *Pool* de soluções na dimensão social do *framework*, com o objetivo de reduzir a ocorrência de passos desnecessários ou ineficientes durante o processo de cooperação. Cada solução inserida no *Pool* recebeu um valor q inicial, atualizado de forma incremental a cada coleta, com base em uma recompensa binária: valor 1 caso o agente tenha melhorado sua solução corrente após aplicar a busca local sobre a solução coletada, e valor 0 caso contrário. A política ϵ -Greedy foi adotada para balancear a diversificação e a intensificação das escolhas. Foram desenvolvidos

dois modos de coleta de solução: **qmax** (escolha do argumento máximo do valor q) e **softmax** (escolha aleatória entre os melhores valores q) e dois critérios de exclusão de solução do *Pool*, **SHI** (exclusão pela menor diversidade) e **Q** (exclusão pelo menor valor q). Os experimentos compararam cenários com e sem o algoritmo, avaliando os valores de função objetivo e o tempo de execução por meio de testes estatísticos não paramétricos e da análise da evolução temporal do processo de busca;

- (ii) *Multi-armed Bandit* Multiagente com cooperação inspirada em PSO (MAMaB-PSO, *Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation*): foi proposto um algoritmo de aprendizado por reforço multiagente que estende o MaB clássico ao cenário com múltiplos agentes interagindo simultaneamente sobre o mesmo *Pool* de soluções, incorporando os componentes cognitivo e social do *Particle Swarm Optimization* à estimativa de valor das soluções. A formulação é genérica em relação à política de seleção, permitindo a instanciagem de variantes das políticas MaB, incluindo estratégias para ambientes não estacionários (DUCB, *Discounted UCB*, e SWUCB, *Sliding-Window UCB*). Os experimentos contrastaram essas variantes com uma configuração de referência sem o componente PSO, avaliando *gap* percentual, tempo de execução, evolução temporal da qualidade das soluções e perfis de desempenho comparativos;
- (iii) *Multi-armed Bandit* Multiagente e Multiobjetivo: foi proposta uma versão multiobjetivo do *Multi-armed Bandit Pool*, na qual o valor escalar q é substituído por um vetor de recompensa que captura, simultaneamente, qualidade de função objetivo, diversidade e número de rotas, permitindo a seleção de soluções segundo critérios de dominância de Pareto sobre estimativas vetoriais;
- (iv) *Pool* de Soluções Multiobjetivo: foi proposta uma extensão da estrutura do *Pool* para suportar a organização multiobjetivo das soluções armazenadas, generalizando os mecanismos de inserção e manutenção com base em dominância de Pareto e *crowding distance* sobre os critérios de função objetivo e diversidade estrutural, complementando o aprendizado vetorial proposto na etapa anterior.

Considerando a complexidade dos temas abordados, todas as premissas foram modeladas tendo o PRVJT como problema-instância. A validação dos objetivos e algoritmos propostos foi realizada por meio da submissão e publicação de artigos em eventos e periódicos científicos.

1.5 Contribuições da Tese

Esta seção sintetiza as principais contribuições desta tese, organizadas segundo seu fluxo lógico: da análise e validação das dimensões de sistemas multiagente para otimização combinatória por meio de meta-heurísticas à incorporação progressiva de aprendizado por reforço sobre a estrutura de cooperação, nas dimensões mono-agente, multiagente e multiobjetivo. As contribuições, validadas por testes estatísticos não paramétricos, articulam-se da seguinte forma:

- (i) Metodologia de análise de escalabilidade, cooperação e diversidade em sistemas multiagente: esta tese propõe uma metodologia de análise de escalabilidade,

cooperação e diversificação em sistemas multiagente para a resolução de problemas de otimização combinatória por meio de meta-heurísticas, validando o comportamento dessas dimensões para o caso avaliado do *framework* AMAM.

- (ii) Modelagem autônoma de recompensas: demonstra-se que o aprendizado por reforço dos agentes pode ser orientado por funções de recompensa baseadas em diferença e em potencial, desvinculadas de conhecimento específico do problema, sem perda de qualidade em relação à recompensa paramétrica de [Silva et al. \(2019\)](#), diminuindo a dependência do mecanismo de aprendizado frente a parâmetros e aumentando suas possibilidades de generalização no contexto de *frameworks* multiagente.
- (iii) Aprendizado por reforço na cooperação (MaB *Pool*): propõe-se um algoritmo baseado no *Multi-armed Bandit* para gerenciar o *Pool* de soluções da dimensão social do *framework*, priorizando soluções que contribuem para o desempenho coletivo. Esse aprendizado escalar preserva ou aprimora a qualidade das soluções e reduz o custo computacional frente às configurações sem aprendizado.
- (iv) Aprendizado por reforço multiagente com cooperação inspirada em PSO (MAMaB-PSO): propõe-se o MAMaB-PSO, uma formulação geral de aprendizado por reforço multiagente que hibridiza o *Multi-armed Bandit* com os componentes cognitivo e social do *Particle Swarm Optimization*, independente do domínio de aplicação. O algoritmo é instanciado na dimensão de cooperação do *Framework* AMAM e avaliado em múltiplas variantes da política de seleção, contrastadas com uma referência sem o componente PSO.
- (v) Aprendizado multiobjetivo sobre o *Pool* de cooperação: estende-se o aprendizado por reforço à formulação multiobjetivo, substituindo a recompensa escalar por um vetor avaliado segundo dominância de Pareto, que considera simultaneamente a qualidade da função objetivo, a diversidade estrutural e o número de rotas. A proposta atua tanto nas políticas de seleção do *Multi-armed Bandit* quanto na própria estrutura do *Pool*, reorganizada por dominância de Pareto e *crowding distance*. Sem alterar a modelagem mono-objetivo do problema, preserva a qualidade das soluções e reduz o custo computacional das variantes multiobjetivo.

Essas contribuições são desenvolvidas ao longo da tese: as contribuições (i)–(ii) são desenvolvidas no Capítulo 4; o aprendizado mono-agente e multiagente, ou seja, contribuições (iii)–(iv), são apresentadas no Capítulo 5; e a extensão multiobjetivo, posta na contribuição (v), está presente no Capítulo 6.

1.6 Estrutura da Tese

Esta tese está estruturada da seguinte forma: o presente capítulo apresenta a introdução ao tema, a justificativa, os objetivos e a metodologia de pesquisa adotada. O Capítulo 2 contextualiza os principais temas tratados neste trabalho, apresentando conceitos de sistemas multiagente, meta-heurísticas e aprendizado de máquina, incluindo as formulações mono-agente, multiagente e multiobjetivo do algoritmo *Multi-armed Bandit*. O Capítulo 3 apresenta os elementos comuns a todos os experimentos

realizados, descrevendo o PRVJT, os agentes implementados no *Framework* AMAM e as configurações experimentais compartilhadas. No Capítulo 4, são apresentados os experimentos e os resultados obtidos na análise do *Framework* AMAM, relativos à modelagem de recompensas, escalabilidade, cooperação e diversidade. O Capítulo 5 apresenta as contribuições de aprendizado por reforço na dimensão de cooperação do *framework*, incluindo o algoritmo MaB Pool para gerenciamento mono-agente do *Pool* de soluções e sua extensão para o cenário multiagente cooperativo com inspiração no PSO. O Capítulo 6 apresenta a extensão multiobjetivo, propondo o algoritmo *Multi-armed Bandit* com recompensas vetoriais e o *Pool* de soluções baseado em dominância de Pareto como suporte estrutural ao aprendizado multiobjetivo. Por fim, o Capítulo 7 finaliza a tese, apresentando as considerações finais e as direções futuras de pesquisa.

Capítulo 2

Fundamentação

Este capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho. Está dividido em três seções principais. A Seção 2.1 aborda o conceito de *frameworks* para a solução de problemas de otimização combinatória. Na Seção 2.2, são apresentados conceitos de sistemas multiagente e suas principais características. Por fim, na Seção 2.3, são discutidos conceitos relacionados à união de meta-heurísticas e aprendizado de máquina.

2.1 *Frameworks* para Otimização

Frameworks são abstrações utilizadas para unir códigos que se repetem em projetos de desenvolvimento de *software*. Ao empregar *frameworks*, o desenvolvimento de *softwares* torna-se menos oneroso e complexo, aumentando a produtividade e a qualidade do processo.

No contexto de problemas de otimização combinatória, a utilização de *frameworks* possibilita a resolução de diversos tipos de desafios, tornando-os ferramentas valiosas para essa finalidade. *Frameworks* para otimização via metaheurísticas (MOF – *Metaheuristic Optimization Frameworks* são ferramentas de software que oferecem implementações dos principais métodos meta-heurísticos disponíveis na literatura, através de códigos reutilizáveis que simplificam o processo de desenvolvimento de aplicações para a resolução de problemas de otimização (Parejo et al., 2012). A utilização de *frameworks* oferece diversas vantagens para os pesquisadores da área, como a disponibilidade de meta-heurísticas pré-implementadas para testes e reutilização, suporte para a avaliação e comparação de diferentes métodos, bem como maior facilidade no desenvolvimento de meta-heurísticas personalizadas para atender a problemas específicos (Silva et al., 2018). Além disso, os *frameworks* também tornam mais acessível a hibridização de meta-heurísticas, permitindo uma integração mais eficiente entre diferentes abordagens.

Uma revisão e análise comparativa dos principais *frameworks* para otimização, encontrados na literatura, é apresentada em Silva et al. (2018). Nessa revisão, os autores identificam características desejáveis e lacunas no desenvolvimento de *Frameworks* de Otimização usando Meta-heurísticas. O estudo avaliou os *frameworks* considerando diversas características, das quais destacamos: (i) Hibridização: refere-se à propriedade do *framework* que permite a implementação de meta-heurísticas híbridas; (ii)

Computação paralela/distribuída: abrange a capacidade de implementação de métodos paralelos ou distribuídos no *framework*, como a execução de sub-tarefas em *threads* ou em máquinas distintas; (iii) Sistemas multiagente: envolve a utilização dos conceitos de sistemas multiagente na definição do *framework*; e; (iv) Aprendizado: diz respeito à capacidade do agente do *framework* de aprender durante o processo de busca da solução. Para uma compreensão mais aprofundada dessas características, recomendamos a leitura de [Silva et al. \(2018\)](#). A partir dessa análise, optamos pela utilização do *Framework AMAM*, que será descrito a seguir.

2.1.1 Framework AMAM

O *Framework AMAM* (Arquitetura MultiAgente para Metaheurística) foi proposto, inicialmente, em [Silva \(2007\)](#), e, em seguida, novas versões foram apresentadas em [Fernandes \(2009\)](#); [Silva et al. \(2015\)](#); [Silva \(2019\)](#). O *Framework AMAM* é uma arquitetura multiagente para metaheurísticas, na qual cada agente implementa uma meta-heurística e age de forma a buscar uma solução para um problema. O espaço de busca do problema corresponde ao ambiente do sistema multiagente e, assim, com uma simples troca do ambiente, altera-se com facilidade o problema a ser tratado. A capacidade de movimentação de cada agente pelo ambiente se dá pelas formas de manipulação da solução que cada problema disponibiliza. A facilidade de inclusão de novos agentes na estrutura multiagente do *framework* garante a sua escalabilidade, e a adição desses agentes afeta minimamente a estrutura. Os agentes possuem a capacidade de interagir e perceber o ambiente de forma cooperativa, além de possuírem capacidades adaptativas. Os principais elementos do *Framework AMAM* são: Ambiente, Agente e Cooperação.

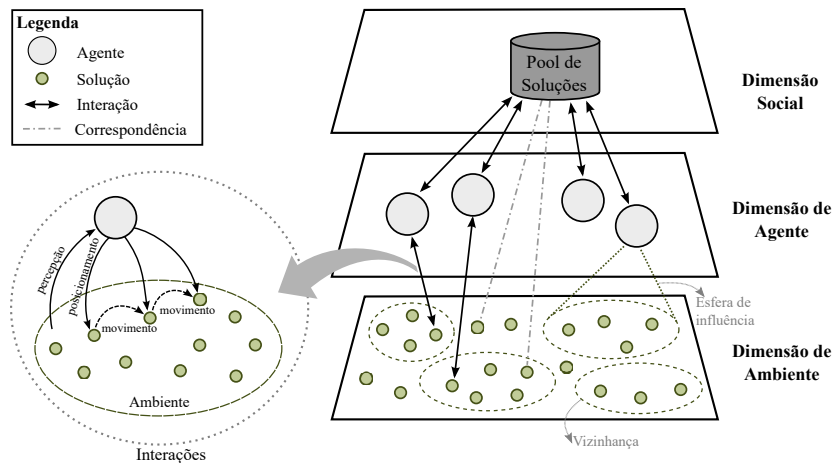


Figura 2.1: Dimensões do *Framework AMAM*. Adaptado de [Silva \(2019\)](#).

Conforme apresentado na Figura 2.1, o *Framework AMAM* possui três dimensões principais para a resolução de um problema de otimização. A Figura 2.1 mostra estas três dimensões do *Framework AMAM* e as suas interações. Estas dimensões compreendem a dimensão social, responsável pelo compartilhamento de soluções; a dimensão de Agente, responsável pela implementação das meta-heurísticas, e a dimensão de

Ambiente, que se trata do próprio espaço de busca do problema. Portanto, é necessário construir uma representação dessas três dimensões a fim de aplicar o *framework* a um problema de otimização. Para isso, as seções seguintes apresentam as principais características dessas dimensões.

2.1.1.1 Dimensão Ambiente

A dimensão **Ambiente** é o próprio espaço de busca do problema a ser tratado. Dessa forma, os elementos do ambiente (espaço de busca das soluções) estão acessíveis a todos os agentes. O ambiente é passado por referência aos agentes para garantir esse acesso, o que implica que todos os agentes têm acesso ao mesmo local na memória e que, além disso, qualquer modificação realizada por um agente fica imediatamente disponível para os demais. Observa-se que os agentes atuam simultaneamente na dimensão **Ambiente**, através da movimentação pelas soluções do problema.

2.1.1.2 Dimensão Agente

A dimensão **Agente** incorpora uma meta-heurística/meta-heurística híbrida e tem a função de buscar a solução para um dado problema. O agente possui as seguintes habilidades de interação com o ambiente: (i) Percepção: característica que o torna capaz de enxergar parcialmente o ambiente do qual faz parte; (ii) Posicionamento: capacidade de se posicionar no espaço de busca, seja através de uma nova solução ou pela escolha de uma solução existente; (iii) Movimento: condição de se movimentar pelo espaço de busca, de uma solução a outra, usando, por exemplo, estruturas de vizinhanças ou operadores genéticos.

O agente apresenta também habilidades autoadaptativas (Silva et al., 2019), que permitem sua adaptação às características específicas do problema de otimização, utilizando conceitos de aprendizado de máquina. Em Silva et al. (2019), o agente construído implementa o algoritmo *Q-learning* de aprendizado por reforço na busca local, denominada *ALS-QLearning* (*Adaptive Local Search* com *Q-Learning*). Esta busca local consiste em um método em que a ordem de vizinhança é aprendida através do método *Q-learning*. Os resultados apresentados em Silva et al. (2019) demonstram que a utilização do algoritmo de aprendizado por reforço alcançou melhores resultados do que a estrutura determinada via complexidade das vizinhanças.

O objetivo do aprendizado é avaliar o ganho obtido com a aplicação de uma sequência de duas vizinhanças da busca local e, a partir daí, premiar as melhores sequências e maximizar a recompensa acumulada. Cada vizinhança a ser utilizada pelo método de busca é considerada como um estado de aprendizagem. Desta forma, o conjunto de estados S é formado pela funções de vizinhanças disponíveis.

No conjunto de ações $A(s)$, uma ação é definida como a mudança de um estado para outro. Dessa forma, o conjunto de ações pode ser representado por um grafo completo, no qual cada ação é representada por um arco conectando dois estados (nós do grafo), como apresentado na Figura 2.2, extraída de Silva et al. (2019). Um exemplo de gráfico representando a relação entre estados (funções de vizinhança) e possíveis ações é mostrado na Figura 2.2(a). Neste exemplo, quatro funções de vizinhança do PRVJT são usadas para facilitar a visualização. A tabela Q referente a este exemplo é mostrada na Figura 2.2(b). Esta tabela tem dimensões dadas por

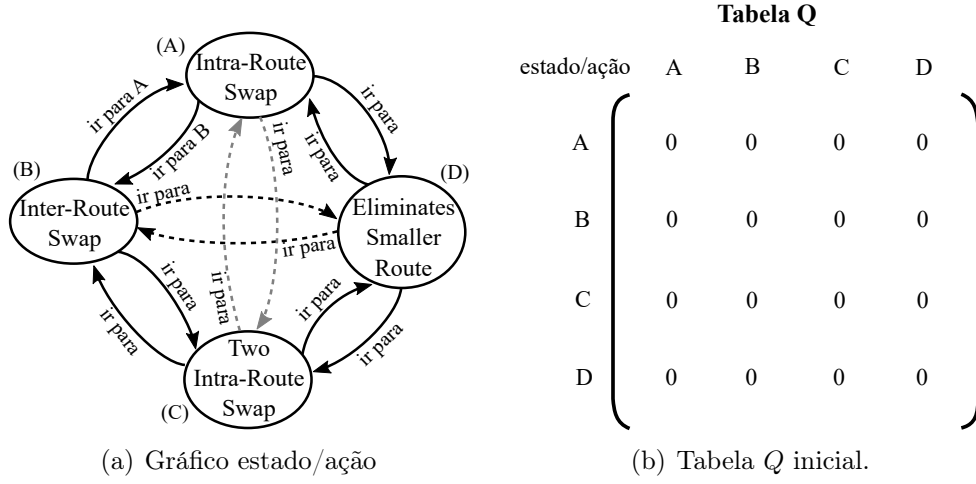


Figura 2.2: Relação entre estados (vizinhanças) e ações possíveis (Silva et al., 2019).

$M \times M$, em que M é o número de estados, ou seja, o número de funções de vizinhança do problema tratado.

A tabela Q é responsável por armazenar os valores dos pares estado-ação no decorrer do processo de aprendizagem. Nesta proposta, cada elemento da tabela Q é inicializado com valor zero, de modo que o processo de aprendizagem seja realizado de forma não tendenciosa. Uma vez que as funções de vizinhança são parâmetros específicos do problema, o objetivo do uso do aprendizado por reforço é permitir que o *framework* se adapte melhor às características próprias do problema. É importante ressaltar que a tabela Q é parte integrante do agente, e não está externa ao seu funcionamento.

O Algoritmo 1 descreve o algoritmo de busca local ALS- $QLearning$ proposto em Silva et al. (2019). Neste algoritmo, o estado inicial da tabela Q é definido na linha 7. O próximo estado é determinado de modo aleatório quando o valor *proximo_estado* é igual a 2; caso contrário, ou seja, caso *proximo_estado*=1, é utilizada a função *ϵ -greedy*. Esse estado inicial é aplicado à solução pela estratégia Descida/Subida Completa, na linha 8. A cada episódio, iniciado na linha 14, o número de estados visitados é reiniciado, e o estado aplicado na iteração anterior é registrado (linhas 15 e 16). Um episódio é executado até a meta ser alcançada (linhas 17 a 38). A meta é encontrar uma solução melhor que a atual e, se alcançada, o episódio corrente é encerrado. Caso não tenha havido melhoria na solução na iteração anterior, é utilizada a função aleatória. O próximo estado também é aplicado à solução pela estratégia Descida/Subida Completa, na linha 23. Quando a solução melhora, uma recompensa é atribuída e seu valor na matriz Q é calculado. O objetivo é determinar o ganho obtido ao aplicar uma sequência de duas vizinhanças. Assim, a atualização da matriz Q considera o estado corrente (selecionado na iteração atual) e o estado da iteração anterior. Dessa forma, se a aplicação de uma vizinhança m_2 após a aplicação de uma vizinhança m_1 resultar em melhoria na solução corrente, a matriz Q é atualizada.

Algoritmo 1 Busca Local Adaptativa - ALS-*QLearning* (Silva, 2019)

```

1: função SOLUÇÃO BUSCALocalAdaptativaQLearning( $x_0, w$ )
2:   inicializa( $Q(\text{estado}, \text{acao})$ );
3:   melhorou  $\leftarrow$  verdadeiro;
4:   sem_melhora  $\leftarrow$  0;
5:    $x^* \leftarrow x_0$ ;
6:    $x \leftarrow x_0$ ;
7:   proximo_estado  $\leftarrow$  escolheUmaAcao(0, 2); ▷ 2: estado inicial definido por função aleatória
8:    $x \leftarrow$  melhorVizinho(proximo_estado,  $x$ );
9:   se ( $x$  é melhor que  $x^*$ ) então
10:     $x^* \leftarrow x$ ;
11:   senão
12:    sem_melhora  $\leftarrow$  1;
13:   fim se
14:   repita ▷ para cada episódio
15:     qtd_estados_visitados  $\leftarrow$  0;
16:     estado  $\leftarrow$  proximo_estado;
17:     enquanto (meta do estado não alcançada) faça ▷ meta do estado: melhorar a solução
18:       se (sem_melhora = 0) então
19:         proximo_estado = escolheUmaAcao(estado, 1); ▷ 1: função epsilon greedy
20:       senão
21:         proximo_estado = escolheUmaAcao(0, 2); ▷ se não houve melhora, a função greedy não deve ser
22:         usada
23:       fim se
24:        $x =$  melhorVizinho(proximo_estado,  $x$ );
25:       se ( $x$  é melhor que  $x^*$ ) então ▷ meta do estado alcançada
26:          $x^* \leftarrow x$ ;
27:         melhorou  $\leftarrow$  verdadeiro;
28:         sem_melhora  $\leftarrow$  0;
29:         calculaValorQ(estado, proximo_estado,  $w$ );
30:       senão
31:         sem_melhora  $\leftarrow$  sem_melhora + 1;
32:         se (o estado já foi visitado) então
33:           qtd_estados_visitados  $\leftarrow$  qtd_estados_visitados + 1;
34:         fim se
35:         se ((sem_melhora > max_iterações_sem_melhora) e (qtd_estados_visitados = tam_q)) então
36:           melhorou  $\leftarrow$  falso;
37:         fim se
38:       fim enquanto
39:        $\epsilon \leftarrow \epsilon * \text{taxa\_decaimento}$ ;
40:       até que (não(melhorou))
41:       retorne  $x$ ;
42:   fim função

```

2.1.1.3 Dimensão Cooperação

A dimensão **Cooperação** possui estruturas que permitem a interação e cooperação entre os agentes. O objetivo dessa estrutura cooperativa é orientar os agentes no espaço de soluções em direção às áreas mais promissoras do espaço de busca e, assim, melhorar o resultado final e reduzir o tempo necessário para solucionar o problema.

A principal ferramenta de cooperação implementada no *framework* é o *Pool* de Soluções. Esta ferramenta permite compartilhar informações entre os agentes. No *Pool* de Soluções, localizado no ambiente do sistema multiagente, são inseridas e armazenadas as soluções disponíveis durante o processo de busca. As soluções são compartilhadas pelos agentes ao final de cada iteração. A comunicação é regida pelas regras de acesso ao *Pool*, tanto para escrita quanto para leitura, visando garantir diversidade no compartilhamento de informações de busca. A seguir são apresentadas informações em relação à implementação da estrutura de cooperação do *framework*, conforme (Silva et al., 2019).

O método de inserção de soluções no *Pool* de soluções segue três passos:

- (i) O tamanho máximo do *Pool* de soluções é um parâmetro informado ao início de cada execução, determinando o tamanho atual do *Pool*. Caso o tamanho atual do *Pool* de soluções seja menor que o tamanho máximo permitido, a solução obrigatoriamente é adicionada ao *Pool*;
- (ii) Se o tamanho máximo do *Pool* já foi alcançado, verifica-se, primeiramente, se aquela solução ainda não está presente no conjunto. Caso ainda não esteja, passa-se, então, a avaliar o valor da função objetivo das soluções. Caso o valor de função objetivo da solução candidata a ser inserida seja melhor que o pior valor de função objetivo das soluções contidas no *Pool*, esta solução candidata é adicionada ao *Pool* de soluções, substituindo a solução com pior valor de função objetivo nele contido. Caso contrário, ou seja, se o valor da função objetivo da solução candidata seja pior que o pior valor de função objetivo dentre as soluções contidas no *Pool* naquele instante de tempo, a solução candidata não será adicionada ao *Pool* de soluções;
- (iii) O terceiro e último passo do compartilhamento de soluções ocorre apenas se o passo (i) for falso e a solução candidata for inserida no passo (ii). Neste caso, deve-se, então, definir qual será a solução que será retirada do *Pool*. A extração desta solução é realizada a partir do grau de diversidade entre as soluções, definido como $g(\phi_i)$, e dado por:

$$g(\phi_i) = \sum_{j=1}^{|P|} \phi(\lambda_{ij}) \quad (2.1)$$

em que $|P|$ representa a cardinalidade do *Pool* de soluções, ou seja, o número de soluções nele contidas, e λ_{ij} é a distância entre as soluções i e j . A função $g(\phi_i)$ estima a concentração de soluções na vizinhança da solução i com base na distância entre essa solução i e as demais soluções presentes no *Pool*. Essa função de avaliação é fundamentada nas técnicas de nicho (Li et al., 2017) para coordenar conjuntos de soluções. A função $\phi(\lambda_{ij})$ é definida como:

$$\phi(\lambda_{ij}) = \begin{cases} 1 - \frac{\lambda_{ij}}{pr}, & \text{se } \lambda_{ij} \leq pr \\ 0, & \text{se } \lambda_{ij} > pr \end{cases} \quad (2.2)$$

O parâmetro pr representa o raio do *Pool*, e controla o grau de dispersão das soluções. Esse valor é específico para cada problema tratado e é determinado como um parâmetro inicial. Para o PRVJT, por exemplo, o valor de pr é definido como o número mínimo de arcos em comum para que duas soluções sejam consideradas próximas uma da outra. Portanto, o valor de pr está diretamente relacionado à dimensão da instância do problema em questão. A medida λ_{ij} indica o quão similares são as soluções i e j , e seu cálculo depende fundamentalmente do problema que está sendo abordado. A Figura 2.3 ilustra dois exemplos de cálculo de distância entre soluções no PRVJT. No primeiro exemplo, a distância entre a solução i , mostrada na Figura 2.3(a), e a solução j , mostrada na Figura 2.3(b), é igual a $\lambda_{ij} = 12$, o que significa que há 12 arcos distintos entre

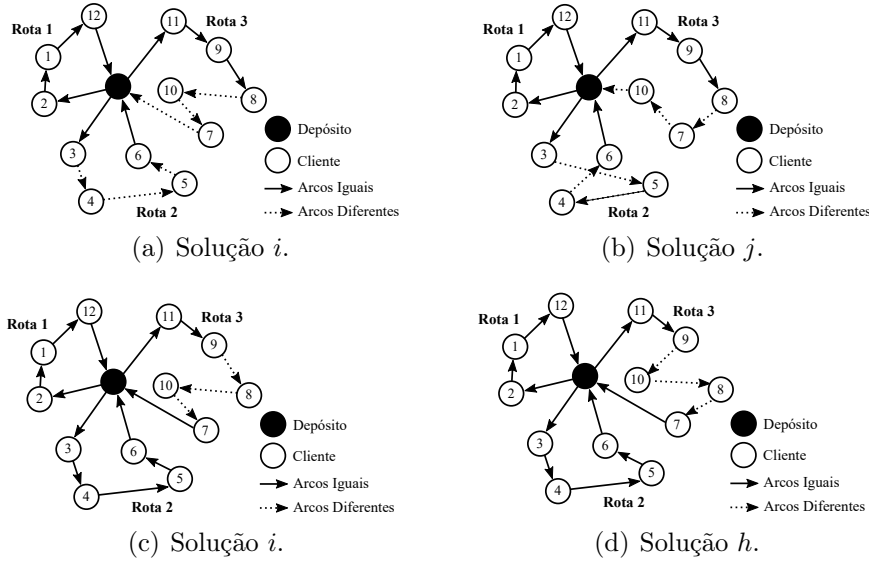


Figura 2.3: Exemplos de cálculo da distância entre as soluções para o PRVJT, conforme Silva et al. (2019)

essas soluções. No segundo exemplo, a distância entre a solução i , mostrada na Figura 2.3(c), e a solução h , mostrada na Figura 2.3(d), é igual a $\lambda_{ih} = 6$, indicando a presença de 6 arcos distintos entre essas soluções.

Como resultado, após avaliar o valor $g(\cdot)$ de cada solução i no *Pool*, a solução que apresentar a pior avaliação nessa análise (ou seja, o maior valor da função $g(\cdot)$, dada pela expressão (2.1)) é removida para permitir a inserção da nova solução. A principal finalidade dessa função de avaliação é preservar a diversidade do conjunto, evitando a manutenção de soluções excessivamente similares ou até mesmo idênticas. Ao mesmo tempo, esta ferramenta é elitista, pois a melhor solução existente no conjunto é sempre preservada.

Desse modo, o processo de inserção e retirada de soluções no *Pool* está definido e sua estrutura de atualização descrita.

2.2 Sistemas Multiagente

Sistemas Multiagente (MAS – *Multiagent System*) são compostos por múltiplos agentes autônomos que interagem em um ambiente compartilhado, com o objetivo de alcançar metas individuais ou coletivas (Wooldridge, 2009). Cada agente em um MAS é capaz de perceber seu ambiente, tomar decisões autônomas, agir e se comunicar com outros agentes. A interação entre esses agentes é fundamental para lidar com tarefas complexas. Os sistemas multiagente são frequentemente utilizados para modelar e resolver problemas que requerem cooperação, coordenação e negociação.

Nesta seção, são apresentadas as principais características encontradas em sistemas multiagente que são de interesse para este trabalho. A literatura discute diversas propriedades desses sistemas, incluindo coordenação, cooperação, diversidade, racionalidade e modelagem de conhecimento, apresentadas a seguir.

2.2.1 Cooperação

A discussão a respeito de cooperação em sistemas multiagente aborda como agentes autônomos podem interagir e colaborar entre si para alcançar objetivos comuns. A cooperação age de forma a possibilitar que os agentes compartilhem, coordenem e tomem decisões conjuntas a partir de informações e ações realizadas, com o propósito de alcançar melhores resultados que em sistemas em que estes agentes trabalhem de forma isolada. Wooldridge (2009) afirma que os interesses dos agentes em sistemas multiagente, assim como nas sociedades humanas, podem entrar em conflito, partindo da premissa de que os agentes podem ter sido projetados e implementados por diferentes indivíduos e com diferentes objetivos. Em geral, sistemas multiagente são orientados a agir em seu próprio interesse. Ele também afirma que a maioria dos trabalhos sobre resoluções cooperativa de problemas faz a suposição de benevolência, ou seja, que os agentes em um sistema partilham implicitamente de um objetivo comum e, sendo assim, não há um conflito potencial entre eles. Instintivamente, o que importa para esses agentes são os objetivos gerais do sistema e não os seus próprios objetivos. Portanto, para que isso seja alcançado, agentes são projetados para ajudar, mesmo que isso signifique que eles tenham que se sacrificar para fazê-lo. A teoria da benevolência é aceitável sob a premissa de que os agentes são propriedades de uma mesma entidade. Quando os agentes se preocupam apenas com a utilidade geral do sistema, pode-se projetar um sistema que a otimize. Mesmo com o potencial para conflitos de interesse, os agentes em um sistema multiagente precisarão cooperar para atingir seus objetivos; novamente, assim como eles fazem nas sociedades humanas. Wooldridge (2009) também observa que a pesquisa de sistemas multiagente está, portanto, preocupada com os problemas mais amplos de projetar sociedades de agentes autônomos, abordando questões como: (i) por que e como os agentes cooperam? (ii) como os agentes podem reconhecer e resolver conflitos? (iii) como os agentes podem negociar ou transigir em situações em que aparentemente estão em desacordo?; dentre outras questões.

Agentes podem cooperar de várias maneiras durante seu processo de busca. Smith e Davis (1981) mostram duas formas cooperativas que um sistema multiagente pode apresentar. Essas formas incluem o compartilhamento de tarefas e o compartilhamento de resultados. O compartilhamento de tarefas consiste em dividir o problema em sub-tarefas para que cada agente possa resolver partes dos problemas contidos no sistema. Já compartilhamento de resultados consiste em compartilhar resultados obtidos pelos agentes durante o seu processo de busca. Diferentes agentes possuem diferentes visões do ambiente ao qual eles estão inseridos. Wooldridge (2009) mostra que, em ferramentas de compartilhamento de resultados (*results sharing*), a solução de problemas ocorre por agentes que trocam informações cooperativas à medida que uma solução é desenvolvida. Além disso, existem caminhos que podem melhorar o desempenho dos grupos de agentes, que são: (i) confiança: soluções podem ser verificadas com o intuito de destacar possíveis erros e aumentar a confiança da solução geral; (ii) integridade: agentes compartilham sua visualização com o intuito de obter uma melhor visão global; (iii) precisão: agentes compartilham resultados para garantir que a precisão da solução geral do sistema seja melhorada; e; (iv) pontualidade: mesmo que um agente seja capaz de alcançar um resultado com o compartilhamento, este resultado poderia ser alcançado mais rapidamente com a cooperação.

Sistemas multiagente são vantajosos por diferentes razões, seja por coordenação, cooperação, racionalidade, modelagem de conhecimento, e escalabilidade, dentre outras características contidas nestes sistemas. Os sistemas multiagente podem ser modelados e implementados com agentes clonados/iguais ou agentes diferentes. Estes agentes possuem diferentes visões do espaço no qual estão inclusos. A heterogeneidade de agentes pode trazer ganhos ao objetivo geral do sistema, aumentando, assim, a sua resiliência e robustez. Ambientes diversos podem proporcionar diferentes perspectivas sobre soluções e métodos de resolução.

Neste trabalho, a cooperação é analisada a partir da validação da eficiência do mecanismo de cooperação do *Framework* AMAM. Essa validação é realizada por meio de implementações, testes computacionais e análises estatísticas, conforme apresentado na Seção 4.2.

2.2.2 Diversidade

A diversidade está diretamente relacionada a um ambiente cheio de diferenças, que envolvem todos os aspectos de concepção, desenvolvimento e sobrevivência. Nesse ambiente, cada agente ou indivíduo respeita suas particularidades e apreende com elas. [Esmaeili et al. \(2016\)](#) mostram que a diversidade, em Ciências Sociais, está direcionada às diferenças de culturas, etnias, características demográficas e conhecimentos dentro de uma sociedade. Essas características são divididas em diversidade de identidade e diversidade cognitiva ([Esmaeili et al., 2016](#)).

Em sistemas de computação e investigação operacional, a diversidade é um atributo almejado em diversos campos de pesquisa. Em algoritmos genéticos ([Golberg, 1989](#)), a diversidade da população de indivíduos é desejada com o intuito de impedir a convergência prematura para soluções com qualidade baixa. A convergência pode ser garantida através da diversificação das soluções correntes pela seleção da nova geração da população e/ou pela mutação de indivíduos durante o processo de busca.

A heterogeneidade de um sistema multiagente consiste em agentes com diferentes características, habilidades e comportamentos relacionados à sua modelagem e/ou implementação. Essa diversidade de agentes pode melhorar o resultado geral do sistema, uma vez que agentes diferentes são capazes de visitar diferentes partes do ambiente, aumentando a resiliência do sistema e melhorando sua eficiência na resolução de problemas complexos. Contudo, conforme apresentado por [Wooldridge \(2009\)](#), a diversidade também apresenta desafios, relacionados à coordenação, à comunicação efetiva entre os agentes, resolução de conflitos, divergências de interesses ou desequilíbrio do sistema. Pesquisas que buscam incorporar diversidades entre agentes exploram estratégias como algoritmos de seleção, mecanismos de aprendizado, incentivos e negociação. O objetivo do desenvolvedor do sistema é encontrar o equilíbrio adequado entre a diversidade e a cooperação, a fim de alcançar soluções melhores e mais robustas nos sistemas multiagente.

A análise dos efeitos da diversificação em sistemas multiagente é uma tarefa fundamental do desenvolvimento de sistemas multiagente. [Wong et al. \(2005\)](#) propõem uma versão de jogos de grandes populações em que os agentes competem por recursos. Esses jogos são utilizados para modelar mercados econômicos, ecossistemas ou controle distribuído. A diversidade de estratégias é introduzida pela atribuição aleatória de vieses às estratégias de diferentes agentes, apresentando melhores resul-

tados, como o tempo de convergência, a fração de agentes inconstantes e a variância da riqueza. Assim, eles apontam que a diversidade entre os agentes aumenta seu comportamento adaptativo. Já em [Chowdhury e Sklar \(2014\)](#) são exploradas propriedades de um sistema multiagente designado para resolver problemas de tarefas sequenciais em ambientes dinâmicos em tempo real e assíncronas. Diversos modelos evolutivos foram implementados e testes computacionais foram realizados em ambiente computacional de alto desempenho. Diferentes mecanismos de interação e modos de diversidade são avaliados populacionalmente de acordo com múltiplas métricas de desempenho. Ao fim, resultados computacionais que expõem sistemas heterogêneos contra sistemas homogêneos mostram que equipes heterogêneas aprendem a coletar recursos de forma mais eficaz que equipes homogêneas, além de aprenderem a se comunicar com mais eficiência.

Neste trabalho, a diversidade é analisada a partir da inserção e implementação de diferentes agentes no *Framework* AMAM. Essa análise considera a capacidade do *framework* de integrar agentes com características distintas, permitindo avaliar seus efeitos nos testes computacionais e nas análises estatísticas apresentadas na Seção 4.3.

2.2.3 Escalabilidade

[Barroso et al. \(2019\)](#) definem escalabilidade como a capacidade de um sistema de lidar e se adaptar a um aumento substancial na carga, tráfego ou demanda, mantendo um desempenho aceitável sem que isso prejudique seu funcionamento. No caso de sistemas multiagente, a escalabilidade é um atributo fundamental a ser considerado em sua implementação e aplicação, uma vez que os requisitos e demandas de um sistema podem variar ao longo do tempo. Além disso, o crescimento no número de agentes ativos é inerente a esses sistemas. Na Ciência da Computação convencional, a escalabilidade é o foco do desenvolvimento de *software* e refere-se a quão bem o sistema é capaz de executar um trabalho útil à medida que o tamanho do sistema aumenta. [Deters \(2001\)](#) oferece três definições do termo escalabilidade:

- (i) escalabilidade fraca: a capacidade de um aplicativo ou produto de computador (*hardware* ou *software*) continuar a funcionar bem à medida que é modificado em tamanho ou volume para atender às necessidades do usuário;
- (ii) escalabilidade forte: não apenas a capacidade de funcionar bem em situações de redimensionamento, mas também a capacidade de otimização máxima;
- (iii) *scale-up* $\times$ *scale-down*: a escala pode ser aumentada ou reduzida. Um programa é considerado capaz de aumentar a escala se a adição de recursos específicos resultar em um aumento de desempenho.

[Rana e Stout \(2000\)](#) descrevem escalabilidade, em sua forma mais geral, como “a capacidade de uma solução para um problema funcionar quando o tamanho do problema aumenta”. Além disso, [Rana e Stout \(2000\)](#) contextualizam a escalabilidade para a comunidade de sistemas multiagente, destacando várias dimensões de escalabilidade fundamentais, sendo as principais: (i) o aumento do número total de agentes em uma plataforma específica; (ii) o crescimento do número total de agentes em vários sistemas ou plataformas; (iii) o aumento no tamanho dos dados nos quais os agentes operam; (iv) o aumento da diversidade de agentes envolvidos.

No contexto específico de sistemas multiagente, não existe um consenso unificado sobre a definição de escalabilidade, uma vez que ela se refere a diversos aspectos, como a complexidade dos agentes ou o volume de mensagens. De acordo com [Lee et al. \(1998\)](#), a escalabilidade é medida pela complexidade do algoritmo avaliado, avaliando o tempo de execução gasto quando novos agentes são adicionados ao sistema. Por sua vez, [Deters \(2001\)](#) apresenta três premissas para medir a escalabilidade: (i) o número de agentes no sistema; (ii) o consumo de recursos por agente; (iii) o número de mensagens ou conversas. [Deters \(2001\)](#) também destaca a importância de distinguir entre sistemas que podem se ajustar dinamicamente em tempo de execução e aqueles que exigem reinicialização ou recompilação. Sistemas que podem ser ajustados em tempo de execução têm uma escalabilidade dinâmica completa; aqueles que exigem reinicialização têm apenas escalabilidade dinâmica parcial; e, por fim, aqueles que exigem recompilação têm escalabilidade estática. Além disso, [Deters \(2001\)](#) observa que o problema de escalabilidade de sistemas multiagente não se limita apenas à questão de recursos computacionais, mas também à complexidade de *software*. O desenvolvimento de agentes baseados em informações da *web* apresenta desafios adicionais, já que esses agentes exigem recursos computacionais significativos, tornando a construção de sistemas escaláveis complexa, especialmente em relação ao número de agentes, uma vez que os sistemas *web* devem fornecer acesso rápido e confiável às informações.

[Turner e Jennings \(2001\)](#) observam uma crescente demanda pelo desenvolvimento de sistemas multiagente. Esses sistemas podem consistir em centenas de milhares de agentes autônomos, operando em ambientes abertos e dinâmicos, e o número de agentes pode variar significativamente durante o processo. Para enfrentar os desafios de escalabilidade e variação no número de agentes, [Turner e Jennings \(2001\)](#) implementam um sistema multiagente aplicado ao domínio de negociação automatizada e desenvolvem uma estrutura gerenciável para superar as dificuldades na construção de sistemas multiagente em grande escala. O objetivo é avaliar o comportamento de sistemas multiagente quando submetidos a um grande e variável número de agentes. Os resultados preliminares indicam que testes mais abrangentes são necessários, pois determinar o número ótimo de agentes funcionais em um sistema multiagente é uma questão que requer experimentação. Além disso, os autores concluem que criar um algoritmo gerenciável não garante que um sistema multiagente seja verdadeiramente escalável.

[Juhász e Paul \(2002\)](#) avaliaram o desempenho e a escalabilidade de um sistema multiagente baseado no Protocolo de Rede de Contrato. Para isso, desenvolveram um sistema multiagente MAS e avaliaram os tempos de execução das tarefas sob diferentes cargas de trabalho. Os resultados destacam que o tempo de resposta do sistema está diretamente relacionado ao número de agentes e ao valor do prazo estabelecido. Por sua vez, [Korba e Song \(2002\)](#) abordam a escalabilidade como um aspecto fundamental no desenvolvimento de sistemas multiagente. Explicam que a escalabilidade pode ser alcançada aumentando os recursos (por exemplo, memória, processador) - escalabilidade vertical - ou reduzindo esses recursos - escalabilidade horizontal. Observa-se, no entanto, que a expansão é o foco mais prevalente na literatura. [Korba e Song \(2002\)](#) também identificam as seguintes características pelas quais os sistemas multiagente podem ser dimensionados: (i) aumento no número total de agentes em uma plataforma específica; (ii) aumento no número total de agentes em

vários sistemas ou plataformas; *(iii)* aumento no tamanho dos dados com os quais os agentes trabalham; *(iv)* maior diversidade de agentes envolvidos. Para avaliar essas características, [Korba e Song \(2002\)](#) modelaram e simularam a escalabilidade de uma aplicação multiagente em um cenário de mercado de trabalho usando a plataforma multiagente JADE. Suas principais conclusões incluem: *(i)* sistemas distribuídos superam sistemas monolíticos; *(ii)* a escalabilidade é mais eficiente quando os agentes são executados em um único bloco computacional; *(iii)* o tempo de processamento aumenta linearmente com o número de agentes; *(iv)* o aumento de blocos computacionais tem um impacto mínimo no tempo de processamento quando os agentes são executados no mesmo bloco. Além disso, [Korba e Song \(2002\)](#) enfatizam que a dinâmica dos sistemas multiagente é difícil de prever, especialmente em aplicações distribuídas em larga escala, nas quais o número de agentes pode variar consideravelmente ao longo do tempo. Portanto, é crucial dimensionar esses sistemas rapidamente sem comprometer significativamente o desempenho ou aumentar a complexidade administrativa. Eles também concluem que a escalabilidade é um aspecto importante, embora pouco investigado, em sistemas multiagente.

[Wang e De Wilde \(2007\)](#) apresentam um modelo de negociação em um contexto do mundo real. Neste estudo, desenvolvem um conjunto de redes de agentes interativos, que têm a capacidade de evoluir automaticamente ao longo do tempo, com base nos comportamentos dos agentes e nas configurações do sistema. Dentro deste modelo, os agentes são dotados de recursos e são recompensados por negociar esses recursos, sendo considerados como jogadores em um jogo. O trabalho propõe uma metodologia para investigar a escalabilidade de redes de agentes evolutivas e realiza uma análise abrangente com diferentes configurações de parâmetros para examinar os efeitos das ações dos agentes na escalabilidade do sistema. De acordo com [Wang e De Wilde \(2007\)](#), a carga e a complexidade são fatores centrais que influenciam a escalabilidade de um sistema multiagente. A carga abrange principalmente o uso de memória, processamento da CPU e comunicação, enquanto a complexidade computacional está diretamente relacionada aos algoritmos usados no sistema. No modelo que eles propõem, o foco está especificamente na carga da CPU como medida de escalabilidade. Para avaliar essa escalabilidade, eles levantam três pressupostos: *(i)* como o tempo de processamento de um único agente varia, dependendo do número fixo de agentes; *(ii)* como o crescimento da população de agentes afeta o desempenho; e *(iii)* como vários parâmetros do sistema afetam o desempenho em termos de escalabilidade. Os resultados obtidos demonstram que o modelo de negociação é, de fato, escalável, uma vez que o tempo de processamento aumenta linearmente com o aumento do número de agentes.

Em outra abordagem, [Lorig et al. \(2015\)](#) enfatizam a crescente importância das simulações de computador como técnica para gerar conhecimento em diversas áreas de pesquisa. Ao mesmo tempo, observam o aumento no tamanho dos modelos de simulação. Diante disso, desenvolvem um método para analisar e comparar a escalabilidade de estruturas de simulação baseadas em agentes, avaliando quatro estruturas específicas: *Aimpulse Spectrum*, JADE, MASON e Repast. [Lorig et al. \(2015\)](#) definem desempenho como a velocidade e eficiência de um sistema de computação, enquanto a escalabilidade é descrita como a capacidade de manter o desempenho à medida que a carga de trabalho aumenta. Um sistema que sofre uma queda significativa no desempenho com o aumento da carga de trabalho é considerado “não escalável”.

Lorig et al. (2015) testam a escalabilidade vertical dessas ferramentas, com base nas definições de escalabilidade conforme descrito por Deters (2001) e explicado com mais detalhes por Michael et al. (2007), que incluem: (i) “*scale-up*” ou escalabilidade vertical, que envolve a implantação de um servidor robusto com muitos núcleos de CPU e RAM suficiente; e (ii) “*scale-out*” ou escalabilidade horizontal, que, ao contrário, integra vários servidores menores em uma única unidade para distribuir a carga do sistema. Os resultados revelam que essas ferramentas nem sempre se mostram escaláveis, destacando a importância de avaliar cuidadosamente a escalabilidade em sistemas baseados em agentes.

Kamdar et al. (2018) apresentam uma revisão abrangente abordando vários aspectos de sistemas multiagente, elucidando os conceitos básicos e as diferentes definições encontradas na literatura. Este artigo enfatiza que uma abordagem de agente único pode levar a um único ponto de falha, ao contrário de sistemas multiagente, nos quais a descentralização se destaca. Kamdar et al. (2018) também destacam os benefícios e limitações do uso de MAS ao tentar identificar direções futuras de pesquisa, além de apresentar as dificuldades e desafios enfrentados na implementação de MAS, com ênfase na escalabilidade, um ponto crítico de pesquisa em sistemas multiagente. Kamdar et al. (2018) enfatizam que a escalabilidade é uma das capacidades computacionais mais relevantes atualmente, permitindo que os pesquisadores modelem redes maiores, com um grande número de agentes coordenando ações em uma única plataforma. No entanto, eles observam que a compreensão da escalabilidade em MAS ainda não está completamente esclarecida, tanto em relação ao aumento das dimensões do problema (número de agentes) quanto à diversidade (agentes de vários tipos). No contexto de sistemas multiagente, Rodrigues (2019) estuda agentes inteligentes em sistemas de protocolo de comunicação da internet. O artigo destaca o desenvolvimento de várias plataformas de MAS nos últimos anos, uma vez que o paradigma de implementar agentes para lidar com sistemas grandes e complexos em grande escala é considerado uma solução promissora na era do *big data*. O principal objetivo de Rodrigues (2019) é avaliar plataformas multiagente em termos de escalabilidade, desempenho e compatibilidade com outras tecnologias de transporte de mensagens em grande escala. O artigo descreve uma comparação entre várias plataformas existentes na literatura, apresentando um cenário realista no qual um MAS escalável pode ser adotado como solução, uma vez que os MAS devem ser capazes de oferecer uma implantação distribuída, uma característica essencial de sistemas escaláveis. Em relação à escalabilidade em MAS, Rodrigues (2019) argumenta que é crucial definir o conceito e a maneira de medir a escalabilidade. Este artigo define a escalabilidade como a relação entre o desempenho do sistema quando o sistema é expandido (aumento no número de agentes e/ou mensagens) e os recursos computacionais utilizados. Neste contexto, o foco está em identificar limitações em tecnologias para construir sistemas em grande escala que fazem uso de agentes.

Portanto, os estudos sobre escalabilidade se concentram em estruturas amplamente citadas na literatura, sendo a maioria dessas ferramentas responsáveis por gerenciar agentes responsáveis por enviar mensagens em servidores de internet. O interesse do trabalho atual, em relação à escalabilidade, é abordar tanto a expansão no número de agentes quanto a escalabilidade vertical. Essa abordagem é justificada, pois, à medida que o número de agentes aumenta, a demanda por recursos computacionais também aumenta. Além disso, a partir da revisão da literatura realizada,

não foram identificados artigos abordando escalabilidade em ambientes multiagente para resolver problemas de otimização combinatória.

A escalabilidade, neste trabalho, é analisada a partir da variação no número de agentes inseridos no *Framework* AMAM. Com isso, busca-se avaliar a capacidade do *framework* de manter seu desempenho e sua eficiência diante do aumento da quantidade de agentes envolvidos nos experimentos na Seção 4.1 e validados pela publicação do trabalho em [Silva et al. \(2025\)](#).

2.3 Meta-heurísticas e Aprendizado de Máquina

Uma forma de se buscar o equilíbrio entre o dilema da diversidade e o compartilhamento de informações é a criação de mecanismos de controle/aprendizado de soluções compartilhadas. Sistemas multiagente e agentes são passíveis de implementar técnicas de aprendizado de máquinas que podem gerenciar esse tipo de metodologia. Como esta tese envolve um *framework* multiagente para meta-heurísticas para solucionar problemas de otimização combinatória, uma junção entre técnicas de aprendizado de máquina e meta-heurísticas deve ser explorada e melhor compreendida.

As meta-heurísticas (MH) e o aprendizado de máquina (ML – *Machine Learning*) são dois campos de pesquisa fundamentais na área da inteligência artificial e na resolução de problemas complexos. Ao contrário das heurísticas, as meta-heurísticas são métodos gerais, não vinculados a problemas específicos. Elas representam estratégias de otimização baseadas em estratégias de trajetória e/ou populacional, nas quais, de um lado, não é possível garantir a otimalidade de uma solução encontrada, mas, de outro, é possível encontrar soluções aproximadas de boa qualidade. As meta-heurísticas são métodos que definem mecanismos para evitar que a busca fique presa em ótimos locais e são aplicadas a problemas de otimização da classe NP-Difícil, nos quais a utilização de métodos exatos é restrita. Um aspecto central de qualquer meta-heurística é o equilíbrio entre dois comportamentos complementares: a *diversificação* (*exploration*), que amplia a busca por regiões ainda inexploradas do espaço de soluções, e a *intensificação* (*exploitation*), que refina as melhores soluções já conhecidas aproveitando o conhecimento acumulado. O equilíbrio adequado entre diversificação e intensificação é determinante para a eficácia do método e reaparece, sob diferentes formas, ao longo deste trabalho ([Blum e Roli, 2003](#)).

Diversas meta-heurísticas podem ser encontradas na literatura. Essas técnicas são divididas em dois grupos: (i) algoritmos de busca populacional, como *Genetic Algorithm* (GA) ([Golberg, 1989](#)), *Ant Colony Optimization* (ACO) ([Dorigo et al., 1996](#)), PSO ([Kennedy e Eberhart, 1995](#)), dentre outros; (ii) algoritmos de trajetória, como *Iterated Local Search* (ILS) ([Lourenço et al., 2001](#)), *Variable Neighborhood Search* (VNS) ([Mladenović e Hansen, 1997](#)), *Greedy Randomized Adaptive Search Procedure* (GRASP) ([Feo e Resende, 1995](#)), *Tabu Search* (TS) ([Glover, 1990](#)), *Simulated Annealing* (SA) ([Kirkpatrick et al., 1983](#)), dentre outros. Para uma compreensão mais aprofundada sobre heurísticas e meta-heurísticas, recomendamos a leitura de [Blum e Roli \(2003\)](#); [Gaspar-Cunha et al. \(2012\)](#); [Silva \(2019\)](#).

O ML concentra-se no desenvolvimento de algoritmos e técnicas que permitem que sistemas computacionais aprendam e melhorem seu desempenho em uma tarefa específica com base na experiência obtida, em vez de serem explicitamente progra-

mados. Conforme [Bishop e Nasrabadi \(2006\)](#), no contexto de agentes, o aprendizado de máquina pode ser dividido em três áreas: (i) Aprendizagem Supervisionada (SL – *Supervised Learning*): modelo de aprendizagem no qual o agente possui a figura de um professor, o qual lhe apresenta exemplos de entradas e saídas desejáveis; (ii) Aprendizagem não Supervisionada (UL – *Unsupervised Learning*): modelo de aprendizagem no qual nenhum padrão de resultado é fornecido como referência, cabendo ao próprio agente encontrar meios para alcançar um resultado; e; (iii) Aprendizagem por Reforço (RL – *Reinforcement learning*): modelo de aprendizagem no qual o agente interage com um ambiente dinâmico, a fim de obter melhores resultados.

A união entre Aprendizado de Máquina e Meta-heurísticas tem como objetivo combinar a capacidade de aprendizado adaptativo do ML com a exploração robusta e a busca global das MH. O termo *busca global*, aqui e no restante do texto dessa tese, designa a *abrangência* da exploração, isto é, a capacidade de a meta-heurística percorrer regiões distantes do espaço de soluções e escapar de ótimos locais, em contraposição às buscas locais, restritas à vizinhança da solução corrente. Não se trata, portanto, de garantia de otimalidade: as meta-heurísticas aqui empregadas são métodos aproximados e não asseguram o encontro do ótimo global, nem oferecem certificado de qualidade em relação à solução devolvida. A avaliação dos resultados desta tese é feita por comparação com os melhores valores conhecidos da literatura, conforme o Capítulo 3. Essa junção pode ser aplicada a uma variedade de cenários, como seleção de operadores em algoritmos genéticos, ajuste de parâmetros de meta-heurísticas, seleção de etapas em hiper-heurísticas, entre outros. Esse campo é altamente relevante na área de inteligência artificial e otimização combinatória, e tem ganhado crescente interesse ([Karimi-Mamaghan et al., 2022](#)). Este interesse visa criar abordagens mais poderosas e versáteis para resolver uma ampla gama de problemas de otimização e tomadas de decisões complexas.

[Karimi-Mamaghan et al. \(2022\)](#) apresentam uma revisão abrangente sobre a integração de técnicas de aprendizado de máquina em meta-heurísticas para diversos propósitos. O foco está na seleção de algoritmos, avaliação de aptidão, geração de soluções iniciais, evolução, ajuste de parâmetros e cooperação entre algoritmos. A revisão destaca o aumento significativo no uso de técnicas de ML em MH ao longo das últimas duas décadas, demonstrando um aumento no conhecimento e na popularidade do tema. O trabalho também discute os desafios e as oportunidades futuras nessa área de pesquisa. Assim como em [Karimi-Mamaghan et al. \(2022\)](#), [da Costa Oliveira et al. \(2023\)](#) apresentam uma revisão da literatura sobre a junção de ML e MH. Essa revisão concentra-se na melhoria de meta-heurísticas aplicadas a problemas de otimização por meio do uso de aprendizado de máquina. A revisão analisa estudos publicados nos últimos anos e os categoriza com base nas técnicas de aprendizado de máquina, estratégias de validação e problemas reais de otimização. Os resultados mostram que algoritmos evolucionários auxiliados por ML são comumente utilizados para melhorar a eficiência da otimização, enquanto algoritmos de estimativa de distribuição são usados para aprimorar a eficácia da otimização. A revisão também destaca a predominância do valor da função objetivo como critério de avaliação para validar os algoritmos. Além disso, [Talbi \(2021\)](#) realizou uma investigação sobre a aplicação de aprendizado de máquina para o desenvolvimento de meta-heurísticas eficientes, eficazes e robustas, mostrando que essa área tem ganhado crescente popularidade. [Talbi \(2021\)](#) explora diversas oportunidades para a integração do ML em

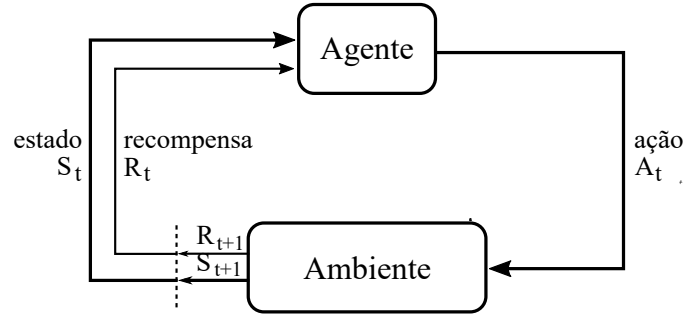


Figura 2.4: Modelo de Aprendizado por Reforço (Sutton e Barto, 2018).

MH e estabelece uma definição abrangente das várias formas de sinergias que podem ser alcançadas. Uma taxonomia detalhada é proposta com base nos componentes de busca abordados: o problema de otimização alvo e os elementos de meta-heurísticas de nível superior e inferior. O autor também motiva pesquisadores na área de otimização a incorporar conceitos de aprendizado de máquina em suas abordagens de MH e identifica questões de pesquisa em aberto nesse domínio, que demandam investigações mais profundas.

Esta tese tem como foco a área de aprendizado por reforço envolvendo sistemas multiagente para a solução de problemas de otimização combinatória. Algoritmos de aprendizado por reforço consistem em um agente tomar decisões com base em sua interação com o ambiente, para maximizar sua recompensa cumulativa ao longo do tempo. O agente aprende por tentativa e erro, ajustando suas ações com base nas recompensas recebidas. A Seção 2.3.1 a seguir detalha, nos termos do interesse da presente tese, as questões associadas a Aprendizado por Reforço.

2.3.1 Aprendizado por Reforço

O problema de aprendizagem por reforço consiste em alcançar metas por meio de recompensas cumulativas (Sutton e Barto, 2018). Uma recompensa é uma observação escalar especial R_t , emitida a cada passo de tempo t por um sinal de recompensa no ambiente, que fornece uma medida instantânea do progresso em direção a um objetivo. Uma instância do problema de aprendizado por reforço é definida por um ambiente, um agente, um estado S_t , um sinal de recompensa R_t , uma ação A_t e um objetivo cumulativo a ser maximizado, conforme apresentado na Figura 2.4.

Na Figura 2.4, observa-se que a ação A_t altera o estado do ambiente, e o valor dessa transição de estado é comunicada ao agente através de um sinal de reforço R_{t+1} . A próxima ação do agente é determinada pelo estado S_{t+1} . O comportamento do agente consiste em escolher ações que tendem a aumentar a soma dos valores de reforço a longo prazo. Um algoritmo de aprendizado por reforço amplamente difundido e de grande importância na literatura é o *Q-learning* (Watkins e Dayan, 1992). Esse algoritmo visa encontrar uma política ótima de tomada de decisões, não requer um modelo do ambiente e pode lidar com problemas transitórios e estocásticos sem modificações. A atualização do valor do estado/ação da tabela Q do *Q-learning* em um determinado tempo t é dada por:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (2.3)$$

em que a é a ação tomada e r corresponde ao valor da recompensa por mudar do estado atual s para o estado resultante s' . A alternância entre letras maiúsculas e minúsculas segue conforme proposto por Sutton e Barto (2018): S_t , A_t e R_t são *variáveis aleatórias*, representando o estado, a ação e a recompensa no passo t , respectivamente, enquanto s , a e r são os *valores* que elas assumem. Por isso, a interação da Figura 2.4 é descrita em maiúsculas e a regra de atualização, que opera sobre valores observados, em minúsculas. Os parâmetros α e γ representam, respectivamente, a taxa de aprendizagem e o fator de desconto. Esses dois parâmetros dependem diretamente do problema tratado. A função $Q(\cdot, \cdot)$ estima a utilidade esperada da tomada de uma ação em um determinado estado s . Um episódio é definido aqui como uma sequência de estados que vão desde um estado inicial até um estado final. A partir desses valores, um mapeamento de política de estado/ação é gerado. Para equilibrar a diversificação e a intensificação, um método de seleção de ação é o método ϵ -greedy. Este método escolhe a ação de maior valor para o estado atual com probabilidade $1 - \epsilon$ e escolhe uma ação aleatória com a probabilidade restante (Sutton e Barto, 2018).

O método ϵ -greedy não é, porém, o único método de seleção de ação, e a escolha entre tais métodos afeta diretamente o equilíbrio entre diversificação e intensificação. Sutton e Barto (2018) apresentam dois outros métodos. O primeiro é a seleção *softmax*, que substitui o sorteio uniforme entre ações não gulosas por uma distribuição de Boltzmann sobre os valores estimados, na forma:

$$\pi(a) = \frac{e^{Q(a)/\tau_B}}{\sum_{b \in \mathcal{K}} e^{Q(b)/\tau_B}}, \quad (2.4)$$

em que $\tau_B > 0$ é a *temperatura*: valores altos aproximam a distribuição da uniforme, favorecendo a diversificação, e valores baixos concentram a probabilidade na ação de maior valor. A vantagem sobre o ϵ -greedy é graduar a escolha pela diferença entre as estimativas, em vez de tratar todas as ações não gulosas como equivalentes. A segunda família é a dos índices de confiança superior, do tipo UCB, que dispensam a aleatoriedade e são tratados na Seção 2.3.2.1. Registre-se, para evitar confusão adiante, que o mecanismo denominado *softmax* na Seção 5.1 desta tese não corresponde ao método de seleção definido pela Expressão (2.4). Trata-se de um sorteio uniforme, restrito às soluções acima de um limiar de qualidade, conforme ali detalhado.

O aprendizado por reforço pode ser intensivo em termos de tempo de execução, uma vez que os algoritmos de aprendizagem precisam avaliar as consequências de suas ações a longo prazo, usando *feedback* ou recompensas atrasadas. Dessa forma, modelar as recompensas recebidas pelos agentes pode se tornar uma técnica eficaz para facilitar a atribuição de recompensas aos agentes, sem a necessidade de um conhecimento aprofundado sobre o problema em relação às necessidades do agente em resolvê-lo. A seguir, apresenta-se uma discussão a respeito de modelagens de recompensas.

2.3.1.1 Modelagem de Recompensas

A modelagem de recompensas é um método que incorpora o conhecimento do domínio no aprendizado por reforço, permitindo que os algoritmos se direcionem mais

rapidamente para soluções promissoras (Grzes, 2017). No entanto, modelar essas recompensas pode exigir conhecimento prévio do domínio ou a aplicação de técnicas de modelagem. Devlin et al. (2014) demonstram que modelagens de recompensas baseadas em diferenças e modelagens de recompensas baseadas em potencial podem melhorar significativamente a política conjunta aprendida por múltiplos agentes de aprendizado por reforço atuando simultaneamente no mesmo ambiente. No aprendizado por reforço de agente único, o agente busca maximizar apenas sua própria recompensa. Entretanto, quando esses agentes operam em um ambiente multiagente cooperativo, o foco do aprendizado deixa de ser puramente individual e passa a ser avaliado com base em uma função global. As duas principais maneiras de recompensar um agente são por meio de recompensas locais, que representam o estado direto do agente, ou recompensas globais, que refletem o desempenho do sistema como um todo (Agogino e Tumer, 2008). Os modelos de aprendizado por reforço aqui avaliados incluem:

- (i) **Recompensa Local:** uma Recompensa Local (L) é uma recompensa baseada na parte do sistema que representa o valor alcançado pelo agente de forma direta. O uso desse sinal de recompensa geralmente incentiva um comportamento “egoísta”, no qual o agente pode agir apenas direcionando suas ações para sua observação direta. Este modelo de recompensa pode levar a uma tomada de decisão que pode ir contra o objetivo do sistema, pois, enquanto tenta cegamente aumentar sua própria recompensa, um desempenho indesejado do sistema pode ocorrer.
- (ii) **Recompensa Global:** uma Recompensa Global (G) é dada através do valor de objetivo alcançado pelo sistema como forma de aprendizado. Isso encoraja o agente a agir no interesse do sistema, mas inclui uma quantidade substancial de ruído de outros agentes agindo simultaneamente. A própria contribuição de um agente para a recompensa global pode ser ofuscada pela contribuição de centenas de outros agentes, resultando em uma baixa relação sinal-ruído (Agogino e Tumer, 2008).
- (iii) **Recompensa por Diferenças:** uma forma de avaliar o quanto o agente está contribuindo para um sistema como um todo é através de modelos de recompensas por diferenças. Os modelos de recompensas por diferenças capturam a contribuição de um agente para o desempenho do sistema. Isso pode se tornar difícil devido ao fato dos demais agentes estarem atuando no ambiente, obscurecendo a contribuição individual do agente para o objetivo do sistema (Devlin et al., 2014). A Recompensa por Diferenças (D) é um sinal de recompensa moldado que ajuda um agente a aprender as consequências de suas ações no objetivo do sistema, removendo grande parte do ruído criado pelas ações de outros agentes ativos no sistema (Agogino e Tumer, 2008). A modelagem de recompensa por diferenças é descrita em Devlin et al. (2014) como:

$$D(z) = G(z) - G(z - i) \quad (2.5)$$

em que z é um termo geral representativo de estados ou pares estado-ação dependendo da aplicação, $G(z)$ é o termo global de desempenho do sistema, e

$G(z - i)$ é $G(z)$ para um sistema teórico sem a contribuição do agente i . Qualquer ação tomada para aumentar D consequentemente aumenta G , enquanto o impacto do agente i em sua própria recompensa é muito maior que seu impacto relativo em G (Devlin et al., 2014). Estas propriedades permitem que o modelo de recompensa por diferença aumente o desempenho do aprendizado em um sistema multiagente em um grau significativo. A recompensa por diferenças geralmente pode ser estimada, mesmo em domínios extremamente complexos. No entanto, no sistema em análise, D é diretamente calculável. Usar modelos de recompensas por diferenças diretamente ou estimá-los provou-se ser uma técnica eficaz para atribuição de crédito em muitas aplicações (Devlin et al., 2014).

A aprendizagem em sistemas multiagente é uma característica relevante, porém de difícil definição e implementação. Portanto, o aprendizado por reforço se destaca como uma técnica importante, visto que não requer um profundo conhecimento do problema. Um algoritmo de aprendizado por reforço amplamente difundido e de grande relevância na literatura é o *Multi-armed Bandit*. Este algoritmo será apresentado a seguir.

2.3.2 Algoritmo *Multi-armed Bandit*

O algoritmo *Multi-armed Bandit* (MaB) (Robbins, 1952) é um modelo de tomada de decisão sequencial que aborda o dilema entre diversificação e intensificação em problemas envolvendo incertezas. O nome *Bandit* é inspirado nas máquinas caça-níqueis, em que um jogador deve decidir em qual braço apostar para maximizar seus ganhos. No contexto do algoritmo *Multi-armed Bandit*, cada braço representa uma opção ou ação disponível, e cada braço possui uma recompensa associada, que é desconhecida para o tomador de decisão. O objetivo é encontrar uma estratégia de seleção de ações que maximize o retorno acumulado ao longo do tempo. A principal questão é que, ao selecionar uma ação, há um *trade-off* entre a diversificação (tentar braços com recompensas incertas para aprender mais sobre eles) e a intensificação (aproveitar o conhecimento existente para obter recompensas mais imediatas). O algoritmo MaB tem aplicações em diversas áreas, como publicidade *online*, otimização de sites, testes clínicos, alocação de recursos, entre outras. Ele fornece uma estrutura útil para resolver problemas de tomada de decisão sequencial com incerteza, buscando maximizar o retorno ao longo do tempo.

A formulação clássica do MaB pressupõe um ambiente *estacionário*: cada braço está associado a uma distribuição de recompensas fixa, cuja média não se altera ao longo do horizonte, e o conjunto de braços permanece o mesmo do início ao fim da dinâmica de aprendizagem. É essa hipótese que sustenta as garantias de convergência dos algoritmos apresentados a seguir, e, de outro lado, é o seu não atendimento que motiva as variantes apresentadas na Seção 2.3.2.1. Em Mahajan e Teneketzis (2008) é reforçado que os problemas MaB constituem uma categoria de desafios envolvendo a alocação sequencial de recursos entre múltiplos projetos concorrentes. Estes problemas destacam um conflito intrínseco entre tomar decisões que resultam em recompensas imediatas mais elevadas versus optar por ações que, embora possam reduzir ganhos imediatos, possibilitem perspectivas de recompensas mais substanciais no futuro. Além disso, Mahajan e Teneketzis (2008) concluem que, em geral,

Algoritmo 2 *Um algoritmo bandit simples*

```

1: função bandit
2:   Inicialize, para  $a = 1$  até  $k$ :
3:      $Q(a) \leftarrow \text{valor inicial}$ ;
4:      $N(a) \leftarrow \text{valor inicial}$ ;
5:   Repita por um número  $T$  de passos:
6:      $A \leftarrow \begin{cases} \arg \max_a Q(a) \text{ com probabilidade } 1 - \epsilon \text{ (desempate aleatório);} \\ \text{uma ação aleatória com probabilidade } \epsilon \end{cases}$ 
7:      $R \leftarrow \text{bandit}(A)$ 
8:      $N(A) \leftarrow N(A) + 1$ ;
9:      $Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$ ;
10: fim função

```

problemas de alocação sequencial de recursos podem ser resolvidos por programação dinâmica, porém estes algoritmos sofrem com a “maldição” da dimensionalidade. Desta forma, os problemas MaB se destacam por serem computacionalmente menos intensos. [Slivkins \(2019\)](#) destaca que o MaB possui uma estrutura simples, porém muito poderosa, e lida com decisões ao longo do tempo e sob incertezas, além de destacar que são interdisciplinares e serem foco de várias pesquisas em torno do assunto durante os últimos anos.

O Algoritmo 2 mostra a estrutura clássica do Algoritmo *Multi-armed Bandit* ([Sutton e Barto, 2018](#)). Nas linhas 3 e 4 inicializa-se os valores de q e do número de visitas n de cada ação. A escolha de ação pelo agente é dada utilizando ϵ -greedy, conforme a linha 6. A recompensa é determinada a partir da ação A do agente na linha 7. As linhas 8 e 9 mostram a atualização dos valores de n e q , respectivamente. Esse processo de escolha da ação, obtenção da recompensa e atualização dos valores é repetido por um número T de passos, adotado como critério de parada do algoritmo. Observa-se, na linha 9, que a atualização é dada pelo número de visitas ocorridas naquela ação, sendo essa forma de atualização utilizada para problemas estacionários. Para problemas não estacionários, o fator de visitas $\frac{1}{N(A)}$ é substituído por um parâmetro de desconto α , para o qual geralmente adota-se o valor $\alpha = 0.99$. Este parâmetro é responsável por diminuir o valor de referência de ações antigas. Assim, a nova forma de atualização é:

$$Q(A) \leftarrow Q(A) + \alpha [R - Q(A)] \quad (2.6)$$

Pode-se observar que o fator α substituiu o termo de repetições de escolha $\frac{1}{N(A)}$ na Equação (2.6) e, desse modo, o algoritmo de *Multi-armed Bandit* pode tratar problemas não estacionários, conforme apresentado em [Sutton e Barto \(2018\)](#).

2.3.2.1 Variantes do *Multi-armed Bandit*

Na literatura, discutem-se inúmeras variações e adaptações do algoritmo *Multi-armed Bandit*, de acordo com as diferentes características dos problemas aos quais ele é aplicado. Essas características podem incluir a estacionariedade ou não do problema, alterações nos valores das recompensas, sejam contínuas ou abruptas, bem como a existência de braços fixos ou variáveis. A Tabela 2.1 sintetiza os principais

Tabela 2.1: Algoritmos *Multi-armed Bandit*.

Algoritmo	Ambiente Não Estacionário	Recompensa			Multiagente
		Fixa	Variável	Varição Abrupta	
Clássico (Robbins, 1952)		✓			
UCB (Auer, 2002)		✓			
DUCB (Kocsis e Szepesvári, 2006)	✓	✓			
SWUCB (Garivier e Moulines, 2011)	✓		✓		
FRRMaB ⁽¹⁾ (Li et al., 2014)	✓		✓		
FRAMaB ⁽²⁾ (Ferreira et al., 2017)	✓			✓	
Multi-player MaB (Liu e Zhao, 2010)		✓			✓
Multi-player UCB (Chang et al., 2021)		✓			✓
Cooperative UCB (Landgren et al., 2021)		✓			✓

⁽¹⁾FRRMaB: *Fitness-Rate-Rank based Multi-armed Bandit*;

⁽²⁾FRAMaB: *Fitness-Rate-Average based Multi-armed Bandit*.

algoritmos MaB, indicando, para cada um, se trata ambientes não estacionários, a natureza da função de recompensa e se possui formulação multiagente.

O UCB (*Upper Confidence Bound*) (Auer, 2002) equilibra diversificação e intensificação de forma determinística, sem depender de um parâmetro de diversificação aleatória como o ϵ -greedy. Ele seleciona o braço com maior potencial considerando não apenas a recompensa média estimada, mas também a incerteza associada a cada braço. Braços menos explorados recebem um bônus de diversificação maior, sendo priorizados até que suas estimativas se tornem mais confiáveis. O índice UCB_a de um braço a é calculado por:

$$UCB_a = \bar{X}_a + \sqrt{\frac{2 \ln t}{N_a}}, \quad (2.7)$$

em que $\bar{X}_a$ é a recompensa média estimada do braço a ; t é o número total de rodadas realizadas; e N_a é o número de vezes que o braço a foi selecionado. Esse índice não está definido enquanto $N_a = 0$, pois o radicando é uma divisão por zero. Adota-se a convenção usual de atribuir índice infinito a todo braço ainda não selecionado, de modo que as primeiras K rodadas percorrem uma vez cada braço do conjunto $\mathcal{K}$ e, a partir daí, $N_a \geq 1$ para todo a . A mesma convenção vale para o DUCB e para o SWUCB, com a ressalva de que, no SWUCB, um braço pode voltar a ter contagem nula quando sai da janela W , situação em que ele é novamente priorizado.

O *Discounted Upper Confidence Bound* (DUCB) (Kocsis e Szepesvári, 2006) é uma extensão do UCB para ambientes não estacionários. O DUCB introduz um fator de desconto $\gamma \in (0, 1]$ que reduz progressivamente o peso das interações passadas, de modo que as observações mais recentes influenciem mais as estimativas atuais. O índice DUCB para o braço a é:

$$DUCB_a = \bar{X}_a^{(t)} + \sqrt{\frac{2 \ln(S_t)}{N_a^{(t)}}}, \quad (2.8)$$

em que:

$$\bar{X}_a^{(t)} = \frac{\sum_{j=1}^t \gamma^{t-j} R_j \cdot \mathbf{1}[A(j) = a]}{\sum_{j=1}^t \gamma^{t-j} \cdot \mathbf{1}[A(j) = a]}$$

é a recompensa média descontada; $S_t = \sum_{j=1}^t \gamma^{t-j}$ é a soma acumulada dos pesos de desconto; e $N_a^{(t)} = \sum_{j=1}^t \gamma^{t-j} \cdot \mathbf{1}[A(j) = a]$ é a contagem efetiva de seleções com desconto. O símbolo $\mathbf{1}[\cdot]$ denota a função indicadora, que assume valor 1 quando a condição entre colchetes é satisfeita, e valor 0 caso contrário; assim, $\mathbf{1}[A(j) = a]$ vale 1 apenas nas rodadas j em que o braço a foi o braço escolhido, de modo que ambos os somatórios percorrem todas as rodadas mas acumulam somente aquelas em que a foi selecionado.

O *Sliding-Window Upper Confidence Bound* (SWUCB) (Garivier e Moulines, 2011) também trata ambientes não estacionários, mas por uma estratégia distinta: considera apenas as recompensas observadas dentro de uma janela deslizante de tamanho W , descartando completamente as interações anteriores. Essa abordagem é especialmente adequada para ambientes com mudanças abruptas. O índice SWUCB para o braço a é:

$$SWUCB_a = \bar{X}_a^{(t,W)} + \sqrt{\frac{2 \ln t}{N_a^{(t,W)}}}, \quad (2.9)$$

em que $\bar{X}_a^{(t,W)}$ é a recompensa média do braço a nos últimos W passos e $N_a^{(t,W)}$ é o número de seleções do braço a dentro da janela. Em síntese, UCB, DUCB e SWUCB formam uma família de algoritmos baseados em índices de confiança, diferindo pela forma como tratam o histórico: o UCB pondera todo o histórico uniformemente, o DUCB aplica desconto exponencial e o SWUCB restringe o histórico a uma janela fixa.

O *Fitness-Rate-Rank based Multi-armed Bandit* (FRRMaB) (Li et al., 2014) e o *Fitness-Rate-Average based Multi-armed Bandit* (FRAMaB) (Ferreira et al., 2017) são variantes desenvolvidas no contexto de hiper-heurísticas, em que os braços representam operadores de busca e as recompensas refletem a melhoria de qualidade produzida por cada operador. O FRRMaB utiliza uma janela deslizante para armazenar as melhorias recentes e atribui créditos com base em um ranqueamento com decaimento, favorecendo operadores com melhor desempenho recente:

$$\begin{aligned} \text{Reward}_h &= \sum_{j=1}^W \delta_{h,j}, \\ \text{Decay}_h &= D^{\text{Rank}_h} \cdot \text{Reward}_h, \\ R_{h,t} &= \frac{\text{Decay}_h}{\sum_{k=1}^K \text{Decay}_k}, \end{aligned}$$

em que $\delta_{h,j}$ é a taxa de melhoria da heurística h na posição j da janela; W é o tamanho da janela; $D \in [0,1]$ é o fator de decaimento; Rank_h é a posição de h no ranqueamento; e K é o número total de heurísticas. O FRAMaB substitui o ranqueamento com decaimento pela média das taxas de melhoria na janela, produzindo uma atribuição

de crédito mais estável:

$$R_{h,t} = \frac{\sum_{j=1}^w \delta_{h,j}}{N_{h,t}},$$

$$Q_{h,t} = R_{h,t} \cdot \frac{1}{N_{h,t}},$$

em que $N_{h,t}$ é o número de vezes que a heurística h aparece na janela no instante t .

A combinação de sistemas de otimização usando meta-heurísticas com algoritmos *Multi-armed Bandit* surge como uma abordagem promissora para integrar técnicas de aprendizado de máquina e estruturas de meta-heurísticas e é um tema de interesse para o presente trabalho. Nesse sentido, a seguir, apresenta-se uma breve lista de trabalhos relacionados.

Ferreira et al. (2017) apresentam um estudo sobre o comportamento de estratégias de seleção de MaB para hiper-heurísticas. Experimentos são conduzidos utilizando o *Framework HyFlex* e compararam o desempenho de três algoritmos MaB. Os resultados demonstraram que a estratégia de seleção MaB pode ser bem-sucedida, porém é necessária uma exploração adicional para aprimorar sua sensibilidade às configurações de parâmetros. A abordagem proposta foi comparada com hiper-heurísticas da literatura e obteve resultados favoráveis. Já em de Oliveira et al. (2018), um estudo de caso sobre a escolha de rotas em redes de tráfego utilizando algoritmos de aprendizado por reforço é apresentada. O problema é modelado como um MaB, em que os motoristas devem aprender a escolher rotas que minimizem seus tempos de viagem. O objetivo do estudo é comparar o desempenho de diferentes algoritmos MaB com o algoritmo *Q-learning*. Os resultados obtidos mostram que o *Q-learning* apresentou os melhores resultados, mas o algoritmo MaB também teve um desempenho interessante. O estudo sugere que mais pesquisas são necessárias para explorar outros algoritmos de MaB e desenvolver funções de recompensa mais adequadas para o problema da escolha de rotas. Para uma discussão a respeito, veja, por exemplo, Huanca-Anquise et al. (2023).

Algoritmos de Seleção de Operador *Multi-armed Bandit* (MaB-OS) são propostos em Meidani et al. (2022) com o intuito de selecionar, de forma inteligente, o otimizador mais adequado para cada problema de otimização durante o processo de busca. A estrutura utiliza o MaB para equilibrar a intensificação e a diversificação na seleção da regra de atualização do algoritmo com a melhoria estimada mais significativa na solução. Experimentos são conduzidos com três algoritmos base e mostram que o MaB-OS pode melhorar significativamente o desempenho dos algoritmos base, combinando suas vantagens de maneira sistemática. Zhang et al. (2023) apresentam uma nova hiper-heurística de seleção, baseada em MaB, para resolver problemas de otimização multiobjetivo. A hiper-heurística proposta é um processo iterativo, que inclui três componentes principais: (i) a estratégia de equilíbrio de recompensa; (ii) a estratégia de aprendizado baseada em MaB; e; (iii) a estratégia de seleção em múltiplos estágios. O desempenho e o comportamento da hiper-heurística proposta são investigados em uma variedade de *benchmarks*, bem como em problemas do mundo real. Os resultados empíricos demonstram a eficácia e a capacidade de aplicação em diferentes domínios da hiper-heurística proposta, comparando-o com vários algoritmos da literatura. Lagos e Pereira (2023) buscam a integração de métodos de Aprendizado

de Máquina e Problemas de Otimização Combinatória utilizando meta-heurísticas. Especificamente, o estudo explora o uso de Hiper-Heurísticas para resolver problemas de otimização com um componente de aprendizado. O algoritmo proposto combina um modelo de Markov com um método baseado no MaB para selecionar e aplicar heurísticas de baixo nível. [Lagos e Pereira \(2023\)](#) apresentam várias melhorias na meta-heurística, incluindo o uso de pesos exponenciais para diversificação e intensificação como método de aprendizado. A eficácia do algoritmo proposto é demonstrada por meio de um estudo computacional.

2.3.3 Aprendizado por Reforço Multiagente

O Aprendizado por Reforço Multiagente (MARL) é um campo de pesquisa ativo e promissor na área de Inteligência Artificial e Aprendizado de Máquina. Ele lida com sistemas compostos por vários agentes autônomos que interagem em um ambiente compartilhado e buscam aprender como agir para alcançar objetivos individuais ou coletivos. Nesse contexto, os agentes devem tomar decisões autônomas, levando em consideração não apenas seu próprio desempenho, mas, também, o impacto de suas ações nos outros agentes e no sistema como um todo ([Buşoniu et al., 2008, 2010](#); [Canese et al., 2021](#); [Zhu et al., 2022](#)).

O aprendizado por reforço é a base dessa abordagem, em que os agentes exploram seu ambiente, tomam ações e recebem *feedback* na forma de recompensas. Eles procuram aprender políticas de ação que maximizem a recompensa total ao longo do tempo ([Sutton e Barto, 2018](#)). No entanto, em cenários multiagente, a dinâmica se torna mais complexa, uma vez que as ações de um agente afetam diretamente as oportunidades e recompensas dos outros. Os desafios do MARL incluem a necessidade de equilibrar a competição e cooperação entre agentes. Agentes podem competir por recursos limitados ou cooperar para alcançar metas comuns. Além disso, os agentes podem compartilhar informações e estratégias para acelerar o aprendizado e melhorar o desempenho ([Leibo et al., 2017](#)). Em sistemas multiagente, o comportamento global pode emergir a partir das ações individuais, tornando imprevisível o resultado final ([Lowe et al., 2017](#)).

Outro desafio envolve o comportamento emergente de sistemas multiagente, em que o comportamento global pode ser imprevisível e surgir a partir das interações individuais dos agentes ([Iqbal e Sha, 2019](#)). À medida que os agentes aprendem, podem desenvolver estratégias complexas, tornando o aprendizado ainda mais desafiador ([Foerster et al., 2016](#)). Além disso, é essencial que os agentes equilibrem a diversificação e a intensificação, escolhendo entre intensificar ações que já conhecem bem e diversificar a busca rumo a novas ações ([Tampuu et al., 2017](#)).

O MARL tem aplicações em uma ampla variedade de campos, como robótica, jogos, economia, sistemas autônomos e muito mais ([Tan, 1993](#)). É uma área de pesquisa ativa, que envolve o desenvolvimento de novos algoritmos, abordagens teóricas e experimentos computacionais para lidar com a complexidade de sistemas multiagente. À medida que a Inteligência Artificial continua a evoluir, o MARL desempenhará um papel cada vez mais importante no desenvolvimento de sistemas inteligentes e cooperativos.

2.3.4 Algoritmo *Multi-armed Bandit* Multiagente

A formulação clássica do MaB, apresentada na Seção 2.3.2, considera um único agente interagindo com o ambiente. O problema *Multi-Agent Multi-armed Bandit* estende essa formulação para um cenário com múltiplos agentes, nos quais diferentes agentes interagem com um mesmo conjunto de braços e realizam escolhas sucessivas em um ambiente compartilhado. Essa modelagem permite representar situações em que o aprendizado não depende apenas de um único agente, mas de um conjunto de agentes que tomam decisões simultaneamente (Liu e Zhao, 2010; Landgren et al., 2021).

Antes da formulação, cabe uma observação sobre estacionariedade, pois a extensão multiagente introduz uma fonte de não estacionariedade que não existe no caso de agente único. No caso individual, o ambiente só deixa de ser estacionário se as distribuições de recompensa dos braços mudarem por causas externas ao agente. No caso multiagente, mesmo que essas distribuições permaneçam fixas, o ambiente percebido por cada agente é *intrinsecamente não estacionário*, porque os demais agentes também aprendem: a recompensa que o agente i observa ao selecionar um braço depende do que os outros agentes selecionaram e do que aprenderam até ali. São, portanto, duas fontes distintas, e elas se somam na instanciiação desta tese: o conjunto de braços do *Pool* muda ao longo da execução, conforme a Seção 5.2, e a política dos demais agentes evolui simultaneamente. A formulação a seguir, como é usual na literatura, é apresentada no cenário estacionário; o tratamento da não estacionariedade fica a cargo das políticas de seleção.

Considere um conjunto de N agentes $\mathcal{N} = \{1, 2, \dots, N\}$ e um conjunto de K braços $\mathcal{K} = \{1, 2, \dots, K\}$. A cada instante de tempo t , cada agente $i \in \mathcal{N}$ seleciona um braço $A_i(t) \in \mathcal{K}$ e recebe uma recompensa $R_i(t)$ associada à escolha realizada. Assume-se que a recompensa de cada braço $a \in \mathcal{K}$ segue uma distribuição de probabilidade com média desconhecida μ_a , de modo que as estimativas $Q_i(a)$ buscam aproximar esse valor ao longo do tempo. Cada agente mantém ainda um contador $N_i(a)$, que indica quantas vezes o braço a foi selecionado pelo agente i .

A seleção dos braços é realizada por meio da estratégia ϵ -greedy: cada agente escolhe o braço com maior valor estimado com probabilidade $1 - \epsilon$ ou seleciona aleatoriamente um braço com probabilidade ϵ . Formalmente, a ação do agente i no instante t é definida por:

$$A_i(t) = \begin{cases} \arg \max_{a \in \mathcal{K}} Q_i(a), & \text{com probabilidade } 1 - \epsilon, \\ a \sim \mathcal{U}(\mathcal{K}), & \text{com probabilidade } \epsilon, \end{cases} \quad (2.10)$$

em que $\mathcal{U}(\mathcal{K})$ representa uma escolha aleatória uniforme entre os braços disponíveis. Após a escolha do braço $A_i(t)$, o número de seleções e o valor estimado do braço são atualizados por:

$$\begin{aligned} N_i(A_i(t)) &\leftarrow N_i(A_i(t)) + 1 \\ Q_i(A_i(t)) &\leftarrow Q_i(A_i(t)) + \frac{1}{N_i(A_i(t))} [R_i(t) - Q_i(A_i(t))]. \end{aligned}$$

O objetivo do sistema cooperativo é maximizar a recompensa acumulada conjunta

dos agentes ao longo de um horizonte de tempo T :

$$\max \mathbb{E}[G(T)] = \max \mathbb{E} \left[\sum_{t=1}^T \sum_{i=1}^N R_i(t) \right]. \quad (2.11)$$

De forma equivalente, o problema pode ser formulado como a minimização do arrependimento coletivo (*regret*), definido como a diferença entre a recompensa acumulada pela política ótima e a recompensa efetivamente acumulada:

$$\mathcal{R}(T) = T \sum_{a \in \mathcal{K}^*} \mu_a - \mathbb{E}[G(T)], \quad (2.12)$$

em que $\mathcal{K}^* \subset \mathcal{K}$, com $|\mathcal{K}^*| = N$, representa o subconjunto dos N braços com maiores médias esperadas, correspondendo à alocação ótima em que cada agente ocupa um braço distinto. Minimizar $\mathcal{R}(T)$ é equivalente a maximizar $\mathbb{E}[G(T)]$, sendo o *regret* uma métrica amplamente utilizada para avaliar o desempenho de algoritmos MaB (Landgren et al., 2021).

Na formulação com escolha global, considera-se uma estimativa única de valor $Q(a)$ construída a partir das informações disponíveis no sistema, de modo que a decisão considera o comportamento coletivo dos agentes. Nesse caso, após a escolha de um braço $A(t)$, as recompensas dos agentes são agregadas pela média aritmética $R(t) = \frac{1}{N} \sum_{i=1}^N R_i(t)$, e o valor estimado global é atualizado de forma análoga ao caso monoagente. A principal diferença em relação à formulação monoagente está em que cada agente atualiza suas próprias estimativas $Q_i(a)$ com base nas recompensas que observa, sem considerar mecanismos adicionais de comunicação ou coordenação (Sutton e Barto, 2018; Liu e Zhao, 2010; Landgren et al., 2021).

2.3.4.1 Variantes multiagente do *Multi-armed Bandit*

No contexto multiagente, o aprendizado distribuído entre agentes introduz desafios adicionais em relação à formulação clássica. Em particular, a atualização de estimativas por um agente pode interferir negativamente no desempenho dos demais, seja por propagar estimativas imprecisas, seja por induzir a seleção de braços subótimos no nível do sistema (Liu e Zhao, 2010). Isso torna necessário o desenvolvimento de técnicas que promovam um aprendizado eficiente em nível sistêmico, e não apenas individual. Nesse sentido, ao projetar um sistema multiagente, três aspectos centrais devem ser considerados: (i) o grau de centralização das decisões, (ii) o nível de cooperação entre agentes e (iii) os mecanismos de compartilhamento de informações (Landgren et al., 2021).

Diversos algoritmos MaB possuem extensões para o cenário multiagente. No caso do UCB, a formulação pode ser adaptada para o *Multi-player UCB* (Chang et al., 2021), no qual múltiplos agentes aprendem simultaneamente a partir de interações com os mesmos braços, seguindo o índice UCB individual:

$$A_i(t) = \arg \max_{a \in \mathcal{K}} \left[Q_i(a) + \sqrt{\frac{2 \ln t}{N_i(a)}} \right]. \quad (2.13)$$

Em abordagens cooperativas, a comunicação entre agentes desempenha papel central, pois possibilita o compartilhamento de recompensas observadas e estimativas

de valor. Nesse contexto, destaca-se o *Cooperative UCB* (Landgren et al., 2021), que agrega informações de múltiplos agentes para reduzir exploração redundante e acelerar a convergência. As estimativas são construídas globalmente a partir das observações de todos os agentes:

$$\bar{Q}(a, t) = \frac{\sum_{i=1}^N \sum_{j \leq t: A_i(j)=a} R_i(j)}{\sum_{i=1}^N N_i(a, t)}, \quad (2.14)$$

de modo que todos os agentes utilizem uma estimativa comum dos valores esperados dos braços. O índice UCB cooperativo assume então a forma:

$$A_i(t) = \arg \max_{a \in \mathcal{K}} \left[\bar{Q}(a, t) + \sqrt{\frac{2 \ln t}{\sum_{i=1}^N N_i(a, t)}} \right], \quad (2.15)$$

o que reduz a exploração redundante e pode acelerar a convergência para uma alocação eficiente dos braços. A cooperação, nesse sentido, aproxima o comportamento do sistema do de um *bandit* centralizado, ainda que a implementação permaneça distribuída (Landgren et al., 2021). Essas formulações constituem a base sobre a qual a adaptação proposta neste trabalho é desenvolvida, conforme detalhado no Capítulo 5.

Neste trabalho, algoritmos baseados em *Multi-armed Bandit* são implementados com o objetivo de aprimorar o processo de cooperação entre os agentes no *Framework AMAM*. Nesse contexto, algoritmos de aprendizado multiagente baseado em MaB são apresentados na Capítulo 5.

2.3.5 Aprendizado por Reforço Multiobjetivo

As formulações clássicas de aprendizado por reforço apresentadas nas Seções 2.3.1 e 2.3.3 assumem que o desempenho do agente em uma trajetória pode ser sintetizado por um único sinal escalar de recompensa. Em diversas aplicações práticas, no entanto, o agente precisa atender a múltiplos critérios simultaneamente, com objetivos potencialmente conflitantes entre si. Em problemas de roteamento de veículos, por exemplo, a qualidade de uma solução pode ser avaliada tanto pela distância total percorrida quanto pelo número de veículos utilizados ou pela diversidade estrutural em relação a soluções já conhecidas. Reduzir essas dimensões a um único valor escalar exige a definição prévia de pesos ou funções de agregação, decisão que carrega o risco de ocultar compromissos relevantes entre os objetivos e de comprometer a generalidade do método de aprendizagem (Roijers et al., 2013).

O aprendizado por reforço multiobjetivo (*Multi-Objective Reinforcement Learning* – MORL) generaliza a formulação clássica ao substituir o sinal escalar de recompensa por um *vetor* de recompensas, no qual cada componente corresponde a um objetivo distinto. Formalmente, o ambiente de aprendizado é descrito por um Processo de Decisão de Markov Multiobjetivo (*Multi-Objective Markov Decision Process* – MOMDP), definido pela tupla $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathbf{R}, \gamma \rangle$, em que $\mathcal{S}$ é o conjunto de estados; $\mathcal{A}$ o conjunto de ações; $\mathcal{T}$ a função de transição; $\gamma \in [0, 1]$ o fator de desconto; e $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ a função de recompensa vetorial associada a d objetivos (Roijers

et al., 2013; Van Moffaert e Nowé, 2014). O valor de uma política π deixa, então, de ser representado por um escalar $V^\pi(s)$ e passa a ser um vetor:

$$\mathbf{V}^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \mathbf{R}(s_t, a_t, s_{t+1}) \mid s_0 = s, \pi \right], \quad (2.16)$$

cujas componentes refletem o desempenho esperado da política em cada um dos objetivos individualmente.

A presença de múltiplos critérios remove a ordem total entre as políticas, pois a comparação entre vetores deixa de ser unívoca. Para retomar uma relação de comparação, utiliza-se a noção de *dominância de Pareto*: dada duas políticas π e π' , diz-se que π domina estritamente π' no estado s , denotado por $\mathbf{V}^\pi(s) \succ \mathbf{V}^{\pi'}(s)$, se $\mathbf{V}^\pi(s)$ é maior ou igual a $\mathbf{V}^{\pi'}(s)$ em todos os componentes e estritamente maior em pelo menos um (Roijers et al., 2013). O conjunto das políticas não dominadas forma a *frente de Pareto* do problema, que representa o conjunto dos melhores compromissos atingíveis entre os objetivos considerados. O objetivo do agente em MORL passa a ser, portanto, aproximar essa frente de Pareto ou selecionar políticas pertencentes a ela, conforme alguma preferência adicional do tomador de decisão.

A literatura organiza as abordagens de MORL em duas grandes famílias, distinguidas pelo momento em que as preferências entre objetivos são incorporadas ao aprendizado (Roijers et al., 2013; Hayes et al., 2022):

- (i) Abordagens de *escalarização a priori*: o vetor de recompensa é convertido em um escalar antes do processo de aprendizado, por meio de uma função de utilidade (frequentemente uma soma ponderada $\mathbf{w}^\top \mathbf{R}$). Essa estratégia recupera os algoritmos clássicos de aprendizado por reforço, mas exige que os pesos sejam conhecidos antecipadamente e produz, em geral, apenas uma política sobre a frente de Pareto.
- (ii) Abordagens *baseadas em Pareto*: o agente aprende e mantém estimativas vetoriais de valor, deixando a decisão sobre o ponto preferido da frente para uma etapa posterior. Essas abordagens recuperam um conjunto de políticas não dominadas em uma única execução, sendo particularmente adequadas a contextos em que as preferências do decisor não são conhecidas antecipadamente ou variam ao longo do tempo.

Entre as abordagens baseadas em Pareto, o algoritmo *Pareto Q-learning* (PQL), proposto por Van Moffaert e Nowé (2014), estende o algoritmo *Q-learning* clássico ao manter, em vez de um único valor $Q(s,a) \in \mathbb{R}$, um conjunto de vetores $\hat{Q}_{set}(s,a) \subseteq \mathbb{R}^d$ associado a cada par estado-ação. Esse conjunto representa as estimativas vetoriais de retorno alcançáveis a partir de (s,a) e é atualizado de forma a preservar apenas os elementos não dominados, recuperando, ao final do processo, a aproximação da frente de Pareto. Versões *stateless* do PQL, adequadas a problemas em que a interação ocorre em um único estado, mantêm apenas um vetor $\mathbf{Q}(a) \in \mathbb{R}^d$ por ação e atualizam essa estimativa por meio de mistura convexa entre o vetor atual e a recompensa vetorial observada (Van Moffaert e Nowé, 2014). Essa formulação será adotada no Capítulo 6 como base para o aprendizado multiobjetivo sobre o *Pool* de soluções da AMAM.

No contexto multiagente, o MORL apresenta desafios adicionais decorrentes da combinação entre não estacionariedade, decorrente da interação entre agentes, e da relação de dominância parcial induzida pelo vetor de recompensas. [Huanca-Anquise et al. \(2023\)](#) discutem essas dificuldades em cenários de aprendizado multiagente multiobjetivo aplicados a problemas de escolha de rotas, demonstrando que estratégias adaptativas baseadas em dominância de Pareto podem coexistir com mecanismos de cooperação entre agentes sem exigir escalarização explícita. Esse cenário é particularmente próximo ao explorado neste trabalho, no qual o *Pool* cooperativo da AMAM é o ambiente compartilhado e o aprendizado vetorial é construído sobre as recompensas observadas pelos diversos agentes durante o processo de busca.

2.3.6 Algoritmo *Multi-armed Bandit* Multiobjetivo

A formulação clássica do MaB apresentada na Seção 2.3.2 assume que cada braço $a \in \mathcal{K}$ retorna uma recompensa escalar $R(a) \in \mathbb{R}$, sobre a qual o agente constrói estimativas $Q(a)$ utilizadas em sua política de seleção. Em muitos contextos práticos, contudo, a qualidade de uma escolha não pode ser sintetizada por um único valor: a recompensa associada a um braço é, na verdade, um vetor de critérios potencialmente conflitantes. O *Multi-Objective Multi-armed Bandit* (MOMaB), originalmente proposto por [Drugan e Nowé \(2013\)](#), estende a formulação clássica a esse cenário, substituindo a recompensa escalar por uma recompensa vetorial $\mathbf{R}(a) \in \mathbb{R}^d$, em que d é o número de objetivos considerados, e adaptando os mecanismos de seleção e atualização de forma análoga ao que ocorre no MORL discutido na Seção 2.3.5. Essa formulação foi posteriormente aplicada e estendida a outros domínios, como em [Huanca-Anquise et al. \(2023\)](#), no contexto de jogos repetidos e escolha de rotas. Cabe esclarecer que, diferentemente da política π otimizada no MORL, cuja frente de Pareto é constituída por um conjunto de políticas não dominadas, no contexto do MOMaB o termo π designa a regra de seleção de braço aplicada sobre as estimativas, sendo a frente de Pareto definida sobre os braços, e não sobre as políticas. No caso *stateless* aqui considerado, em que a interação ocorre em um único estado, ambos os conceitos coincidem, uma vez que uma política sobre estados se reduz a uma regra de seleção de ações.

Considere um conjunto de K braços $\mathcal{K} = \{1, 2, \dots, K\}$ em que cada braço a está associado a uma distribuição de probabilidade com vetor de médias desconhecidas $\boldsymbol{\mu}_a \in \mathbb{R}^d$. A cada instante t , o agente seleciona um braço $A(t) \in \mathcal{K}$ e observa uma recompensa vetorial $\mathbf{R}(t) \in \mathbb{R}^d$, cuja média é $\boldsymbol{\mu}_{A(t)}$. As estimativas do agente sobre cada braço passam, então, a ser vetoriais:

$$\bar{\mathbf{Q}}(a, t) = \frac{1}{N(a, t)} \sum_{j \leq t: A(j)=a} \mathbf{R}(j), \quad (2.17)$$

em que $N(a, t)$ é o número de vezes que o braço a foi selecionado até o instante t . A ausência de ordem total entre vetores de $\mathbb{R}^d$ impede a definição de uma noção única de braço ótimo, e a comparação entre as estimativas passa a ser realizada via dominância de Pareto: o conjunto de braços não dominados, $\mathcal{K}^*(t) \subseteq \mathcal{K}$, é constituído pelos braços a para os quais não existe outro braço $b \in \mathcal{K}$ com $\bar{\mathbf{Q}}(b, t) \succ \bar{\mathbf{Q}}(a, t)$ ([Drugan e Nowé, 2013](#)).

A noção de arrependimento (*regret*) também é adaptada ao caso multiobjetivo. Em vez de comparar a recompensa acumulada do agente com aquela de um único braço ótimo, o *Pareto regret* avalia a distância entre o braço selecionado e a frente de Pareto dos braços. Formalmente, para um braço a selecionado no instante t , o arrependimento de Pareto $\mathcal{R}_{\text{Pareto}}(T)$ é definido como o menor incremento escalar $\varepsilon \geq 0$ necessário para que o vetor de médias $\boldsymbol{\mu}_a$, somado a ε em todos os componentes, atinja a frente de Pareto dos braços; o desempenho do algoritmo é então avaliado por

$$\mathcal{R}_{\text{Pareto}}(T) = \mathbb{E} \left[\sum_{t=1}^T \varepsilon_{A(t)} \right], \quad (2.18)$$

que mede o quão distante, em média, o agente esteve da frente de Pareto ao longo do horizonte T (Drugan e Nowé, 2013). O símbolo $\varepsilon_{A(t)}$ segue a notação de Drugan e Nowé (2013) e designa a distância do braço escolhido para a frente de Pareto, não devendo ser confundido, portanto, com a taxa de diversificação ϵ da política ϵ -greedy. Minimizar $\mathcal{R}_{\text{Pareto}}(T)$ corresponde a manter as escolhas do agente concentradas no conjunto de braços não dominados.

Observe que a multiplicidade de objetivos nem sempre decorre de critérios declarados como conflitantes desde a modelagem. Ela também pode emergir de recompensas *não estacionárias*: quando a importância relativa dos critérios se altera ao longo da busca, um escalar único deixa de representar o que se deseja otimizar em cada fase, e a manutenção de estimativas vetoriais preserva essa informação em vez de descartá-la. É o caso do *Pool* tratado no Capítulo 6, em que a contribuição de uma solução para a qualidade e para a diversidade do conjunto varia conforme a composição do próprio *Pool* no instante da coleta.

A literatura organiza as estratégias de seleção em MOMaB em dois grupos principais, em paralelo ao que ocorre no MORL (Drugan e Nowé, 2013; Roijers et al., 2013):

- (i) Abordagens *escalarizadas*, em que o vetor de recompensa é convertido em um valor escalar por meio de uma função de utilidade $f(\mathbf{R}) \in \mathbb{R}$, recuperando-se um problema MaB clássico. Funções de escalarização lineares (somas ponderadas) e não lineares (*Chebyshev*) são utilizadas conforme as preferências entre objetivos sejam previamente conhecidas e fixas;
- (ii) Abordagens *baseadas em Pareto*, em que o agente mantém estimativas vetoriais e seleciona, a cada instante, um braço pertencente ao conjunto não dominado $\mathcal{K}^*(t)$, sem necessidade de definir uma função de utilidade.

Entre as abordagens baseadas em Pareto, o algoritmo *Pareto-UCB1*, proposto por Drugan e Nowé (2013), generaliza o índice UCB (Auer, 2002) para o caso vetorial. Para cada braço a , define-se um vetor de índices de confiança superior:

$$\mathbf{U}(a, t) = \bar{\mathbf{Q}}(a, t) + \sqrt{\frac{2 \ln(t \cdot d^4 \sqrt{K})}{N(a, t)}} \mathbf{1}, \quad (2.19)$$

em que $\mathbf{1} \in \mathbb{R}^d$ é o vetor unitário e o bônus de diversificação é somado a todos os componentes do vetor de média. A seleção é, então, feita em duas etapas: primeiro,

identifica-se o conjunto $\mathcal{K}^*(t) = \{a \in \mathcal{K} \mid \nexists b \in \mathcal{K}, \mathbf{U}(b, t) \succ \mathbf{U}(a, t)\}$ de braços não dominados segundo $\mathbf{U}$; em seguida, escolhe-se uniformemente um braço de $\mathcal{K}^*(t)$. Drugan e Nowé (2013) demonstram que o algoritmo possui limite superior logarítmico para o arrependimento de Pareto, em consonância com o resultado clássico do UCB no caso escalar.

Variações do Pareto-UCB1 incorporam, ao caso vetorial, os mesmos mecanismos de adaptação a ambientes não estacionários discutidos na Seção 2.3.2.1. O DUCB (Kocsis e Szepesvári, 2006) e o SWUCB (Garivier e Moulines, 2011), originalmente definidos para recompensas escalares, podem ser estendidos componente a componente para o caso multiobjetivo, mantendo estimativas vetoriais descontadas ou restritas a uma janela deslizante. Essas variantes são especialmente úteis em ambientes em que a composição ou a qualidade relativa dos braços muda ao longo do tempo, situação característica do *Pool* cooperativo da AMAM, em que soluções são inseridas e removidas dinamicamente. As formulações específicas dessas variantes vetoriais, bem como sua aplicação à dimensão social do *framework*, são apresentadas no Capítulo 6.

Neste trabalho, algoritmos baseados em *Multi-armed Bandit* multiobjetivo são implementados com o objetivo de aprimorar o processo de cooperação entre os agentes no *Framework* AMAM, considerando múltiplos critérios de avaliação. Nesse contexto, algoritmos de aprendizado multiagente baseados em MaB multiobjetivo são apresentados no Capítulo 6.

2.4 Considerações Parciais: *Q-learning* e *Multi-armed Bandit*

As seções anteriores apresentaram duas formulações de aprendizado por reforço: *Q-learning*, na Seção 2.3.1, e *Multi-armed Bandit*, nas Seções 2.3.2, 2.3.4 e 2.3.6. Convém esclarecer que, na presente tese, que não há substituição de uma formulação pela outra. As duas coexistem ao longo do trabalho, atuando sobre decisões distintas e em dimensões distintas do *framework* AMAM, conforme exposto a seguir.

O *Q-learning* opera no *aprendizado individual* do agente, na dimensão **Agente** descrita na Seção 2.1.1. É o mecanismo do ALS-*QLearning* do Algoritmo 1: cada agente aprende qual sequência de estruturas de vizinhança aplicar em sua busca local, e, nesse problema, o estado está bem definido, pois corresponde à estrutura de vizinhança corrente, e a ação leva de um estado a outro de forma determinística e observável pelo próprio agente. Já o *Multi-armed Bandit* opera no *aprendizado cooperativo*, na dimensão **Social**: cada agente aprende qual solução coletar do *Pool* compartilhado. Os dois algoritmos são executados simultaneamente, por todos os agentes, em todas as execuções relatadas nos Capítulos 4, 5 e 6.

A escolha do *Multi-armed Bandit* para a dimensão **Social** não é uma simplificação do *Q-learning*, adotada por conveniência ou custo computacional: decorre da natureza da cooperação implementada, que não constitui um processo de decisão de Markov, uma vez que:

- (i) *Não há transição de estado controlada pelo agente.* Ao coletar uma solução do *Pool*, o agente não conduz o sistema de uma configuração a outra: a composição do *Pool* é alterada de forma assíncrona por todos os agentes, conforme as regras

de inserção e exclusão da Seção 2.1.1. O agente apenas escolhe dentre as alternativas disponíveis naquele instante, sem determinar quais estarão disponíveis no instante seguinte;

- (ii) *A propriedade de Markov não se sustenta.* Se o estado fosse definido como a composição do *Pool*, ele mudaria entre o instante da decisão e o instante em que a recompensa é observada, uma vez que a recompensa só se materializa após o agente executar sua busca local sobre a solução coletada. A transição observada não dependeria apenas do par estado-ação do agente, mas também das ações concorrentes dos demais agentes;
- (iii) *A recompensa é imediata e local.* Conforme a Expressão (5.2), a recompensa mede a melhoria que a solução coletada produziu na busca daquele agente, sem propagação para decisões futuras. Não há retorno descontado a maximizar ao longo de uma trajetória.

Restando apenas a escolha repetida entre alternativas com retorno incerto e imediato, o problema é, por definição, de estado único, que é exatamente a classe modelada pelo método *Multi-armed Bandit*. Trata-se, portanto, de um problema de natureza diferente daquele que o método *Q-learning* resolve, e não de uma versão reduzida dele.

A formulação *stateless* tem três consequências que percorrem o restante da tese. A primeira é que não existe política ótima de longo prazo a ser aprendida na dimensão Social, nem faz sentido falar em iteração de valor ou em equação de Bellman sobre o *Pool*: o que se aprende é uma estimativa de valor por braço, atualizada a cada coleta. A segunda é que o critério natural de qualidade deixa de ser a convergência para a política ótima e passa a ser o *arrependimento* acumulado, isto é, a perda em relação a sempre ter escolhido a melhor alternativa disponível, conforme a Expressão (2.12). A terceira é que toda a dificuldade se concentra no equilíbrio entre diversificação e intensificação, o que explica por que as contribuições dos Capítulos 5 e 6 se organizam em torno de políticas de seleção, e não de estruturas de estado.

Cabe registrar que a ausência de estados não elimina a dificuldade, mas a desloca. Como o conjunto de braços do *Pool* não é estacionário, as garantias assintóticas dos algoritmos MaB clássicos, obtidas para conjuntos fixos de braços com distribuições estacionárias, não se transferem diretamente. É essa lacuna que motiva as variantes com desconto e com janela deslizante da Seção 2.3.2.1 e a avaliação empírica adotada nesta tese.

Capítulo 3

Descrição dos Experimentos

Este capítulo reúne os elementos comuns a todos os experimentos computacionais realizados nesta tese. Está organizado da seguinte forma: a Seção 3.1 descreve o Problema de Roteamento de Veículo com Janela de Tempo (PRVJT), utilizado como instância-problema em todos os experimentos; a Seção 3.2 apresenta os agentes implementados no *Framework* AMAM para a solução do PRVJT; e a Seção 3.3 descreve as configurações e dados comuns utilizados em todos os experimentos computacionais.

3.1 Problema de Roteamento de Veículo com Janela de Tempo

O problema utilizado como referência nos experimentos desta tese é o Problema de Roteamento de Veículo com Janela de Tempo (PRVJT) (Toth e Vigo, 2014). No PRVJT, um conjunto $V = \{v : v = 1, 2, \dots, |V|\}$ de veículos está localizado em um único depósito e deve servir a um conjunto $C = \{i : i = 1, 2, \dots, N\}$ de clientes espalhados geograficamente. No caso aqui considerado, a frota de veículos é homogênea, ou seja, todos os veículos são iguais e possuem a mesma capacidade Q . Cada cliente i tem uma dada demanda q_i e deve ser atendido dentro de uma janela de tempo especificada $[a_i, b_i]$ (veja a Figura 3.1(a)). Uma solução para o PRVJT é um conjunto de rotas (veja a Figura 3.1(b)), em que cada rota é representada por uma lista ordenada de clientes, que determina a sequência na qual eles devem ser atendidos por um veículo. Os arcos mostram a conexão entre os clientes e têm um valor associado c_{ij} , que representa o custo de viagem entre o cliente i e o cliente j , que pode ser relacionado à distância entre os clientes. A solução s , mostrada na Figura 3.1(b), pode ser descrita como:

$$\begin{aligned} s &= [0, 2, 1, 12, 0, 3, 4, 5, 6, 0, 10, 7, 8, 9, 11, 0] \\ &= [rota_1, rota_2, rota_3] \end{aligned}$$

na qual o índice 0 indica o depósito e $rota_1 = [0, 2, 1, 12, 0]$, $rota_2 = [0, 3, 4, 5, 6, 0]$ e $rota_3 = [0, 10, 7, 8, 9, 11, 0]$ são as três rotas desta solução. Assim, a solução s também pode ser descrita como $s = [rota_1, rota_2, rota_3]$.

O objetivo do PRVJT é determinar um conjunto de rotas para minimizar o custo total envolvido com essa operação. Cada rota está associada a um único veículo. As

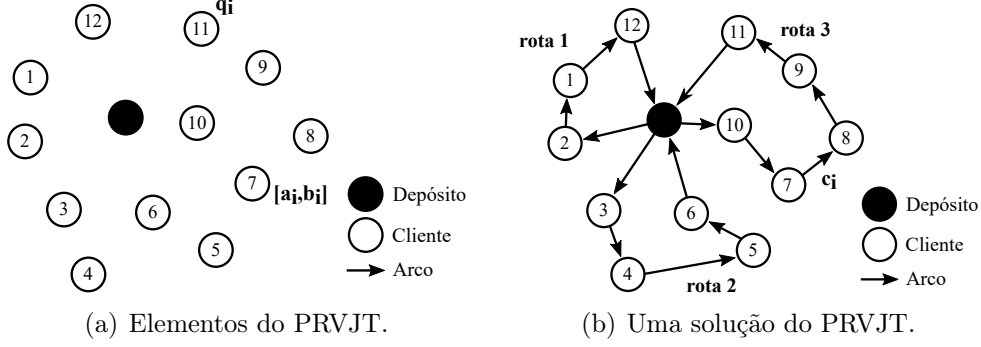


Figura 3.1: Uma solução s para o PRVJT (Silva et al., 2019).

rotas devem começar e terminar no depósito. Neste caso, o custo de uma solução s é calculado de acordo com:

$$f(s) = \omega V(s) + \sum_{(i,j) \in E} c_{ij} \quad (3.1)$$

em que E é o conjunto de arcos pertencentes à solução s ; $V(s)$ é o número de veículos na solução s ; e ω é um fator de penalidade de alto valor, não negativo e arbitrário. Nesta função, a prioridade é minimizar o número de veículos (ou rotas, consequentemente). Em caso de empate no número de veículos, a distância total percorrida deve ser minimizada. Este trabalho não tem como objetivo competir pelos melhores valores de função objetivo no contexto do problema em questão. No entanto, é necessário explicar as técnicas heurísticas empregadas nesse processo, a fim de compreender os resultados computacionais obtidos durante o desenvolvimento deste trabalho.

Como apresentado na descrição da arquitetura AMAM, mostrada na Seção 2.1.1, e levando em consideração que este trabalho busca solucionar o PRVJT, o ambiente a ser instanciado é o espaço de soluções possíveis para o PRVJT. Já as habilidades do agente (veja a Seção 2.1.1.2) em se movimentar pelo espaço de busca são adquiridas através da representação da solução do problema a ser tratado. Neste trabalho, para a resolução do PRVJT, a percepção e o posicionamento no ambiente são alcançados por meio da representação de uma solução para o problema (conforme descrito na Seção 3.1). Já a movimentação ocorre através das estruturas de vizinhança (conforme explicado na Seção 3.1.1) e perturbações (conforme detalhado na Seção 3.1.2).

3.1.1 Estruturas de vizinhança do PRVJT

As estruturas de vizinhanças de um problema de otimização são responsáveis pela movimentação do agente no ambiente do *framework*. Para explorar o espaço de soluções, oito diferentes funções de vizinhança, já definidas em Silva et al. (2019), foram usadas na instanciação do *Framework* AMAM para resolver o PRVJT:

- (i) *Intra-Route Swap*: função de vizinhança que realiza o movimento de troca de um cliente com outro cliente da mesma rota;
- (ii) *Inter-Route Swap*: função de vizinhança que realiza o movimento de troca de um cliente de uma rota com um cliente de outra rota;
- (iii) *Intra-Route Shift*: função de vizinhança que realiza o movimento de realocação de um cliente para outra posição na mesma rota;

- (iv) *Inter-Route Shift*: função de vizinhança que realiza a realocação de um cliente de uma rota para outra;
- (v) *Two Intra-Route Swap*: função de vizinhança que consiste na troca de clientes na mesma rota, assim como na função de vizinhança *Intra-Route Swap*. No entanto, na função *Two Intra-Route Swap*, dois clientes consecutivos são trocados por dois outros clientes consecutivos da mesma rota;
- (vi) *Two Intra-Route Shift*: função de vizinhança que consiste na realocação de clientes na mesma rota, assim como na função de vizinhança *Intra-Route Shift*. No entanto, na função *Two Intra-Route Shift*, dois clientes consecutivos são removidos de suas posições e reinseridos em outra posição da mesma rota;
- (vii) *Eliminates Smaller Route*: função de vizinhança que procura eliminar a menor rota da solução. A menor rota é definida como a rota que possui o menor número de clientes. Para isso, os clientes da menor rota da solução são removidos e reinseridos em outras rotas da solução;
- (viii) *Eliminates Random Route*: função de vizinhança *Eliminates Random Route* opera de forma semelhante à função *Eliminates Smaller Route*, mas a rota a ser excluída é escolhida aleatoriamente.

Ao longo do texto, a vizinhança de uma solução s segundo a k -ésima estrutura é denotada por $\mathcal{V}(k)$, e $\mathcal{V}(s)$ designa o conjunto dos vizinhos de s na estrutura corrente. O símbolo $\mathcal{N}$ fica reservado ao conjunto de agentes, empregado nos Capítulos 2 e 5, e N_a à contagem de seleções do braço a , de modo que nenhum dos três usos se confunde. Nos Algoritmos 5 e 6, o conjunto de vizinhos candidatos passou a ser denotado por M , para não se confundir com a vizinhança $\mathcal{V}$.

Para a resolução do PRVJT, o conjunto de estados do algoritmo *Q-learning* do agente é constituído pelo conjunto de funções de vizinhança apresentadas anteriormente e definido por S_{prvjt} . Após a explicação das estruturas de vizinhança do problema, as técnicas de perturbação também desempenham um papel crucial na compreensão do funcionamento da estrutura e nos resultados computacionais obtidos, pois também são responsáveis pela movimentação dos agentes no espaço de busca.

3.1.2 Perturbações para o PRVJT

Perturbações são alterações sistemáticas realizadas nas soluções com o intuito de explorar regiões de busca mais distantes durante o processo de busca. Como os agentes implementados neste trabalho utilizarão essas técnicas de perturbação na solução corrente, elas serão apresentadas a seguir:

- (i) *TwoShiftInterRandom*: perturbação que consiste em realizar dois movimentos aleatórios utilizando a estrutura de vizinhança *Inter-Route Shift*;
- (ii) *TwoSwapInterRandom*: perturbação que realiza dois movimentos aleatórios utilizando-se da estrutura de vizinhança *Inter-Route Swap*;

- (iii) *ThreeShiftInterRandom*: perturbação que consiste em realizar três movimentos aleatórios utilizando a estrutura de vizinhança *Inter-Route Shift*;
- (iv) *ThreeSwapInterRandom*: perturbação que realiza três movimentos aleatórios utilizando-se da estrutura de vizinhança *Inter-Route Swap*;
- (v) *ThreeShifInterConsecutive*: perturbação que consiste em realizar três movimentos utilizando a estrutura de vizinhança *Inter-Route Shift*. Sendo que os outros dois clientes a serem visitados são consecutivos ao cliente escolhido aleatoriamente;
- (vi) *EliminatesSmallRoute*: perturbação que consiste em eliminar as menores rotas da solução e alocar os clientes nas outras rotas disponíveis;
- (vii) *EliminatesRandomRoute*: esta perturbação consiste em eliminar rotas da solução. Porém essas rotas são selecionadas aleatoriamente e não definida pelo seu tamanho.

Descrevemos aqui as principais técnicas heurísticas utilizadas neste trabalho para a solução do PRVJT. Para um melhor entendimento das técnicas utilizadas aconselha-se a leitura de [Silva et al. \(2019\)](#), em que são descritos todos os métodos utilizados para implementação das meta-heurísticas aqui utilizadas.

3.2 Agentes implementados

Para alcançar os objetivos deste trabalho, foram utilizadas instanciações do *Framework* AMAM ([Silva et al., 2019](#)), nas quais os agentes implementam, para a solução do PRVJT, uma meta-heurística híbrida baseada na meta-heurística ILS (*Iterated Local Search*), que gera soluções iniciais baseadas na fase de construção da meta-heurística GRASP (*Greedy randomized adaptive search procedure*) e busca local adaptativa baseada no método VND (*Variable Neighborhood Descent*) ([Mladenović e Hansen, 1997](#)). Em [Silva \(2019\)](#), os agentes são denominados ILS_ALS (*Iterated Local Search-Adaptive Local Search*), pois são baseados no algoritmo ILS com Busca Local Adaptativa ALS-*QLearning*, apresentada também em [Silva et al. \(2019\)](#).

O Algoritmo 3 apresenta o ILS implementado pelo agente. Na linha 2, uma solução inicial é construída, utilizando a fase de construção da metaheurística *GRASP*. A linha 5 é responsável pela perturbação da solução corrente. A função *busca_local_vnd(so)* (linhas 3 e 6) implementa a versão da meta-heurística de busca local VND, utilizando o método ALS-*QLearning*, na definição das sequência de estruturas de vizinhanças. Essa técnica foi introduzida na Seção 2.1.1.2. Neste algoritmo, a perturbação da solução, realizada a partir de mudanças na solução atual, é implementada em níveis, ou seja, em cada iteração a função de perturbação é alterada se não houver melhoria na solução (linha 11), e retorna ao seu primeiro nível (linha 9), se uma melhor solução é encontrada. Na linha 14, uma solução final é retornada. O Algoritmo 4 mostra o comportamento do método de busca local VND implementado pelos agentes.

O Algoritmo 4 mostra o procedimento da busca local VND com critério de escolha do melhor vizinho. Pode-se observar que a linha 3 apresenta o parâmetro k_{max} , que define o número de estruturas de vizinhanças para o problema tratado. A linha 4

Algoritmo 3 Busca Local Iterada (*Iterated Local Search* – ILS)

```

1: função ils()
2:    $so \leftarrow construcao\_grasp();$ 
3:    $s \leftarrow busca\_local\_vnd(so);$ 
4:   enquanto critério de parada não satisfeito faça
5:      $s' \leftarrow perturbacao(s, nivel\_perturbacao);$ 
6:      $s'' \leftarrow busca\_local\_vnd(s');$ 
7:     se (critério_aceitacao( $s, s''$ )) então
8:        $s \leftarrow s'';$ 
9:        $nivel\_perturbacao \leftarrow 1;$ 
10:    senão
11:       $nivel\_perturbacao \leftarrow nivel\_perturbacao + 1;$ 
12:    fim se
13:  fim enquanto
14:  retorne  $s$ 
15: fim função

```

Algoritmo 4 Descida em Vizinhança Variável (*Variable Neighborhood Descent* – VND)

```

1: função busca_local_vnd( $s$ )
2:    $k \leftarrow 1;$ 
3:   enquanto  $k \leq k_{max}$  faça       $\triangleright k_{max}$  é o número de estruturas de vizinhança
    diferentes
4:      $s' \leftarrow BestImprovement(s, \mathcal{V}(k));$ 
5:     se  $f(s') < f(s)$  então
6:        $s \leftarrow s';$ 
7:        $k \leftarrow 1;$ 
8:     senão
9:        $k \leftarrow k + 1;$ 
10:    fim se
11:  fim enquanto
12:  retorne  $s$ 
13: fim função

```

introduz o método de busca local com critério do melhor vizinho, que consiste em encontrar o melhor vizinho de dada estrutura de vizinhança $\mathcal{V}(k)$. Esta proposta buscou implementar agentes que fossem distintos. Para isso, foram definidos agentes com técnicas de busca local distintas. Portanto, cada agente implementa a chamada de um método de busca local diferente na linha 4, que são responsáveis por diversificar os agentes implementados na estrutura. Os métodos de busca local implementados são *BestImprovement*, *FirstImprovement* e *RandomImprovement*. Desta forma, os Algoritmos 5, 6 e 7 apresentam os procedimentos para estes três métodos, respectivamente.

O Algoritmo 5 mostra o método de descida *BestImprovement*, no qual todos os vizinhos são visitados (linha 2) e o cálculo da função de avaliação é realizado para cada um deles. Em seguida, a solução com o melhor valor de função de avaliação é

Algoritmo 5 Descida pela Melhor Melhoria (*Best Improvement Descent*)

```

1: função BestImprovement( $s, \mathcal{V}$ )
2:    $M \leftarrow \{s' \in \mathcal{V}(s) \mid f(s') < f(s)\};$ 
3:   se  $|M| > 0$  então
4:      $s' \leftarrow \arg \min\{f(s') \mid s' \in M\};$ 
5:      $s \leftarrow s';$ 
6:   fim se
7:   retorne  $s;$ 
8: fim função

```

Algoritmo 6 Descida pela Primeira Melhoria (*First Improvement Descent*)

```

1: função FirstImprovement( $s, \mathcal{V}$ )
2:    $M \leftarrow \mathcal{V}(s);$ 
3:   enquanto  $|M| > 0$  faça
4:     Selecione aleatoriamente  $s' \in M;$ 
5:      $M \leftarrow M \setminus \{s'\}$ 
6:     se  $f(s') < f(s)$  então
7:        $s \leftarrow s';$ 
8:     retorne  $s;$ 
9:   fim se
10:  fim enquanto
11:  retorne  $s;$ 
12: fim função

```

retornada, conforme mostrado na linha 4.

O Algoritmo 6 apresenta o método *FirstImprovement*, caracterizado por evitar uma busca exaustiva em todos os vizinhos disponíveis para uma dada estrutura de vizinhança, o que permite reduzir a frequência de cálculo da função de avaliação, como mostrado na linha 6. Para garantir que este método não seja determinístico, é inserido um fator de aleatoriedade na linha 4. Conforme linha 8, o método retorna a primeira solução com um valor de função de avaliação melhor que o valor da solução s recebida.

Algoritmo 7 Descida pela Melhoria Aleatória (*Random Improvement Descent*)

```

1: função RandomImprovement( $s, \mathcal{V}$ )
2:    $Iter \leftarrow 0;$  ▷ Contador de quantidade de interações
3:   enquanto  $Iter < IterMax$  faça
4:     Selecione aleatoriamente  $s' \in \mathcal{V}(s);$ 
5:     se  $f(s') < f(s)$  então
6:        $s \leftarrow s';$ 
7:     fim se
8:      $Iter \leftarrow Iter + 1;$ 
9:   fim enquanto
10:  retorne  $s;$ 
11: fim função

```

O Algoritmo 7 mostra o pseudocódigo do método de descida randômica, que, assim como o método *FirstImprovement*, busca reduzir a busca exaustiva por todos os vizinhos da solução. Para atingir esse objetivo, o método visita apenas um percentual dos vizinhos possíveis (linha 3) e também inclui um fator de aleatoriedade, semelhante ao método *FirstImprovement*, na linha 4.

Os três métodos de busca local apresentados foram utilizados na definição dos seguintes agentes:

- (i) Agente ILS_ALS_NO: agente ILS_ALS com busca local *BestImprovement* sem a utilização da estrutura de cooperação do *framework*;
- (ii) Agente ILS_ALS_BI: agente ILS_ALS com busca local *BestImprovement* com a utilização da estrutura de cooperação do *framework*;
- (iii) Agente ILS_ALS_FI: agente ILS_ALS com busca local *FirstImprovement* com a utilização da estrutura de cooperação do *framework*;
- (iv) Agente ILS_ALS_RI: agente ILS_ALS com busca local *RandomImprovement* com a utilização da estrutura de cooperação do *framework*;

Tabela 3.1: Características gerais dos agentes implementados

Agente	Aprendizado	Cooperação	Busca Local		
			<i>BestImprovement</i>	<i>FirstImprovement</i>	<i>RandomImprovement</i>
<i>ILS_ALS_NO</i>	X		X		
<i>ILS_ALS_BI</i>	X	X	X		
<i>ILS_ALS_FI</i>	X	X		X	
<i>ILS_ALS_RI</i>	X	X			X

A Tabela 3.1 resume as características dos agentes implementados. A coluna “Agente” lista os agentes implementados; a coluna “Aprendizado” indica se os agentes utilizam o algoritmo ALS-*QLearning*; a coluna “Cooperação” indica se o agente utiliza a dimensão social do *framework*; e a coluna “Busca Local” especifica a busca local implementada por cada agente.

A Figura 3.2 ilustra um exemplo de instância da aplicação do *Framework* AMAM. Essa instanciação é definida com apenas quatro agentes (*ILS_ALS_NO*, *ILS_ALS_BI*, *ILS_ALS_FI* e *ILS_ALS_RI*), a título de demonstração. Neste exemplo, os quatro agentes executam métodos distintos e cada um conduz sua própria busca de forma independente, sem coordenação central. A cooperação ocorre por meio do *Pool* de soluções, estrutura compartilhada por todos os agentes. Assim, ao longo da busca, cada agente deposita no *Pool* as boas soluções que encontra e dele coleta soluções produzidas por outros agentes, retomando a busca a partir delas. É esse intercâmbio de soluções que constitui a dimensão social do *framework* e que a coluna *Cooperação* da Tabela 3.1 indica. O agente *ILS_ALS_NO*, por não acessar o *Pool*, executa de forma isolada e serve como referência de agente sem cooperação. Além disso, os métodos executados pelos agentes interagem diretamente com o ambiente, por meio de um conjunto de movimentos que operam nas soluções. Também é observado que o agente *ILS_ALS_NO* não interage com o *Pool* de soluções.

Para os experimentos computacionais com maior escala, define-se o cenário ILS_MIX, composto por:

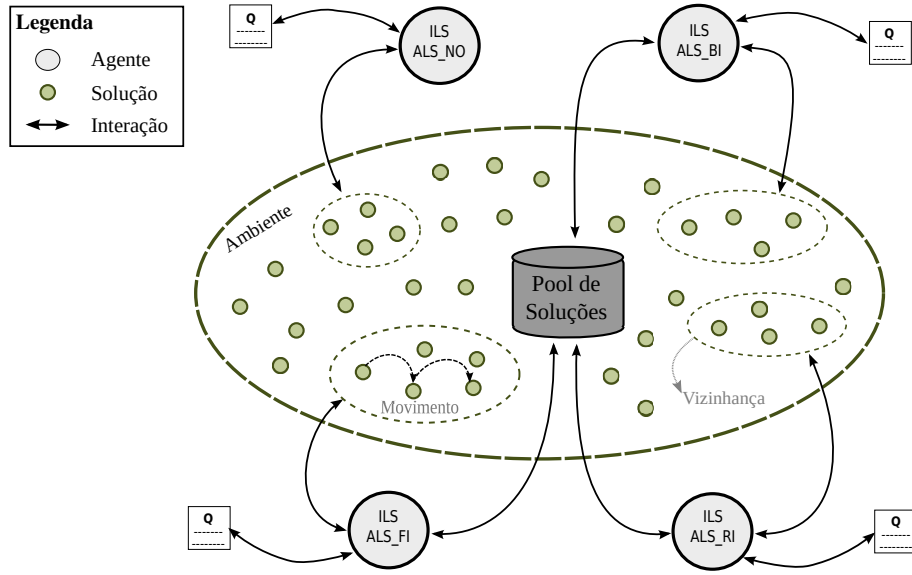


Figura 3.2: Instancição do *Framework* AMAM para o PRVJT com 4 agentes. Note que o agente ILS_ALS_NO não possui acesso ao *Pool* de soluções.

- sete agentes ILS_ALS_BI,
- sete agentes ILS_ALS_FI e
- seis agentes ILS_ALS_RI,

totalizando 20 agentes por execução. Essa configuração é adotada com base em [Silva et al. \(2025\)](#), que demonstra que a escolha de 20 agentes representa um bom equilíbrio entre qualidade da solução e tempo de execução no *Framework* AMAM. A composição heterogênea dos agentes garante diversidade comportamental durante o processo de busca, uma vez que cada tipo de agente explora o espaço de soluções por meio de uma estratégia de busca local distinta.

3.3 Dados comuns dos Experimentos

Nesta seção são apresentados dados que foram utilizados em todos os experimentos realizados. Para execução dos testes computacionais, esta tese utiliza-se do *Framework* AMAM. O *Framework* AMAM foi implementado em JAVA, com o JDK 8, utilizando a Eclipse IDE for JAVA Developers version - 4.26.0. A versão mais recente do *framework* AMAM está disponível, sob a licença GNU LGPLv3, em <https://github.com/mamelials/AMAM-Multiagente-Architecture-for-Metaheuristics.git>.

Para execução dos experimentos, foi utilizado um conjunto de instâncias para o PRVJT, proposto inicialmente em [Solomon \(1987\)](#) e referencial para a análise de técnicas de solução para essa classe de problemas. Cada instância deste conjunto de instâncias contém 100 clientes. Essas instâncias são separadas em três diferentes conjuntos de clientes, conforme a sua distribuição geográfica (*C-Cluster*, *R-Random* e *RC-Random Cluster*). As posições geográficas dos clientes são geradas aleatoriamente no conjunto de problemas R. Nos conjunto de problemas C, as posições geográficas

são agrupadas. Além disso, uma mistura de estruturas aleatórias e agrupadas é usada no conjunto de problemas RC. Cada uma dessas três famílias divide-se ainda em duas subséries, identificadas pelo primeiro dígito do código da instância: a subsérie 1xx reúne instâncias com horizonte de planejamento curto e janelas de tempo estreitas, ao passo que a subsérie 2xx reúne instâncias com horizonte longo e janelas amplas, o que admite rotas com mais clientes por veículo. Nos experimentos que empregam um subconjunto do *benchmark*, as instâncias foram selecionadas de modo a preservar essa estrutura, tomando-se a mesma quantidade de instâncias de cada família e de cada subsérie. O subconjunto de seis instâncias (C109, C208, R112, R211, RC108 e RC208), utilizado nos experimentos de cooperação, diversidade, modelagem de recompensas e do *MaB Pool*, contém uma instância de cada combinação de família e subsérie; o subconjunto de vinte e quatro instâncias, utilizado nos experimentos de aprendizado multiagente e multiobjetivo, contém quatro instâncias de cada combinação. Dessa forma, os subconjuntos mantêm a diversidade de distribuição geográfica e de restrição temporal do conjunto completo, mantendo o custo computacional compatível com o número de execuções e de agentes avaliados.

Todos os resultados computacionais foram obtidos utilizando-se de computadores disponíveis no *cluster* da Universidade Federal de Viçosa - UFV, mantido pela Divisão de Suporte ao Desenvolvimento Científico e Tecnológico da Universidade Federal de Viçosa (<https://dct.ufv.br>). Com o intuito de diminuir os ruídos das execuções, cada agente do *framework* é executado em sua própria *thread* JAVA e em sua própria *thread* física (processador/núcleo). Para que isso fosse possível, utilizou-se de ferramentas de administração Linux para que cada *thread* JAVA fosse alocada a uma *thread* física. É digno de nota ressaltar que a coleta de dados foi feita de forma casualizada, porém não totalmente aleatória, pois os grupos de agentes tiveram que ser obedecidos com o intuito de respeitar a ideia de redução de ruído causada pela disponibilidade de *hardware* e gerenciamento de *software* JAVA.

Em todos os experimentos computacionais, em relação a todos os cenários desenvolvidos nesta tese, os testes computacionais foram executados 30 vezes. Após isso, com o intuito de determinar os testes estatísticos apropriados para avaliar a distribuição dos dados obtidos nas execuções, foram inicialmente verificadas as premissas exigidas pelos testes paramétricos. A independência dos resíduos foi avaliada pelo teste de Durbin-Watson; a homogeneidade das variâncias pelo teste de Fligner-Killeen; e a normalidade pelo teste de Shapiro-Wilk, todos aplicados sobre os resíduos do modelo de análise de variância ajustado para cada conjunto de dados. Os resultados indicaram que, em nenhum dos conjuntos de dados gerados, essas premissas foram atendidas. Portanto, foram aplicados testes não paramétricos a estes conjuntos de dados, que representam os resultados obtidos.

O delineamento experimental é de blocos completos: todos os cenários são executados sobre o mesmo conjunto de instâncias, de modo que cada instância funciona como um bloco e as observações dos diferentes cenários são emparelhadas dentro dela. Os testes empregados respeitam esse emparelhamento. Para a comparação global entre os cenários utiliza-se o teste de Friedman (Montgomery e Runger, 2010), que é o análogo não paramétrico da análise de variância para blocos e se verifica se ao menos um cenário difere dos demais. Quando essa diferença é detectada, aplica-se, como análise *post-hoc*, o teste de Wilcoxon *signed-rank* (Wilcoxon, 1992) em cada par de cenários, também de forma pareada. As comparações par a par são bilaterais e ser-

vem para decidir se existe diferença; quando ela existe, o sentido da diferença, isto é, qual cenário é melhor, é determinado pela comparação das medianas e é o que os símbolos $>$, $<$ e $=$ representam nas tabelas de teste. A única exceção a esse procedimento está na Seção 4.1.5, em que três das quatro situações confrontam conjuntos de tamanhos diferentes e sem correspondência entre as observações, não admitindo emparelhamento; nesses casos, utiliza-se o teste de Wilcoxon-Mann-Whitney (Mann e Whitney, 1947), que é a versão para amostras independentes do teste de Wilcoxon *signed-rank*. Para ajustar o nível de significância das comparações múltiplas e evitar o erro do tipo I (rejeitar uma hipótese nula verdadeira), emprega-se a correção de Bonferroni (Montgomery e Runger, 2010). Salvo indicação em contrário, adota-se um nível de significância de 5% (95% de confiança). Todas as análises estatísticas foram realizadas utilizando a linguagem R versão 4.3.3.

Por fim, precisamos, resumir alguns dados para facilitar a interpretação dos resultados apresentados nas próximas seções. O critério de parada dos agentes ILS desenvolvidos foi o número de iterações sem melhora, tendo valor igual a 100 iterações. Sempre que nos referirmos ao *gap*(%), esse valor é obtido pelo cálculo apresentado na Equação (3.2):

$$gap(\%) = \frac{valor - valor\ literatura}{valor\ literatura} \times 100 \quad (3.2)$$

em que *valor* é o valor obtido com a execução do referido teste e *valor_literatura* é o melhor valor encontrado na literatura para aquela determinada instância. Também é de suma importância destacar que competir com os melhores resultados da literatura para instâncias do problema está fora do escopo destes experimentos.

Capítulo 4

Análise do *Framework* AMAM

Este capítulo apresenta os experimentos computacionais realizados para a análise do *Framework* AMAM. Esses experimentos têm caráter validatório: antes de propor as contribuições principais desta tese, é necessário verificar que o *framework* opera de forma adequada em relação às suas propriedades fundamentais, como modelagem de recompensas, escalabilidade, cooperação e diversidade de agentes. Cada seção deste capítulo descreve os objetivos, a configuração experimental e as principais conclusões obtidas. As seções 4.1, 4.2, 4.3 e 4.4 apresentam, respectivamente, os experimentos relativos à escalabilidade, cooperação, diversidade e à modelagem de recompensas.

4.1 Escalabilidade

Em geral, escalabilidade se refere à capacidade de um sistema executar trabalho útil de forma uniforme à medida que a dimensão do sistema em si aumenta. Rana e Stout (2000) discutem diversos problemas relacionados ao desenvolvimento de sistemas multiagente e, dentre eles, destacamos a seguinte pergunta: “*Quantos agentes podem ser usados com sucesso em uma aplicação?*”. Assim, observa-se a importância de investigar o comportamento de um sistema multiagente quando novos agentes são adicionados, ou seja, avaliar as consequências (limites, vantagens e desvantagens) de se adicionar novos agentes a um sistema multiagente.

Os experimentos aqui apresentados foram conduzidos com o objetivo de investigar o desempenho do *framework* AMAM, dada a inclusão de agentes em sua estrutura. O objetivo é avaliar a estrutura quanto a sua escalabilidade. Partindo da heterogeneidade das formas de mensuração da escalabilidade, descritas na Seção 2.2.3, o objetivo principal deste experimento é avaliar se a inclusão de agentes no *framework* tem influência direta no desempenho do mesmo em relação à qualidade dos resultados obtidos e em relação aos tempos de execução. Adicionalmente, é avaliado o mecanismo de cooperação entre os próprios agentes, a partir da existência de maior número de agentes na estrutura.

Nesta seção, são apresentadas as análises e os resultados obtidos a partir do trabalho desenvolvido por Silva et al. (2025), com o objetivo de contextualizar e discutir os principais achados relacionados à proposta investigada.

4.1.1 Definição da Experimentação

Os experimentos utilizaram as 56 instâncias de PRVJT introduzidas em [Solomon \(1987\)](#). O computador (nó de cálculo) utilizado possui as seguintes configurações: processador Intel Xeon Silver 4216 (Cache de 22MB, 2.1GHz, 16 núcleos, 384 GB de RAM), totalizando 64 núcleos de execução, executando o sistema operacional Linux CentOS.

Os testes computacionais foram realizados com um, dois, quatro, oito, dez, vinte, trinta, quarenta, cinquenta e sessenta agentes. Com o intuito de diminuir o ruído e tornar as execuções mais justas frente ao que será avaliado, os testes foram separados da seguinte forma: utilizou-se de um computador com 64 núcleos, sendo que, a cada execução, 60 núcleos foram utilizados pelos agentes e os outros 4 núcleos foram deixados à disposição da JVM do JAVA e do sistema operacional. Para isso, os testes foram separados em 5 grupos, sendo que cada grupo utiliza 60 *threads* e cada agente é direcionado para uma *thread*. Os grupos de testes foram:

- (i) 4 execuções por vez dos cenários com um, dois, quatro e oito agentes;
- (ii) 1 execução por vez dos cenários de dez e cinquenta agentes;
- (iii) 1 execução por vez dos cenários de vinte e quarenta agentes;
- (iv) 2 execuções por vez do cenário de trinta agentes;
- (v) 1 execução por vez do cenário de sessenta agentes.

Observe que, em cada grupo de teste, o total de agentes instanciados é igual ao número de *threads* disponíveis, ou seja, 60.

Para investigar o efeito da escalabilidade da estrutura, projetou-se um experimento em que o *Framework* AMAM foi configurado em 10 cenários diferentes em relação ao número de agentes. Um cenário é compreendido como cada configuração usada em experimentos envolvendo diferentes números de agentes, que podem ou não incluir recursos de cooperação entre os agentes. A partir daí, foram definidos dez cenários para avaliar o *framework*:

- (i) um cenário de um único agente;
- (ii) um cenário com dois agentes;
- (iii) um cenário com quatro agentes;
- (iv) um cenário com oito agentes;
- (v) um cenário com dez agentes;
- (vi) um cenário com vinte agentes;
- (vii) um cenário com trinta agentes;
- (viii) um cenário com quarenta agentes;
- (ix) um cenário com cinquenta agentes e

(x) um cenário com sessenta agentes.

Os cenários utilizaram agentes ILS_ALS_BI e a aplicação para estes testes foi nomeada App_PRVJT. Como estes testes buscam avaliar a escalabilidade do *framework* AMAM, foram utilizados agentes idênticos para não corromper essa análise.

4.1.2 Resultados Computacionais

Esta seção apresenta os resultados obtidos nas execuções do cenário apresentado para a aplicação App_PRVJT, seguindo as diretrizes já descritas. Além disso, esta seção inclui a análise e discussão desses resultados.

A Tabela 4.1 apresenta as melhores soluções obtidas para cada cenário após 30 execuções para cada uma das 56 instâncias testadas. O valor atribuído ao parâmetro ω na Expressão (3.1), que define o cálculo do custo da solução, foi $\omega = 1000$. Nesta tabela, a coluna “Inst” corresponde à instância testada, e a coluna “BKV” (*Best Known Value*) corresponde à melhor solução encontrada na literatura para a respectiva instância. As colunas seguintes apresentam os cenários de teste e os números correspondentes de agentes utilizados em cada um, ou seja, um, dois, quatro, oito, dez, vinte, trinta, quarenta, cinquenta e sessenta agentes, respectivamente. O valor BKV fornecido corresponde ao melhor número conhecido de veículos, bem como ao melhor valor conhecido de distância percorrida para uma determinada instância, de acordo com <https://www.sintef.no/projectweb/top/vrptw/100-customers/>, aplicado à Expressão (3.1).

A Tabela 4.2 apresenta os tempos médios em segundos obtidos com as 30 execuções para cada cenário em todas as 56 instâncias testadas. Nesta Tabela, a coluna “Ins” informa a instância testada, e as outras colunas mostram a média, em segundos, dos tempos de execução associados a cada um dos cenários avaliados em termos do número de agentes considerados.

A Tabela 4.3 resume os resultados das Tabelas 4.1 e 4.2. Nesta tabela, a coluna “Cenário” apresenta as medidas usadas para analisar o comportamento dos resultados, ou seja: “*gap*(%) melhor”, “*gap*(%) médio” e “*gap*(%) mediano”, que correspondem à média das diferenças (%) entre as medidas encontradas e a melhor solução conhecida na literatura. Esses valores são calculados como:

$$gap(\%) = \frac{resultado_medida - melhor_resultado}{melhor_resultado} \times 100,$$

em que *medida* é a métrica de interesse e *melhor_resultado* é o melhor resultado conhecido na literatura. A linha “número de vezes do melhor” corresponde a quantas vezes o cenário avaliado alcançou a melhor solução conhecida na literatura. Por sua vez, as linhas “tempo médio de execução” e “tempo mediano de execução” correspondem, respectivamente, à média e à mediana do tempo médio obtido com 30 execuções de cada instância.

O objetivo é avaliar o desempenho do *Framework* AMAM e a escalabilidade da proposta. A escalabilidade é avaliada identificando se o aumento no número de agentes envolvidos na solução influencia o desempenho do sistema multiagente. As medidas utilizadas aqui para essa análise são a qualidade da função objetivo e o tempo de execução. A Figura 4.1, gerada a partir de dados extraídos da Tabela 4.3, apresenta

Tabela 4.1: Valor de função objetivo de cada instância avaliada do PRVJT $\times$ Número de Agentes. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. O destaque é feito por linha, entre os dez cenários de número de agentes; a coluna BKV traz o melhor valor conhecido na literatura e não entra na comparação.

Inst	BKV	Número de Agentes									
		1	2	4	8	10	20	30	40	50	60
C101	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C102	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C103	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06	10828,06
C104	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78	10824,78
C105	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C106	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C107	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C108	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C109	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94	10828,94
C201	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56
C202	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56	3591,56
C203	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17	3591,17
C204	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60	3590,60
C205	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88	3588,88
C206	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49	3588,49
C207	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29	3588,29
C208	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32	3588,32
R101	20650,80	20651,10	20651,40	20650,80	20650,80	20650,80	20650,80	20650,80	20650,80	20650,80	20650,80
R102	18486,12	18496,60	18492,50	18487,90	18491,30	18486,20	18487,60	18486,12	18486,12	18486,12	18486,12
R103	14292,68	15226,20	14313,40	14310,80	14307,90	14301,30	14293,30	14297,20	14297,80	14303,60	14296,70
R104	10007,31	11000,50	10989,20	10986,30	10984,90	10983,10	10008,00	10978,00	10020,80	10978,90	10977,20
R105	15377,11	15399,80	15392,50	15383,50	15377,11	15377,30	15378,50	15377,11	15377,11	15377,11	15377,11
R106	13252,03	13296,50	13263,30	13270,70	13256,50	13263,40	13257,30	13260,80	13258,60	13258,50	13256,60
R107	11104,66	12084,10	12089,90	12068,30	12074,70	12080,00	11138,00	11130,50	11125,70	11113,20	11125,30
R108	9960,88	10001,80	10954,70	9997,50	9979,10	9987,20	9974,40	9966,20	9966,80	9961,80	9961,30
R109	12194,73	13173,50	13165,70	13165,40	13163,10	13161,30	13155,90	13158,60	13152,60	13159,10	13149,60
R110	11118,84	12105,20	12093,00	12100,70	12085,60	12098,10	12086,30	12082,90	12088,00	12084,20	12085,90
R111	11096,72	12083,20	12067,30	11112,00	12070,50	12066,50	12060,10	11159,80	11105,50	11125,00	11099,30
R112	9982,14	10994,10	10967,70	10970,20	10968,20	10967,30	10954,70	10963,40	10960,60	10960,80	10956,30
R201	5252,37	5267,70	5256,00	5263,60	5254,40	5252,37	5252,60	5253,80	5252,37	5252,37	5252,37
R202	4191,70	5096,80	5094,90	5098,20	5090,00	5093,70	4205,50	4207,10	4202,50	4207,10	4204,30
R203	3939,50	3947,80	3952,20	3944,70	3947,20	3942,60	3941,00	3942,30	3943,40	3939,50	3940,80
R204	2825,52	2838,10	3759,70	2853,30	2832,00	2831,80	2837,30	2829,50	2829,60	2828,80	2828,90
R205	3994,43	4063,40	4063,80	4030,20	4022,40	4022,80	4009,70	4023,40	4017,70	4015,80	4018,90
R206	3906,14	3938,80	3919,20	3923,80	3910,70	3916,00	3918,30	3911,50	3911,00	3911,90	3906,40
R207	2890,61	3825,70	3830,70	2915,40	2901,50	2955,50	3814,90	2895,80	2926,40	2905,80	2906,10
R208	2726,82	2738,00	2738,80	2730,20	2734,80	2730,90	2730,00	2726,82	2726,82	2726,82	2726,82
R209	3909,16	3964,10	3925,90	3929,00	3922,40	3926,70	3910,10	3921,80	3914,00	3909,16	3917,70
R210	3939,37	3973,10	3966,60	3964,20	3944,80	3952,90	3960,00	3949,70	3956,40	3947,40	3946,70
R211	2885,71	3814,30	3797,40	3813,90	3794,60	3797,10	3803,90	3789,60	3787,20	3790,50	3778,70
RC101	15696,95	16673,60	16624,90	16640,70	16642,20	16642,60	16632,30	16631,40	16633,40	16631,40	16637,00
RC102	13554,75	14518,10	14474,30	14507,10	14505,80	14508,00	14500,60	14493,20	14494,40	14480,10	13554,75
RC103	12261,67	12300,50	12293,90	12275,80	12272,40	12274,80	12261,67	12264,10	12263,00	12267,80	12265,20
RC104	11135,48	11177,00	11153,40	11144,70	11135,48	11138,60	11135,48	11135,48	11135,48	11135,48	11135,48
RC105	14629,44	15559,60	15581,70	15552,30	15536,70	15556,00	15547,70	15543,10	15536,90	15543,60	15543,60
RC106	12424,73	13409,20	13382,70	13386,80	13382,70	13389,10	13381,80	13382,70	13382,70	13380,50	13382,70
RC107	12230,48	12270,70	12230,48	12240,80	12230,48	12235,70	12231,70	12230,48	12230,48	12230,48	12230,48
RC108	11139,82	12146,30	11208,90	11152,30	11164,70	11149,10	11144,30	11149,40	11149,30	11150,60	11145,90
RC201	5406,94	5471,20	5457,70	5440,00	5431,30	5421,60	5443,10	5434,00	5440,10	5414,50	5431,50
RC202	4365,65	5219,70	5190,50	5185,90	5196,80	5194,70	4446,20	5169,70	5169,60	5158,40	5161,90
RC203	4049,62	4082,80	4092,60	4075,90	4082,50	4050,00	4073,20	4068,00	4050,80	4074,70	4070,40
RC204	3798,46	3826,20	3815,60	3818,40	3800,20	3802,00	3802,10	3800,40	3803,60	3801,00	3799,40
RC205	5297,65	5337,40	5366,10	5322,50	5315,20	5344,90	5317,80	5331,00	5313,50	5309,40	5305,20
RC206	4146,32	4225,40	4156,80	4176,70	4191,70	4167,40	4155,70	4146,32	4154,70	4161,50	4146,32
RC207	4061,14	4120,00	4092,00	4118,30	4074,60	4077,20	4086,40	4075,80	4065,60	4068,40	4067,10
RC208	3828,14	3877,60	3888,30	3877,60	3863,30	3853,20	3843,90	3849,80	3843,50	3855,00	3841,30

a média das diferenças (%) em relação às melhores soluções conhecidas, à média das soluções e à mediana das soluções para os 10 cenários. Observa-se uma melhora na qualidade da função objetivo à medida que o número de agentes utilizados na solução aumenta. Por outro lado, a Figura 4.2 mostra o comportamento do tempo de execução, representado na forma de um *boxplot*, incluindo o tempo médio de execução em cada cenário. Os resultados mostram uma leve tendência de crescimento no tempo de

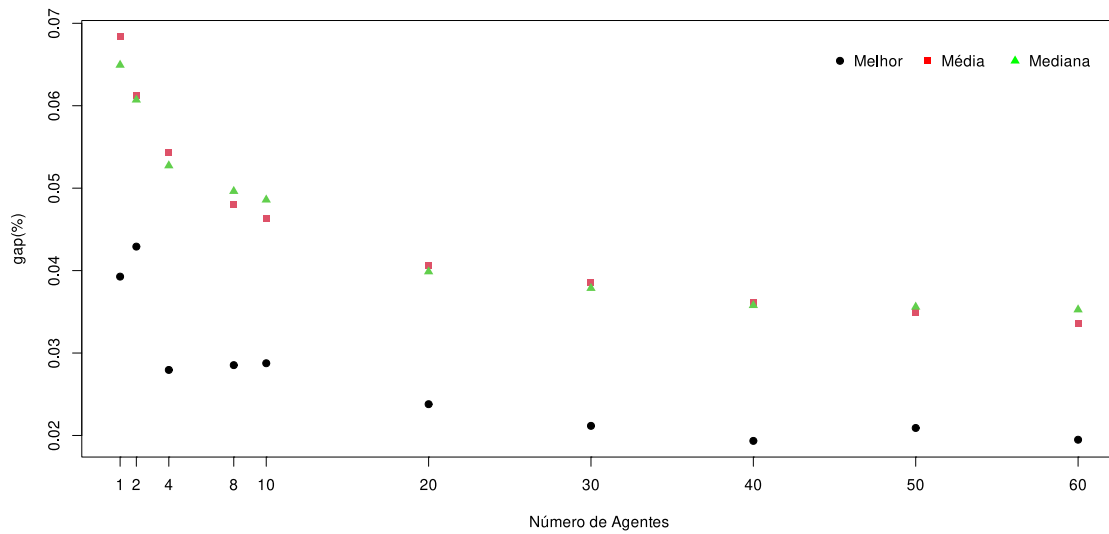
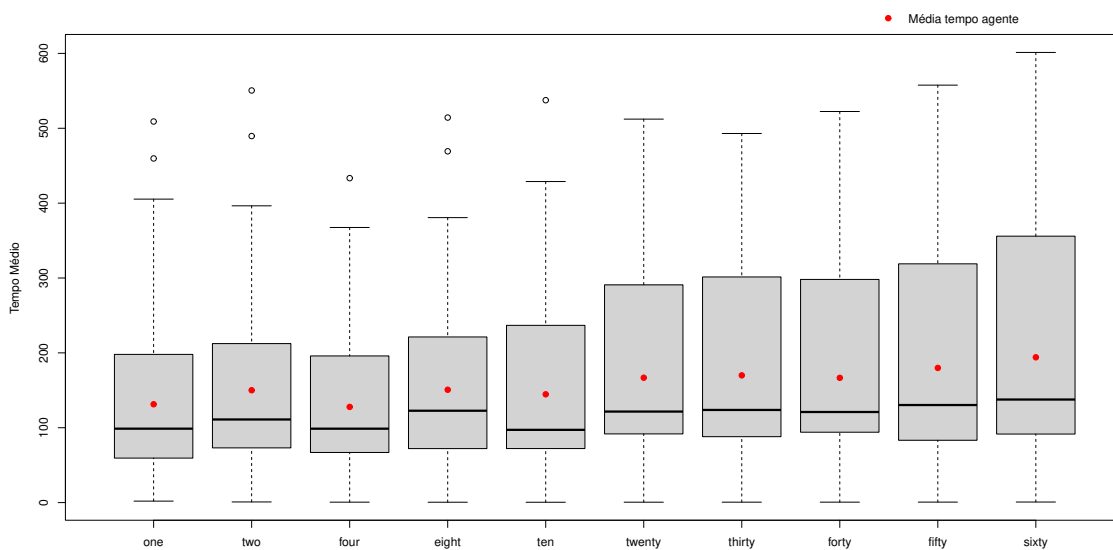
Tabela 4.2: Tempo de execução médio $\times$ Número de agentes

Inst	Número de Agentes									
	1	2	4	8	10	20	30	40	50	60
C101	17,19	13,85	9,87	5,93	4,33	2,81	2,90	2,37	2,35	2,99
C102	79,83	81,82	56,71	66,46	37,27	32,72	35,20	27,46	30,32	34,50
C103	179,12	165,38	164,11	169,85	96,58	92,89	88,66	92,31	80,50	89,51
C104	143,23	177,20	178,75	182,60	213,50	221,35	267,61	235,07	256,59	240,73
C105	12,87	17,21	11,04	9,71	5,75	3,73	3,94	3,09	3,36	4,76
C106	12,72	13,10	13,60	9,39	7,35	4,67	5,84	4,19	6,55	5,33
C107	12,98	7,57	5,07	4,15	2,89	1,96	2,69	2,38	2,67	2,82
C108	33,95	30,73	27,22	17,61	14,06	12,67	12,87	12,81	12,25	12,87
C109	115,23	106,59	87,53	84,13	88,95	93,36	91,83	96,05	86,02	93,60
C201	1,98	0,86	0,58	0,50	0,47	0,54	0,57	0,60	0,65	0,75
C202	11,45	9,95	8,09	4,81	4,29	2,90	2,25	2,48	2,56	1,61
C203	22,70	30,33	26,40	27,41	18,10	18,90	16,59	17,46	14,86	10,93
C204	66,06	86,36	82,67	105,60	82,72	97,75	109,49	100,58	108,81	127,00
C205	9,81	4,78	2,90	1,79	2,13	1,10	1,08	1,00	1,06	1,08
C206	10,10	7,08	3,23	3,28	2,32	1,75	1,26	1,20	1,20	1,44
C207	16,17	15,83	10,01	10,18	3,28	3,52	2,84	2,67	2,43	2,51
C208	17,71	18,92	12,60	7,80	8,63	5,53	3,11	4,07	3,61	2,21
RC101	263,16	299,00	245,90	275,05	255,04	292,58	279,71	295,61	318,31	356,20
RC102	310,83	363,34	271,26	337,46	344,33	330,81	326,98	328,20	342,65	396,22
RC103	116,38	140,13	166,04	154,85	207,81	260,54	295,04	304,24	350,27	292,83
RC104	110,08	133,07	146,41	168,44	187,80	246,82	293,59	271,48	334,44	310,11
RC105	288,64	362,77	272,31	335,12	332,70	368,57	356,52	341,01	387,01	400,47
RC106	252,93	271,95	228,79	301,26	265,93	321,12	310,35	313,41	337,66	369,99
RC107	216,82	247,55	193,31	246,88	223,46	291,13	278,19	310,14	313,92	370,35
RC108	120,46	129,44	149,35	156,77	192,90	236,24	292,55	265,11	267,14	340,66
RC201	62,52	77,78	89,35	100,92	90,58	107,26	100,61	99,47	113,77	116,75
RC202	99,22	142,54	111,63	143,59	120,58	139,14	142,31	133,97	136,66	157,48
RC203	57,06	65,13	65,45	68,98	76,55	92,19	106,41	109,87	112,59	113,12
RC204	67,00	92,03	83,74	89,24	97,56	114,27	122,36	122,22	131,78	133,19
RC205	98,39	116,77	112,38	131,05	118,18	137,33	128,04	123,75	135,73	156,50
RC206	76,57	90,51	68,46	82,76	76,70	91,31	90,43	98,13	98,40	103,32
RC207	98,65	105,27	85,38	105,59	88,13	122,70	112,21	114,20	125,20	129,59
RC208	61,96	77,13	78,81	82,41	70,06	118,32	125,68	118,67	126,88	139,43
R101	405,37	389,19	307,89	380,65	317,44	358,04	340,56	337,29	360,27	351,26
R102	509,02	550,48	433,39	514,31	537,50	512,22	493,02	522,40	557,62	601,28
R103	459,69	489,62	367,48	469,24	428,86	425,02	444,92	424,31	454,33	487,44
R104	241,11	288,61	238,17	282,27	244,39	307,05	320,20	291,64	319,57	387,62
R105	238,25	307,18	215,34	294,94	283,68	313,19	316,52	316,66	364,53	378,00
R106	336,01	396,42	300,66	359,45	326,34	361,63	336,70	361,16	357,45	374,01
R107	133,96	153,32	177,29	190,66	263,06	290,51	302,30	311,73	355,58	355,57
R108	275,03	321,44	250,78	331,69	320,21	381,38	360,17	322,49	382,45	409,62
R109	247,58	293,72	205,58	274,71	281,82	340,77	317,82	288,03	313,07	393,87
R110	274,86	303,81	244,14	300,50	276,67	334,37	352,23	333,55	354,84	409,35
R111	120,03	147,63	198,51	195,63	229,13	271,51	300,50	276,50	285,95	313,10
R112	105,01	131,53	125,60	155,66	202,30	278,04	318,68	300,61	313,08	420,95
R201	89,06	105,37	96,53	118,70	106,68	120,46	119,20	116,89	122,72	127,84
R202	127,87	161,12	134,28	157,29	142,47	151,83	148,85	136,49	152,06	161,69
R203	98,84	115,47	100,89	126,69	96,77	127,75	125,13	121,62	134,93	144,07
R204	113,89	126,92	103,32	128,77	98,10	115,84	118,34	118,07	116,46	131,58
R205	56,52	78,78	79,75	86,63	74,31	94,58	99,65	97,56	108,90	118,35
R206	63,82	73,17	78,22	81,24	84,04	105,80	121,33	120,60	128,89	117,29
R207	115,81	134,01	111,30	137,93	106,72	143,42	134,48	140,56	150,53	144,62
R208	63,08	72,98	72,76	75,27	76,63	98,51	87,43	95,69	106,84	109,78
R209	75,75	93,77	88,04	104,13	96,33	114,33	104,18	108,33	121,64	135,89
R210	78,45	90,41	73,36	100,57	80,38	102,67	113,55	121,42	118,70	125,57
R211	61,80	76,74	85,25	83,31	84,50	123,91	131,83	113,59	145,40	148,80

execução à medida que o número de agentes utilizados aumenta. Deve-se notar que o tempo médio de execução para as 30 execuções com 60 agentes é de 137,66 segundos, enquanto para a execução de um único agente, o tempo médio de execução é de 98,75 segundos. Portanto, o aumento no tempo de execução é pequeno em comparação

Tabela 4.3: Análise dos resultados apresentados nas Tabelas 4.1 e 4.2

Cenário	Número de Agentes									
	1	2	4	8	10	20	30	40	50	60
<i>gap</i> (%) melhor	0,43	0,33	0,14	0,13	0,09	0,04	0,05	0,05	0,05	0,01
<i>gap</i> (%) média	3,41	2,39	1,57	1,21	1,21	0,87	0,62	0,60	0,53	0,36
<i>gap</i> (%) mediana	2,85	2,19	1,38	1,12	1,09	0,60	0,50	0,30	0,39	0,29
número de vezes melhor valor	16	18	18	21	19	20	24	24	26	26
média tempo de execução	98,75	111,03	98,71	122,70	97,16	121,58	123,75	121,01	130,33	137,66
mediana tempo de execução	105,45	115,19	98,31	128,41	99,50	128,02	128,15	125,45	132,93	142,21

Figura 4.1: Comportamento do *gap* médio em relação às melhores soluções conhecidas, à média das soluções e à mediana das soluções para os 10 cenários nas 30 execuções.Figura 4.2: Tempo de execução médio $\times$ número de agentes.

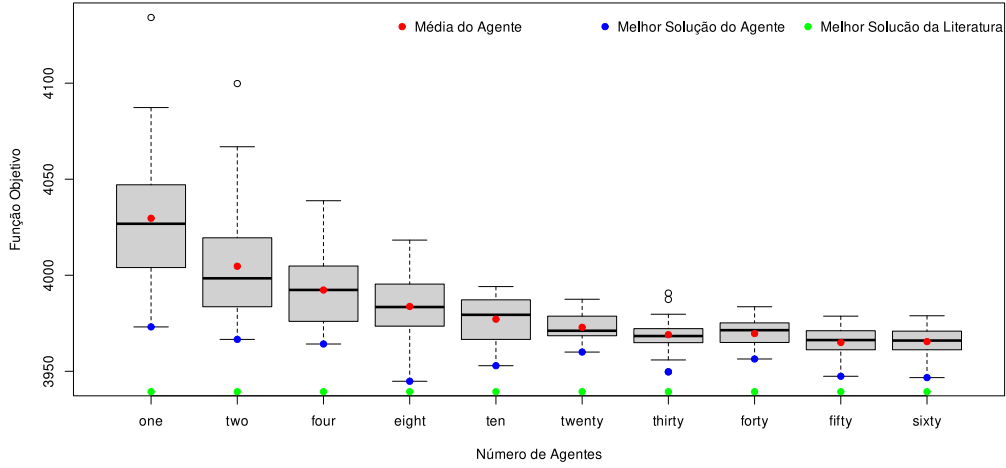
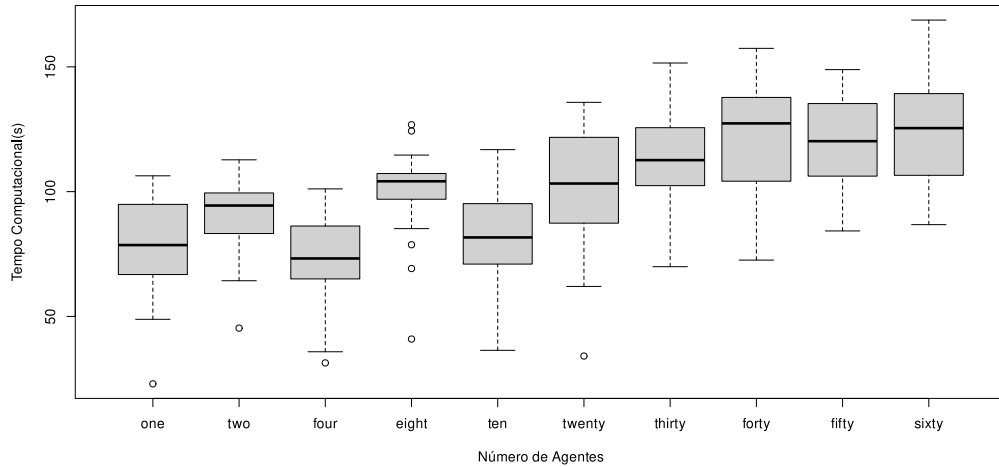
(a) Instância R210: Função Objetivo $\times$ Número de Agentes.(b) Instância R210: Tempo de Execução $\times$ Número de Agentes.

Figura 4.3: Comportamento do *Framework* AMAM ao resolver a instância R210 do PRVJT em relação à função objetivo e valores de tempo de execução $\times$ Número de agentes.

com a melhoria nas soluções obtida com a adição de agentes.

A instância R210 é utilizada para ilustrar o comportamento do *framework* em relação à função objetivo e ao tempo de execução, à medida que o número de agentes cresce. A Figura 4.3(a) apresenta um gráfico *boxplot* dos valores da função objetivo obtidos na execução dessa instância para todos os cenários de teste. Além disso, inclui os valores da média das soluções, a melhor solução obtida e a melhor solução da literatura. Esta figura claramente mostra que a adição de agentes proporciona a melhoria na qualidade das soluções obtidas, tanto no tocante ao valor médio das soluções obtidas quanto aos melhores valores encontrados pelos agentes, aproximando-os dos melhores valores da literatura e, portanto, validando a hipótese da escalabilidade para a instância em questão. Além disso, vale ressaltar também a melhoria da ro-

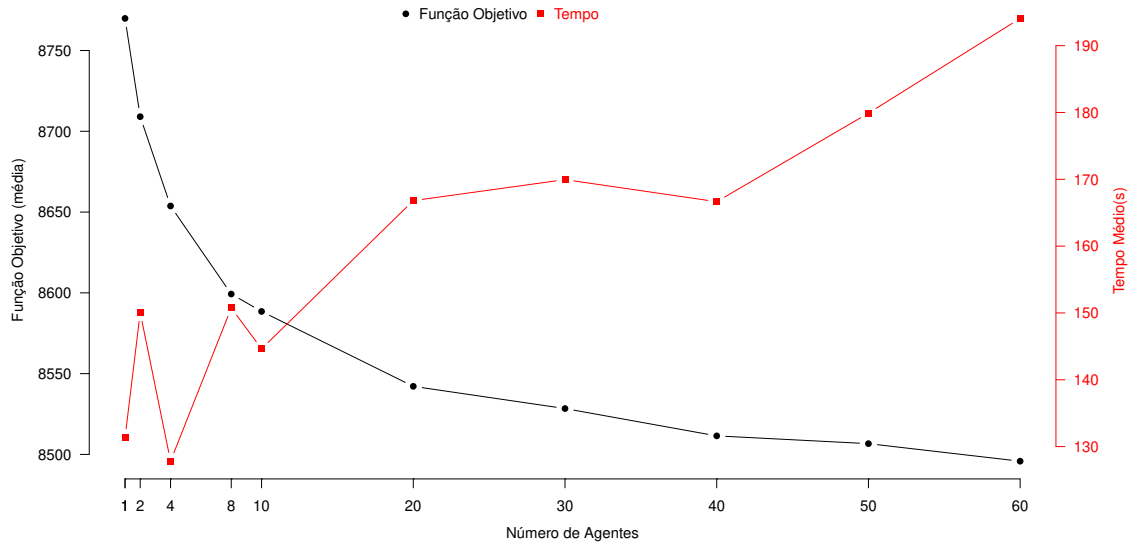


Figura 4.4: Comparação dos valores médios da função objetivo ($gap(\%)$) e do tempo de execução em relação ao aumento no número de agentes.

bustez e da eficiência do *framework* quando exposto ao funcionamento envolvendo maior número de agentes, uma vez que há uma forte diminuição na dispersão de valores quando se observa a evolução dos *boxplots* frente ao número de agentes. A Figura 4.3(b), por outro lado, mostra, também para a instância R210, um gráfico *boxplot* para o tempo de execução obtido das 30 execuções de cada cenário. Este gráfico evidencia uma leve tendência no aumento do tempo de execução à medida em que agentes são adicionados ao *framework* AMAM. Porém, surpreendentemente, este aumento não ocorre de forma acentuada, pois, apesar do número de agentes ser aumentado em 60 vezes, o tempo de execução aumenta em média em 45 segundos, o que corresponde apenas a 1,56 vezes o tempo de 1 agente.

A Figura 4.4 compara os valores médios da função objetivo e do tempo de execução para os 10 cenários avaliados, usando um gráfico com dois eixos das ordenadas. A primeira ordenada mostra o valor médio da função objetivo encontrada e a segunda o valor do tempo de execução médio obtido. Claramente, há melhoria no valor médio da função objetivo à medida que o número de agentes aumenta. Contudo, observa-se também um aumento do tempo de execução médio encontrado. Porém, este aumento pode ser considerado pequeno, pois a média do tempo de execução obtido ficou abaixo de 200 segundos na execução de 60 agentes, com um aumento de 70 segundos entre os cenários *one* e *sixty*. Já o ganho da função objetivo fica em torno de 300 unidades entre os cenários *one* e *sixty*.

4.1.3 Análise estatística

Com o intuito de verificar a existência de diferenças significativas entre os cenários implementados e, consequentemente, demonstrar a escalabilidade desta estrutura, foi realizada uma análise estatística, buscando verificar se a adição de agentes interfere na qualidade das soluções e/ou nos tempos de execução. Para isso, seguiu-se a metodolo-

gia estatística não paramétrica descrita na Seção 3.3: utilizou-se o teste de Friedman, para verificar se existem diferenças significativas entre os cenários propostos, e, em seguida, o teste pareado de Wilcoxon *signed-rank* com correção de Bonferroni, para identificar em qual ou quais dos tratamentos essas diferenças ocorrem, tomando cada instância como bloco.

Inicialmente, foi realizado o teste para verificar a qualidade dos valores da função objetivo em relação a adição de agentes. O teste de Friedman apresentou $p\text{-value} < 2e-16$, demonstrando que existe diferença significativa entre pelo menos dois dos cenários, a um nível de significância de 5%. Desta forma, rejeita-se a hipótese nula de igualdade. Em seguida, foi realizado o teste de Wilcoxon pareado com correção de Bonferroni para identificar em quais pares de cenários a diferença ocorre. Dos 45 pares possíveis, 29 apresentaram diferença estatisticamente significativa, sempre com o cenário de maior número de agentes obtendo os menores valores. Os 16 pares restantes concentram-se em duas situações: os cenários imediatamente vizinhos na escala (1×2 , 2×4 , 4×8 e 8×10 agentes) e todos os pares formados entre 20, 30, 40, 50 e 60 agentes. Ou seja, o ganho de qualidade com a adição de agentes é estatisticamente sustentado, mas satura a partir de aproximadamente 20 agentes, de modo que acrescentar novos agentes deixa de produzir diferença significativa. Os resultados indicam, em relação à qualidade das soluções, que a inclusão de agentes torna a estrutura mais robusta e eficiente, demonstrando a escalabilidade do *framework*.

O teste não paramétrico foi realizado em relação ao tempo de execução. O teste de Friedman apresentou $p\text{-value} < 2e-16$, indicando diferença significativa entre pelo menos dois cenários quanto ao tempo de execução, ao nível de significância de 5%. Vale notar que o emparelhamento é determinante nesse caso: a variabilidade do tempo entre instâncias é muito superior à variação provocada pelo número de agentes e, sem bloquear por instância, o efeito fica encoberto. Em seguida, o teste de Wilcoxon pareado com correção de Bonferroni foi realizado para verificar em quais pares existe diferença significativa, tendo apontado 28 dos 45 pares. O teste foi configurado para encontrar os cenários nos quais o tempo de computação foi menor e, em seguida, verificar em quais cenários o tempo foi maior. O tempo de computação foi maior nos seguintes cenários:

- (i) dois > um;
- (ii) oito > um e quatro;
- (iii) dez > quatro;
- (iv) vinte > dois, quatro, oito e dez;
- (v) trinta > dois, quatro, oito e dez;
- (vi) quarenta > dois, quatro, oito e dez;
- (vii) cinquenta > dois, quatro, oito, dez, vinte, trinta e quarenta;
- (viii) sessenta > dois, quatro, oito, dez, vinte, trinta, quarenta e cinquenta;

Os resultados encontrados mostram que a utilização de quatro agentes foi melhor que um e dois agentes e a utilização de dez agentes foi melhor que a utilização de

dois e oito agentes. Ou seja, os tempos de execução foram menores entre estes cenários. Destaca-se também que os cenários com 20, 30 e 40 agentes não apresentaram diferenças significativas em relação ao tempo. A partir daí, os testes mostram que, na maioria dos casos, a inclusão de agentes aumenta o tempo de execução ao nível de significância de 5%. Porém, como já mencionado anteriormente, este aumento não invalida a utilização do sistema multiagente, pois o ganho em relação à função objetivo é mais relevante do que o ganho de tempo. Além disso, de certa forma, o tempo de execução é baixo.

4.1.4 Ajuste Linear

A partir dos resultados já discutidos, tentou-se realizar um ajuste linear em relação à inclusão de agentes. Inicialmente, para realizar o ajuste linear entre o número de agentes e o valor da função objetivo, se fez necessário a alteração dos valores da função objetivo e do número de agentes para uma escala logarítmica, com o intuito de verificar se os dados assumiriam um comportamento linear. O gráfico da Figura 4.5 foi criado a partir do gráfico da Figura 4.1.

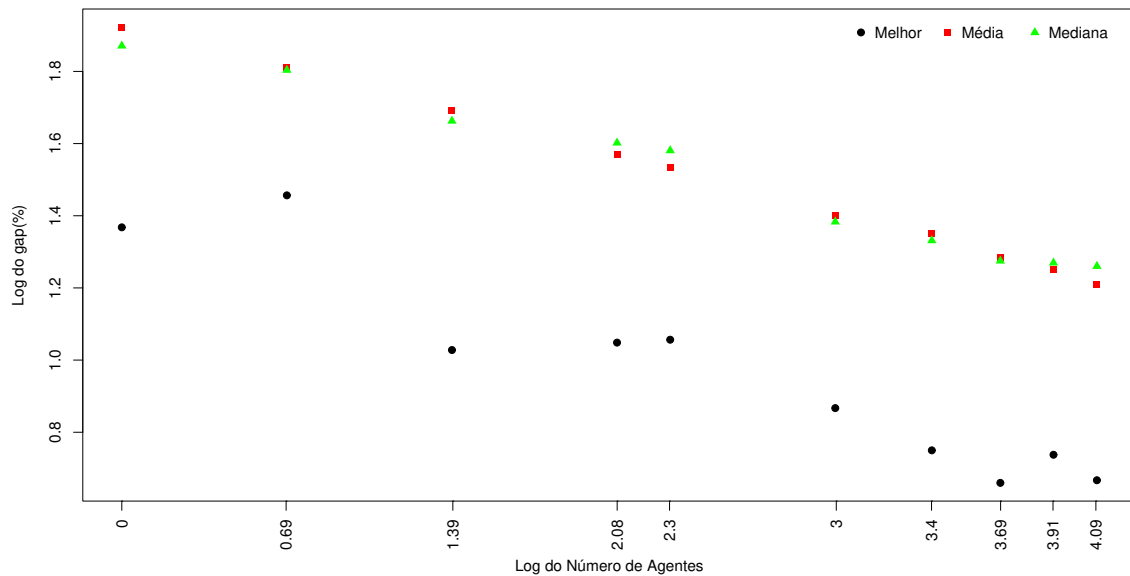


Figura 4.5: Comportamento do *gap* em uma escala logarítmica.

A Figura 4.5 apresenta os resultados em uma escala logarítmica. Observa-se um comportamento próximo ao linear para os valores médios e medianos das soluções, mas o mesmo não ocorre para os valores da melhor solução. Portanto, procedimentos de regressão linear foram realizados nos valores das três medidas estatísticas (média, mediana e melhor solução), e os resultados encontrados são mostrados na Figura 4.6. Esta figura, portanto, apresenta um gráfico de regressão linear para o valor da função objetivo da média, da mediana e da melhor solução obtida. As regressões lineares encontraram as seguintes funções: média = $-0,1737173x + 1,9290996$; mediana = $-0,161440x + 1,900294$; e melhor = $-0,1904809x + 1,4315237$. O ajuste de regressão linear R^2 apresentou valores de 0,9992, 0,9805 e 0,9034, respectivamente, para a média, mediana e melhor solução. O comportamento das retas apresentou um bom

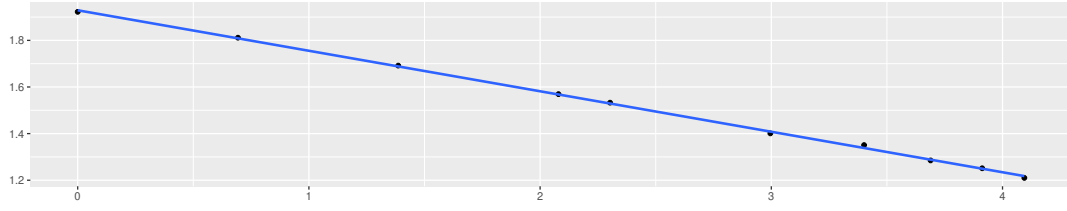
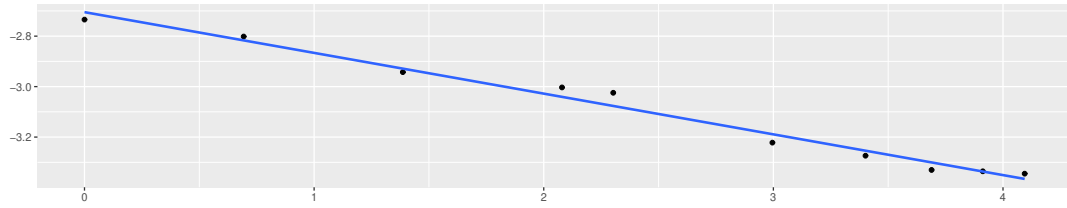
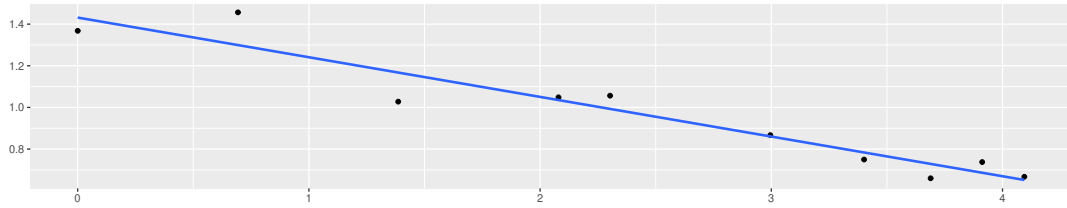
(a) Regressão Linear para Média. Média = $-0,1737173x + 1,9290996$.(b) Regressão Linear para Mediana. Mediana = $-0,161440x + 1,900294$ (c) Regressão Linear para Melhor. Melhor = $-0,1904809x + 1,4315237$

Figura 4.6: Regressão Linear para Valores de Função Objetivo.

ajuste linear para todas as medidas estatísticas, e o melhor ajuste encontrado foi para a média das soluções. Portanto, a melhor maneira de representar a inclusão de agentes no *Framework AMAM* é a função linear referente à média das soluções.

O mesmo procedimento é adotado para analisar o comportamento do tempo de execução em função do número de agentes. A Figura 4.7 apresenta um gráfico da regressão linear em relação ao tempo de execução médio, mediano e melhor obtido em função do número de agentes. O ajuste à regressão linear apresentou valores de 0,7361; 0,6745; e 0,8176 para o tempo médio, mediano e melhor tempo de execução, respectivamente. As funções de linha reta encontradas foram $13,0450x + 126,1668$; $13,54566x + 131,13084$; e $8,522198x + 45,208538$. Seguindo os valores de ajuste linear, as linhas retas não representaram bem o tempo de execução em relação à inclusão de agentes. A melhor linha reta apresentou um ajuste linear de 0,8176. No entanto, esse ajuste ocorreu apenas para os melhores tempos de execução obtidos, o que significa que as funções lineares não são boas aproximações do comportamento do *Framework AMAM*.

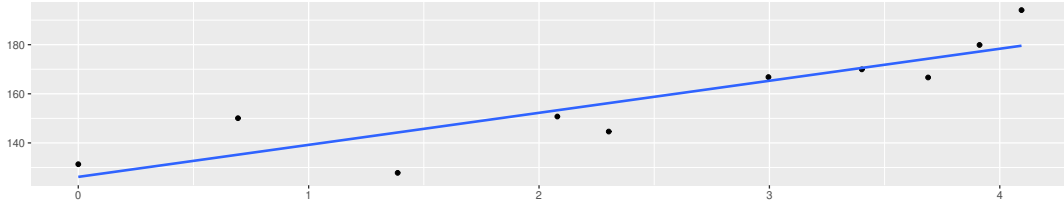
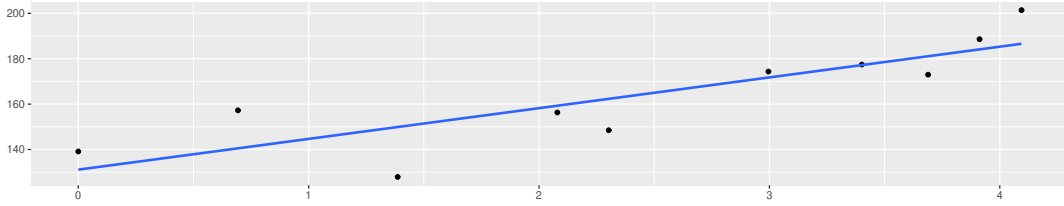
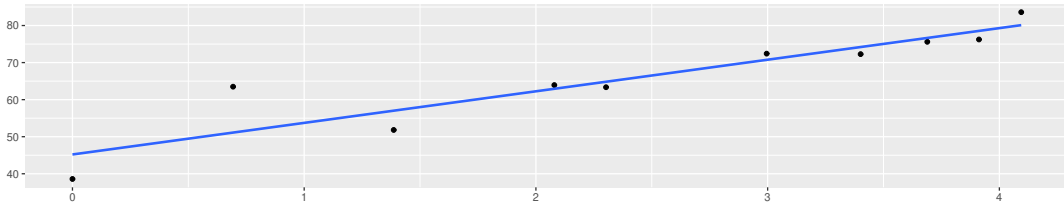
(a) Regressão Linear para Média. Média = $13,0450x + 126,1668$ (b) Regressão Linear para Mediana. Mediana = $13,54566x + 131,13084$ (c) Regressão Linear para Melhor. Melhor = $8,522198x + 45,208538$

Figura 4.7: Regressão Linear para Valores de Tempo de Execução

4.1.5 Agente Único $\times$ Multiagente

Esta seção apresenta os testes realizados para comparar o desempenho do *Framework* AMAM com um único agente e com sessenta agentes. As quatro situações diferem em dois aspectos: quantas observações de cada caso (ou agente único ou sessenta agentes) entram na comparação e, principalmente, se o sistema de agente único recebe ou não o mesmo orçamento computacional do sistema multiagente. O sistema multiagente foi executado 30 vezes com 60 agentes, o que produz $30 \times 60 = 1800$ resultados individuais de agente ou, se for tomada a melhor solução de cada execução, 30 resultados. O sistema de agente único, por sua vez, foi executado 1800 vezes, de modo a igualar o esforço total. Essa comparação envolve a avaliação dos valores da função objetivo e do tempo de execução em quatro situações diferentes:

- (i) os valores de função objetivo obtidos com 1800 execuções do sistema de agente único $\times$ os valores de função objetivo obtidos por cada um dos 60 agentes de cada uma das 30 execuções do sistema multiagente (1800×1800);
- (ii) os valores de função objetivo obtidos pelas 30 execuções do sistema de agente único $\times$ os valores de função objetivo obtidos com 30 execuções do sistema multiagente, sendo, neste caso, utilizada a melhor solução encontrada em cada execução do sistema multiagente com 60 agentes (30×30);

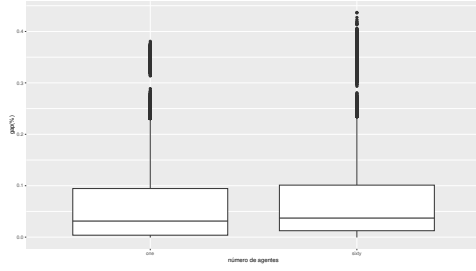
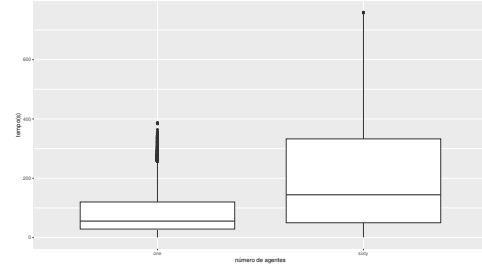
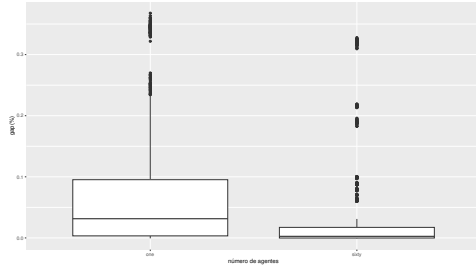
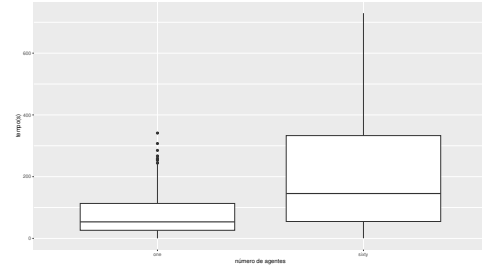
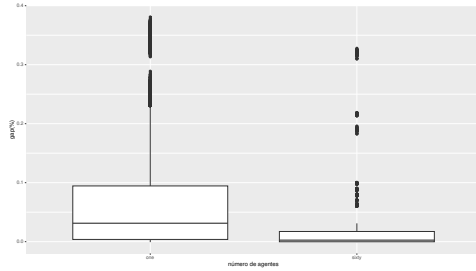
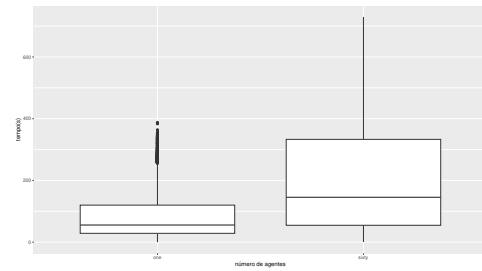
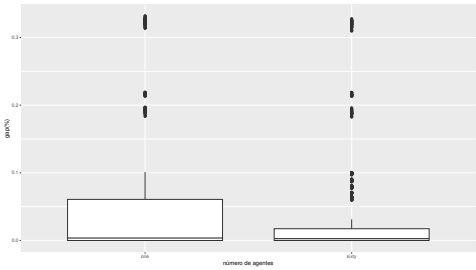
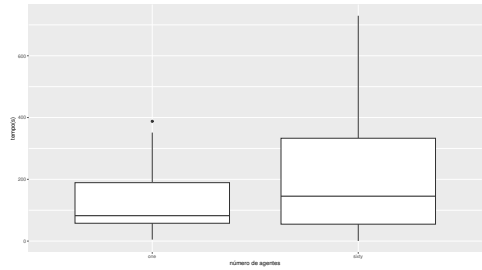
- (iii) os valores de função objetivo obtidos com as 1800 execuções do sistema de agente único $\times$ os valores de função objetivo obtidos com 30 execuções do sistema multiagente, sendo, neste caso, utilizada a melhor solução encontrada em cada execução do sistema multiagente com 60 agentes (1800×30);
- (iv) as 1800 execuções do sistema de agente único agrupadas em 30 blocos de 60 execuções, tomando-se de cada bloco o melhor valor de função objetivo e o pior tempo de execução $\times$ os valores obtidos nas 30 execuções do sistema multiagente, novamente pela melhor solução de cada execução (30×30 máquina). Cada bloco de 60 execuções independentes emula uma máquina rodando 60 agentes em paralelo sem cooperação: o melhor valor porque a máquina devolveria a melhor das 60 soluções e o pior tempo porque só terminaria quando o último agente terminasse.

As situações (ii) e (iv) são ambas rotuladas 30×30 pois comparam 30 observações de cada lado, mas respondem a perguntas diferentes. Em (ii), o agente único dispõe de uma única execução, enquanto o sistema multiagente dispõe de 60 execuções, de modo que a comparação é entre esforços desiguais e mede o ganho de se usar mais agentes. Na situação (iv), os dois lados recebem 60 execuções, e a comparação passa a medir o que a cooperação acrescenta em relação a simplesmente executar 60 agentes independentes no mesmo tempo de máquina. Assim, a situação (iv) é a comparação mais exigente dentre as quatro avaliadas.

Diferentemente das demais análises desta tese, três das quatro situações aqui avaliadas confrontam conjuntos de tamanhos distintos e sem correspondência entre as observações, o que impede o emparelhamento. Nelas, foi empregado o teste de Wilcoxon-Mann-Whitney ([Mann e Whitney, 1947](#)), para amostras independentes; apenas a situação (ii), em que as 30 execuções de cada sistema se correspondem, foi avaliada com o teste de Wilcoxon pareado. Os p -valores obtidos para os dados referentes à função objetivo e ao tempo de execução foram, respectivamente:

- (i) $< 2e-16$ e $< 2e-16$;
- (ii) $< 2e-16$ e $< 2e-16$;
- (iii) $< 2e-16$ e $< 2e-16$;
- (iv) 0,091 e $< 2e-16$.

Conclui-se, portanto, que as situações (i), (ii) e (iii) apresentam diferença significativa, ao nível de confiança de 5%, quanto aos valores da função objetivo, e que a situação (iv) não apresenta tal conclusão. Nas situações (ii) e (iii), que confrontam o resultado entregue por cada sistema, a diferença favorece o sistema multiagente. Já na situação (i), que confronta execuções individuais, a diferença favorece o agente único, cuja mediana de *gap* é de 3,12% contra 3,70% dos agentes tomados individualmente dentro do sistema multiagente: isoladamente, um agente sem compartilhamento chega a soluções melhores do que cada agente do enxame, e o ganho do sistema multiagente está em selecionar a melhor entre as 60 buscas, não em tornar cada busca individualmente melhor. Em relação ao tempo de execução, em todas as situações, o uso do sistema de um único agente foi melhor. No entanto, como mencionado anteriormente, esse aumento no tempo não é significativo no contexto da aplicação desenvolvida.

(a) 1800×1800 função objetivo(b) 1800×1800 tempo de execução(c) 1800×30 função objetivo(d) 1800×30 tempo de execução(e) 30×30 função objetivo(f) 30×30 tempo de execução(g) Machine 30×30 função objetivo(h) Machine 30×30 tempo de execuçãoFigura 4.8: Agente único $\times$ Multiagente.

A Figura 4.8 apresenta os resultados encontrados para as quatro situações avaliadas. Os *boxplots* mostram que o sistema multiagente teve um desempenho superior nas situações (ii) (ou seja, 30×30) e (iii) (ou seja, 1800×30). Embora não haja diferenças significativas na situação (iv) (ou seja, 30×30 máquina), o *boxplot* mostra que o sistema multiagente foi mais robusto do que o sistema de agente único. No entanto, o tempo de execução foi maior ao usar sistemas multiagente nas quatro situações avaliadas, mas, como já explicado, com um aumento proporcionalmente pequeno. A Figura 4.9 mostra os resultados do tempo de execução para as quatro si-

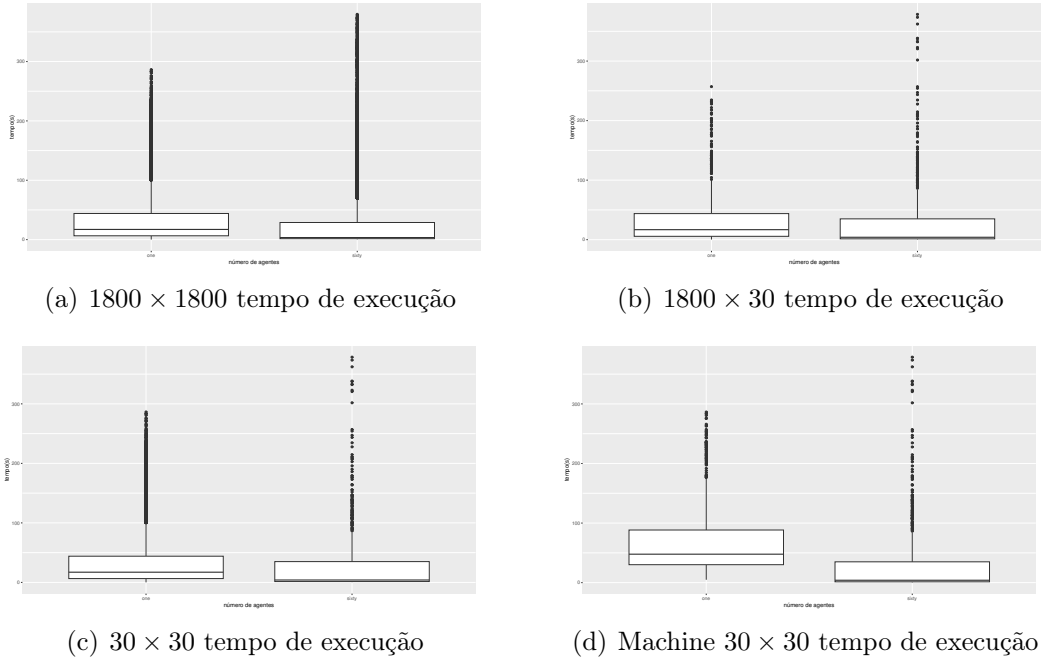


Figura 4.9: Agente único × Multiagente tempo de execução (C - Cluster).

tuações avaliadas no caso do conjunto de instâncias C. Em todos os casos, o tempo de execução mediano do sistema multiagente permaneceu abaixo do tempo de execução mediano do sistema de agente único. Quando realizamos o teste de Wilcoxon, apenas as situações (i) e (iv) mostraram uma diferença significativa para este conjunto de instâncias. Portanto, concluímos que o tempo de execução não é um problema que implique dificuldades na aplicação do *Framework AMAM*.

4.1.6 Considerações Parciais

Os resultados indicam, em relação à qualidade das soluções, que a inclusão de agentes torna a estrutura mais robusta e eficiente, demonstrando a escalabilidade do *framework*. Em relação ao tempo de execução, observa-se uma leve tendência de crescimento à medida que o número de agentes aumenta. Porém, este crescimento não invalida a utilização do sistema multiagente, pois o ganho associado à melhoria dos valores da função objetivo é mais relevante do que o crescimento do tempo de execução. Portanto, concluímos que o tempo de execução não é um problema que implique dificuldades na aplicação do *Framework AMAM*.

4.2 Cooperação

Os conceitos de sistema multiagente e de cooperação empregados nesta seção são os apresentados na Seção 2.2.1 do Capítulo 2 e não são aqui repetidos. Interessa, para os experimentos que se seguem, apenas a forma concreta como a cooperação é realizada no *Framework AMAM*: a dimensão *Cooperação*, descrita na Seção 2.1.1.3, é implementada por um *Pool* de Soluções mantido no ambiente, no qual cada agente deposita, ao final de cada iteração, a solução que encontrou e do qual pode coletar

uma solução produzida por outro agente para reiniciar a sua busca. A cooperação, portanto, resume-se à escrita e à leitura de soluções nesse conjunto compartilhado.

O principal objetivo da presente seção é avaliar o comportamento do *Framework* AMAM quanto à cooperação entre agentes. Para isso, executamos testes com e sem a dimensão de cooperação inclusa à estrutura do *framework*. Por *estrutura sem cooperação* entende-se a execução em que a dimensão **Cooperação** é desativada: os agentes são instanciados e executados em paralelo, mas não acessam o *Pool* de Soluções, de modo que cada agente realiza uma busca independente e o resultado do cenário é a melhor solução dentre as buscas isoladas. Por *estrutura com cooperação* entende-se a execução em que a mesma dimensão está ativa: os agentes inserem e coletam soluções do *Pool* durante a busca, de modo que a solução encontrada por um agente pode ser usada como ponto de partida por outro. Assim, a única diferença entre os dois cenários comparados a seguir é a ativação da dimensão de cooperação; o problema, as instâncias, os parâmetros, o número de agentes e o critério de parada são idênticos. As medidas avaliadas são os valores de função objetivo e de tempo de execução. Experimentos computacionais são realizados e, em seguida, testes estatísticos são aplicados para avaliar os resultados obtidos.

4.2.1 Definição da Experimentação

Os experimentos computacionais desta seção utilizaram as instâncias C109, C208, R112, R211, RC108 e RC208 do PRVJT. Os testes foram realizados em um computador (nó de cálculo) com processador Intel(R) Xeon(R) CPU X5650 (12M Cache, 2.66 GHz, 6.40 GT/s Intel(R) QPI, 6 cores, 12 threads), totalizando 24 núcleos de execução, sob o sistema operacional Linux CentOS.

Os testes foram executados com dois, quatro, oito e dezesseis agentes. Dos 24 núcleos disponíveis na máquina utilizada, 20 foram destinados aos agentes do *framework* e os 4 restantes foram reservados para a JVM do JAVA e para o sistema operacional. Cada agente do *framework* foi executado em sua própria *thread* JAVA e em sua própria *thread* física (processador/núcleo). Para que as execuções com diferentes números de agentes utilizassem os mesmos 20 núcleos disponíveis, foram organizados dois cenários de execução: o primeiro com duas execuções de 2 agentes em paralelo somadas a duas execuções de 8 agentes em paralelo, totalizando 20 *threads* simultâneas; e o segundo com uma execução de 4 agentes em paralelo somada a uma execução de 16 agentes, totalizando também 20 *threads*. Essa medida foi adotada com o objetivo de reduzir o ruído das execuções e manter os recursos computacionais disponibilizados a cada cenário o mais equitativos possível. Esta mesma configuração é adotada na Seção 4.3 e no Capítulo 5, que a ela remetem.

Para a experimentação da cooperação foram criados três cenários de execução do *framework*, denominados ILSN, ILSBI e ILSNMIX, utilizando os agentes definidos na Seção 3.2 e Tabela 3.1. Estes cenários são apresentados a seguir:

- (i) ILSN: este cenário utiliza-se de agentes ILS_ALS_NO em sua execução; desta forma, todos os agentes implementados neste cenários são iguais. Assim, este cenário não se utiliza da estrutura da cooperação do *framework*; e;
- (ii) ILSBI: este cenário implementa agentes ILS_ALS_BI. Da mesma forma que o cenário anterior, este cenário utiliza-se de agentes iguais, porém, agentes

ILS_ALS_BI; diferentemente do cenário anterior, estes agentes acessam o *Pool* de Soluções, inserindo nele a solução obtida ao final de cada iteração e dele coletando uma solução para reiniciar a busca, conforme as regras de inserção e retirada descritas na Seção 2.1.1.3. É esse acesso, e somente ele, que caracteriza o uso da cooperação neste cenário. A comparação entre ILSN e ILSBI isola, assim, o efeito da cooperação sobre a qualidade das soluções e sobre o tempo de execução.

- (iii) ILSNMIX: este cenário é parcialmente cooperativo. Metade dos agentes é do tipo ILS_ALS_BI, com acesso ao *Pool*, e a outra metade é do tipo ILS_ALS_NO, sem acesso. Ele foi incluído para verificar se o benefício da cooperação exige que todos os agentes compartilhem soluções ou se já se manifesta quando apenas parte deles o faz.

A execução destes três cenários buscam validar a cooperação utilizada na AMAM. Para isso, os resultados destas execuções são apresentados a seguir.

4.2.2 Resultados Computacionais

Para introduzir os resultados dos cenários ILSN, ILSBI e ILSNMIX definidos acima, inicialmente são apresentadas as Tabelas 4.4 e 4.5.

A Tabela 4.4 apresenta os valores médios da função objetivo e do *gap* (%) para cada instância executada, considerando 30 execuções de cada cenário. Os melhores resultados foram obtidos nas instâncias do tipo *C*, enquanto os piores resultados foram encontrados nas instâncias do tipo *R* em ambos os cenários. A escalabilidade em relação à adição de agentes foi mais eficaz na execução do cenário ILSN do que no cenário ILSBI. Em relação à cooperação, os resultados indicam uma tendência de melhora nos resultados alcançados pelo cenário ILSBI, que faz uso da cooperação.

A Tabela 4.5 apresenta os resultados referentes aos valores de tempos de execução obtidos nas 30 execuções de cada cenários. Como era esperado, a execução do cenário ILSN foi mais rápida em todas as execuções. Os resultados indicam que o uso do cenário ILSBI resulta em um tempo proporcionalmente maior. No entanto, esse aumento é previsível e não é motivo de preocupação, uma vez que esse tempo ainda é consideravelmente baixo. Além disso, o cenário ILSBI alcança melhores resultados em relação à função objetivo do que o ILSN. Enquanto o cenário ILSN possui um *gap* médio de 15,52%, o cenário ILSBI mantém um *gap* médio de 8,88%. O cenário ILSNMIX, em que apenas metade dos agentes coopera, alcança um *gap* médio de 8,88%, praticamente idêntico ao do cenário totalmente cooperativo e muito abaixo do cenário sem cooperação. Isso indica que o benefício da cooperação já é obtido quando somente parte dos agentes compartilha soluções.

A Figura 4.10 apresenta o resultado do *gap* percentual médio e do tempo de execução médio da execução dos três cenários sem a separação por instâncias. Os gráficos mostram a melhora nos valores da função objetivo obtidos com os cenários ILSBI e ILSNMIX, em que ambos utilizam-se da cooperação. No entanto, os melhores valores de tempo de execução foram alcançados com o cenário ILSN. O gráfico 4.10(b), que mostra os resultados do tempo de execução, evidencia a diferença estatística entre os métodos. No entanto, o gráfico 4.10(a), referente ao *gap*(%), não é capaz de comprovar a diferença estatística entre os cenários.

Tabela 4.4: Função Objetivo e $gap(\%)$ ILSN $\times$ ILSBI

Cenário	Instância		Número de Agentes				Média
			2	4	8	16	
ILSN	C109	FO	11132,10	11086,75	11064,72	11020,61	11076,05
		<i>gap</i>	2,80	2,38	2,18	1,77	2,28
	C208	FO	3719,96	3677,17	3636,85	3614,04	3662,01
		<i>gap</i>	3,67	2,48	1,35	0,72	2,05
	R112	FO	12394,90	12162,70	12175,42	12016,53	12187,39
		<i>gap</i>	24,17	21,84	21,97	20,38	22,09
	R211	FO	4031,39	4004,81	3970,21	3963,21	3992,40
		<i>gap</i>	39,70	38,78	37,58	37,34	38,35
	RC108	FO	13581,60	13363,51	13287,14	13164,24	13349,12
		<i>gap</i>	21,92	19,96	19,28	18,17	19,83
	RC208	FO	4193,56	4158,16	4156,93	4105,41	4153,52
		<i>gap</i>	9,55	8,62	8,59	7,24	8,50
	Média	<i>gap</i>	16,97	15,68	15,16	14,27	15,52
ILSBI	C109	FO	10846,36	10838,96	10828,94	10829,56	10835,95
		<i>gap</i>	0,16	0,09	0,00	0,01	0,06
	C208	FO	3597,35	3588,48	3588,32	3588,32	3590,62
		<i>gap</i>	0,25	0,00	0,00	0,00	0,06
	R112	FO	11007,69	10998,73	10989,13	10981,21	10994,19
		<i>gap</i>	10,27	10,18	10,09	10,01	10,14
	R211	FO	3843,32	3840,28	3819,28	3814,40	3829,32
		<i>gap</i>	33,18	33,08	32,35	32,18	32,70
	RC108	FO	12179,10	11912,77	11993,45	12053,51	12034,71
		<i>gap</i>	9,33	6,94	7,66	8,20	8,03
	RC208	FO	3946,91	3927,76	3894,37	3889,11	3914,54
		<i>gap</i>	3,10	2,60	1,73	1,59	2,26
	Média	<i>gap</i>	9,38	8,81	8,64	8,66	8,88
ILSNMIX	C109	FO	10845,39	10832,90	10829,30	10827,41	10833,75
		<i>gap</i>	0,15	0,04	0,00	-0,01	0,04
	C208	FO	3595,81	3586,83	3586,04	3586,01	3588,67
		<i>gap</i>	0,21	-0,04	-0,06	-0,06	0,01
	R112	FO	11006,36	10999,41	10986,66	10983,82	10994,06
		<i>gap</i>	10,26	10,19	10,06	10,03	10,13
	R211	FO	3851,28	3837,74	3824,02	3812,60	3831,41
		<i>gap</i>	33,46	32,99	32,52	32,12	32,77
	RC108	FO	12148,02	12097,23	12059,63	11857,21	12040,52
		<i>gap</i>	9,05	8,59	8,26	6,44	8,09
	RC208	FO	3948,59	3920,90	3896,43	3887,13	3913,26
		<i>gap</i>	3,15	2,42	1,78	1,54	2,22
	Média	<i>gap</i>	9,38	9,03	8,76	8,34	8,88

A Figura 4.11 apresenta a evolução dos cenários ILSN, ILSBI e ILSNMIX. Os gráficos ilustram a evolução de uma execução da instância RC108 com 4 agentes. O gráfico 4.11(a) mostra a evolução do cenário ILSN, em que é possível observar que a evolução de cada agente é curta, com apenas algumas melhorias ocorrendo durante o processo de busca. Por outro lado, o gráfico 4.11(b) apresenta a evolução do cenário ILSBI, em que podemos observar que a evolução resultou em melhores valores de função objetivo. Além disso, a melhoria dos valores de função objetivo de cada agente ocorre em um número maior de vezes do que no método ILSN. Também é possível observar o momento em que cada agente coloca uma solução no *Pool* e em que momento essa solução foi coletada por outro agente durante o processo de busca. O gráfico 4.11(c) apresenta o cenário ILSNMIX: os agentes cooperativos inserem e

Tabela 4.5: Tempo médio de execução (s) ILSN $\times$ ILSBI

Cenário	Instância	Número de Agentes				Média
		2	4	8	16	
ILSN	C109	16,90	19,20	19,69	20,50	19,07
	C208	4,30	4,87	5,88	7,39	5,61
	R112	24,49	25,35	26,39	27,93	26,04
	R211	5,65	6,91	7,33	8,15	7,01
	RC108	25,21	26,81	28,50	29,32	27,46
	RC208	5,65	6,47	6,86	8,35	6,83
	Média	13,70	14,93	15,77	16,94	15,34
ILSBI	C109	170,33	151,41	120,34	85,14	131,80
	C208	25,41	10,62	12,47	6,54	13,76
	R112	223,43	243,63	236,08	264,88	242,01
	R211	137,89	140,18	143,98	150,45	143,12
	RC108	246,04	259,41	257,59	272,46	258,88
	RC208	122,98	127,41	131,04	139,22	130,16
	Média	154,35	155,44	150,25	153,12	153,29
ILSNMIX	C109	222,12	152,58	180,48	113,96	167,28
	C208	24,70	16,04	16,63	7,24	16,15
	R112	284,20	306,14	291,51	328,28	302,53
	R211	130,48	138,63	138,51	154,94	140,64
	RC108	241,01	287,40	249,41	318,02	273,96
	RC208	117,28	129,29	125,28	139,73	127,89
	Média	169,96	171,68	166,97	177,03	171,41

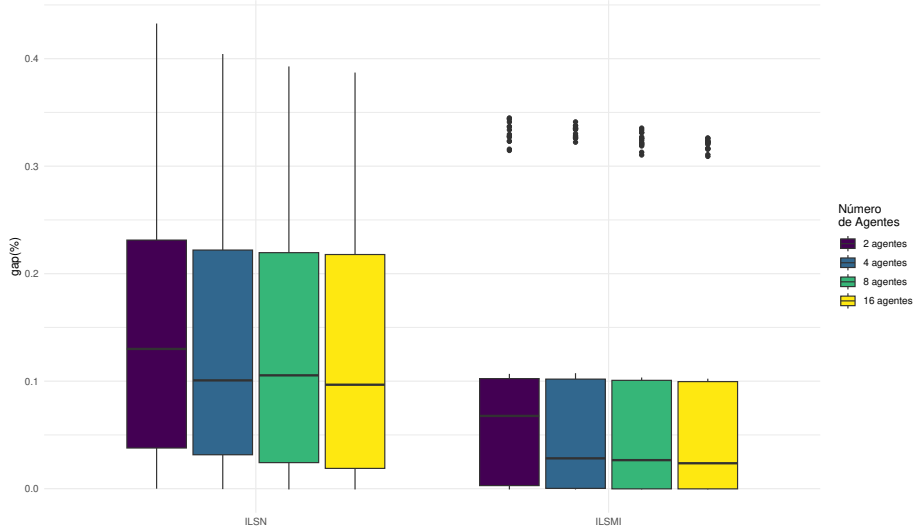
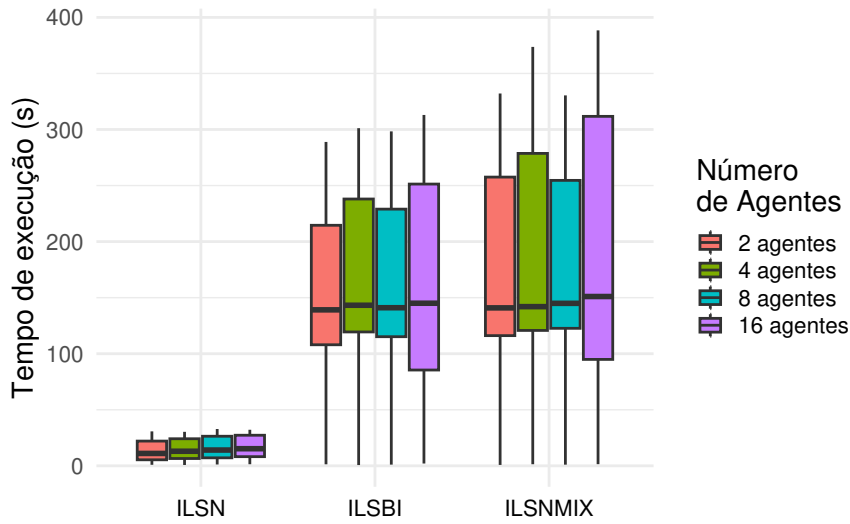
coletam soluções do *Pool* e evoluem de forma semelhante à do cenário ILSBI, enquanto os agentes não cooperativos evoluem menos, o que é coerente com a natureza parcialmente cooperativa desse cenário.

Os resultados de tempo de execução levantam a seguinte questão:

O cenário ILSN alcançaria valores de função objetivo iguais ou até melhores caso fosse executado com um tempo equivalente ao gasto pelo cenário ILSBI?

Para responder a essa questão e, ao mesmo tempo, tornar a avaliação da cooperação mais justa, foi realizado um novo experimento computacional em que o critério de parada passa a ser o tempo de execução, e não o número de iterações sem melhora. Dessa forma, os dois cenários dispõem da mesma oportunidade de encontrar boas soluções. Os testes nesse formato foram executados 30 vezes, com critério de parada estabelecido em 300 segundos para cada execução isolada. Esse valor foi definido tomando como referência o maior tempo médio observado para o cenário ILSBI na Tabela 4.5, de 272,46 segundos. A Tabela 4.6 apresenta os resultados desse novo experimento.

A Figura 4.12 compara os valores de *gap*(%) obtidos com o critério de parada por tempo de execução e os obtidos na execução anterior, com critério de parada por número de iterações sem melhora. Não se observa diferença expressiva entre as duas situações, e o teste não paramétrico de Wilcoxon pareado, com nível de confiança de 95%, confirmou que não há diferença estatisticamente significativa entre

(a) $Gap(\%)$.

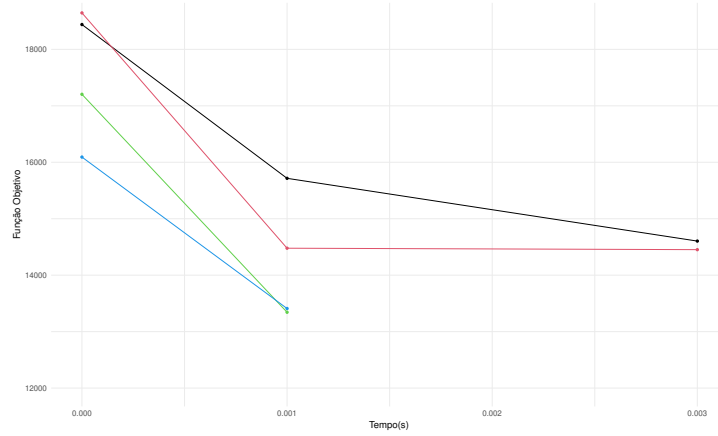
(b) Tempo de Execução(s)

Figura 4.10: ILSN $\times$ ILSBI $\times$ ILSNMIX

elas. Portanto, mesmo dispondo de um tempo de execução muito maior, o cenário ILSN não supera o cenário ILSBI quanto à qualidade das soluções, o que evidencia a efetividade da estrutura de cooperação do *Framework* AMAM.

4.2.3 Considerações Parciais

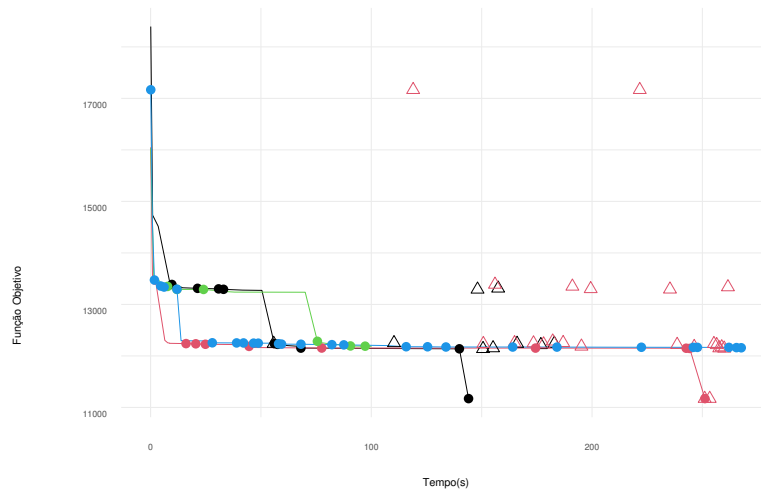
A principal contribuição desta seção é a avaliação da cooperação do *Framework* AMAM, demonstrando sua capacidade de melhora no desempenho do sistema multi-agente à medida a cooperação é incluída. Desta forma, esta seção buscou analisar a dimensão cooperação deste *framework* com o intuito de validá-la. Para avaliar a cooperação do *Framework* AMAM avaliou-se: (i) a existência de diferenças significati-



(a) ILSN



(b) ILSBI

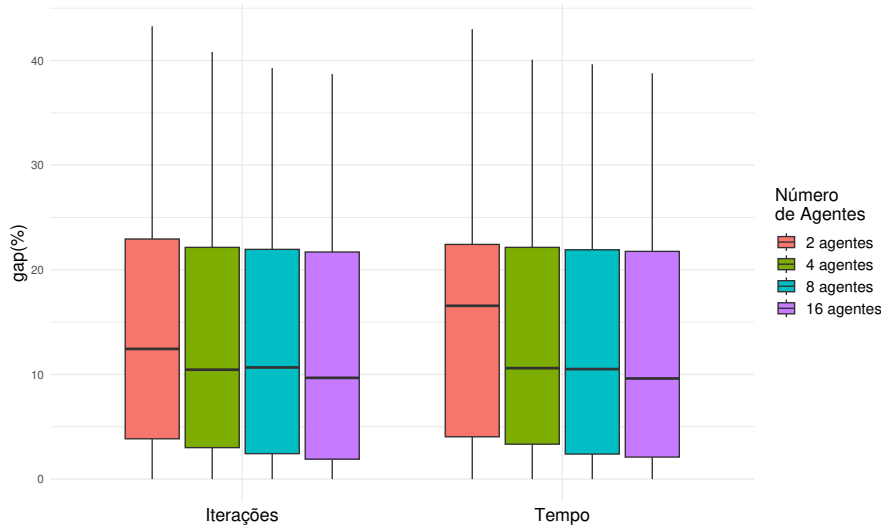


(c) ILSNMIX

Figura 4.11: Evolução ILSN $\times$ ILSBI $\times$ ILSNMIX - instância RC108 com 4 agentes. Cada cor identifica um agente; $\bullet$ marca o momento em que uma solução é inserida no *Pool* e $\triangle$ o momento em que uma solução é coletada do *Pool* por um agente.

Tabela 4.6: Função Objetivo e $gap(\%)$ do cenário ILSN com o tempo de execução como critério de parada

Cenário	Instância		Número de Agentes				Média
			2	4	8	16	
ILSN	C109	FO	11122,14	11104,12	11057,79	11028,90	11078,24
		gap	2,71	2,54	2,11	1,85	2,30
	C208	FO	3715,15	3687,09	3638,47	3626,24	3666,74
		gap	3,53	2,75	1,40	1,06	2,19
	R112	FO	12243,94	12260,04	12130,65	12154,92	12197,39
		gap	22,66	22,82	21,52	21,77	22,19
	R211	FO	4028,96	3982,37	3979,58	3964,28	3988,80
		gap	39,62	38,00	37,91	37,38	38,23
	RC108	FO	13544,64	13334,47	13386,83	13224,61	13372,64
		gap	21,59	19,70	20,17	18,71	20,04
	RC208	FO	4202,48	4167,36	4131,86	4111,93	4153,41
		gap	9,78	8,86	7,93	7,41	8,50
	Média	gap	16,65	15,78	15,17	14,70	15,57

Figura 4.12: Cenário ILSN e critério de parada: iterações sem melhora $\times$ tempo de execução, $gap(\%)$

vas entre as médias dos valores de função objetivo e tempo de execução obtidos a partir da inserção de novos agentes; (ii) os resultados obtidos utilizando o sistema com cooperação e sem cooperação, com o intuito de validar a utilização de sistemas multiagente com cooperação para a solução de problemas de otimização; (iii) comportamento do tempo de evolução da estrutura à medida que a cooperação é incluída no processo de busca do *framework*. Os resultados foram avaliados a partir dos testes estatísticos não paramétricos descritos na Seção 3.3.

Os testes estatísticos foram aplicados em todos os cenários, com as 6 instâncias e os 4 números diferentes de agentes. Os resultados mostraram que houve uma diferença estatisticamente significativa em todos os cenários em relação ao valor da função objetivo. O agente ILSBI obteve os melhores resultados em comparação com o agente ILSN, o que confirma a eficiência da cooperação na estrutura do *framework*. Como os dados não apresentaram distribuição normal, foram empregados testes não

paramétricos pareados, tomando cada combinação de instância e número de agentes como bloco, com correção de Bonferroni e nível de confiança de 95%, conforme a Seção 3.3. O teste de Friedman confirmou a existência de diferença entre os cenários ($p < 0,001$) e o teste de Wilcoxon pareado apontou o ILSBI como significativamente melhor que o ILSN tanto em função objetivo quanto em tempo ($p = 2 \times 10^{-5}$ em ambos, após a correção). Quanto à função objetivo, o teste de Wilcoxon pareado não indicou diferença estatisticamente significativa entre o cenário parcialmente cooperativo ILSNMIX e o totalmente cooperativo ILSBI ($p = 0,11$), ao passo que o ILSNMIX foi significativamente melhor que o cenário sem cooperação ILSN ($p < 0,001$). Isso confirma que a configuração parcialmente cooperativa atinge a mesma qualidade de solução da configuração totalmente cooperativa. Quanto ao tempo de execução, o teste de Wilcoxon pareado acusou diferença significativa ao nível de 95%: o ILSN é mais rápido que o ILSBI e o ILSNMIX consome significativamente mais tempo que o ILSBI ($p < 0,001$), pois os agentes cooperativos continuam evoluindo enquanto compartilham soluções.

4.3 Diversidade

Esta seção analisa a diversidade dos agentes durante o processo de busca do *Framework* AMAM, ou seja, analisa o comportamento do *framework* quando agentes com comportamentos diferentes são instanciados em sua estrutura. Para isso, cenários com agentes diferentes foram implementados e executados computacionalmente, de modo a permitir compreender o comportamento da estrutura com agentes não clonados. Esse teste não havia sido realizado até então no *framework*. Os resultados dos experimentos computacionais realizados foram avaliados quanto aos valores de função objetivo e de tempo de execução obtidos.

Cabe distinguir os dois sentidos em que o termo *diversidade* é empregado neste trabalho. O primeiro é a *diversidade de agentes*, objeto desta seção: refere-se à instanciação, em uma mesma execução, de agentes que empregam estratégias de busca local distintas, em oposição aos cenários anteriores, nos quais todos os agentes eram cópias de um mesmo agente. O segundo é a *diversidade de soluções*, que é a dispersão do conjunto de soluções mantido no *Pool* e que é medida pela função de nicho $g(\phi_i)$ definida na Expressão (2.1) da Seção 2.1.1.3: quanto maior o valor de $g(\phi_i)$, maior a concentração de soluções na vizinhança da solução i e, portanto, menor a sua contribuição para a diversidade do conjunto. As duas noções estão ligadas, pois são os agentes que alimentam o *Pool*: agentes com comportamentos diferentes tendem a submeter soluções mais afastadas entre si.

Os momentos de inclusão e de exclusão de soluções no *Pool*, detalhados na Seção 2.1.1.3, ocorrem da seguinte forma. A *inclusão* é avaliada ao final de cada iteração de cada agente, quando este submete ao *Pool* a solução que acabou de obter. Se o *Pool* ainda não atingiu o tamanho máximo definido como parâmetro, a solução é inserida diretamente. Se o tamanho máximo já foi atingido, a solução candidata é inserida apenas se ainda não estiver presente no conjunto e se o seu valor de função objetivo for melhor que o pior valor presente no *Pool*; caso contrário, ela é descartada e o *Pool* permanece inalterado. A *exclusão* ocorre somente quando uma inserção é aceita com o *Pool* cheio, isto é, uma solução entra e outra necessariamente sai. A

solução retirada não é a de pior função objetivo, mas a de menor contribuição para a diversidade: aquela com o maior valor de $g(\phi_i)$, ou seja, a mais próxima das demais soluções do conjunto. A melhor solução do *Pool* é sempre preservada, o que torna o mecanismo elitista. O efeito combinado das duas regras é manter no *Pool* um conjunto de soluções de boa qualidade e, ao mesmo tempo, afastadas entre si, evitando que o conjunto compartilhado convirja para cópias de uma mesma solução.

4.3.1 Definição da Experimentação

Os experimentos computacionais utilizaram-se da mesma configuração de execução descrita na Seção 4.2.1.

Para validação das premissas de diversidade da presente seção, foram construídos quatro cenários de testes, utilizando os agentes definidos na Seção 3.2 e Tabela 3.1. Estes cenários são apresentados a seguir:

- (i) ILSBI: este cenário utiliza-se de agentes ILS_ALS_BI em sua execução. Desta forma, todos os agentes implementados neste cenários são iguais e apresentam comportamentos iguais;
- (ii) ILSFI: este cenário implementa agentes ILS_ALS_FI, iguais em sua execução. Porém, é importante destacar que estes agentes possuem comportamentos não determinísticos em relação a sua busca local;
- (iii) ILSRI: este cenário implementa agentes ILS_ALS_RI, iguais em sua execução. Porém, é importante destacar que estes agentes possuem comportamentos não determinísticos em relação a sua busca local;
- (iv) ILSMI: este cenário utiliza agentes diferentes. Para execuções com número diferentes de agentes, os agentes ILS_ALS_BI, ILS_ALS_FI e ILS_ALS_RI são utilizados, segundo as seguintes configurações:
 - (a) 2 agentes: um agente ILS_ALS_BI e um agente ILS_ALS_FI;
 - (b) 4 agentes: dois agentes ILS_ALS_BI, um agente ILS_ALS_FI e um agente ILS_ALS_RI;
 - (c) 8 agentes: três agentes ILS_ALS_BI, três agentes ILS_ALS_FI e dois agentes ILS_ALS_RI;
 - (d) 16 agentes: seis agentes ILS_ALS_BI, cinco agentes ILS_ALS_FI e cinco agentes ILS_ALS_RI.

Cabe ressaltar que essas quatro configurações para o cenário ILSMI foram definidas de maneira empírica, segundo os limites dos recursos computacionais disponíveis.

Esses cenários buscam verificar a eficácia da diversidade dos agentes na execução do *framework*. Os agentes implementados são minimamente diferentes, conforme descrito na Seção 3.2. O cenário ILSBI apresenta agentes com comportamentos iguais em relação à busca local, enquanto os outros 3 cenários apresentam agentes com comportamento diferentes em relação à busca local.

4.3.2 Resultados Computacionais – Diversidade

Após a execução dos testes computacionais, os resultados obtidos são apresentados nas Tabelas 4.7 e 4.8, que contém os valores da função objetivo e do tempo de execução, respectivamente.

A Tabela 4.7 mostra os valores da função objetivo e o *gap* % obtidos com a execução dos testes computacionais. Da mesma forma que nos resultados da cooperação, os melhores resultados foram obtidos nas instâncias do tipo *C*, enquanto os piores resultados foram encontrados nas instâncias do tipo *R* em ambos os cenários. Observa-se que a inclusão de agentes melhora, mesmo que minimamente, o valor do *gap*%. A análise da Tabela 4.7 revela uma tendência de melhores resultados com os cenários ILSFI e ILSMI.

A Tabela 4.8 apresenta os resultados computacionais obtidos. Observa-se que os maiores valores de tempo de execução foram obtidos nas instâncias R112 e RC108, enquanto os menores valores de tempo de execução foram obtidos na instância C208. Além disso, é notável que os cenários com maior diversidade de agentes obtiveram médias menores de tempo de execução. O cenário ILSRI apresentou a menor média computacional. Para uma visualização mais clara dos resultados computacionais, a Figura 4.13 apresenta gráficos de *boxplot* com os resultados do *gap*(%) e do tempo de execução.

A Figura 4.13 exibe *boxplots* com o *gap*(%) obtido. A partir do gráfico da Figura 4.13(a), não é possível identificar uma tendência clara nos valores obtidos. No entanto, no gráfico da Figura 4.13(b), observa-se que os valores de tempo de execução apresentam uma tendência de melhores resultados com os cenários ILSRI e ILSMI.

Com intuito de ilustrar a evolução do algoritmo, a Figura 4.14 mostra a evolução de uma execução da instância RC108 com quatro agentes para os quatro cenários apresentados. Os gráficos exibem os valores da função objetivo em relação ao tempo de execução. Com base nos gráficos apresentados, é possível observar que a execução do cenário ILSRI teve um tempo de execução menor. Além disso, os cenários ILSFI e ILSMI mostraram uma convergência mais rápida em comparação com os outros cenários nessa execução. Com base na análise apresentada, algumas tendências nos resultados são observadas. No entanto, uma análise estatística mais aprofundada é necessária para validar esses resultados. Portanto, testes estatísticos não paramétricos, conforme a metodologia descrita na Seção 3.3, foram realizados nas amostras para verificar a significância estatística dos resultados. Os resultados desses testes são apresentados nas Tabelas 4.9 e 4.10.

A Tabela 4.9 exibe os resultados dos testes estatísticos, em que o símbolo “>” indica que os valores do primeiro cenário foram maiores (piores) que os valores do segundo cenário; o símbolo “<” indica que os valores foram menores (melhores) e o símbolo “=” representa que não houve diferença significativa entre os cenários. A partir dos dados de 72 testes para cada cenário, conclui-se que:

- (i) ILSBI não foi melhor em nenhum dos testes executados e foi pior em 39 deles;
- (ii) ILSFI foi melhor em 31 das vezes e não obteve valores inferiores em nenhum;
- (iii) ILSRI foi melhor em 9 dos testes realizados e apresentou igualdade em 50 testes;

Tabela 4.7: Função Objetivo e $gap(\%)$

Cenário	Instância		Número de Agentes				Média
			2	4	8	16	
ILSBI	C109	FO	10846,36	10838,96	10828,94	10829,56	10835,95
		<i>gap</i>	0,16	0,09	0,00	0,01	0,06
	C208	FO	3597,35	3588,48	3588,32	3588,32	3590,62
		<i>gap</i>	0,25	0,00	0,00	0,00	0,06
	R112	FO	11007,69	10998,73	10989,13	10981,21	10994,19
		<i>gap</i>	10,27	10,18	10,09	10,01	10,14
	R211	FO	3843,32	3840,28	3819,28	3814,40	3829,32
		<i>gap</i>	33,18	33,08	32,35	32,18	32,70
	RC108	FO	12179,10	11912,77	11993,45	12053,51	12034,71
		<i>gap</i>	9,33	6,94	7,66	8,20	8,03
	RC208	FO	3946,91	3927,76	3894,37	3889,11	3914,54
		<i>gap</i>	3,10	2,60	1,73	1,59	2,26
	Média	<i>gap</i>	9,38	8,81	8,64	8,66	8,88
ILSFI	C109	FO	10831,38	10828,94	10828,94	10828,94	10829,55
		<i>gap</i>	0,02	0,00	0,00	0,00	0,01
	C208	FO	3588,32	3588,32	3588,32	3588,32	3588,32
		<i>gap</i>	0,00	0,00	0,00	0,00	0,00
	R112	FO	10980,85	10967,15	10964,64	10957,48	10967,53
		<i>gap</i>	10,00	9,87	9,84	9,77	9,87
	R211	FO	3808,16	3793,58	3791,03	3783,72	3794,12
		<i>gap</i>	31,97	31,46	31,37	31,12	31,48
	RC108	FO	11967,10	12018,53	11849,44	11711,86	11886,73
		<i>gap</i>	7,43	7,89	6,37	5,14	6,70
	RC208	FO	3886,20	3863,64	3862,01	3849,57	3865,36
		<i>gap</i>	1,52	0,93	0,88	0,56	0,97
	Média	<i>gap</i>	8,49	8,36	8,08	7,77	8,17
ILSRI	C109	FO	10841,70	10833,00	10828,94	10829,75	10833,35
		<i>gap</i>	0,12	0,04	0,00	0,01	0,04
	C208	FO	3594,33	3588,32	3588,32	3588,32	3589,82
		<i>gap</i>	0,17	0,00	0,00	0,00	0,04
	R112	FO	10999,10	10984,53	10975,34	10972,05	10982,76
		<i>gap</i>	10,19	10,04	9,95	9,92	10,02
	R211	FO	3824,43	3809,31	3803,04	3793,93	3807,68
		<i>gap</i>	32,53	32,01	31,79	31,47	31,95
	RC108	FO	12180,59	12030,60	11924,20	11981,32	12029,18
		<i>gap</i>	9,34	8,00	7,04	7,55	7,98
	RC208	FO	3899,39	3895,43	3877,93	3875,34	3887,02
		<i>gap</i>	1,86	1,76	1,30	1,23	1,54
	Média	<i>gap</i>	9,04	8,64	8,35	8,36	8,60
ILSMI	C109	FO	10837,70	10830,56	10829,75	10828,94	10831,74
		<i>gap</i>	0,08	0,01	0,01	0,00	0,03
	C208	FO	3593,38	3588,32	3588,32	3588,32	3589,58
		<i>gap</i>	0,14	0,00	0,00	0,00	0,04
	R112	FO	10984,40	10980,34	10971,23	10965,96	10975,48
		<i>gap</i>	10,04	10,00	9,91	9,86	9,95
	R211	FO	3817,70	3812,43	3800,36	3797,68	3807,04
		<i>gap</i>	32,30	32,11	31,70	31,60	31,93
	RC108	FO	12034,80	12156,32	11855,41	11786,24	11958,19
		<i>gap</i>	8,03	9,12	6,42	5,80	7,35
	RC208	FO	3900,92	3899,28	3868,60	3861,01	3882,45
		<i>gap</i>	1,90	1,86	1,06	0,86	1,42
	Média	<i>gap</i>	8,75	8,85	8,18	8,02	8,45

Tabela 4.8: Tempo médio de execução (s)

Cenário	Instância	Número de Agentes				Média
		2	4	8	16	
ILSBI	C109	170,33	151,41	120,34	85,14	131,80
	C208	25,41	10,62	12,47	6,54	13,76
	R112	223,43	243,63	236,08	264,88	242,01
	R211	137,89	140,18	143,98	150,45	143,12
	RC108	246,04	259,41	257,59	272,46	258,88
	RC208	122,98	127,41	131,04	139,22	130,16
	Média	154,35	155,44	150,25	153,12	153,29
ILSFI	C109	44,31	32,52	30,89	24,87	33,15
	C208	4,05	2,48	3,02	3,18	3,18
	R112	217,09	242,64	242,88	254,07	239,17
	R211	120,02	128,10	131,24	132,14	127,87
	RC108	249,10	255,23	254,58	284,38	260,82
	RC208	114,57	118,09	122,29	118,97	118,48
	Média	124,85	129,84	130,82	136,27	130,45
ILSRI	C109	52,39	42,92	29,03	33,38	39,43
	C208	3,62	3,45	3,51	3,96	3,63
	R112	108,12	118,12	121,23	125,89	118,34
	R211	61,33	70,27	67,71	74,72	68,51
	RC108	111,39	142,99	136,28	145,71	134,09
	RC208	57,56	63,78	59,86	67,50	62,18
	Média	65,74	73,59	69,61	75,19	71,03
ILSMI	C109	69,73	48,96	33,63	29,09	45,35
	C208	8,41	6,01	4,21	3,90	5,63
	R112	221,94	208,30	220,32	209,74	215,07
	R211	117,24	113,13	124,08	114,54	117,25
	RC108	227,07	228,37	229,39	238,54	230,84
	RC208	101,83	114,54	114,09	111,31	110,44
	Média	124,37	119,89	120,95	117,85	120,77

- (iv) ILSMI este cenário foi igual em 50 testes, assim como o anterior. Porém, ele foi melhor em 17 dos testes.

Com base nos testes estatísticos apresentados, concluímos que os cenários com diversidade de agentes (ILSFI, ILSRI e ILSMI) encontraram melhores resultados para as hipóteses apresentadas. Além disso, o cenário ILSFI teve o melhor desempenho, pois obteve maior número de melhores resultados e não perdeu em nenhum dos testes realizados.

Os mesmos testes estatísticos foram realizados para o tempo de execução, tendo os resultados apresentados na Tabela 4.10.

A Tabela 4.10 segue a mesma estrutura de representação quanto a simbologia de comparação de resultados adotada para a Tabela 4.9. Portanto, a partir dos 72 testes para cada cenário, extraímos que:

- (i) ILSBI não foi melhor em nenhum dos testes executados e foi pior em 56 deles;

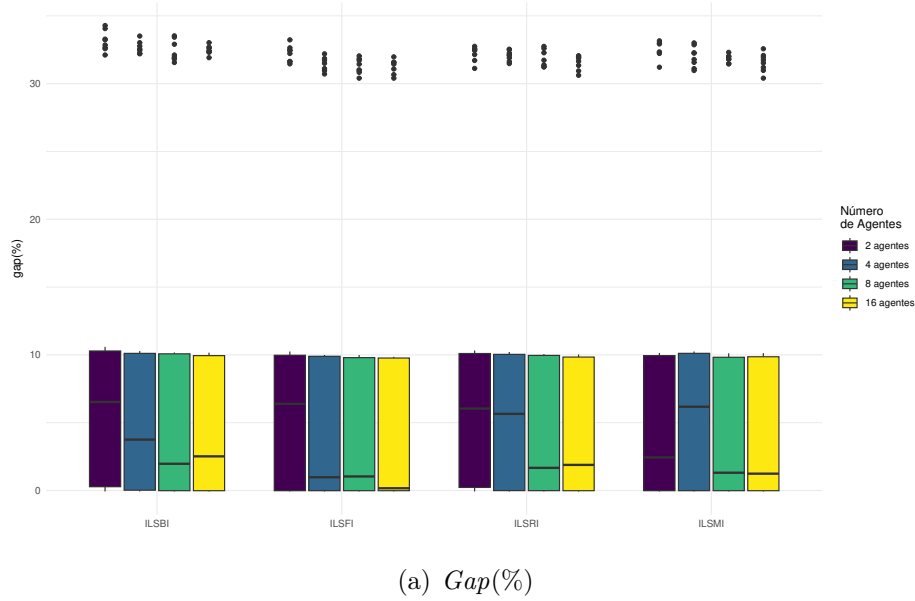
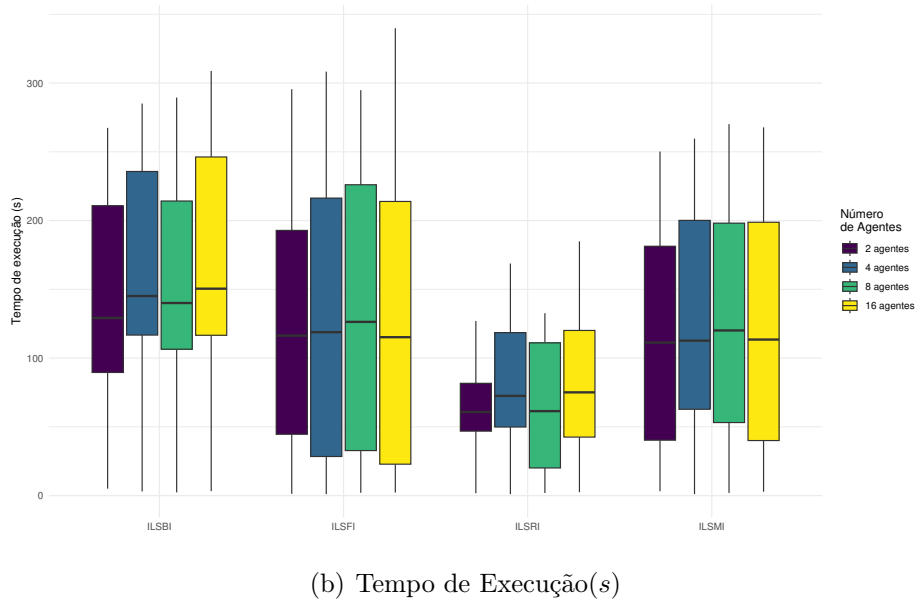


Figura 4.13: Diversidade: $Gap(\%)$ e tempo de execução.

- (ii) ILSFI foi melhor em 19 das vezes e não obteve diferença estatísticas em 30 deles;
- (iii) ILSRI foi melhor em 56 dos testes realizados e foi pior em apenas 1 dos testes;
- (iv) ILSMI foi melhor em 25 testes e apresentou igualdade em 27 deles.

Com base nos testes estatísticos, concluímos que os cenários com diversidade de agentes encontraram melhores resultados também em relação ao tempo de execução. É importante destacar que o cenário ILSRI obteve os melhores resultados em tempo de execução, com 56 testes em que foi melhor e apenas 1 teste em que foi pior. Os testes com os cenários ILSFI e ILSMI resultaram em 19 e 25 melhores resultados, respectivamente, além de 41,67% e 37,50% de empates. Refeitas as comparações de forma pareada, com o teste de Wilcoxon *signed-rank* e correção de Bonferroni, o teste

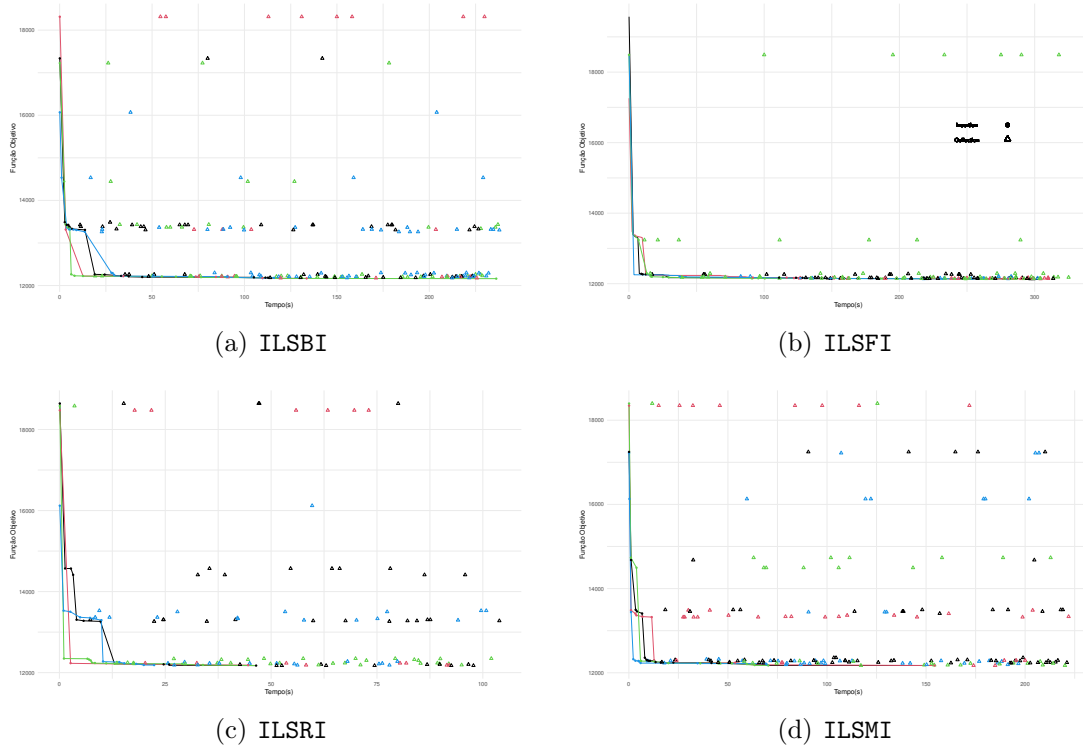


Figura 4.14: Evolução diversidade - instância RC108 com 4 agentes. Cada cor identifica um agente; ● marca o momento em que uma solução é inserida no *Pool* e △ o momento em que uma solução é coletada do *Pool* por um agente.

de Friedman confirma a diferença global entre os quatro cenários ($p < 0,001$) e cinco dos seis pares permanecem significativos quanto ao tempo de execução; a exceção é o par ILSFI e ILSMI ($p = 0,23$ após a correção), que passa a não apresentar diferença estatisticamente significativa entre si. Quanto à função objetivo, o resultado pareado reproduz exatamente a análise anterior: apenas o par ILSRI e ILSMI não apresenta diferença significativa ($p = 0,41$).

4.3.3 Considerações Parciais

A principal contribuição desta seção é avaliar o comportamento do *Framework* AMAM em relação à diversidade de agentes, demonstrando sua capacidade de melhora no desempenho do sistema multiagente à medida que a diversidade é incluída. Desta forma, esta seção busca analisar se a diversidade tem influência na qualidade dos resultados obtidos pelo *framework*. Para avaliar a diversidade do *Framework* AMAM, busca-se avaliar: (i) a existência de diferenças significativas entre as médias dos valores de função objetivo e tempo de execução obtidos a partir da inserção de agentes distintos; (ii) os resultados obtidos utilizando cenários do *framework* com agentes diferentes, com o intuito de validar a utilização de sistemas multiagente com agentes distintos na solução de problemas de otimização; (iii) comparar o tempo de evolução da estrutura à medida que a diversidade é incluída no processo de busca do *framework*. Para todos os resultados obtidos nesta análise foram utilizados testes estatísticos não paramétricos. Os testes utilizados foram os mesmos apresentados na Seção 4.2.1.

Tabela 4.9: Teste 2 a 2 - Função Objetivo

Instância	Número Agentes	ILSBI	ILSBI	ILSBI	ILSFI	ILSFI	ILSRI
		×	×	×	×	×	×
		ILSFI	ILSRI	ILSMI	ILSRI	ILSMI	ILSMI
C109	2	>	=	=	<	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
C208	2	>	=	>	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
R112	2	>	=	>	<	=	=
	4	>	>	>	<	<	=
	8	>	>	>	=	=	>
	16	>	=	>	<	<	=
R211	2	>	>	>	<	=	=
	4	>	>	>	<	<	=
	8	>	>	>	<	=	=
	16	>	>	>	=	<	=
RC108	2	>	=	>	<	=	>
	4	=	=	=	=	=	=
	8	=	=	>	=	=	=
	16	>	>	=	=	=	>
RC208	2	>	>	>	=	=	=
	4	>	>	=	<	<	=
	8	>	=	>	=	=	=
	16	>	=	>	<	=	=

Por fim, um teste estatístico foi realizado com base no *gap* das execuções, com o objetivo de verificar os resultados sem considerar as instâncias e os números de agentes diferentes. Novamente, o teste estatístico não paramétrico foi utilizado. As comparações foram realizadas par a par entre os quatro cenários. Entre todos os pares avaliados, apenas ILSRI e ILSMI não apresentaram diferença estatisticamente significativa entre si; em todos os demais pares houve diferença. Nessas comparações, o cenário ILSFI foi superior a todos os outros cenários, enquanto o cenário ILSBI foi inferior em todos os testes. Isso comprova que a diversidade de agentes é uma habilidade de suma importância para o *framework*.

Portanto, em conclusão, podemos afirmar que a diversidade de agentes leva a resultados melhores, tanto em relação à função objetivo quanto em relação ao tempo de execução, validando, assim, a hipótese que a diversidade é uma ferramenta crucial em sistemas multiagente e em sua aplicação à solução de problemas de otimização combinatória.

4.4 Modelagem de Recompensas

Maximizar a recompensa de um agente, atendendo às diversas finalidades para as quais sua inteligência é direcionada, é suficiente para expressar uma ampla variedade de objetivos contidos neste agente. [Silver et al. \(2021\)](#) discutem se as distintas formas de inteligência podem estar sujeitas à maximização de recompensas e se as várias ha-

Tabela 4.10: Teste 2 a 2 - Tempo de Execução

Instância	Número Agentes	ILSBI	ILSBI	ILSBI	ILSFI	ILSFI	ILSRI
		×	×	×	×	×	×
		ILSFI	ILSRI	ILSMI	ILSRI	ILSMI	ILSMI
C109	2	>	>	>	=	=	=
	4	>	>	>	<	<	=
	8	>	>	>	=	<	=
	16	>	>	>	=	=	=
C208	2	>	>	>	=	<	=
	4	>	>	>	=	<	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	=
R112	2	=	>	=	>	=	<
	4	=	>	>	>	>	<
	8	=	>	=	>	=	<
	16	=	>	>	>	>	<
R211	2	=	>	=	>	=	<
	4	>	>	>	>	>	<
	8	>	>	>	>	=	<
	16	>	>	>	>	>	<
RC108	2	=	>	=	>	=	<
	4	=	>	>	>	>	<
	8	=	>	=	>	=	<
	16	=	>	>	>	>	<
RC208	2	=	>	=	>	=	<
	4	>	>	>	>	=	<
	8	>	>	>	>	=	<
	16	>	>	>	>	>	<

bilidades, associadas às formas de inteligência do agente, podem surgir subjacente da busca de recompensas, através da atuação do agente em seu ambiente. [Silver et al. \(2021\)](#), além disso, discutem a importante questão de como construir um agente que maximize a recompensa e indica que a resposta pode ser alcançada através da habilidade do agente em aprender a maximizar sua recompensa, mediante sua experiência contínua de interação com o ambiente. O aprendizado do agente através de suas experiências com o ambiente é a forma mais natural de maximizar suas recompensas.

Em relação à discussão sobre aprendizado por reforço e modelagem de recompensas, [Agogino e Tumer \(2008\)](#) discutem sobre a importância de analisar a eficácia das estruturas de recompensas do agente, além de destacarem que o sucesso da política de um bom comportamento é fortemente dependente do domínio. Avaliam também que a modelagem potencial é fortemente ligada ao domínio do problema e que a modelagem de diferença, não necessariamente, requer um forte domínio do problema. Em [Devlin et al. \(2014\)](#), são discutidas e implementadas modelagens de recompensas baseadas em diferenças, em potencial e uma hibridização destas duas. [Grzes \(2017\)](#) discutem técnicas de aprendizado por reforço que podem resolver problemas de grande escala, levando a uma tomada de decisão autônoma e de alta qualidade. Porém, algumas destas técnicas de aprendizagem por reforço podem ser demoradas computacionalmente, devido à complexidade em recompensar os agentes de forma eficiente.

As análises e os resultados apresentados nesta seção constituem uma exposição do trabalho apresentado em [Silva et al. \(2022\)](#).

4.4.1 Definição da Experimentação

Os experimentos dessa seção foram realizados com 6 instâncias do PRVJT (C109, C208, R112, R211, RC108 e RC208). Para isto utilizou-se de um computador (nó de cálculo) com processador AMD Opteron(tm) Processor 6376 (16M Cache, 2.3 GHz, 32 cores), totalizando 64 núcleos de execução, com o sistema operacional Linux CentOS.

A composição do sistema multiagente usada nos experimentos envolveu agentes ILS_ALS_BI idênticos. Como a máquina na qual foram executados os experimentos possui 64 núcleos, a cada execução, 60 núcleos foram utilizados pelos agentes e os outros 4 núcleos foram deixados à disposição da JVM do JAVA e do sistema operacional. Desta forma, os testes utilizam 60 *threads* em um cenário em que cada iteração implementa 4 execuções por vez dos cenários com um, dois, quatro e oito agentes.

Em [Silva et al. \(2019\)](#), a recompensa r é determinada de forma paramétrica, pelo conhecimento específico do problema. No presente texto propõe-se modelar a recompensa r do algoritmo ALS-*QLearning*. Para isso, seis cenários de recompensa foram criados e divididos em três categorias de modelagem de recompensas: locais (2 cenários), globais (1 cenário) e diferenciais (3 cenários). Os cenários de recompensas locais são:

- (i) r_linha : recompensa que consiste em atribuir o valor da função objetivo corrente do método VND (Algoritmo 4), como forma de recompensa para o agente; e;
- (ii) r_local : recompensa que consiste em atribuir o valor da melhor função objetivo da busca local do agente, como forma de recompensa para o agente.

O cenário de recompensa global é:

- (iii) r_global : recompensa que consiste em atribuir o valor da melhor função objetivo encontrada pelo sistema, como forma de recompensa para o agente.

Por fim, os cenários baseado em recompensas por diferenças são:

- (iv) rd_lilo : recompensa que consiste em atribuir o valor da diferença entre r_linha por r_local , como forma de recompensa para o agente;
- (v) rd_ligl : recompensa que consiste em atribuir o valor da diferença entre r_linha por r_global , como forma de recompensa para o agente;
- (vi) rd_logl : recompensa que consiste em atribuir o valor da diferença entre r_local por r_global , como forma de recompensa para o agente.

As modelagens de recompensas por diferença expressam a distância entre a solução corrente do agente e a melhor solução objetivo encontrada pelo *Framework* AMAM em determinado instante de tempo.

4.4.2 Resultados Computacionais

A Tabela 4.11 apresenta o $gap(\%)$ médio das 30 execuções para as instâncias testadas nos 6 modelos de recompensas implementados, dado por:

$$gap(\%) = \frac{res_{mod} - res_{amam}}{res_{amam}} \times 100$$

em que res_{mod} corresponde ao valor obtido pela modelagem de recompensa mod e res_{amam} corresponde ao resultado obtido utilizando o algoritmo de Silva et al. (2019). A partir desta tabela, conclui-se que a modelagem de recompensa rd_lilo apresentou os menores valores médios de $gap(\%)$, enquanto a modelagem de recompensa r_gl apresentou os maiores valores de $gap(\%)$. De forma coerente, as modelagens que usam recompensa global apresentaram os piores resultados quanto ao $gap(\%)$. Por outro lado, os melhores resultados foram alcançados com a recompensa r_linha . Além disso, as execuções com oito agentes levaram aos menores valores de $gap(\%)$.

Tabela 4.11: $Gap(\%)$ médio para as instâncias testadas. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. O destaque é feito por linha, entre as quatro colunas de número de agentes. Valores negativos indicam resultados melhores que os de Silva et al. (2019), tomados como referência no cálculo do gap apresentado no início desta seção.

Recompensa	Instância	Número de Agentes				$gap(\%)$		Recompensa	Instância	Número de Agentes				$gap(\%)$
		1	2	4	8					1	2	4	8	
r_li	C109	0,31	0,21	0,11	0,09	0,18		rd_lilo	C109	0,30	0,14	0,15	0,11	0,18
	C208	0,03	-0,13	-0,04	0,00	-0,03			C208	0,05	-0,10	-0,06	0,00	-0,03
	R112	0,11	0,34	0,11	0,04	0,15			R112	0,72	0,04	0,07	0,09	0,23
	R211	-0,12	-0,11	-0,10	0,18	-0,04			R211	0,00	0,11	-0,03	0,20	0,07
	RC108	-0,04	0,03	0,87	-0,25	0,15			RC108	-0,05	0,62	0,92	-1,32	0,04
	RC208	0,05	0,43	0,09	-0,05	0,13			RC208	-0,01	-0,10	-0,29	-0,01	-0,10
	$gap(\%)$	0,06	0,13	0,17	0,00	0,09			$gap(\%)$	0,17	0,12	0,13	-0,16	0,06
r_lo	C109	0,46	0,13	0,16	0,10	0,21		rd_ligl	C109	0,31	0,15	0,11	0,06	0,15
	C208	0,04	0,05	-0,06	0,00	0,01			C208	-0,32	-0,03	-0,06	0,00	-0,10
	R112	0,14	0,00	0,05	0,06	0,06			R112	0,75	-0,05	0,03	0,08	0,20
	R211	0,11	0,01	-0,10	0,16	0,04			R211	0,09	0,22	-0,01	0,14	0,11
	RC108	-0,35	0,82	0,65	0,66	0,45			RC108	-0,08	0,60	0,37	-0,21	0,17
	RC208	-0,23	-0,08	-0,07	-0,06	-0,11			RC208	0,16	0,13	0,23	-0,03	0,12
	$gap(\%)$	0,03	0,16	0,10	0,15	0,11			$gap(\%)$	0,15	0,17	0,11	0,01	0,11
r_gl	C109	0,50	0,24	0,12	0,16	0,25		rd_logl	C109	0,41	0,07	0,07	0,07	0,16
	C208	0,06	-0,12	0,00	0,00	-0,02			C208	-0,41	0,16	-0,08	0,00	-0,08
	R112	0,16	0,00	0,06	0,08	0,08			R112	0,12	-0,04	0,07	0,03	0,04
	R211	0,23	0,05	-0,07	0,15	0,09			R211	0,08	-0,01	-0,04	0,09	0,03
	RC108	-0,27	0,79	1,74	0,07	0,58			RC108	-0,02	0,25	1,20	0,07	0,37
	RC208	0,48	0,12	0,22	0,13	0,23			RC208	0,18	0,16	0,01	0,02	0,09
	$gap(\%)$	0,19	0,18	0,34	0,10	0,20			$gap(\%)$	0,06	0,10	0,21	0,05	0,10

A Figura 4.15(a) apresenta um *boxplot* com os valores de $gap(\%)$ obtidos com as modelagens de recompensas implementadas. Observa-se que a tendência é que não existam diferenças significativas entre as modelagens. Já a Figura 4.15(b) mostra os valores de $gap(\%)$ encontrados ao se alterar o número de agentes utilizados pelo AMAM. Neste caso, claramente identifica-se a superioridade das soluções geradas com oito agentes frente aos demais cenários. Além disso, este é o único cenário em que as soluções obtidas foram melhores que as postas em Silva et al. (2019).

Com a intenção de verificar a existência ou não de diferenças significativas entre os modelos de recompensas implementados, foi realizado o teste de Friedman, conforme a metodologia estatística não paramétrica descrita na Seção 3.3. Este teste foi realizado para verificar a qualidade dos valores da função objetivo nas diferentes modelagens de

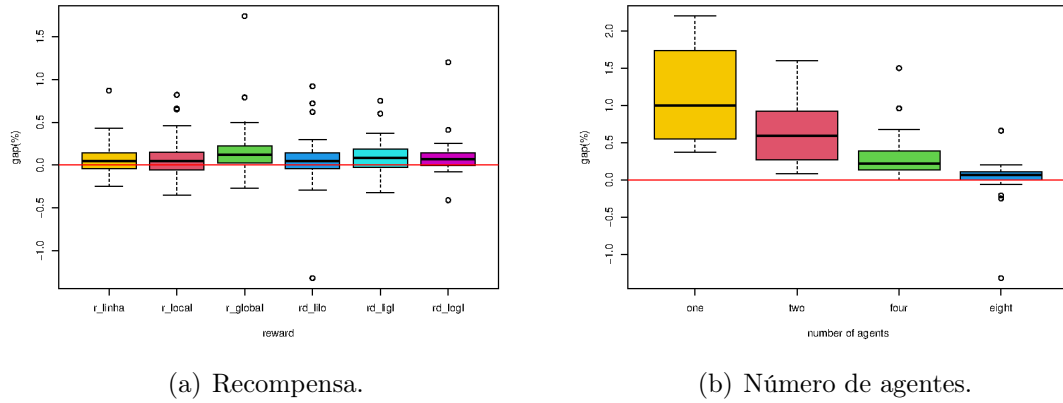


Figura 4.15: $Gap(\%)$ médio por recompensa e por número de agentes.

recompensa. O teste de Friedman apresentou $p\text{-value} = 0,070$, e o teste de Wilcoxon pareado com correção de Bonferroni não apontou diferença em nenhum dos 15 pares. Desta forma, constata-se que não há diferenças significativas entre as modelagens para um nível de significância de 5%. Assim, não se rejeita a hipótese nula de igualdade das médias. Desta forma, comprova-se que a utilização de modelagens de recompensas é eficiente e pode substituir modelagens fortemente ligadas ao problema e, portanto, a modelagem de recompensas torna a estrutura mais autônoma, uma vez que o uso das modelagens eliminam a definição da recompensa em forma de parâmetro.

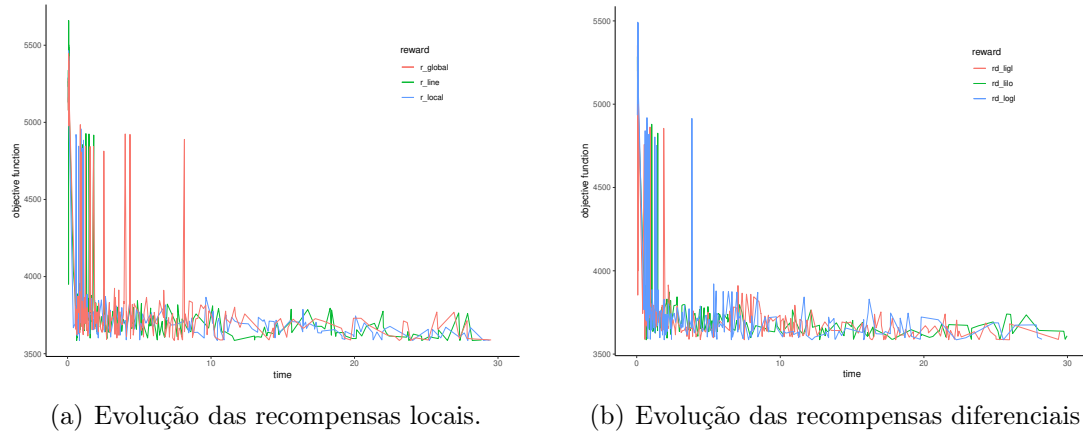


Figura 4.16: Evolução das modelagens de recompensas - instância C208.

Uma comparação do tempo de execução obtido com cada modelagem de recompensas em relação ao tempo de execução obtido com a estrutura de [Silva et al. \(2019\)](#) mostra que (i) a estrutura que obteve maior tempo de execução médio foi a estrutura com quatro agentes; e, (ii) o tempo de execução foi, em média, 16,33% maior com as modelagens de recompensas. O aumento do tempo de execução não é significativo, pois, em média, os 16,33% correspondem a 25 segundos.

A Figura 4.16 apresenta os testes obtidos com a execução da instância C208 com um agente. A Figura 4.16(a) mostra a evolução do valor da função objetivo em relação ao tempo de execução, para as modelagens r_linha , r_local e r_global . Pode-se

observar que a modelagem de recompensa r_global é menos estável e sua evolução é levemente menor. Observa-se também que a evolução da modelagem r_linha é levemente mais rápida. A Figura 4.16(b) apresenta a evolução das modelagens de recompensas rd_lilo , rd_ligl e rd_logl . Observa-se que a modelagem rd_logl se mostra menos estável e a modelagem de recompensa rd_lilo se mostra mais estável, porém, a evolução da modelagem rd_ligl aparentemente se mostra mais rápida.

4.4.3 Considerações Parciais

O principal objetivo desta seção é avaliar o comportamento do *Framework AMAM* quanto à modelagem de recompensas do seu algoritmo de aprendizagem por reforço. O propósito desta experimentação é demonstrar que a modelagem de recompensas pode facilitar o desenvolvimento destas aplicações, pois definir recompensas reais e eficientes pode requerer um conhecimento mais profundo do problema. A partir daí, testes computacionais e estatísticos foram realizados, a fim de avaliar estes modelos. Para isso, busca-se (i) analisar os dados individualmente e unificadamente, com o intuito de avaliar alguma tendência no comportamento dos dados em relação ao valor da função objetivo e ao tempo de execução; (ii) realizar testes para verificar a existência de diferenças significativas entre as médias dos modelos criados; (iii) comparar o tempo de aprendizado dos diferentes modelos de recompensas; por fim, (iv) validar as modelagens de recompensas como formas mais autônomas de aprendizagem dos agentes.

Para investigar os resultados obtidos com a aplicação das modelagens de recompensas do algoritmo *Q-learning* do agente no *Framework AMAM*, foi realizado um experimento no qual o algoritmo foi comparado com os resultados obtidos utilizando a instanciãção do AMAM apresentada em Silva et al. (2019), na qual a recompensa r do agente é definida de forma paramétrica, a partir de conhecimento prévio do problema.

Conclui-se, a partir desta análise, que (i) o melhor cenário, em relação ao número de agentes, é a utilização da estrutura com oito agentes, pois ela se mostra mais estável, obteve os menores $gap(\%)$ e foi o único cenário a melhorar as soluções de Silva et al. (2019), quando comparado às melhores soluções obtidas, independente do número de agentes utilizados; e (ii) em relação às modelagens de recompensas, indica-se a utilização das modelagens de diferenças, pois elas se mostram mais eficientes e encontraram os menores $gap(\%)$. Ainda em relação às modelagens de recompensas, indica-se a utilização da modelagem rd_lilo , pois a mesma se mostrou mais estável e obteve os menores $gap(\%)$.

Capítulo 5

Aprendizado por Reforço Multiagente no *Framework* AMAM

Este capítulo apresenta as técnicas de aprendizado por reforço desenvolvidas e incorporadas à dimensão social do *Framework* AMAM, com foco no gerenciamento cooperativo do *Pool* de soluções. As contribuições evoluem progressivamente de um cenário mono-agente para um cenário com múltiplos agentes aprendendo de forma cooperativa. O capítulo está organizado da seguinte forma: a Seção 5.1 apresenta o algoritmo *Multi-armed Bandit Pool*, que introduz o aprendizado por reforço no gerenciamento das soluções compartilhadas no *Pool* da AMAM; a Seção 5.2 estende essa abordagem para o cenário multiagente, propondo o algoritmo MAMaB-PSO, que incorpora os componentes cognitivo e social do PSO ao processo de seleção cooperativa das soluções do *Pool*.

5.1 *Multi-armed Bandit Pool*

Gerenciar a estrutura de cooperação sem uma ferramenta que beneficie soluções que possam ser mais eficientes em relação ao objetivo do sistema pode fazer com que os agentes percorram soluções indesejadas ou ineficientes para sua busca, tornando assim o sistema mais lento, menos eficiente e menos robusto. Uma alternativa para esses problemas se encontra em algoritmos de aprendizado por reforço. Técnicas de aprendizado por reforço fazem com que o sistema seja mais eficiente e eficaz na busca por soluções em um problema de otimização combinatorial.

Sutton e Barto (2018) definem o aprendizado por reforço em como um agente pode aprender as suas ações a partir de interações com ambiente, de modo a maximizar um sinal numérico de recompensa. O agente não é informado sobre quais ações realizar e, sim, deve descobrir quais são as melhores ações que vão lhe retornar melhores recompensas. As ações podem afetar não somente as ações imediatas, como também suas ações futuras. Wooldridge (2009) mostra que a cooperação pode ocorrer a partir do compartilhamento de resultados (*results sharing*).

Sistemas de cooperação possuem estruturas de armazenamento de soluções que podem ser compartilhadas com outros agentes. Os agentes devem repetidamente escolher entre k soluções/ações disponíveis nestes espaços de compartilhamento. A partir desta escolha, o agente recebe uma recompensa numérica, que está diretamente

ligada à sua escolha. O objetivo é maximizar a recompensa final do sistema. Essas ferramentas podem reduzir os passos desnecessários e ou ineficientes, acelerando, assim, a busca por soluções mais promissoras. Para isso, este trabalho propõe um algoritmo de aprendizado baseado no procedimento *Multi-armed Bandit* (Robbins, 1952), apresentado na Seção 2.3.2. Essa proposta busca minimizar os passos ineficazes do agente durante o seu processo de escolha das soluções no *Pool*, indicando soluções que obtêm melhores recompensas durante o processo de busca para os agentes.

A formulação original do *Multi-armed Bandit*, mostrada no Algoritmo 2, exige que o agente escolha entre as máquinas caça-níqueis disponíveis para apostas. A proposta do algoritmo para o problema de cooperação de soluções consiste em indicar quais são as soluções que possuem as maiores recompensas atribuídas durante o período que se encontram no *Pool* de soluções, com o intuito de obter os melhores ganhos. Dessa forma, as soluções disponíveis para a escolha são as ações disponíveis para os agentes, sendo que cada solução possuirá um valor q acumulado de ganhos. Cada ação retornará uma recompensa r , que depende das tarefas aplicadas pelo agente na dada solução. Portanto, cada solução inserida no *Pool* de soluções receberá um valor inicial q e, à medida que são coletadas no *Pool* e utilizadas, esse valor é atualizado, conforme a linha 9 do Algoritmo 2.

Esta seção introduz o algoritmo *Multi-armed Bandit Pool*, mostrado no Algoritmo 8, que possui três funções principais : (i) inserir solução no *Pool*; (ii) coletar solução no *Pool*; (iii) atualizar o valor q da solução. Como todos os algoritmos de busca e de aprendizado por reforço, este algoritmo convive com o paradoxo entre diversificação e intensificação e utiliza a função ϵ -Greedy para balancear estas escolhas. A atualização do valor q é realizada após toda coleta de solução no *Pool* de soluções. O agente coleta a solução e aplica suas técnicas heurísticas; após isso, é verificado se o agente conseguiu melhorar a sua solução corrente. Caso o agente melhore a sua solução corrente após a coleta de uma solução no *Pool*, ele atualizará o valor q da solução coletada, enviando uma bonificação por este resultado. Caso contrário, ele envia um valor de recompensa de valor 0 para atualização do valor q .

A principal contribuição desta seção é avaliar a implementação de um algoritmo de aprendizado por reforço no *Framework* AMAM em relação ao gerenciamento do compartilhamento de soluções na estrutura do *framework*. Este algoritmo é baseado no método *Multi-armed Bandit*, demonstrando sua capacidade de melhora no desempenho do sistema multiagente à medida que o algoritmo é inserido. Sendo assim, essa seção avalia a aplicação do *Multi-armed Bandit* no *Framework* AMAM. Para isso, avalia-se: (i) a existência de diferenças significativas entre as médias dos valores de função objetivo e tempo de execução obtidos a partir da implementação do MaB; (ii) os resultados obtidos nos cenários do *framework* utilizando diferentes configurações do algoritmo, com o intuito de validar a utilização de algoritmos MaB junto a meta-heurísticas na solução de problemas de otimização; (iii) comparar o tempo de evolução da estrutura à partir da introdução do algoritmo no processo de busca do *framework*. Os resultados foram avaliados a partir de testes estatísticos não paramétricos. Os testes utilizados foram os mesmos testes da cooperação apresentados na Seção 4.2.1.

5.1.1 Algoritmo *Multi-armed Bandit Pool*

Esta seção apresenta o algoritmo *Multi-armed Bandit Pool* proposto para a realização do procedimento de aprendizagem por reforço no *framework* AMAM. Inicialmente, são apresentadas as estruturas de coleta de solução e de exclusão de solução. Em seguida, o algoritmo completo é descrito.

5.1.2 Alterações na coleta de soluções

Para coletar uma solução no *Pool* e para que essa escolha não fosse realizada de forma aleatória, definem-se dois novos caminhos de escolha de soluções. As duas formas implementadas são **qmax** e **softmax**. Essas funções de escolha são baseadas no algoritmo de *Q-learning* e são definidas como:

- (a) **qmax**: consiste em escolher o argumento máximo em relação ao valor de q ;
- (b) **softmax**: consiste em escolher aleatoriamente entre os $x\%$ melhores valores de q dentre o conjunto de soluções, conforme a expressão:

$$V \leftarrow \{s \in C(s) \mid q > q_{min} + x(q_{max} - q_{min})\} \quad (5.1)$$

em que q_{min} e q_{max} representam, respectivamente, o menor e o maior valor de q das soluções contidas no conjunto de soluções $C()$. A partir daí, escolhe-se aleatoriamente uma solução contida no vetor V .

Cabe um esclarecimento quanto à nomenclatura. O nome **softmax** é usado aqui apenas por analogia e não corresponde à função *softmax* da literatura de aprendizado por reforço, descrita por Sutton e Barto (2018), em que a probabilidade de selecionar uma ação é proporcional a $e^{q(a)/\tau} / \sum_b e^{q(b)/\tau}$, com o parâmetro de temperatura τ controlando o grau de aleatoriedade. Na Expressão (5.1) não há distribuição de Boltzmann nem parâmetro de temperatura: define-se um limiar a partir da amplitude $q_{max} - q_{min}$ e sorteia-se uniformemente entre as soluções que o superam. Trata-se, portanto, de uma seleção aleatória, restrita a uma faixa dos melhores valores de q , e não de uma seleção proporcional ao valor de q . A diferença prática é que, na formulação adotada, todas as soluções acima do limiar têm a mesma chance de serem escolhidas, independentemente de quão superiores sejam entre si, ao passo que, na função *softmax* definida em Sutton e Barto (2018), as soluções com maior valor de q seriam proporcionalmente mais prováveis. O nome foi mantido por já estar consolidado nos cenários experimentais SOFTSHI e SOFTQ, definidos na Seção 5.1.5 e usados em todas as tabelas e figuras deste capítulo.

Cabe esclarecer também como essas duas formas de escolha se relacionam com a política ϵ -greedy, apresentada na Seção 2.3.2. Assim, política ϵ -greedy continua sendo utilizada e não é substituída pela função **qmax** nem pela função **softmax**. A política ϵ -greedy mantém os seus dois ramos: com probabilidade ϵ , a solução é coletada aleatoriamente do *Pool*, preservando a diversificação; com probabilidade $1 - \epsilon$, a coleta é gulosa. O que as duas formas alteram é apenas o que se entende por *coleta gulosa*. Na função **qmax**, o ramo guloso retorna o argumento máximo de q , como na formulação clássica. Na função **softmax**, esse mesmo ramo passa a executar a Expressão (5.1), sorteando uniformemente entre as soluções cujo valor de q supera o limiar, em vez de

retornar sempre a de maior q . Assim, **qmax** e **softmax** não são alternativas à política ϵ -greedy, e sim duas implementações do seu ramo de intensificação. A motivação para o **softmax** é que o ramo guloso determinístico tende a devolver repetidamente a mesma solução do *Pool* enquanto o seu valor de q permanecer o maior, o que concentra a busca dos agentes em uma única região; o sorteio dentro da faixa superior mantém a preferência pelas melhores soluções e, ao mesmo tempo, evita essa concentração.

5.1.3 Alterações na exclusão de soluções

Em relação à escolha das soluções a serem excluídas do *Pool* de soluções, o algoritmo avalia duas formas de seleção. A primeira, denominada SHI, consiste em retirar a solução com o maior valor de similaridade estrutural $g(\phi_i)$, definido pela Expressão (2.1). A função $g(\phi_i)$ soma as proximidades da solução i às demais soluções do *Pool*, de modo que um valor alto indica uma solução cercada de soluções parecidas. Retirar a de maior $g(\phi_i)$ é, portanto, retirar a que menos contribui para a diversidade do conjunto. No entanto, a segunda forma de seleção, denominada Q, consiste em retirar a solução que possui o menor valor q atribuído à solução. Assim, a seleção é realizada tanto em relação à diversidade quanto ao valor de contribuição durante o processo de cooperação. Se houver empate em relação à medida principal, o algoritmo avalia o valor da medida secundária em cada caso.

5.1.4 Funcionamento do *Multi-armed Bandit Pool*

Em linhas gerais, o algoritmo trata cada solução armazenada no *Pool* como um braço de um problema *Multi-armed Bandit*. Toda solução inserida recebe um valor q inicial, que mede sua contribuição estimada para o processo de busca coletivo. Quando um agente solicita uma solução, a política de coleta escolhe um braço segundo o valor q , pela regra **qmax**, que toma o maior valor, ou pela regra **softmax**, que sorteia entre os melhores. O agente aplica busca local sobre a solução coletada e devolve uma recompensa binária, igual a 1 se sua solução corrente melhorou, e 0, caso contrário, com a qual o valor q daquele braço é atualizado de forma incremental. Como o conjunto de braços muda ao longo da execução, a atualização emprega taxa de aprendizado constante, apropriada a ambientes não estacionários. Quando o *Pool* atinge seu tamanho máximo, uma solução precisa ser descartada para que a nova entre, e esse descarte segue o critério de menor diversidade ou o de menor valor q .

O Algoritmo 8, denominado *Multi-armed Bandit Pool*, apresenta a implementação proposta do método *Multi-armed Bandit* para aprendizado em sistemas multiagente. As linhas 2-17 apresentam a função de inserção das soluções no *Pool*. Pode-se observar, nas linhas 9 e 11, que a exclusão das soluções do *Pool* ocorre pela avaliação da diversidade. Esta abordagem pode ser alterada e passa-se a avaliar o valor q de representatividade. As linhas 19-21 e 23-29 mostram, respectivamente, as estratégias **qmax** e **softmax** de escolha da solução. As linhas 31-33 mostram a função de atualização do valor q de cada solução. A atualização do valor q é realizada pela linha 32 do Algoritmo 8, devido a não estacionariedade do problema tratado.

Algoritmo 8 *Multi-armed Bandit Pool*

```

1:  $P$ ; ▷  $Pool$  de soluções
2: função InserirNoPool( $s$ )
3:   se  $|P| < TamanhoMax$  então
4:      $q(s) \leftarrow ValorInicial$ ;
5:      $P \leftarrow P \cup \{s\}$ ;
6:   senão
7:     se  $f(s) < \max\{f(s') \mid s' \in P\}$  então
8:       para  $i \leftarrow 1$  até  $|P|$  faça
9:          $sh(s_i) \leftarrow g(\phi_i)$ ;
10:      fim para
11:       $pior \leftarrow \arg \max\{sh(s') \mid s' \in P\}$ ;
12:       $P \leftarrow P \setminus \{pior\}$ ;
13:       $q(s) \leftarrow ValorInicial$ ;
14:       $P \leftarrow P \cup \{s\}$ ;
15:   fim se
16: fim se
17: fim função
18:
19: função ColetarQMax
20:   retorne  $\arg \max\{q(s') \mid s' \in P\}$ ;
21: fim função
22:
23: função ColetarSoftMax
24:    $q_{\min} \leftarrow \min\{q(s') \mid s' \in P\}$ ;
25:    $q_{\max} \leftarrow \max\{q(s') \mid s' \in P\}$ ;
26:    $V \leftarrow \{s' \in P \mid q(s') > q_{\min} + \beta \times (q_{\max} - q_{\min})\}$ ;
27:   Selecione aleatoriamente  $s \in V$ ;
28:   retorne  $s$ ;
29: fim função
30:
31: função AtualizarValorQ( $s$ )
32:    $q(s) \leftarrow q(s) + \alpha [R - q(s)]$ ;
33: fim função

```

5.1.5 Definição da Experimentação

A configuração de execução adotada nestes experimentos, incluindo instâncias utilizadas, *hardware*, isolamento entre *threads* JAVA e *threads* físicas e estratégia de distribuição dos núcleos para os diferentes números de agentes, é a mesma descrita na Seção 4.2.1 do Capítulo 4, à qual se remete o leitor.

Para melhor calibrar a execução do algoritmo proposto, foram testados parâmetros e verificadas as suas influências nos resultados obtidos. Os parâmetros testados foram: escolha da solução (**qmax** ou **softmax**) e forma de exclusão da solução (**shi** ou **q**). Os testes foram realizados utilizando o cenário ILSMI usado na experimentação da Seção 4.3, uma vez que este cenário apresenta uma maior diversidade entre os agentes. Assim, foram propostos os cenários MAXSHI, MAXQ, SOFTSHI e SOFTQ, descritos a seguir:

- (i) **MAXSHI**: cenário que utiliza o método **qmax** como forma de escolha da solução cooperada e o método **shi** como forma de exclusão de uma solução do *Pool*;
- (ii) **MAXQ**: cenário que utiliza o método **qmax** como forma de escolha da solução cooperada e o método **q** como forma de exclusão de uma solução do *Pool*;
- (iii) **SOFTSHI**: cenário que utiliza o método **softmax** como forma de escolha da solução cooperada e o método **shi** como forma de exclusão de uma solução do *Pool*;
- (iv) **SOFTQ**: cenário que utiliza o método **softmax** como forma de escolha da solução cooperada e o método **q** como forma de exclusão de uma solução do *Pool*.

Os resultados obtidos com a aplicação do Algoritmo *Multi-armed Bandit Pool*, quando incorporado à dimensão *cooperação* do *framework* **AMAM**, são apresentados a seguir.

5.1.6 Resultados Computacionais

As Tabelas 5.1 e 5.2 apresentam, respectivamente, os valores de função objetivo e tempo de execução obtidos com a aplicação do Algoritmo *Multi-armed Bandit Pool*. Os resultados para cada um dos quatro cenários avaliados estão incluídos nessas tabelas.

A Tabela 5.1 apresenta os valores de função objetivo e *gap*(%) encontrados com a execução dos cenários descritos. Os resultados mostram que os melhores resultados foram encontrados para o grupo de instâncias C e os piores valores foram encontrados para o grupo de instâncias R. Portanto, não é possível concluir nenhuma tendência nos resultados quanto a estes cenários. O valor de *gap*(%) geral foi mantido em relação ao valor obtido no cenário **ILSMI** nos experimentos computacionais da Seção 4.3.

Na Tabela 5.2 são mostrados os valores de tempo de execução obtidos. Pode-se observar que os menores valores de tempo de execução foram encontrados nas instâncias do grupo C. Já os cenários **MAXQ** e **MAXSHI** obtiveram um tempo médio menor que os outros dois cenários.

Para avaliar se há diferenças significativas para os valores de função objetivo entre os cenários, foram realizados testes estatísticos não paramétricos e os resultados desses testes estão apresentados na Tabela 5.3. O símbolo “>” indica que os valores do primeiro cenário foram maiores (piores) que os resultados do segundo cenário. Já o símbolo “<” simboliza que os valores foram menores (melhores) que o segundo cenário. Já o símbolo “=” representa que os valores dos cenários não apresentaram diferença significativa entre eles. Dentre o total de 72 testes para cada cenário, conclui-se que:

- (a) **MAXSHI** não foi melhor em nenhum dos testes executados e foi pior em apenas 3 deles;
- (b) **MAXQ** não foi melhor em nenhum dos testes executados e foi pior em apenas 4 deles;
- (c) **SOFTSHI** foi melhor e pior em 3 testes e empatou nos outros 66 testes realizados;
- (d) **SOFTQ** este cenário foi melhor em 7 testes e não apresentou nenhum pior resultado.

Tabela 5.1: Função Objetivo e $gap(\%)$ - *Multi-armed Bandit*

Cenário	Instância		2	4	8	16	Média
MAXSHI	C109	FO	10834,44	10830,81	10828,19	10827,40	10830,21
		gap	0,00	0,00	0,00	0,00	0,00
	C208	FO	3589,84	3586,00	3585,89	3586,01	3586,94
		gap	0,00	0,00	0,00	0,00	0,00
	R112	FO	10985,48	10978,56	10972,45	10971,45	10976,99
		gap	0,10	0,10	0,10	0,10	0,10
	R211	FO	3816,66	3812,76	3795,90	3799,69	3806,25
		gap	0,32	0,32	0,32	0,32	0,32
	RC108	FO	12104,37	12093,64	11987,41	11820,00	12001,36
		gap	0,09	0,09	0,08	0,06	0,08
	RC208	FO	3910,85	3889,88	3875,77	3870,59	3886,77
		gap	0,02	0,02	0,01	0,01	0,02
	Média	gap	0,09	0,09	0,09	0,08	0,09
MAXQ	C109	FO	10836,62	10830,78	10828,26	10827,32	10830,75
		gap	0,00	0,00	0,00	0,00	0,00
	C208	FO	3590,41	3586,11	3585,86	3585,82	3587,05
		gap	0,00	0,00	0,00	0,00	0,00
	R112	FO	10991,75	10979,30	10974,71	10967,19	10978,24
		gap	0,10	0,10	0,10	0,10	0,10
	R211	FO	3817,23	3814,72	3801,55	3797,65	3807,79
		gap	0,32	0,32	0,32	0,32	0,32
	RC108	FO	12168,58	12057,26	12050,01	11917,01	12048,22
		gap	0,09	0,08	0,08	0,07	0,08
	RC208	FO	3926,71	3895,12	3875,43	3862,56	3889,96
		gap	0,03	0,02	0,01	0,01	0,02
	Média	gap	0,09	0,09	0,09	0,08	0,09
SOFTSHI	C109	FO	10834,41	10830,73	10827,49	10827,30	10829,98
		gap	0,00	0,00	0,00	0,00	0,00
	C208	FO	3585,91	3587,67	3585,91	3585,88	3586,34
		gap	0,00	0,00	0,00	0,00	0,00
	R112	FO	10989,39	10980,77	10971,77	10966,22	10977,04
		gap	0,10	0,10	0,10	0,10	0,10
	R211	FO	3816,81	3814,02	3794,56	3788,17	3803,39
		gap	0,32	0,32	0,31	0,31	0,32
	RC108	FO	12097,32	12083,84	11917,73	11853,93	11988,21
		gap	0,09	0,08	0,07	0,06	0,08
	RC208	FO	3911,37	3899,02	3872,26	3867,08	3887,43
		gap	0,02	0,02	0,01	0,01	0,02
	Média	gap	0,09	0,09	0,08	0,08	0,08
SOFTQ	C109	FO	10834,31	10829,93	10827,37	10827,33	10829,74
		gap	0,00	0,00	0,00	0,00	0,00
	C208	FO	3589,93	3585,85	3585,92	3585,90	3586,90
		gap	0,00	0,00	0,00	0,00	0,00
	R112	FO	10989,33	10982,57	10970,07	10965,33	10976,83
		gap	0,10	0,10	0,10	0,10	0,10
	R211	FO	3820,01	3808,76	3796,24	3791,42	3804,11
		gap	0,32	0,32	0,32	0,31	0,32
	RC108	FO	12066,54	12091,17	11881,81	11653,53	11923,26
		gap	0,08	0,09	0,07	0,05	0,07
	RC208	FO	3904,75	3888,07	3869,61	3862,18	3881,15
		gap	0,02	0,02	0,01	0,01	0,02
	Média	gap	0,09	0,09	0,08	0,08	0,08

Tabela 5.2: Tempo de execução *Multi-armed Bandit Pool*

Cenário	Instância	2	4	8	16	Média
MAXQ	C109	67,51	47,44	46,91	29,62	47,87
	C208	5,35	4,06	3,78	4,40	4,40
	R112	197,11	186,48	203,34	175,07	190,50
	R211	108,50	110,57	101,86	102,83	105,94
	RC108	211,16	205,62	203,95	190,40	202,78
	RC208	97,95	98,21	94,35	90,16	95,17
	Média	114,60	108,73	109,03	98,75	107,78
MAXSHI	C109	82,78	47,26	38,06	31,78	49,97
	C208	5,20	4,66	4,25	3,91	4,50
	R112	191,31	189,11	175,48	175,73	182,91
	R211	113,74	112,73	102,67	100,48	107,41
	RC108	221,70	196,78	223,43	189,00	207,73
	RC208	88,04	98,20	89,40	96,22	92,96
	Média	117,13	108,12	105,55	99,52	107,58
SOFTQ	C109	87,77	68,12	41,77	37,34	58,75
	C208	7,70	4,95	3,73	3,90	5,07
	R112	211,73	194,37	213,41	195,08	203,65
	R211	109,33	124,01	125,64	118,94	119,48
	RC108	244,47	229,45	243,83	235,85	238,40
	RC208	107,22	111,09	115,12	105,54	109,74
	Média	128,04	122,00	123,92	116,11	122,51
SOFTSHI	C109	54,38	40,49	34,48	29,55	39,73
	C208	6,39	5,00	3,46	3,42	4,57
	R112	210,15	204,44	212,07	202,63	207,33
	R211	107,90	122,55	125,04	113,20	117,17
	RC108	215,31	223,45	228,38	223,68	222,70
	RC208	114,96	112,32	114,87	107,62	112,44
	Média	118,18	118,04	119,72	113,35	117,32

Desta forma, os melhores resultados foram obtidos utilizando o cenário **SOFTQ**, uma vez que obteve 7 melhores resultados e não perdeu nenhum teste realizado. Além disso, deve-se observar que os métodos de escolha **SOFT** obtiveram melhores resultados.

Após a realização dos testes em relação ao valor de função objetivo, avalia-se os valores de tempo de execução. A Tabela 5.4 mostra os resultados dos testes estatísticos. O símbolo “>” indica que os valores do primeiro cenário foram maiores (piores) que os resultados do segundo cenário. O símbolo “<” indica que os valores foram menores (melhores) que o segundo cenário. Já o símbolo “=” representa que os valores dos cenários não apresentaram diferença significativa entre eles. Dentre os 72 testes realizados para cada cenário, extraímos que:

- (a) **MAXSHI** foi melhor em 18 dos testes executados e pior em apenas um deles;
- (b) **MAXQ** foi melhor em 15 das vezes e foi pior em apenas dois testes executados;

Tabela 5.3: Ajuste de parâmetros *Multi-armed Bandit Pool* - Função objetivo

Instância	Número Agentes	MAXSHI	MAXSHI	MAXSHI	MAXQ	MAXQ	SOFTSHI
		×	×	×	×	×	×
		MAXQ	SOFTSHI	SOFTQ	SOFTSHI	SOFTQ	SOFTQ
C109	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	>
	16	=	=	=	=	=	=
C208	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
R112	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	>	=	=	=	=
R211	2	=	=	=	=	=	=
	4	=	=	=	=	=	>
	8	=	=	=	>	=	=
	16	=	>	=	=	>	=
RC108	2	=	=	=	=	>	=
	4	=	=	=	=	=	=
	8	=	=	=	=	>	=
	16	=	=	=	=	=	=
RC208	2	=	=	=	=	=	=
	4	=	=	=	=	=	>
	8	=	=	=	=	=	=
	16	=	=	>	=	=	=

- (c) **SOFTSHI** foi melhor em apenas um teste, pior em 16 testes e não houve diferença em 55 testes;
- (d) **SOFTQ** foi melhor em apenas 2 testes, pior em 17 testes e não houve diferença em 53 testes.

Os resultados mostram que os melhores valores de tempo de execução foram obtidos utilizando os cenários de escolha **MAX**, sendo que os melhores resultados foram obtidos utilizando o cenário **MAXSHI**. Estes resultados não seguiram a tendência dos resultados de valor de função objetivo, já que os melhores resultados foram obtidos utilizando métodos de escolha **SOFT**.

Adicionalmente, um teste estatístico não paramétrico, conforme descrito na seção 3.3, voltado para a análise dos valores de *gap* foi realizando. Este teste busca retirar os efeitos gerados pelas instâncias e pelo número de agentes, avaliando-se diretamente os cenários. Os resultados obtidos mostram que: (i) os métodos **SOFT** têm diferença significativa em relação ao cenário **MAXQ**; e (ii) não foram encontradas diferenças significativas nos demais cenários. O mesmo teste foi realizado em relação ao tempo de execução e os resultados corroboram com os resultados dos testes 2 a 2. Assim, os resultados indicam que os métodos **MAX** obtiveram melhores resultados em relação aos métodos **SOFT**, pois os resultados dos testes estatísticos mostraram diferença significativa entre estes métodos.

A partir dos resultados mostrados, foram selecionados os dois melhores métodos para compararmos com os cenários em que o *Pool* de soluções não apresentam

Tabela 5.4: Ajuste de parâmetros *Multi-armed Bandit Pool* - Tempo de Execução

Instância	Número Agentes	MAXSHI	MAXSHI	MAXSHI	MAXQ	MAXQ	SOFTSHI
		× MAXQ	× SOFTSHI	× SOFTQ	× SOFTSHI	× SOFTQ	× SOFTQ
C109	2	=	=	=	=	=	=
	4	=	=	=	=	=	>
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
C208	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	>	=	=	=
R112	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	<	<	<	<	=
	16	=	=	=	=	<	=
R211	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	<	<	<	<	<	=
	16	=	=	=	<	<	=
RC108	2	=	<	=	=	=	=
	4	=	<	<	<	<	=
	8	<	<	<	<	=	=
	16	=	<	<	<	<	=
RC208	2	=	=	=	=	=	=
	4	=	=	<	=	<	=
	8	=	<	<	<	<	<
	16	=	<	<	=	=	=

um algoritmo de aprendizado. Desta forma, a análise baseia-se nos cenários ILSBI, ILSMI, MAXSHI e SOFTQ. Foram escolhidos os cenários MAXSHI e SOFTQ, devidos aos seus resultados em relação ao tempo de execução e ao valor de função objetivo, respectivamente. Para a visualização dos resultados obtidos, dois gráficos *boxplot* são apresentados na Figura 5.1.

A Figura 5.1(a) mostra um gráfico com valores de *gap*(%). A partir do gráfico não é possível concluir quais foram os melhores resultados obtidos. A Figura 5.1(b) mostra o gráfico dos resultados para o tempo de execução. Pode-se observar uma tendência de melhora no cenário MAXSHI em relação aos outros cenários. Porém, para verificar e validar os resultados, é necessário a execução de testes estatísticos. Para avaliar o comportamento da evolução dos cenários, a Figura 5.2 mostra gráficos com a evolução do valor de função objetivo em relação ao tempo de execução.

A Figura 5.2 apresenta os resultados de uma execução com quatro agentes dos cenários ILSBI, ILSMI, MAXSHI e SOFTQ para a solução da instância R112. Pode-se observar a convergência mais rápida dos métodos com aprendizado, além do tempo de execução final ser menor. Pode-se observar um maior direcionamento das soluções do *Pool* no cenário SOFTQ. Assim, é possível verificar uma melhor convergência dos cenários com aprendizado e um ganho em relação ao tempo de execução.

Os testes estatísticos foram realizados tomando os cenários dois a dois, de forma pareada, conforme a metodologia da Seção 3.3: cada combinação de instância e número de agentes constitui um bloco, o teste de Friedman avalia a diferença global e o teste de Wilcoxon *signed-rank*, com correção de Bonferroni, avalia cada par. Quanto

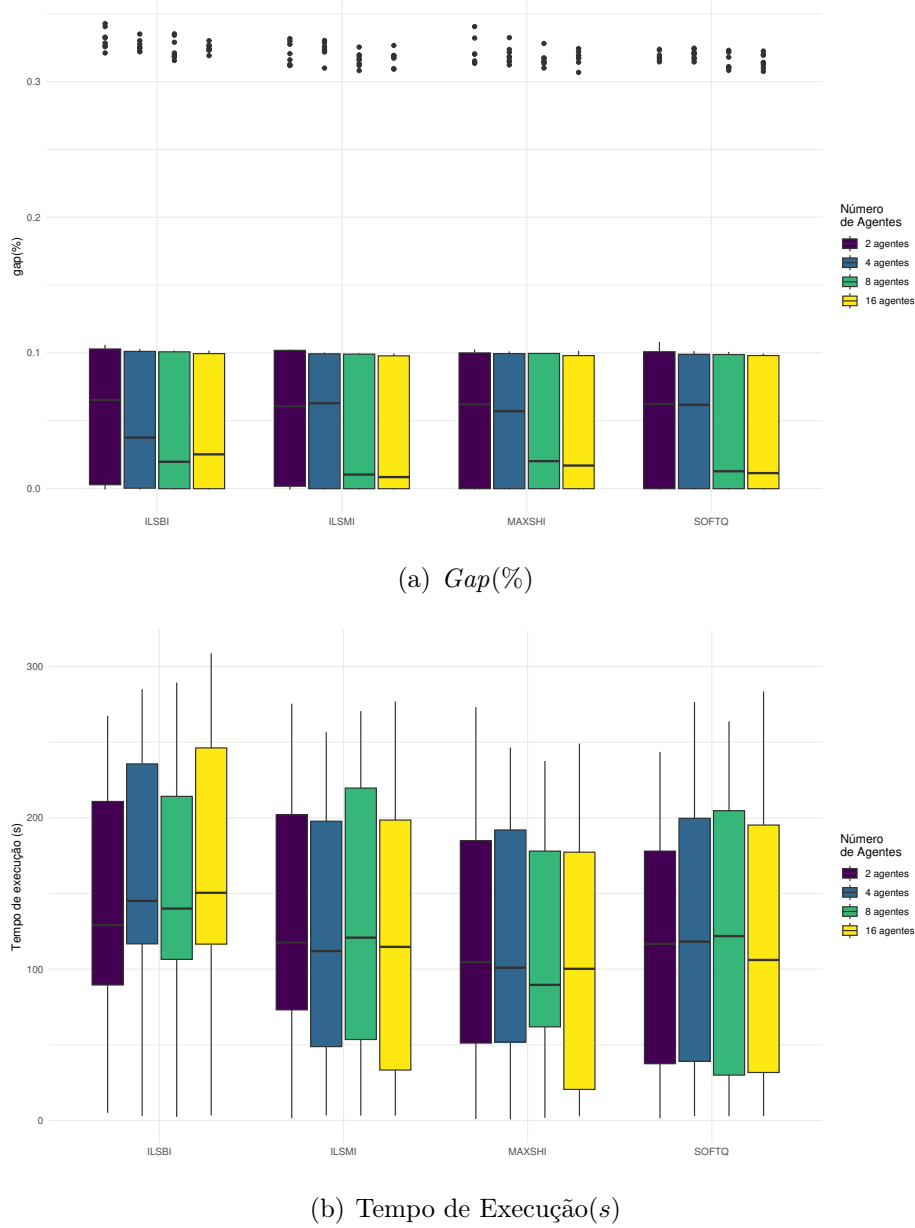
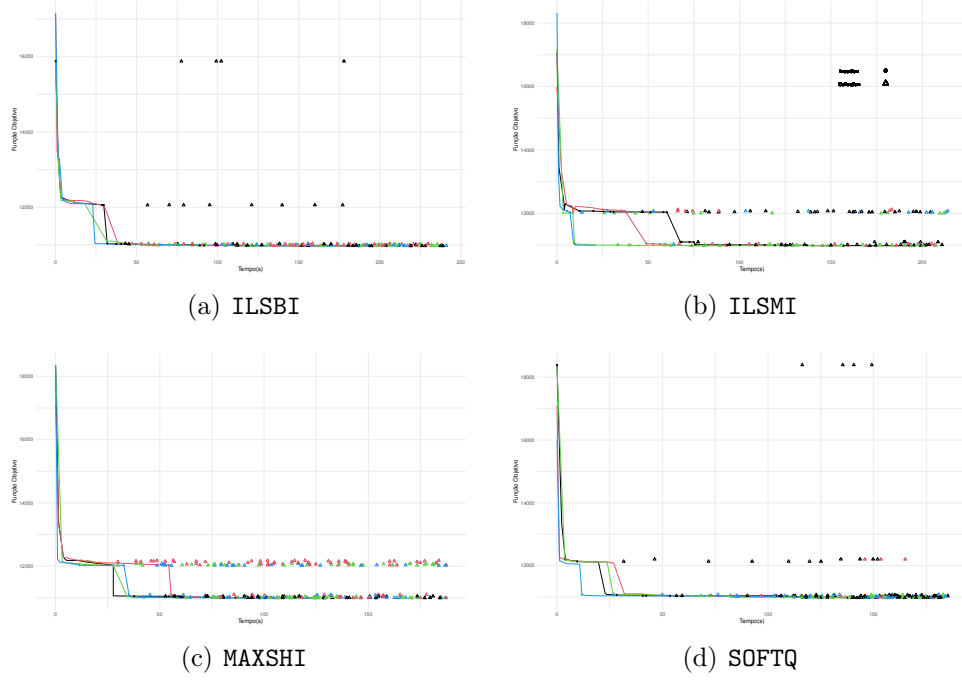


Figura 5.1: Gap(%) e Tempo de Execução - *Multi-armed Bandit Pool*

à função objetivo, o teste de Friedman acusa diferença global ($p = 0,003$), mas, após a correção de Bonferroni, nenhum dos 10 pares apresenta diferença estatisticamente significativa, de modo que os quatro cenários do *Multi-armed Bandit Pool* devem ser considerados equivalentes em qualidade de solução. Quanto ao tempo de execução, o teste de Friedman também acusa diferença ($p < 0,001$) e, nesse caso, 6 dos 10 pares permanecem significativos após a correção. Repetindo a análise por bloco de instâncias, a equivalência em função objetivo se mantém para os blocos C ($p = 0,127$), R (0,393) e RC (0,024, sem nenhum par significativo após a correção), e a diferença em tempo é detectada nos três blocos, ou seja, C ($p = 0,050$), R (0,0022) e RC (0,00086). A Tabela 5.5 exibe os resultados dos testes estatísticos em relação aos valores de função objetivo, em que o símbolo “>” mostra que os valores do primeiro cenário foram maiores (piores) que os resultados do segundo cenário; o símbolo < mostra que

Figura 5.2: Evolução *Multi-armed Bandit Pool* - instância R112 com 4 agentesTabela 5.5: *Framework AMAM* com *Multi-armed Bandit Pool* - Função objetivo

Instância	Número Agentes	ILSBI	ILSBI	ILSBI	ILSMI	ILSMI	MAXSHI
		×	×	×	×	×	×
		ILSMI	MAXSHI	SOFTQ	MAXSHI	SOFTQ	SOFTQ
C109	2	=	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
C208	2	>	=	=	=	=	=
	4	=	=	=	=	=	=
	8	=	=	=	=	=	=
	16	=	=	=	=	=	=
R112	2	>	>	=	=	=	=
	4	>	>	=	=	=	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	=
R211	2	>	>	>	=	=	=
	4	>	>	>	=	=	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	=
RC108	2	>	=	>	=	=	=
	4	=	=	=	=	=	=
	8	>	=	>	=	=	=
	16	=	=	>	=	=	=
RC208	2	>	>	>	=	=	=
	4	=	>	>	=	>	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	>

os valores foram menores (melhores) que o segundo cenário; e o símbolo “=” mostra que os valores dos cenários não apresentaram diferença significativa entre eles. Dos 72 testes para cada cenário, conclui-se que:

- (a) ILSBI: este cenário foi pior em 39 dos testes e igual em 33 deles;
- (b) ILSMI: este cenário obteve melhores resultados em 14 teste e foi pior em apenas 1 teste;
- (c) MAXSHI: este cenário foi pior em apenas um teste e melhor 12 deles;
- (d) SOFTQ: este cenário foi melhor em 15 dos testes e não foi pior em nenhum deles.

O cenário que obteve os melhores resultados foi o cenário **SOFTQ**, sendo que o cenário **ILSBI** não conseguiu ser melhor em nenhum dos testes. Os cenários **ILSMI** e **MAXSHI** também encontraram bons resultados. A seguir são apresentados os testes estatísticos realizados para avaliar os valores de tempos de execução.

Tabela 5.6: *Framework AMAM com Multi-armed Bandit Pool* - Tempo de Execução

Instância	Número Agentes	ILSBI	ILSBI	ILSBI	ILSMI	ILSMI	MAXSHI
		×	×	×	×	×	×
		ILSMI	MAXSHI	SOFTQ	MAXSHI	SOFTQ	SOFTQ
C109	2	>	>	>	=	=	=
	4	>	>	>	=	=	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	=
C208	2	>	>	>	=	=	=
	4	>	>	>	=	=	=
	8	>	>	>	=	=	=
	16	>	>	>	=	=	>
R112	2	=	=	=	=	=	=
	4	>	>	>	>	=	=
	8	=	>	=	=	=	<
	16	>	>	>	>	=	=
R211	2	=	>	>	=	=	=
	4	>	>	>	=	=	=
	8	>	>	>	=	=	<
	16	>	>	>	>	=	=
RC108	2	=	>	=	=	=	=
	4	>	>	>	=	=	<
	8	=	>	>	=	=	<
	16	>	>	=	>	=	<
RC208	2	=	>	=	=	=	=
	4	>	>	>	>	=	<
	8	>	>	>	=	=	<
	16	>	>	>	>	=	<

A Tabela 5.6 mostra os resultados dos testes estatísticos em relação ao tempo de execução, em que > simboliza que os valores do primeiro cenário foram maiores

(piores) que os resultados do segundo cenário. Já $<$ simboliza que os valores foram menores (melhores) que o segundo cenário e $=$ representa que os valores dos cenários não apresentaram diferença significativa entre eles. Conclui-se, então, que:

- (a) ILSBI: este cenário foi pior em 60 dos testes executados e igual em 12 deles;
- (b) ILSMI: conseguiu ser melhor em 18 testes e pior em apenas 6 testes;
- (c) MAXSHI: conseguiu ser melhor em 37 testes e foi pior apenas 1 vez;
- (d) SOFTQ: este cenário foi melhor em 20 testes e pior 8 vezes.

Podemos observar que os resultados com o cenário ILSBI foi pior que todos os cenários. Os cenários com aprendizado apresentaram os melhores resultados, sendo o cenário MAXSHI o mais eficiente deles, pois alcançou 37 melhores resultados. Desta forma, validou-se que o aprendizado melhora a evolução e torna o *framework* mais robusto, tornando atrativo o uso deste tipo de técnica.

Testes estatísticos não paramétricos foram realizados para avaliar os valores de *gap*(%). Estes testes desconsideram o efeito das instâncias e do número de agentes, verificando apenas o efeito da aplicação dos métodos. Os testes estatísticos mostram que todos os cenários foram melhores que o cenário ILSBI. O cenário SOFTQ foi melhor que o cenário MAXSHI. Porém, não houve diferença entre os métodos ILSMI e com aprendizado (MAXSHI e SOFTQ). Em relação ao tempo, assim como anteriormente, todos os cenários foram melhores que o cenário ILSBI. O cenário SOFTQ foi pior que o cenário MAXSHI. Neste caso, contudo, os métodos com aprendizado foram melhores que o método ILSMI, mostrando, então, que os métodos com aprendizado (MAXSHI e SOFTQ) reduzem o número de passos e, portanto, melhoram a evolução do algoritmo.

Portanto, é interessante verificar diretamente os resultados obtidos pelos métodos de aprendizado. Para isso, foram utilizadas as melhores soluções encontradas em cada cenário, de forma a avaliar se os cenários com aprendizado melhoram as soluções obtidas. A Tabela 5.7 mostra os melhores resultados obtidos para as 30 execuções dos cenários. Podemos observar que, nas instâncias do grupo C, todos os métodos encontram as melhores soluções da literatura. Já nos grupos R e RC, em que não foram obtidos os melhores resultados da literatura, os métodos de aprendizado alcançam 15 vezes melhores soluções que o método ILSMI, sendo que o cenário MAXSHI obteve 6 vezes melhores soluções e o método SOFTQ obteve 9 vezes melhores soluções que o cenário ILSMI. Portanto, pode-se concluir que os métodos de aprendizado, além de reduzir o tempo, podem melhorar os valores de função objetivo obtidos em certas ocasiões.

5.1.7 Considerações Parciais

Os resultados desta seção mostram que a incorporação do algoritmo *Multi-armed Bandit* ao gerenciamento do *Pool* de soluções do *Framework* AMAM constitui um avanço efetivo na dimensão de cooperação do sistema frente aos encontrados em Silva et al. (2019), representados pelo cenário ILSBI. A análise comparativa entre os quatro cenários de configuração revelou uma dicotomia clara entre qualidade de solução e eficiência computacional: os métodos de escolha SOFT mostraram-se superiores em relação ao valor de função objetivo, com o cenário SOFTQ alcançando 7 vitórias nos

Tabela 5.7: Melhores soluções por Cenários - *Multi-armed Bandit Pool*

Instância	Cenário	2	4	8	16
C109	ILSMI	10827,30	10827,30	10827,30	10827,30
	MAXSHI	10827,30	10827,30	10827,30	10827,30
	SOFTQ	10827,30	10827,30	10827,30	10827,30
C208	ILSMI	3585,80	3585,80	3585,80	3585,80
	MAXSHI	3585,80	3585,80	3585,80	3585,80
	SOFTQ	3585,80	3585,80	3585,80	3585,80
R112	ILSMI	10954,60	10951,50	10950,90	10955,40
	MAXSHI	10962,80	10955,40	10955,20	10952,20
	SOFTQ	10955,40	10950,80	10948,60	10950,40
R211	ILSMI	3768,50	3779,40	3768,50	3763,10
	MAXSHI	3770,00	3777,80	3774,00	3763,10
	SOFTQ	3775,90	3778,80	3763,10	3769,50
RC108	ILSMI	11171,00	12136,40	11156,50	11147,30
	MAXSHI	11181,40	11174,70	11162,00	11144,90
	SOFTQ	11179,50	11186,60	11148,20	11138,90
RC208	ILSMI	3836,20	3857,60	3829,40	3827,90
	MAXSHI	3857,10	3831,60	3832,80	3827,70
	SOFTQ	3860,30	3831,30	3831,00	3839,00

testes estatísticos sem nenhuma derrota, ao passo que os métodos MAX, em especial o MAXSHI, apresentaram vantagem expressiva em tempo de execução, vencendo 37 dos 72 testes realizados. A comparação com os cenários de referência sem aprendizado evidenciou que todos os métodos com *Multi-armed Bandit* superam o cenário ILSBI e que, em relação ao tempo, os cenários com aprendizado superam estatisticamente o ILSMI, indicando que o mecanismo de recompensa direciona os agentes para soluções mais promissoras e reduz passos ineficientes ao longo da busca. A análise das melhores soluções encontradas confirmou que, nos grupos de instâncias mais desafiadores (R e RC), os métodos com aprendizado alcançaram resultados superiores ao ILSMI em 15 oportunidades, reforçando que o algoritmo *Multi-armed Bandit Pool*, além de acelerar a convergência, amplia a capacidade do *framework* de explorar regiões do espaço de soluções que o gerenciamento aleatório não consegue explorar de forma eficiente.

5.2 *Multi-armed Bandit* Multiagente

A Seção 5.1 mostrou que o algoritmo *Multi-armed Bandit* aplicado ao *Pool* de soluções reduz o tempo de execução e melhora a qualidade das soluções obtidas pelo *framework* AMAM. Naquela análise, contudo, o desempenho do algoritmo foi avaliado no nível global do sistema, isto é, considerando a recompensa e o desempenho do conjunto de agentes como um todo, sem examinar como cada agente conduz individualmente suas decisões nem como o conhecimento acumulado poderia ser compartilhado entre eles. Em cenários com múltiplos agentes explorando simultaneamente o

mesmo *Pool*, a ausência de um mecanismo explícito de cooperação no nível individual pode gerar exploração redundante, na qual diferentes agentes selecionam as mesmas soluções pouco promissoras sem se beneficiar das experiências alheias, retardando a convergência do sistema.

O desafio central em sistemas multiagente cooperativos é definir como cada agente deve equilibrar seu aprendizado individual com o conhecimento coletivo produzido pelo grupo (Landgren et al., 2021). Priorizar apenas o aprendizado individual pode levar a comportamentos especializados mas redundantes entre agentes; ao contrário, privilegiar exclusivamente a informação social pode suprimir a diversidade e induzir convergência prematura (Liu e Zhao, 2010). Esse dilema é análogo ao mecanismo cognitivo-social do *Particle Swarm Optimization* (PSO) (Kennedy e Eberhart, 1995), em que cada partícula ajusta sua trajetória a partir de sua melhor experiência individual e da melhor experiência coletiva observada pelo grupo.

Motivado por essa analogia, esta seção propõe o *Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation* (MAMaB-PSO), um algoritmo de aprendizado por reforço multiagente que incorpora os componentes cognitivo e social do PSO ao processo de seleção do *Multi-armed Bandit* no *Pool* de soluções da AMAM. A formulação é geral e independente de domínio: a política de seleção de alternativas é mantida como um componente genérico, permitindo que diferentes variantes do MaB sejam instanciadas. Quando os coeficientes PSO são nulos, o MAMaB-PSO se reduz ao MaB multiagente clássico, de modo que a formulação proposta generaliza o modelo padrão.

5.2.1 Modelagem do sistema multiagente

A modelagem multiagente adotada neste trabalho segue a formulação geral do *Multi-Agent Multi-armed Bandit* apresentada na Seção 2.3.4, na qual um conjunto de N agentes $\mathcal{N} = \{1, 2, \dots, N\}$ interage simultaneamente com um mesmo conjunto de braços. A cada instante t , cada agente $i \in \mathcal{N}$ seleciona um braço $A_i(t)$ por meio da estratégia ϵ -greedy, recebe uma recompensa $R_i(t)$ associada à decisão tomada e atualiza, pela média incremental das recompensas observadas, suas estimativas locais de valor $Q_i(a)$ e o contador de seleções $N_i(a)$ de cada braço a . O objetivo do sistema é maximizar a recompensa acumulada conjunta dos agentes ou, de forma equivalente, minimizar o arrependimento coletivo (*regret*) em relação à alocação ótima dos braços.

A particularidade da instância aqui considerada está na natureza do conjunto de braços. Cada braço $a \in \mathcal{K}(t)$ corresponde a uma solução candidata disponível no *Solutions Pool* da AMAM no instante t , de modo que o conjunto $\mathcal{K}(t) = \{1, 2, \dots, K(t)\}$ é dinâmico: $K(t)$ varia ao longo da execução conforme soluções são inseridas ou removidas do *Pool* pelos agentes. Essa característica, ausente na formulação clássica, introduz a não estacionariedade tratada nas subseções seguintes e motiva as adaptações propostas no MAMaB-PSO.

5.2.2 Cenários de recompensa

A recompensa recebida pelo agente i após selecionar e utilizar a solução $A_i(t)$ do *Pool* é definida com base na melhoria de qualidade produzida por essa solução no processo de busca do agente. Para problemas de minimização, a recompensa é

formulada como:

$$R_i(t) = f(best_i(t-1)) - f(best_i(t)), \quad (5.2)$$

em que $best_i(t-1)$ é a melhor solução conhecida pelo agente i antes da utilização da solução selecionada, e $best_i(t)$ é a melhor solução após a execução da meta-heurística no instante t . Valores positivos de $R_i(t)$ indicam que a solução selecionada contribuiu para a melhoria do desempenho do agente; valores nulos ou negativos indicam ausência de melhoria. Essa definição está em conformidade com o esquema de recompensas da AMAM descrito na Seção 4.4.

5.2.3 Aprendizado multiagente em ambiente não estacionário

O conjunto de braços disponíveis no *Pool* não é estacionário, propriedade já identificada na Seção 5.1: soluções são inseridas e removidas ao longo da execução conforme os agentes evoluem sua busca, alterando tanto a composição quanto a qualidade relativa das alternativas disponíveis. Essa dinâmica torna as observações passadas progressivamente menos representativas da utilidade atual de cada solução, o que invalida as garantias de convergência dos algoritmos MaB clássicos projetados para ambientes estacionários (Garivier e Moulines, 2011).

Para lidar com a não estacionariedade, duas estratégias são adotadas em variantes do MAMaB-PSO. A estratégia *Discounted UCB* (DUCB) (Kocsis e Szepesvári, 2006) introduz um fator de desconto $\gamma \in (0, 1]$ que reduz progressivamente o peso das observações antigas, conferindo maior influência às recompensas recentes. A estratégia *Sliding-Window UCB* (SWUCB) (Garivier e Moulines, 2011) restringe o histórico de observações a uma janela de tamanho W , descartando completamente interações anteriores e sendo especialmente adequado para ambientes com mudanças abruptas. Essas estratégias são instanciadas como políticas de seleção no algoritmo MAMaB-PSO, conforme descrito na Subseção 5.2.6.

Antes disso, cabe registrar como a estimativa local de cada agente é mantida, uma vez que é sobre ela que todo o mecanismo cooperativo da Seção 5.2.4 seguinte é construído. A estimativa local de cada agente é atualizada com taxa de aprendizado $\alpha \in (0, 1)$, que atribui maior peso às observações recentes e confere ao modelo uma capacidade natural de adaptação a ambientes não estacionários:

$$Q_i(a, t) = (1 - \alpha) Q_i(a, t-1) + \alpha R_i(t) \cdot 1[A_i(t) = a]. \quad (5.3)$$

em que $Q_i(a, t)$ é a estimativa que o agente i mantém para o braço a no instante t ; $R_i(t)$ é a recompensa observada, conforme a Equação (5.2); e a função indicadora $1[A_i(t) = a]$ restringe a atualização ao braço efetivamente selecionado. O uso de α fixo, em lugar da média aritmética $1/N_i(a, t)$ do MaB clássico, é o que impede que observações antigas dominem a estimativa em um *Pool* cujo conteúdo muda ao longo da execução.

5.2.4 Aprendizado cooperativo via *Pool* de soluções

O mecanismo de cooperação proposto é inspirado na meta-heurística PSO (Kennedy e Eberhart, 1995). No PSO, cada partícula ajusta sua posição considerando dois componentes:

- (i) o *componente cognitivo*, que representa a melhor posição já encontrada pela própria partícula; e
- (ii) o *componente social*, que representa a melhor posição encontrada por qualquer partícula do enxame.

Esse mecanismo promove um equilíbrio dinâmico entre autonomia individual e cooperação coletiva.

No contexto do MaB multiagente, essa formulação é aplicada da seguinte forma. O algoritmo MAMaB-PSO define, para cada agente i e cada braço a , dois componentes de influência sobre a estimativa de valor:

- (i) *componente cognitivo*: captura a melhor experiência acumulada pelo próprio agente para cada braço, na forma:

$$P_i(a, t) = \max_{j \leq t} Q_i(a, j), \quad (5.4)$$

em que $P_i(a, t)$ é a melhor avaliação histórica do agente i para o braço a até o instante t ;

- (ii) *componente social*: captura a melhor avaliação observada entre todos os agentes para cada braço:

$$\mathcal{G}(a, t) = \max_{i \in \mathcal{N}} P_i(a, t), \quad (5.5)$$

em que $\mathcal{G}(a, t)$ sintetiza o conhecimento coletivo do sistema sobre o braço a até o instante t .

Os termos “*componente cognitivo*” e “*componente social*” mantêm a mesma nomenclatura adotada originalmente na literatura referente à meta-heurística PSO.

Com base nesses dois componentes, é definida a *estimativa híbrida* $\tilde{Q}_i(a, t)$ do algoritmo MAMaB-PSO para o agente i e o braço a , dada por:

$$\tilde{Q}_i(a, t) = Q_i(a, t) + c_1 r_{1,i}(t) [P_i(a, t) - Q_i(a, t)] + c_2 r_{2,i}(t) [\mathcal{G}(a, t) - Q_i(a, t)] \quad (5.6)$$

em que $c_1 \geq 0$ e $c_2 \geq 0$ são coeficientes que controlam, respectivamente, a intensidade da influência cognitiva e da influência social; e $r_{1,i}(t), r_{2,i}(t) \sim \mathcal{U}(0,1)$ são variáveis aleatórias independentes amostradas a cada instante, conforme a formulação clássica do PSO (Kennedy e Eberhart, 1995). Assim, a estimativa $\tilde{Q}_i(a, t)$ combina três elementos:

- (i) o conhecimento atual do agente, dado pelo termo $Q_i(a, t)$;
- (ii) sua melhor experiência individual, dada pelo termo $P_i(a, t)$; e
- (iii) a melhor experiência global observada pelo sistema, dada pelo termo $\mathcal{G}(a, t)$.

Rearranjando os termos da Equação (5.6) em torno de cada elemento-atrator, obtém-se a forma:

$$\tilde{Q}_i(a, t) = [1 - c_1 r_{1,i}(t) - c_2 r_{2,i}(t)] Q_i(a, t) + c_1 r_{1,i}(t) P_i(a, t) + c_2 r_{2,i}(t) \mathcal{G}(a, t), \quad (5.7)$$

que evidencia a estimativa híbrida como uma combinação *afim* dos três elementos-atratores, uma vez que as ponderações associadas a cada termo somam 1, mas não estão necessariamente contidos no intervalo $[0,1]$. Com os valores adotados nos experimentos, ou seja, $c_1 = c_2 = 2$, o valor $c_1 r_{1,i}(t) + c_2 r_{2,i}(t)$, que pondera o elemento $Q_i(a, t)$ na Expressão 5.7, percorre o intervalo $[0,4]$ e o peso de $Q_i(a, t)$ assume valor negativo na maior parte das iterações, com valor esperado igual a -1 . Portanto, não se trata de uma interpolação entre os três termos, mas, sim, de uma extrapolação na direção do melhor valor individual e do melhor valor global, comportamento herdado da atualização de velocidade do PSO e que é justamente o que confere, à meta-heurística e ao método MAMaB-PSO proposto, a capacidade de diversificação.

Os parâmetros c_1 e c_2 controlam o equilíbrio entre individualismo e cooperação: valores maiores de c_2 aproximam o comportamento dos agentes de uma decisão centralizada, enquanto valores maiores de c_1 favorecem a especialização individual. Quando $c_1 = c_2 = 0$, o algoritmo MAMaB-PSO se reduz ao algoritmo MaB multiagente clássico, cuja regra de seleção é a da Equação (2.10): sem os termos cognitivo e social, $\tilde{Q}_i(a, t)$ colapsa em $Q_i(a, t)$, e cada agente passa a decidir apenas com sua própria estimativa, sem qualquer troca de informação com os demais.

5.2.5 Algoritmo MAMaB-PSO

O Algoritmo 9 apresenta a estrutura geral do MAMaB-PSO. A linha 1 faz a inicialização de valores: as estimativas locais $Q_i(a, 0)$, os componentes cognitivo $P_i(a, 0)$ e social $\mathcal{G}(a, 0)$ e os contadores de seleção $N_i(a, 0)$ têm valor inicial igual a zero, para todo agente $i \in \mathcal{N}$ e todo braço $a \in \mathcal{K}$.

A linha 2 percorre o horizonte de T iterações e a linha 3 percorre os agentes dentro de cada iteração. Para cada agente, a linha 4 calcula o fator de consenso $\theta(t)$ da Equação (5.9), que mede quanto a visão individual pesa em relação à social naquele instante. Em seguida, a linha 5 percorre os braços disponíveis: a linha 6 calcula a estimativa híbrida $\tilde{Q}_i(a, t)$ da Equação (5.6), que combina o conhecimento

Algoritmo 9 MAMaB-PSO - Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation

Entrada: conjunto de agentes $\mathcal{N}$, conjunto de braços $\mathcal{K}$, política π , parâmetros α , c_1 , c_2 , horizonte T

```

1: Inicializar  $Q_i(a, 0) \leftarrow 0$ ,  $P_i(a, 0) \leftarrow 0$ ,  $N_i(a, 0) \leftarrow 0$ ,  $\mathcal{G}(a, 0) \leftarrow 0$  para todo  $i \in \mathcal{N}$ ,  $a \in \mathcal{K}$ 
2: para  $t = 1, 2, \dots, T$  faça
3:   para todo agentes  $i \in \mathcal{N}$  faça
4:     Calcular o fator de consenso  $\theta(t)$  conforme a Equação (5.9) ▷ instânciação AMAM
5:     para todo braços  $a \in \mathcal{K}(t)$  faça
6:       Calcular  $\tilde{Q}_i(a, t)$  conforme a Equação (5.6)
7:       Calcular a pontuação  $\tilde{S}_{a,i}(t)$  conforme a Equação (5.10) ▷ instânciação AMAM
8:     fim para
9:     Selecionar  $A_i(t) \leftarrow \pi(\tilde{Q}_i(\cdot, t), N_i(\cdot, t))$  ▷ ou  $\tilde{S}_{\cdot,i}(t)$  na instânciação AMAM
10:    Observar recompensa  $R_i(t)$  conforme a Equação (5.2)
11:    Atualizar  $Q_i(A_i(t), t)$  conforme a Equação (5.3)
12:    Atualizar  $N_i(A_i(t), t) \leftarrow N_i(A_i(t), t-1) + 1$ 
13:    Atualizar  $P_i(a, t)$  conforme a Equação (5.4)
14:  fim para
15:  para todo braços  $a \in \mathcal{K}(t)$  faça
16:    Atualizar  $\mathcal{G}(a, t)$  conforme a Equação (5.5)
17:  fim para
18: fim para
19: retorne  $\arg \max_{a \in \mathcal{K}} \mathcal{G}(a, T)$ 

```

atual do agente com sua melhor experiência individual e com a melhor experiência global, e a linha 7 calcula a pontuação $\tilde{S}_{a,i}(t)$ da Equação (5.10), que é a forma reduzida empregada na instanciãção no *framework* AMAM: dois termos em vez de três, pesos convexas em vez de afins e pontuações médias com bônus de diversificação em vez dos máximos P_i e $\mathcal{G}$. As linhas 4 e 7 são executadas apenas nessa instanciãção; na formulação geral, a seleção recai diretamente sobre $\tilde{Q}_i(a, t)$.

A linha 9 aplica a política π sobre a pontuação em uso, $\tilde{Q}_i$ na formulação geral ou $\tilde{S}_{\cdot,i}$ na instanciãção no AMAM, e seleciona o braço $A_i(t)$. A seleção do braço por cada agente i é realizada por uma política MaB genérica π aplicada sobre a estimativa híbrida, conforme a expressão geral:

$$A_i(t) = \pi\left(\tilde{Q}_i(\cdot, t), N_i(\cdot, t)\right), \quad (5.8)$$

em que $\pi(\cdot, \cdot)$ pode ser instanciada por qualquer algoritmo MaB (ϵ -greedy, UCB, DUCB, SWUCB, dentre outros), conforme descrito na Subseção 5.2.6.

Na linha 10, o agente i recebe a recompensa $R_i(t)$, segundo o cenário de recompensa adotado, definido na Equação (5.2). As linhas 11 a 13 fazem a atualização local do agente: a estimativa Q_i é recalculada com a taxa de aprendizado α , o contador N_i do braço escolhido é incrementado e o componente cognitivo P_i passa a guardar a melhor avaliação histórica daquele agente.

Encerrado o laço dos agentes, as linhas 15 e 16 atualizam o componente social $\mathcal{G}(a, t)$ de cada braço, tomando o máximo das melhores experiências individuais de todos os agentes. É esse passo que realiza a cooperação: o que um agente descobre passa a estar disponível para os demais na iteração seguinte. A linha 19 devolve o braço com a melhor avaliação global acumulada ao longo de todo o processo de aprendizagem.

5.2.6 Variantes da Política de Seleção MAMaB-PSO

Conforme a formulação do algoritmo MAMaB-PSO, a política de seleção π é mantida como um componente genérico. Na instanciãção do *framework* AMAM, cada solução $a \in \mathcal{K}(t)$ armazena dois tipos de informação de aprendizado: uma informação *social*, compartilhada entre todos os agentes, e uma informação *individual*, específica para cada agente i . Formalmente, para cada solução a , são mantidos o valor acumulado social Q_a^{soc} , o contador de utilizações sociais N_a^{soc} , o valor acumulado individual $Q_{a,i}^{ind}$ e o contador de utilizações individuais $N_{a,i}^{ind}$. Esses termos têm a seguinte descrição:

- Q_a^{soc} acumula a soma das recompensas obtidas por *todos* os agentes sempre que a solução a foi coletada;
- N_a^{soc} conta quantas vezes isso ocorreu, de modo que Q_a^{soc}/N_a^{soc} é a recompensa média social da solução;
- $Q_{a,i}^{ind}$ e $N_{a,i}^{ind}$ representam os mesmos dois valores, restritos, no entanto, às coletas realizadas pelo agente i , sendo $Q_{a,i}^{ind}/N_{a,i}^{ind}$ a recompensa média que aquele agente, individualmente, obteve com a solução a .

Inicialmente, os contadores têm valor $N_a^{soc} = 0$ e $N_{a,i}^{ind} = 0$. Enquanto $N_a^{soc} = 0$ ou $N_{a,i}^{ind} = 0$, a média correspondente não está definida, situação em que a solução recebe escore infinito e é priorizada, seguindo a mesma convenção da Seção 2.3.2.1. Note que os contadores N_a^{soc} e $N_{a,i}^{ind}$ registram *número de utilizações* de cada braço, e não se confundem com $|\mathcal{K}(t)|$, que é a quantidade de braços disponíveis no instante t .

A aplicação do MAMaB-PSO adota, contudo, uma forma alternativa da Expressão (5.6), e convém explicitar em que ela difere da formulação geral. Assim, a Expressão (5.6) combina três atratores por braço, dados por $Q_i(a,t)$, $P_i(a,t)$ e $\mathcal{G}(a,t)$, com pesos afins, que podem assumir valores negativos. A forma alternativa combina apenas dois fatores, quais sejam, a pontuação individual, dada por $Q_{a,i}^{ind}$, e a pontuação social de cada solução, dada por Q_a^{soc} , na forma de uma combinação convexa, com pesos que somam 1 e permanecem no intervalo $[0,1]$. Na expressão adotada, além disso, substitui-se os valores de médias acumuladas pelos valores máximos acrescidos do bônus de diversificação próprio de cada variante. Define-se, então, $\max_a Q_{a,i}^{ind}$ e $\max_a Q_a^{soc}$ como os valores máximos individual e global, respectivamente, sendo ponderações escalares que representam o peso relativo de cada uma das duas visões, e não como atratores por braço. A redução de elementos na expressão mantém a ideia central do PSO, ou seja, o equilíbrio dinâmico entre a experiência própria e a coletiva, e evita a extrapolação de pesos negativos sobre pontuações que já incorporam um termo de diversificação.

Portanto, para as variantes baseadas no MAMaB-PSO definidas a seguir, a pontuação final de uma solução a para o agente i é obtida pela combinação das visões individual e social por um fator de consenso dinâmico $\theta(t) \in [0,1]$, na forma:

$$\theta(t) = \frac{c_1 r_{1,i}(t) \max_a Q_{a,i}^{ind}}{c_1 r_{1,i}(t) \max_a Q_{a,i}^{ind} + c_2 r_{2,i}(t) \max_a Q_a^{soc} + \xi} \quad (5.9)$$

em que $\xi > 0$ é uma constante de estabilidade numérica, que evita a divisão por zero quando ambos os termos do denominador se anulam. A pontuação híbrida da solução a para o agente i é, então:

$$\tilde{S}_{a,i}(t) = \theta(t) S_{a,i}^{ind}(t) + (1 - \theta(t)) S_a^{soc}(t). \quad (5.10)$$

As variantes se diferenciam pela forma de calcular os termos $S_{a,i}^{ind}$ e S_a^{soc} . As variantes avaliadas são:

- (a) MONO-EG (`mono_epsilon_greedy`): Variante de referência, sem o componente PSO e sem compartilhamento de aprendizado entre os agentes. Opera exclusivamente sobre a informação individual de cada agente: com probabilidade ϵ , coleta uma solução aleatória; com probabilidade $1 - \epsilon$, seleciona $A_i(t) = \arg \max_a Q_{a,i}^{ind}$.
- (b) MULTI-EG (`multi_epsilon_greedy`): Variante de referência, sem o componente PSO. Opera exclusivamente sobre a informação social. Com probabilidade ϵ , coleta uma solução aleatória; com probabilidade $1 - \epsilon$, seleciona $A_i(t) = \arg \max_a Q_a^{soc}$. Por não incorporar a visão individual e em oposição ao MONO-EG, serve como *baseline* para avaliar o ganho proporcionado pelo mecanismo PSO.

- (c) HYBRID-EG (`hybrid_epsilon_greedy`): Variante de referência que combina as visões individual e social com a mesma estrutura da Equação (5.10), porém substituindo o fator de consenso dinâmico $\theta(t)$ por um coeficiente fixo $\theta \in [0,1]$, definido antes da execução. A seleção segue a estratégia ϵ -greedy sobre a pontuação combinada resultante. Diferentemente do MAMaB-PSO, o peso entre as visões não se adapta ao longo da busca, servindo como baseline para isolar o efeito da ponderação adaptativa frente a uma ponderação estática.
- (d) PSO-EG (`pso_epsilon_greedy`): Incorpora o mecanismo PSO ao ϵ -greedy: com probabilidade ϵ , coleta uma solução aleatória; com probabilidade $1 - \epsilon$, seleciona pelo maior fator $\tilde{S}_{a,i}(t)$, definido na Expressão (5.10).
- (e) PSO-UCB (`pso_ucb`): Aplica o índice UCB, descrito na Seção 2.3.2.1, separadamente sobre as visões social e individual:

$$S_a^{soc} = \frac{Q_a^{soc}}{N_a^{soc}} + c \sqrt{\frac{2 \ln T^{soc}}{N_a^{soc}}}$$

$$S_{a,i}^{ind} = \frac{Q_{a,i}^{ind}}{N_{a,i}^{ind}} + c \sqrt{\frac{2 \ln T_i^{ind}}{N_{a,i}^{ind}}},$$

em que T^{soc} e T_i^{ind} são os totais de coletas social e individual do agente i . Soluções não selecionadas recebem pontuação infinita para forçar diversificação inicial.

- (f) PSO-DUCB (`pso_ducb`): Implementa a versão descontada do UCB, o DUCB da Seção 2.3.2.1. As estatísticas são reconstruídas com fator de desconto $\gamma \in [0,1]$, atribuindo maior peso às recompensas recentes:

$$S_a^{soc} = \bar{R}_a^{soc,\gamma} + c \sqrt{\frac{2 \ln T^{soc,\gamma}}{N_a^{soc,\gamma}}}$$

$$S_{a,i}^{ind} = \bar{R}_{a,i}^{ind,\gamma} + c \sqrt{\frac{2 \ln T_i^{ind,\gamma}}{N_{a,i}^{ind,\gamma}}}.$$

- (g) PSO-SWUCB (`pso_swucb`): Implementa o UCB com janela deslizante de tamanho W , o SWUCB da Seção 2.3.2.1. As estatísticas consideram apenas as últimas W recompensas observadas:

$$S_a^{soc} = \bar{R}_a^{soc,W} + c \sqrt{\frac{2 \ln T^{soc,W}}{N_a^{soc,W}}}$$

$$S_{a,i}^{ind} = \bar{R}_{a,i}^{ind,W} + c \sqrt{\frac{2 \ln T_i^{ind,W}}{N_{a,i}^{ind,W}}}.$$

- (h) PSO-FRR (`pso_frrmab`): Aplica a estratégia FRRMaB, descrita na Seção 2.3.2.1, sobre janelas deslizantes. A recompensa agregada da solução a é ponderada por decaimento conforme ranqueamento:

$$\begin{aligned} Decay_a &= d^{\text{Rank}_a} \cdot R_a \\ S_a^{soc} &= F_a^{FRR} + c \sqrt{\frac{2 \ln W}{n_a}} \\ S_{a,i}^{ind} &= F_{a,i}^{FRR,ind} + c \sqrt{\frac{2 \ln W}{n_{a,i}}}. \end{aligned}$$

- (i) PSO-FRA (`pso_framab`): Segue a mesma estrutura do PSO-FRR, substituindo o ranqueamento com decaimento pela média das melhorias, conforme o FRAMaB da Seção 2.3.2.1:

$$\begin{aligned} S_a^{soc} &= F_a^{FRA} + c \sqrt{\frac{2 \ln W}{n_a}} \\ S_{a,i}^{ind} &= F_{a,i}^{FRA,ind} + c \sqrt{\frac{2 \ln W}{n_{a,i}}}. \end{aligned}$$

A Tabela 5.8 sintetiza todos os cenários experimentais avaliados, organizados por cenário. Os cenários das famílias MONO, MULTI e HYBRID servem como referências para validar a abordagem PSO, enquanto os cenários PSO representam as variantes do algoritmo MAMaB-PSO proposto.

Tabela 5.8: Cenários experimentais avaliados, organizados por família de política. Todos os cenários utilizam a mesma composição de agentes, ou seja, ILS_MIX com 20 agentes, descrita na Seção 5.2.7.

Família	Cenário	Política
MONO	MONO-EG	mono_epsilon_greedy
MULTI	MULTI-EG	multi_epsilon_greedy
HYBRID	HYBRID-EG	hybrid_epsilon_greedy
PSO	PSO-EG	pso_epsilon_greedy
	PSO-UCB	pso_ucb
	PSO-DUCB	pso_ducb
	PSO-SWUCB	pso_swucb
	PSO-FRR	pso_frrmab
	PSO-FRA	pso_framab

5.2.7 Definição da Experimentação

Os experimentos utilizam o cenário ILS_MIX. Este cenário é composto por sete agentes ILS_ALS_BI, sete agentes ILS_ALS_FI e seis agentes ILS_ALS_RI, totalizando

20 agentes por execução, conforme descrito na Seção 3.2. As instâncias utilizadas são as mesmas do conjunto de Solomon (1987) descritas na Seção 3.3: 24 instâncias representativas dos blocos C, R e RC, com 100 clientes cada.

O ambiente computacional e o critério de parada são os mesmos descritos na Seção 3.3. O critério de parada para cada agente ILS foi definido como 100 iterações consecutivas sem melhoria. Para cada cenário, os experimentos foram repetidos 30 vezes em cada instância, garantindo robustez estatística aos resultados. A avaliação computacional considera duas métricas complementares: a qualidade das soluções, medida pelo *gap* percentual em relação ao melhor valor conhecido na literatura, calculado conforme a Equação (3.2), e o esforço computacional, mensurado pelo tempo médio de execução em segundos. A metodologia de análise estatística segue a mesma abordagem não paramétrica adotada nos demais experimentos desta tese, justificada na Seção 3.3, com teste de Friedman (Montgomery e Runger, 2010) para comparação global e post-hoc de Wilcoxon pareado (Wilcoxon, 1992) com correção de Bonferroni (Montgomery e Runger, 2010).

5.2.8 Comparação entre os Cenários das Famílias de Políticas

Para validar a contribuição do mecanismo PSO proposto, os experimentos avaliam quatro cenários representativos, um por cenário de seleção, que diferem na forma como combinam o aprendizado individual e o aprendizado social de cada agente durante a coleta de soluções no *Pool*:

- (i) **MONO-EG**: cada agente i mantém suas próprias estimativas individuais $Q_{a,i}^{ind}$ para cada solução a do *Pool* e realiza a seleção exclusivamente com base no seu próprio histórico. Essa formulação é equivalente a N instâncias independentes do MaB clássico, uma por agente, sem qualquer troca de conhecimento aprendido entre os agentes. Embora as soluções sejam compartilhadas no *Pool*, o aprendizado sobre a utilidade de cada solução não é compartilhado;
- (ii) **MULTI-EG**: todos os agentes utilizam uma única estimativa social Q_a^{soc} , construída a partir das observações acumuladas de todo o sistema. A seleção é realizada exclusivamente com base nessa visão coletiva, descartando o aprendizado individual de cada agente. Essa formulação corresponde ao MaB multiagente com escolha centralizada;
- (iii) **HYBRID-EG**: combina as visões individual e social por uma ponderação linear com coeficiente fixo $\theta \in [0,1]$, definido como parâmetro antes da execução:

$$\tilde{S}_{a,i} = \theta \cdot S_{a,i}^{ind} + (1 - \theta) \cdot S_a^{soc}. \quad (5.11)$$

Diferentemente do MAMaB-PSO, o coeficiente θ é estático e não se adapta ao longo do processo de busca;

- (iv) **PSO-EG (MAMaB-PSO)**: combina as visões individual e social por meio do coeficiente dinâmico $\theta(t)$, calculado a partir dos componentes cognitivo e social do PSO, conforme a Equação (5.9). O coeficiente é recalculado de forma adaptativa a cada coleta, ponderando as melhores experiências individuais e sociais acumuladas até aquele instante pelos coeficientes c_1 e c_2 , com fatores aleatórios $r_{1,i}(t), r_{2,i}(t) \sim \mathcal{U}(0,1)$.

Tabela 5.9: Parâmetros das políticas MaB utilizados nos experimentos.

Parâmetro	Símbolo	Valor	Políticas
Taxa de diversificação	ϵ	0,10	ϵ -greedy
Constante de diversificação UCB	c	1,0	UCB, FRRMaB, FRAMaB
Fator de desconto temporal	γ	0,77	DUCB, FRRMaB
Tamanho da janela deslizante	W	100	SWUCB, FRRMaB, FRAMaB
Peso fixo individual	θ	0,5	HYBRID
Coefficiente cognitivo	c_1	2,0	MAMaB-PSO
Coefficiente social	c_2	2,0	MAMaB-PSO

Os cenários **MONO-EG**, **MULTI-EG** e **HYBRID-EG** empregam a política de base ϵ -greedy (EG) e servem como referências para a abordagem PSO. A família PSO, por sua vez, é avaliada com seis políticas de seleção (o próprio **PSO-EG** e as variantes UCB, DUCB, SWUCB, FRRMaB e FRAMaB), totalizando nove cenários, descritos na Tabela 5.8.

A Tabela 5.9 apresenta os parâmetros específicos de cada política MaB utilizados nos experimentos. Cada parâmetro controla um aspecto distinto do equilíbrio entre diversificação e intensificação: a taxa ϵ define a probabilidade de seleção aleatória nas políticas ϵ -greedy; a constante c regula o bônus de diversificação nas políticas baseadas em UCB; o fator γ e o tamanho de janela W governam, respectivamente, o desconto temporal e a janela deslizante nas variantes não estacionárias, sendo compartilhados entre DUCB, SWUCB, FRRMaB e FRAMaB para garantir consistência nas comparações; o coeficiente θ determina o peso fixo da visão individual na política HYBRID; e os coeficientes c_1 e c_2 do MAMaB-PSO controlam a intensidade dos componentes cognitivo e social, seguindo os valores convencionais da literatura de PSO (Kennedy e Eberhart, 1995).

5.2.8.1 Validação da Abordagem PSO

Esta análise compara as quatro famílias de políticas, utilizando os cenários representativos com ϵ -greedy como política de base: **MONO-EG**, **MULTI-EG**, **HYBRID-EG** e **PSO-EG**. O objetivo é verificar se o mecanismo de ponderação adaptativa do PSO produz resultados superiores à ponderação fixa do HYBRID e às abordagens puramente individuais ou puramente sociais. A comparação é conduzida separadamente sob dois critérios: a qualidade das soluções, medida pelo *gap* percentual, e o esforço computacional, medido pelo tempo médio de execução.

Análise do *gap* percentual

A Tabela 5.10 apresenta a média do *gap* percentual por instância para os quatro cenários representativos, e a Tabela 5.11 agrega esses valores por bloco. Os valores em negrito indicam o melhor resultado.

A Figura 5.3 exibe os *boxplots* do *gap* percentual para os quatro cenários. O **PSO-EG** apresenta a menor mediana e menor dispersão entre as quatro abordagens. O teste de Friedman, que respeita o emparelhamento por instância, não identificou diferença estatisticamente significativa na qualidade das soluções entre os cenários ($p = 0,73$). Além disso, o teste de Wilcoxon *signed-rank* com correção de Bonferroni não apontou diferença em nenhum dos seis pares, indicando, portanto, equivalên-

Tabela 5.10: Média do *gap* (%) por instância: comparação entre os cenários representativos. Os valores após $\pm$ são o desvio-padrão das 30 execuções independentes de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MONO-EG	MULTI-EG	HYBRID-EG	PSO-EG
C101	0,00 $\pm$ 0,01	0,01 $\pm$ 0,02	0,01 $\pm$ 0,02	0,00 $\pm$ 0,01
C103	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C107	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C109	0,01 $\pm$ 0,04	0,01 $\pm$ 0,04	0,01 $\pm$ 0,04	0,00 $\pm$ 0,00
C204	0,08 $\pm$ 0,08	0,12 $\pm$ 0,09	0,11 $\pm$ 0,09	0,10 $\pm$ 0,10
C205	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C206	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C208	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
R102	1,47 $\pm$ 2,39	0,59 $\pm$ 1,62	0,94 $\pm$ 2,01	0,94 $\pm$ 2,00
R104	9,93 $\pm$ 0,10	9,93 $\pm$ 0,10	9,93 $\pm$ 0,09	9,98 $\pm$ 0,12
R106	5,88 $\pm$ 3,14	6,60 $\pm$ 2,56	5,12 $\pm$ 3,57	4,90 $\pm$ 3,61
R112	10,10 $\pm$ 0,14	10,03 $\pm$ 0,14	10,07 $\pm$ 0,11	10,03 $\pm$ 0,13
R201	0,23 $\pm$ 0,19	0,23 $\pm$ 0,17	0,23 $\pm$ 0,15	0,25 $\pm$ 0,19
R208	0,46 $\pm$ 0,25	0,54 $\pm$ 0,33	0,44 $\pm$ 0,21	0,55 $\pm$ 0,29
R209	0,63 $\pm$ 0,40	0,86 $\pm$ 0,37	0,85 $\pm$ 0,39	0,71 $\pm$ 0,37
R211	31,81 $\pm$ 0,51	31,72 $\pm$ 0,50	31,69 $\pm$ 0,55	31,73 $\pm$ 0,56
RC102	10,75 $\pm$ 3,65	11,04 $\pm$ 3,61	10,29 $\pm$ 3,58	10,75 $\pm$ 3,65
RC103	3,65 $\pm$ 4,07	3,67 $\pm$ 4,04	3,97 $\pm$ 4,11	3,12 $\pm$ 3,92
RC105	8,33 $\pm$ 2,98	7,45 $\pm$ 2,33	8,11 $\pm$ 2,87	7,84 $\pm$ 2,73
RC108	8,24 $\pm$ 2,68	8,81 $\pm$ 1,60	8,84 $\pm$ 1,65	8,25 $\pm$ 2,69
RC201	0,78 $\pm$ 0,32	0,73 $\pm$ 0,39	0,82 $\pm$ 0,41	0,76 $\pm$ 0,33
RC202	19,01 $\pm$ 0,57	18,92 $\pm$ 0,60	18,94 $\pm$ 0,55	19,18 $\pm$ 0,62
RC206	1,36 $\pm$ 0,90	1,59 $\pm$ 0,80	1,53 $\pm$ 0,83	1,43 $\pm$ 0,91
RC208	1,22 $\pm$ 0,51	1,38 $\pm$ 0,56	1,09 $\pm$ 0,52	1,08 $\pm$ 0,60
Média	4,75 $\pm$ 7,65	4,76 $\pm$ 7,60	4,71 $\pm$ 7,61	4,65 $\pm$ 7,65

Tabela 5.11: Média do *gap* (%) por bloco de instâncias e cenário. Os valores após $\pm$ são o desvio-padrão das 30 execuções independentes de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Bloco	MONO-EG	MULTI-EG	HYBRID-EG	PSO-EG
C	0,01 $\pm$ 0,04	0,02 $\pm$ 0,05	0,02 $\pm$ 0,05	0,01 $\pm$ 0,05
R	7,56 $\pm$ 10,07	7,56 $\pm$ 10,03	7,41 $\pm$ 10,08	7,39 $\pm$ 10,10
RC	6,67 $\pm$ 6,36	6,70 $\pm$ 6,24	6,70 $\pm$ 6,24	6,55 $\pm$ 6,40
Geral	4,75 $\pm$ 7,65	4,76 $\pm$ 7,60	4,71 $\pm$ 7,61	4,65 $\pm$ 7,65

cia estatística em termos de *gap*. A mesma análise repetida por bloco confirma este resultado: nenhum par é estatisticamente significativo diferente nos blocos C ($p_{\text{Friedman}} = 0,086$), R (0,440) e RC (0,682), o que mostra que a equivalência estatística vale também nos conjuntos mais desafiadores. Ainda assim, a análise por bloco na Tabela 5.11 revela que o PSO-EG obteve consistentemente o menor *gap* médio nos blocos R e RC, que concentram as instâncias mais desafiadoras. Em termos de média global, o PSO-EG lidera (4,65%), seguido de perto pelo HYBRID-EG (4,71%), pelo

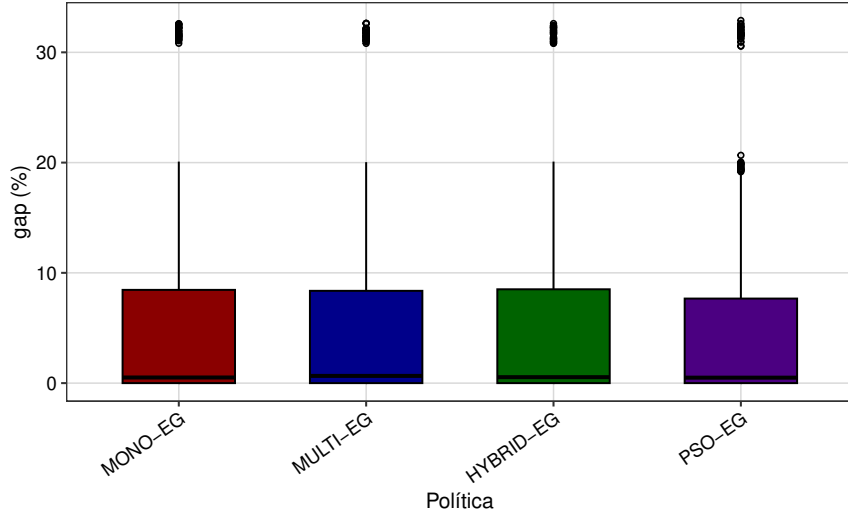


Figura 5.3: Distribuição do *gap* (%) por cenário ao longo das 24 instâncias avaliadas.

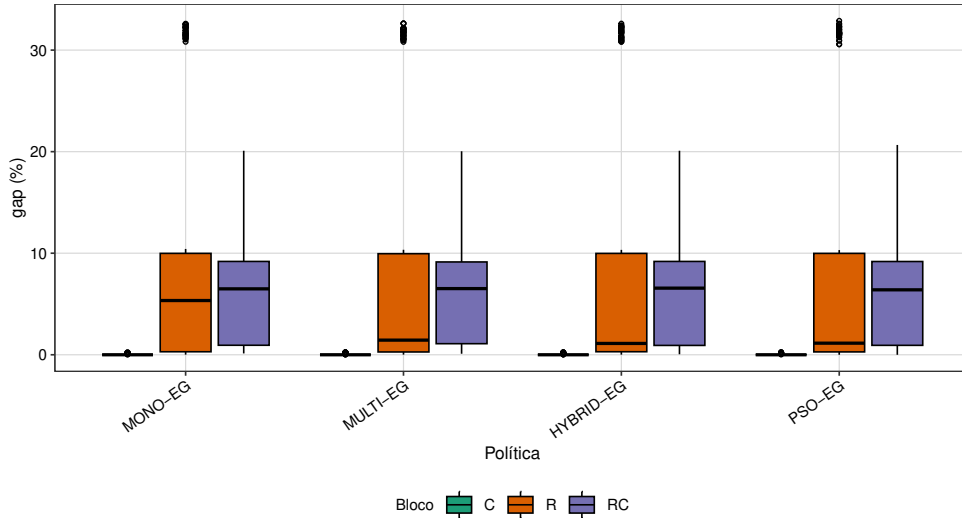


Figura 5.4: Distribuição do *gap* (%) por cenário e bloco de instâncias (C, R e RC).

MONO-EG (4,75%) e pelo MULTI-EG (4,76%).

Os valores de desvio-padrão incluídos nas Tabelas 5.10 e 5.11 ajudam a interpretar essa equivalência estatística. A dispersão entre as execuções depende fortemente do bloco de instâncias: no bloco C, ela é praticamente nula, com desvio-padrão médio de 0,02, tendo cinco instâncias (C103, C107, C205, C206 e C208) em que, nas 30 execuções, todos os cenários convergem sempre para o mesmo valor; no bloco R, o desvio-padrão médio sobe para cerca de 0,8; e, no bloco RC, o desvio-padrão médio chega a aproximadamente 1,9, com casos extremos como RC103 ($\pm 3,92$ no PSO-EG) e RC102 ($\pm 3,65$). O aspecto mais relevante é a ordem de grandeza: a diferença entre o melhor e o pior cenário na média global é de apenas 0,11 ponto percentual, enquanto o desvio-padrão médio por instância fica entre 0,83 e 0,96, quase uma ordem de grandeza maior. Isso é coerente com o resultado do teste de Friedman e indica que, para a qualidade final das soluções, a escolha entre as quatro políticas é de menor importância que a variabilidade das próprias execuções. Nas tabelas por bloco, o desvio

Tabela 5.12: Menor *gap* (%) por instância e cenário (melhor solução entre as 30 execuções). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. A linha *Vitórias* indica em quantas instâncias cada cenário atinge o menor *gap* (empates incluídos).

Instância	MONO-EG	MULTI-EG	HYBRID-EG	PSO-EG
C101	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00
C109	0,00	0,00	0,00	0,00
C204	0,00	0,00	0,00	0,00
C205	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00
R102	0,01	0,01	0,01	0,00
R104	9,78	9,79	9,77	9,80
R106	0,18	0,10	0,08	0,06
R112	9,83	9,82	9,87	9,81
R201	0,02	0,01	0,02	0,02
R208	0,03	0,03	0,03	0,03
R209	0,03	0,03	0,16	0,01
R211	30,82	30,81	30,81	30,57
RC102	6,96	6,81	7,01	6,92
RC103	0,13	0,10	0,05	0,09
RC105	6,33	6,32	6,33	6,23
RC108	0,28	0,34	0,13	0,17
RC201	0,22	0,21	0,15	0,20
RC202	18,22	18,22	18,22	18,22
RC206	0,18	0,37	0,18	0,00
RC208	0,23	0,40	0,28	0,16
Vitórias	10	12	14	18

é calculado sobre todas as execuções das instâncias do bloco e incorpora, portanto, a heterogeneidade entre instâncias, o que explica valores superiores à própria média, como $7,39 \pm 10,10$ no bloco R.

Para avaliar o potencial de qualidade de cada família, a Tabela 5.12 apresenta o menor *gap* obtido em cada instância, ou seja, a melhor solução encontrada entre as 30 execuções. A linha *Vitórias* contabiliza em quantas instâncias cada cenário atinge o menor *gap* (empates incluídos), contagem também resumida na Figura 5.5.

O PSO-EG encontrou a melhor solução no maior número de instâncias (18 das 24), seguido pelo HYBRID-EG (14), pelo MULTI-EG (12) e pelo MONO-EG (10). A diferença é ainda mais nítida nas vitórias estritas (Figura 5.5): o PSO-EG é o único mínimo em 8 instâncias, contra 4 do HYBRID-EG, 2 do MULTI-EG e nenhuma do MONO-EG. Esse resultado reforça a vantagem da abordagem PSO também na análise da melhor solução, em linha com a recomendação desta seção.

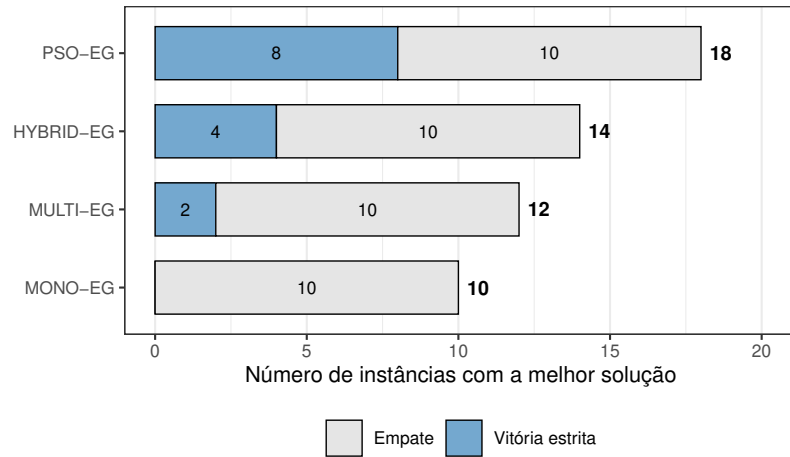


Figura 5.5: Número de instâncias em que cada cenário encontra a melhor solução (menor *gap*), separando vitórias estritas (único mínimo) de empates.

Análise do tempo de execução

A Tabela 5.13 apresenta a média do tempo de execução por instância, enquanto a Tabela 5.14 agrega esses valores por bloco. Os valores em negrito indicam o menor tempo.

As Figuras 5.6 e 5.7 apresentam a distribuição do tempo de execução. Em média global, o MONO-EG é o mais rápido (66,46s), seguido pelo MULTI-EG (67,77s), pelo PSO-EG (68,60s) e pelo HYBRID-EG (69,13s). O MONO-EG também é o mais eficiente no bloco C (81,95s); nos blocos R e RC, as diferenças entre os cenários são pequenas (inferiores a 1,5s) e estatisticamente não significativas ($p > 0,05$), com o MULTI-EG ligeiramente à frente em R e o PSO-EG em RC. O menor tempo do MONO-EG é coerente com sua formulação puramente individual, que dispensa o cálculo e a sincronização das estimativas sociais exigidos pelas demais famílias.

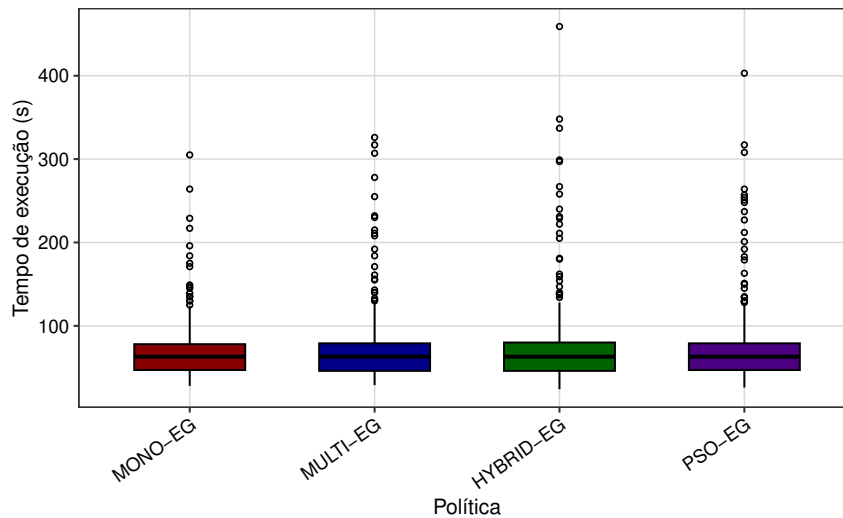


Figura 5.6: Distribuição do tempo de execução (s) por cenário ao longo das 24 instâncias avaliadas.

Tabela 5.13: Média do tempo de execução (s) por instância: comparação entre os cenários representativos. Os melhores valores são os menores e estão destacados em negrito

Instância	MONO-EG	MULTI-EG	HYBRID-EG	PSO-EG
C101	71,83	74,20	75,63	76,97
C103	51,07	50,33	49,33	51,07
C107	59,43	63,40	62,33	63,10
C109	72,00	69,50	70,17	72,03
C204	91,20	91,40	91,43	89,60
C205	127,30	137,93	126,40	138,43
C206	91,50	109,23	132,23	134,07
C208	91,30	94,33	114,00	89,03
R102	46,73	44,27	46,20	46,07
R104	41,73	41,00	43,00	44,33
R106	42,97	43,63	40,10	41,93
R112	35,20	34,33	34,20	33,83
R201	66,17	64,23	68,43	66,43
R208	110,97	109,93	110,33	108,37
R209	66,93	66,73	67,23	65,40
R211	80,50	83,33	81,87	82,70
RC102	48,43	50,97	51,00	49,23
RC103	40,90	42,30	41,23	40,13
RC105	49,73	45,10	47,50	45,17
RC108	41,83	41,17	41,60	43,67
RC201	59,40	60,00	58,67	60,13
RC202	64,47	64,73	66,30	64,53
RC206	64,43	64,93	62,40	62,30
RC208	78,97	79,47	77,47	77,97
Média	66,46	67,77	69,13	68,60

Tabela 5.14: Média do tempo de execução (s) por bloco de instâncias e cenário.

Bloco	MONO-EG	MULTI-EG	HYBRID-EG	PSO-EG
C	81,95	86,29	90,19	89,29
R	61,40	60,93	61,42	61,13
RC	56,02	56,08	55,77	55,39
Geral	66,46	67,77	69,13	68,60

Síntese e recomendação

O PSO-EG obteve o menor *gap* médio geral (4,65%), com vantagem consistente nos blocos mais difíceis (R e RC), validando o MAMaB-PSO como a abordagem de melhor qualidade entre as formulações avaliadas. Como o *gap* é estatisticamente equivalente entre as quatro famílias (Friedman, $p = 0,73$) e as diferenças de tempo também não são significativas, qualquer das demais famílias constitui uma alternativa viável em termos de desempenho. Dentre elas, indica-se o MULTI-EG como a segunda opção mais adequada. O MULTI-EG (*multi_epsilon_greedy*) representa a cooperação puramente social, mecanismo central do *framework* e linha de base natural para mensurar o ganho proporcionado pelo componente adaptativo do PSO. Além disso, sua qualidade é estatisticamente equivalente à do PSO-EG e seu tempo

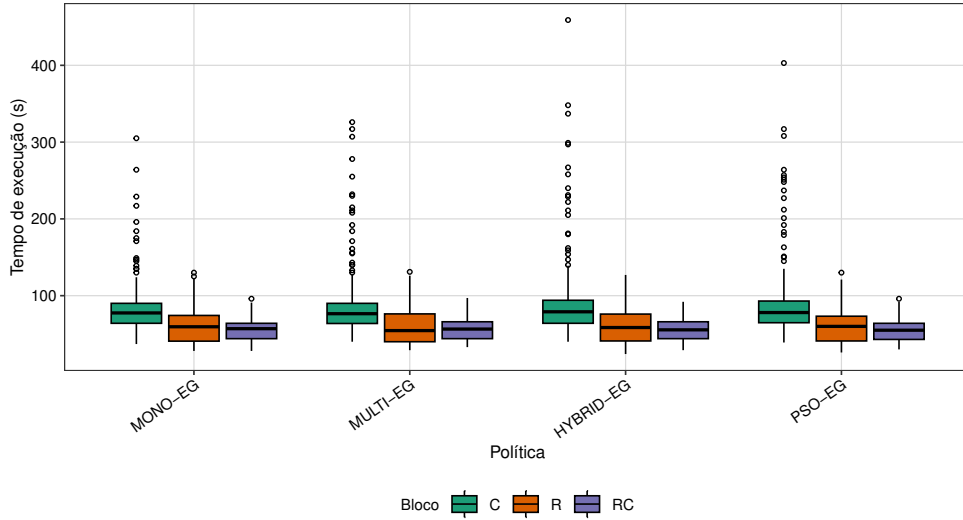


Figura 5.7: Distribuição do tempo de execução (s) por cenário e bloco de instâncias (C, R e RC).

médio (67,77s) é o segundo menor, inferior ao do próprio PSO-EG (68,60s) e ao do HYBRID-EG (69,13s). Por reunir uma formulação cooperativa simples, desempenho estatisticamente equivalente ao da melhor abordagem e baixo custo computacional, o MULTI-EG é estabelecido como a referência escalar para as análises subsequentes deste trabalho. A análise aprofundada das variantes do MAMaB-PSO é apresentada na subseção seguinte.

5.2.9 Análise das Variantes do MAMaB-PSO

Esta subseção apresenta os resultados computacionais obtidos com as seis variantes do MAMaB-PSO, comparando-as entre si e com o cenário de referência MULTI-EG (`multi_epsilon_greedy`), que representa o MaB multiagente sem o componente PSO.

5.2.9.1 Análise do *gap* percentual

A Tabela 5.15 apresenta a média do *gap* percentual obtida nas 30 execuções de cada instância para as sete variantes do MAMaB-PSO.

As instâncias do grupo C apresentaram *gaps* nulos ou próximos de zero em todas as configurações, evidenciando que esse conjunto é resolvido de forma quase ótima por qualquer política. As instâncias R211 ($gap \approx 31,7\%$) e RC202 ($\approx 19\%$) foram as mais desafiadoras. Em termos de média global, PSO-EG e PSO-DUCB empataram com o menor valor (4,65%), enquanto PSO-SWUCB registrou a maior média (4,77%). MULTI-EG e PSO-UCB (ambos 4,76%) posicionam-se em nível intermediário, seguidos por PSO-FRA (4,72%) e PSO-FRR (4,67%).

A Figura 5.8 apresenta os *boxplots* do *gap* percentual de cada configuração. Em todas as configurações, a mediana situa-se próxima de zero, indicando que a maioria das instâncias é resolvida com alta qualidade. Os *outliers* em torno de 31% correspondem à instância R211 e ocorrem em todos os métodos. A comparação global

Tabela 5.15: Média do *gap* (%) por instância e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-EG	PSO-UCB	PSO-DUCB	PSO-SWUCB	PSO-FRR	PSO-FRA
C101	0,01	0,00	0,00	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C109	0,01	0,00	0,01	0,01	0,00	0,01	0,00
C204	0,12	0,10	0,11	0,10	0,11	0,11	0,11
C205	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R102	0,59	0,94	0,77	0,93	0,77	0,42	0,77
R104	9,93	9,98	9,97	9,97	9,97	9,94	9,95
R106	6,60	4,90	4,62	4,20	6,13	4,13	5,60
R112	10,03	10,03	10,05	10,07	10,04	10,06	10,07
R201	0,23	0,25	0,21	0,26	0,25	0,21	0,29
R208	0,54	0,55	0,49	0,51	0,45	0,51	0,45
R209	0,86	0,71	0,85	0,71	0,77	0,80	0,68
R211	31,72	31,73	31,81	31,95	31,87	31,82	31,97
RC102	11,04	10,75	11,05	9,81	9,81	10,29	8,88
RC103	3,67	3,12	4,43	2,90	4,20	4,24	4,49
RC105	7,45	7,84	8,37	9,01	8,79	9,22	8,79
RC108	8,81	8,25	8,56	8,54	8,50	7,91	8,53
RC201	0,73	0,76	0,80	0,84	0,79	0,85	0,84
RC202	18,92	19,18	18,99	18,83	19,02	18,92	19,03
RC206	1,59	1,43	1,84	1,64	1,85	1,50	1,74
RC208	1,38	1,08	1,28	1,25	1,25	1,10	1,11
Média	4,76	4,65	4,76	4,65	4,77	4,67	4,72

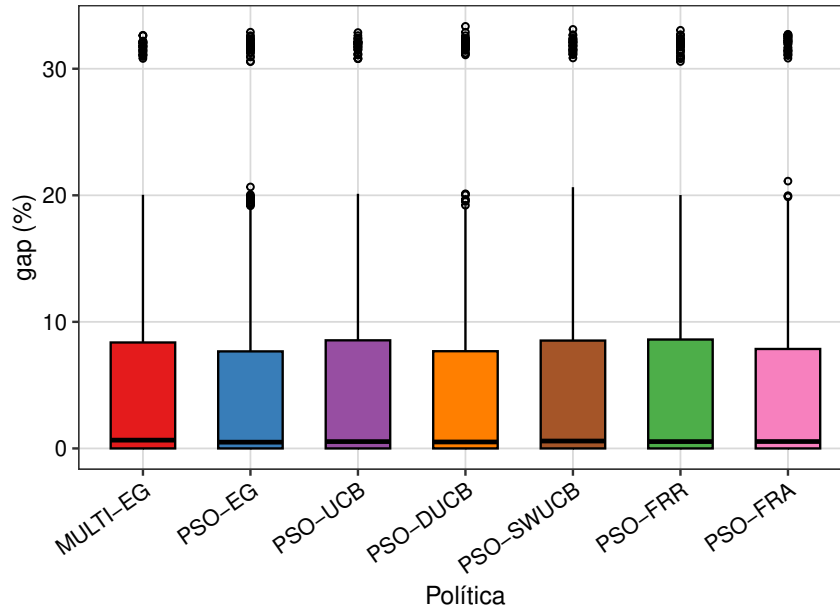


Figura 5.8: Distribuição do *gap* (%) por configuração ao longo das 24 instâncias avaliadas.

pelo teste de Friedman, pareado por instância, não revelou diferença estatisticamente significativa ($p = 0,81$), e nenhum dos 21 pares apresentou diferença no teste de

Tabela 5.16: Média do *gap* (%) por bloco de instâncias e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Bloco	MULTI-EG	PSO-EG	PSO-UCB	PSO-DUCB	PSO-SWUCB	PSO-FRR	PSO-FRA
C	0,02	0,01	0,01	0,01	0,01	0,01	0,01
R	7,56	7,39	7,35	7,33	7,53	7,24	7,47
RC	6,70	6,55	6,92	6,60	6,78	6,75	6,68
Média	4,76	4,65	4,76	4,65	4,77	4,67	4,72

Wilcoxon *signed-rank* com correção de Bonferroni, confirmando que todas as políticas MaB propostas produzem soluções de qualidade estatisticamente equivalente. A análise por bloco reproduz o resultado, sem diferença significativa nos blocos C ($p = 0,220$), R (0,512) e RC (0,350).

A Tabela 5.16 confirma o padrão observado por bloco de instâncias. O bloco C é resolvido de forma quase ótima por todos os métodos. No bloco R, PSO-FRR (7,24%) obteve o menor *gap* médio, seguido por PSO-DUCB (7,33%) e PSO-UCB (7,35%). No bloco RC, PSO-EG destacou-se com a menor média (6,55%). Nenhuma dessas diferenças é estatisticamente significativa, conforme os p -valores do teste de Friedman por bloco já reportados: as médias apenas ordenam descritivamente as configurações dentro de cada grupo. A Figura 5.9 apresenta a distribuição do *gap*(%) por configuração e por bloco de instância. Observe que o bloco C alcança *gap*(%) próximo a zero para a maioria absoluta das execuções em todas as configurações.

As análises anteriores avaliaram o desempenho médio das configurações. Para examinar o potencial de qualidade de cada uma, a Tabela 5.17 apresenta o menor *gap* obtido em cada instância, ou seja, a melhor solução encontrada entre as 30 execuções. A linha *Vitórias* contabiliza em quantas instâncias cada configuração atinge o menor *gap* (empates incluídos), contagem também resumida na Figura 5.10.

As instâncias do grupo C, além de R208 e RC202, são resolvidas com qualidade

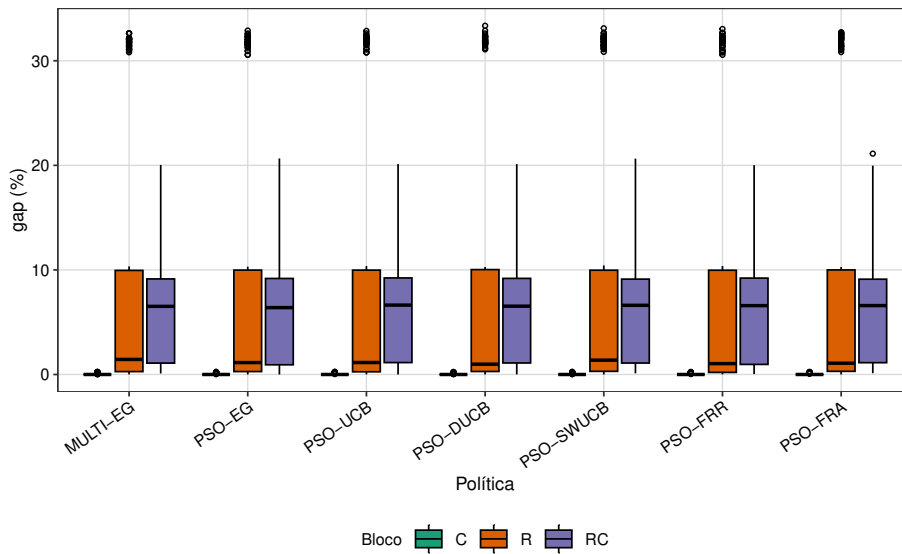


Figura 5.9: Distribuição do *gap*(%) por configuração e bloco de instâncias (C, R e RC).

Tabela 5.17: Menor *gap* (%) por instância e configuração (melhor solução entre as 30 execuções). A linha *Vitórias* indica em quantas instâncias cada configuração atinge o menor *gap* (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-EG	PSO-UCB	PSO-DUCB	PSO-SWUCB	PSO-FRR	PSO-FRA
C101	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C109	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C204	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C205	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R102	0,01	0,00	0,01	0,00	0,00	0,01	0,00
R104	9,79	9,80	9,73	9,79	9,78	9,73	9,73
R106	0,10	0,06	0,07	0,09	0,15	0,04	0,11
R112	9,82	9,81	9,84	9,92	9,80	9,86	9,84
R201	0,01	0,02	0,02	0,02	0,02	0,02	0,02
R208	0,03	0,03	0,03	0,03	0,03	0,03	0,11
R209	0,03	0,01	0,24	0,18	0,03	0,21	0,16
R211	30,81	30,57	30,79	31,09	30,85	30,57	30,83
RC102	6,81	6,92	6,99	6,96	6,84	6,81	6,83
RC103	0,10	0,09	0,09	0,12	0,11	0,07	0,12
RC105	6,32	6,23	6,32	6,31	6,39	6,24	6,24
RC108	0,34	0,17	0,17	0,34	0,26	0,05	0,23
RC201	0,21	0,20	0,00	0,10	0,15	0,34	0,12
RC202	18,22	18,22	18,22	18,22	18,22	18,22	18,22
RC206	0,37	0,00	0,24	0,18	0,22	0,18	0,24
RC208	0,40	0,16	0,27	0,01	0,12	0,04	0,15
Vitórias	12	15	12	12	12	16	11

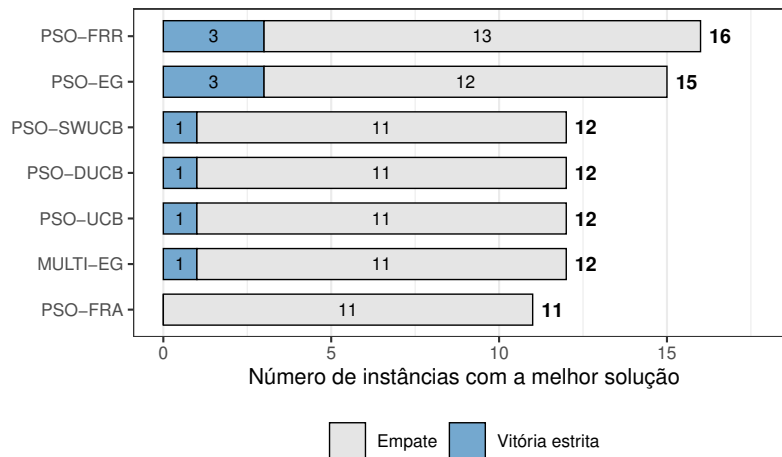


Figura 5.10: Número de instâncias em que cada configuração encontra a melhor solução (menor *gap*), separando vitórias estritas (único mínimo) de empates.

idêntica por todas as configurações, o que eleva por empate a contagem de vitórias de todos os métodos. Nas instâncias mais sensíveis, o PSO-FRR encontrou a melhor solução no maior número de instâncias (16 das 24), seguido pelo PSO-EG (15). Esse padrão indica que, no melhor caso, a aleatoriedade do ϵ -greedy e a atribuição de

crédito por janela do FRRMaB favorecem a obtenção de soluções de alta qualidade, ainda que as diferenças entre todas as configurações permaneçam pequenas.

5.2.9.2 Análise do tempo de execução

A Tabela 5.18 apresenta a média do tempo de execução de cada configuração. Em contraste com o *gap*, o tempo de execução revela diferenças expressivas entre as políticas.

PSO-UCB destacou-se como a configuração mais eficiente, com média global de 57,54s e menor tempo em 13 das 24 instâncias. PSO-DUCB foi o segundo método mais rápido (58,01s). As variantes MULTI-EG (67,77s) e PSO-EG (68,60s) e as políticas FRR/FRA/SWUCB (≈ 65 s) apresentaram médias intermediárias. As maiores discrepâncias ocorreram nas instâncias C205 e C206, em que PSO-UCB concluiu em ≈ 50 –58s contra ≈ 110 –138s do MULTI-EG e PSO-EG.

A Figura 5.11 apresenta os *boxplots* do tempo de execução de cada configuração, reunindo as execuções das 24 instâncias. A leitura gráfica acrescenta às médias uma informação que elas não revelam: o que separa o MULTI-EG e o PSO-EG das demais políticas não é a posição da caixa, e sim a cauda superior, com *outliers* que ultrapassam 300s. O PSO-UCB e o PSO-DUCB apresentam as caixas mais baixas e compactas, com valores extremos contidos em torno de 120s, enquanto as variantes com janela deslizante (PSO-SWUCB, PSO-FRR e PSO-FRA) ocupam posição intermediária, porém já com dispersão reduzida. A vantagem das políticas baseadas em UCB está tanto na redução do tempo médio quanto na maior previsibilidade do custo computacional.

Tabela 5.18: Média do tempo de execução (s) por instância e configuração.

Instância	MULTI-EG	PSO-EG	PSO-UCB	PSO-DUCB	PSO-SWUCB	PSO-FRR	PSO-FRA
C101	74,20	76,97	54,90	56,23	58,17	58,47	57,90
C103	50,33	51,07	44,10	45,67	53,03	53,33	53,43
C107	63,40	63,10	47,70	49,27	56,10	56,70	55,60
C109	69,50	72,03	62,57	62,87	74,33	76,23	74,63
C204	91,40	89,60	91,67	89,90	112,40	115,10	113,83
C205	137,93	138,43	50,13	50,70	62,33	64,30	63,90
C206	109,23	134,07	57,67	59,00	75,27	76,80	74,73
C208	94,33	89,03	62,43	62,63	79,73	81,00	79,47
R102	44,27	46,07	46,70	45,83	46,70	42,80	46,40
R104	41,00	44,33	42,63	42,37	48,10	47,27	48,80
R106	43,63	41,93	41,37	40,80	45,10	41,60	44,40
R112	34,33	33,83	34,60	33,93	39,50	38,60	40,23
R201	64,23	66,43	60,83	61,07	66,93	68,73	67,17
R208	109,93	108,37	109,43	110,70	117,20	117,00	118,20
R209	66,73	65,40	66,63	64,87	70,97	70,17	67,77
R211	83,33	82,70	80,27	83,77	83,97	85,17	85,67
RC102	50,97	49,23	48,80	47,07	52,43	51,67	48,93
RC103	42,30	40,13	40,83	41,63	45,10	44,67	44,30
RC105	45,10	45,17	45,07	45,20	51,27	52,20	49,37
RC108	41,17	43,67	42,23	42,50	49,43	50,53	48,50
RC201	60,00	60,13	52,87	54,27	59,50	60,70	60,10
RC202	64,73	64,53	63,37	64,23	65,63	67,33	66,00
RC206	64,93	62,30	59,43	59,23	66,27	68,83	68,07
RC208	79,47	77,97	74,73	78,47	81,53	81,97	81,23
Média	67,77	68,60	57,54	58,01	65,04	65,47	64,94

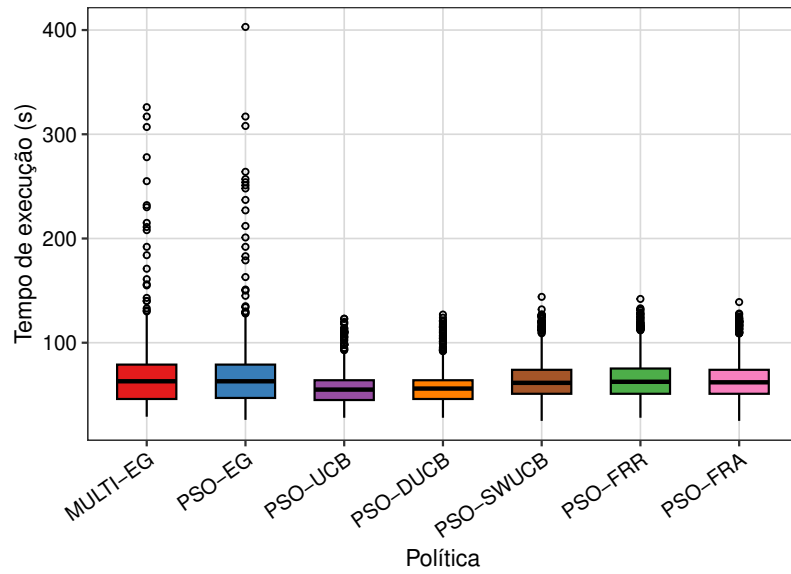


Figura 5.11: Distribuição do tempo de execução por configuração ao longo das 24 instâncias avaliadas.

Tabela 5.19: Média do tempo de execução (s) por bloco de instâncias e configuração.

Bloco	MULTI-EG	PSO-EG	PSO-UCB	PSO-DUCB	PSO-SWUCB	PSO-FRR	PSO-FRA
C	86,29	89,29	58,90	59,53	71,42	72,74	71,69
R	60,93	61,13	60,31	60,42	64,81	63,92	64,83
RC	56,08	55,39	53,42	54,08	58,90	59,74	58,31
Média	67,77	68,60	57,54	58,01	65,04	65,47	64,94

A Tabela 5.19 revela o padrão de eficiência computacional por bloco. No bloco C, PSO-UCB é substancialmente mais rápido que as demais configurações (58,90 s), com economia de aproximadamente 27 s em relação ao MULTI-EG (86,29 s) e PSO-EG (89,29 s). Nos blocos R e RC, as diferenças são menos pronunciadas: PSO-UCB mantém ligeira vantagem (60,31 s em R e 53,42 s em RC), seguido por PSO-DUCB, enquanto MULTI-EG apresenta tempo comparável em R (60,93 s) mas inferior em RC (56,08 s).

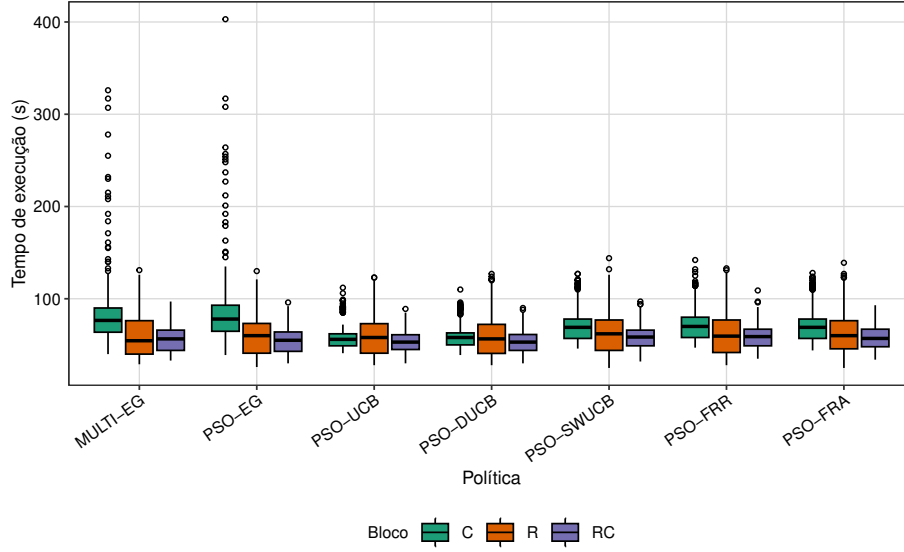


Figura 5.12: Distribuição do tempo de execução (s) por configuração e bloco de instâncias (C, R e RC).

A Figura 5.12 desdobra essa distribuição pelos três blocos de instâncias e localiza a origem das caudas observadas na Figura 5.11. Os valores extremos do MULTI-EG e do PSO-EG concentram-se quase integralmente no bloco C, cujas caixas são visivelmente mais altas que as dos blocos R e RC nessas duas configurações. Nas demais políticas, o bloco C deixa de ser o mais custoso e as três distribuições tornam-se comparáveis entre si. Isso explica graficamente por que a economia de tempo do PSO-UCB registrada na Tabela 5.19 é expressiva no bloco C e apenas marginal em R e RC.

O teste de Friedman para o tempo de execução, pareado por instância, revelou diferença altamente significativa entre os métodos ($p = 3,46 \times 10^{-13}$), e o teste de Wilcoxon *signed-rank* com correção de Bonferroni apontou 9 dos 21 pares como significativos. Por bloco, o teste de Friedman também acusa diferença nos três grupos, tendo p-valores $p = 7,9 \times 10^{-5}$ para o bloco C; $8,1 \times 10^{-5}$ para o bloco R; e $3,0 \times 10^{-6}$ para o bloco RC. Porém, nenhum par isolado sobrevive à correção de Bonferroni dentro de um bloco: com apenas 8 instâncias por grupo, o *post-hoc* não tem potência suficiente para as 21 comparações, ainda que o efeito global seja claramente detectado. O *post-hoc* de Wilcoxon com correção de Bonferroni identificou dois grupos homogêneos: um *cluster rápido* (PSO-UCB e PSO-DUCB), com todas as comparações entre grupos sendo significativas ($p < 10^{-8}$), e um *cluster intermediário* (MULTI-EG, PSO-EG, PSO-SWUCB, PSO-FRR e PSO-FRA).

Embora as médias sugiram uma leve tendência de menor tempo nas variantes baseadas em janela deslizante (PSO-SWUCB, PSO-FRR e PSO-FRA, com ≈ 65 s) frente ao MULTI-EG e ao PSO-EG (≈ 68 s), essa diferença é apenas descritiva e não encontra respaldo estatístico. Dentro do *cluster intermediário*, todas as comparações par a par foram não significativas ($p \approx 1,000$ após a correção de Bonferroni). Por esse motivo, não é possível subdividir esse grupo, mantendo-se a separação em apenas dois *clusters*, e não em três (rápido, mediano e mais lento), justamente pela ausência de diferença estatística entre os métodos de tempo intermediário.

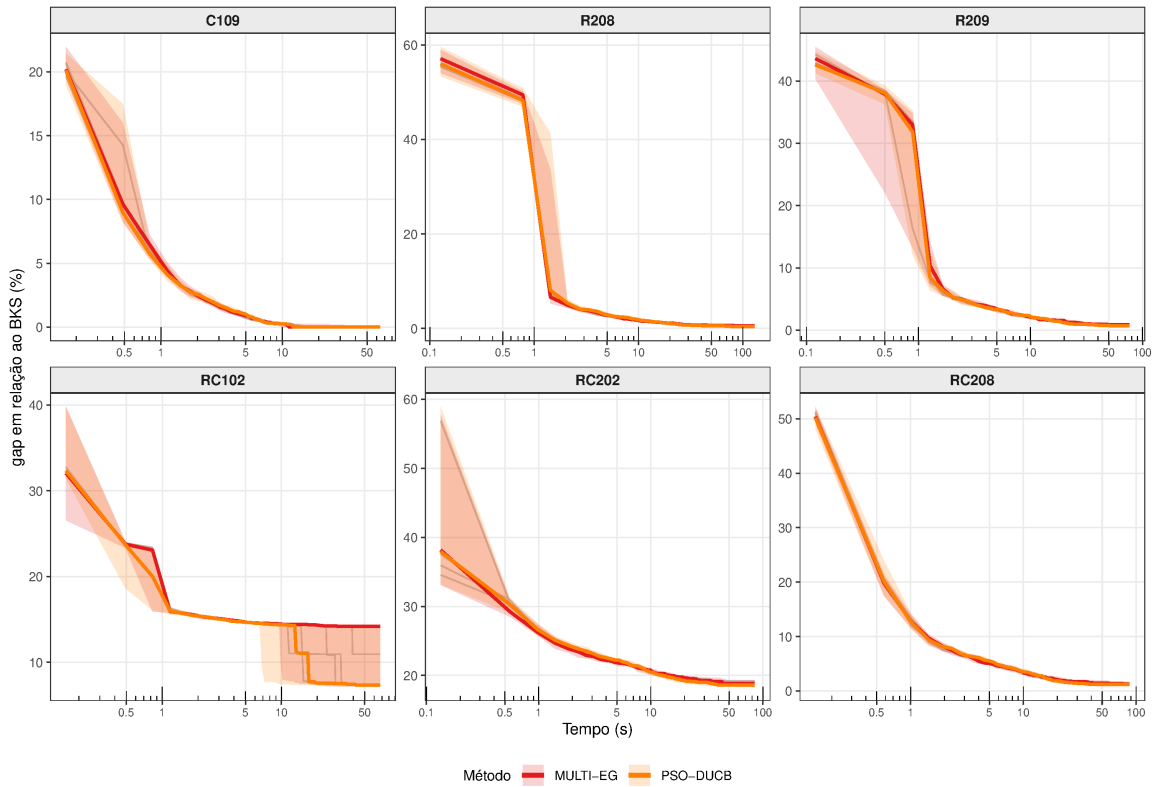


Figura 5.13: Evolução temporal do *gap* em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem ao MULTI-EG (vermelho) e ao PSO-DUCB (laranja); as demais variantes PSO são mostradas em cinza. As faixas sombreadas indicam o intervalo interquartilício (IQR) entre as 30 execuções independentes. O eixo x está em escala logarítmica.

5.2.9.3 Análise da evolução da qualidade das soluções

A Figura 5.13 apresenta a evolução do *gap* para seis instâncias representativas, selecionadas de modo a cobrir as três famílias de instâncias de Solomon (C, R e RC) e evidenciar os cenários de maior contraste entre os métodos avaliados. As curvas destacadas correspondem ao MULTI-EG (referência sem PSO) e ao PSO-DUCB (variante com desconto temporal); as demais variantes PSO são exibidas em cinza para contexto. As faixas sombreadas indicam o intervalo interquartilício (IQR, percentis 25–75) entre as 30 execuções independentes, e o eixo horizontal está em escala logarítmica.

Na instância C109 (família C), ambos os métodos convergem rapidamente para *gap* igual a zero ainda no primeiro segundo de execução, com variância praticamente nula, confirmando que todas as variantes são capazes de igualar o melhor valor conhecido nesse grupo.

Nas instâncias da família R, o PSO-DUCB apresenta vantagem consistente. Em R208, a mediana do PSO-DUCB atinge 0,36% contra 0,54% do MULTI-EG, uma redução de 33% no *gap* final, mantida ao longo de toda a execução. Em R209, o PSO-DUCB obtém mediana de 0,71% versus 0,89% do MULTI-EG (redução de 20%), com IQR inferior (0,41% contra 0,50%), indicando convergência mais robusta. Esses resultados evidenciam que o fator de desconto do DUCB permite ao algoritmo

adaptar-se à não estacionariedade do *Pool*, reavaliando com maior peso as soluções coletadas mais recentemente.

O contraste mais expressivo ocorre na instância RC102 (família RC1xx). O MULTI-EG apresenta mediana de 14,2% de *gap*, enquanto o PSO-DUCB reduz esse valor para 7,33%, uma melhoria de aproximadamente 48%. A análise das trajetórias individuais revela que o MULTI-EG frequentemente estagna em soluções de qualidade inferior ao longo da execução, enquanto o PSO-DUCB, ao ponderar o histórico recente de recompensas, consegue redirecionar os agentes para soluções do *Pool* que ainda apresentam potencial de melhoria. Em RC208, o PSO-DUCB também supera o MULTI-EG (mediana 1,21% contra 1,33%, com q_{25} de 0,76% contra 1,07%), indicando que a variante DUCB converge para soluções de alta qualidade em mais execuções. Por fim, em RC202 (subfamília RC2xx), ambos os métodos estabilizam em torno de 19% de *gap*, com PSO-DUCB apresentando IQR ligeiramente menor (0,75% contra 0,94%).

Em síntese, o PSO-DUCB demonstra superioridade consistente sobre o MULTI-EG nas instâncias de dificuldade intermediária das famílias R e RC, com ganhos que variam de 20% a 48% na qualidade mediana das soluções finais. Nas instâncias de fácil resolução (família C), ambos os métodos atingem o ótimo. Esses resultados confirmam que a incorporação do mecanismo de desconto temporal do PSO ao processo de seleção de soluções no *Pool* melhora a qualidade e a robustez das soluções obtidas pelo *Framework* AMAM.

5.2.9.4 Perfil de desempenho comparativo

A Figura 5.14 apresenta o perfil de desempenho proposto por Dolan e Moré (2002), que permite comparar múltiplos algoritmos em múltiplas instâncias em um único gráfico. Para cada par (método m , instância i), é calculada a razão de desempenho $r(m, i)$ em relação ao melhor resultado obtido por qualquer método naquela instância. O perfil $\rho_m(\tau)$ fornece a fração das instâncias em que o método m possui razão $r(m, i) \leq \tau$: curvas mais altas e à esquerda indicam desempenho superior.

A análise do perfil revela um panorama de hierarquia entre os métodos. Em $\tau = 1,01$, limiar que exige estar praticamente empatado com o melhor resultado, o PSO-DUCB lidera com 70,8% das instâncias, seguido pelo PSO-FRR (66,7%) e pelos demais (58,3%). Em $\tau = 1,05$, PSO-EG e PSO-DUCB empatam na liderança com 79,2%, seguidos pelo MULTI-EG e PSO-FRR (75,0% cada). Em $\tau = 1,10$, o PSO-EG destaca-se com 87,5% das instâncias, indicando que, quando se tolera uma pequena margem de 10% acima do melhor, essa variante cobre o maior número de instâncias.

Em relação ao limiar τ necessário para cobrir 90% das instâncias, o PSO-DUCB apresenta o menor valor ($\tau = 1,223$), evidenciando maior robustez geral entre os métodos avaliados. O PSO-SWUCB é o segundo mais robusto ($\tau = 1,232$), seguido pelo PSO-EG e PSO-FRR ($\tau = 1,264$). O MULTI-EG requer $\tau = 1,290$ para atingir a mesma cobertura, e o PSO-UCB apresenta o maior valor ($\tau = 1,346$), indicando instâncias em que seu desempenho se afasta mais substancialmente do melhor método. Em termos de vitórias diretas, PSO-EG, PSO-DUCB e PSO-FRR lideram com 11 instâncias em que produzem o melhor resultado entre os sete métodos, contra 10 do MULTI-EG e 8 do PSO-UCB.

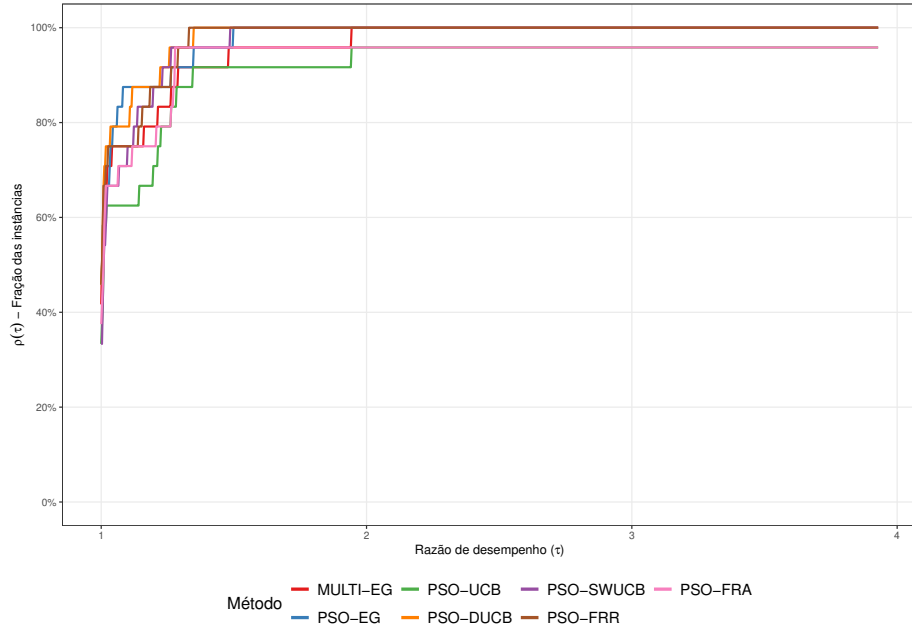


Figura 5.14: Perfil de desempenho (Dolan e Moré, 2002) dos sete métodos avaliados nas 24 instâncias de Solomon. O eixo x representa a razão de desempenho τ em relação ao melhor método em cada instância; o eixo y indica a fração das instâncias em que o método possui razão $\leq \tau$.

Em síntese, os resultados confirmam que a incorporação dos mecanismos PSO ao MaB multiagente não compromete a qualidade das soluções (equivalência estatística em $p = 0,998$), e produz ganhos mensuráveis na qualidade final nas variantes que incorporam desconto temporal (PSO-DUCB) ou janela deslizante (PSO-SWUCB). O PSO-DUCB configura-se como a variante mais robusta do experimento, combinando liderança em $\tau = 1,01$ com o menor limiar para atingir 90% de cobertura. Esses resultados respondem ao objetivo específico (vii) da Seção 1.3.2, demonstrando a viabilidade e a eficácia de técnicas de aprendizado por reforço multiagente incorporadas ao *Framework* AMAM.

5.2.10 Considerações Parciais

Os resultados desta seção evidenciam que a extensão multiagente do *Multi-armed Bandit* por meio do algoritmo MAMaB-PSO representa um avanço significativo em relação à abordagem individual da Seção 5.1. A validação da abordagem PSO demonstrou que a ponderação adaptativa entre os componentes cognitivo e social, por meio do coeficiente dinâmico $\theta(t)$, produz resultados superiores às abordagens puramente individuais (MONO-EG) e puramente sociais (MULTI-EG): o PSO-EG liderou com 18 vitórias diretas nas 24 instâncias, confirmando que o equilíbrio dinâmico entre aprendizado individual e cooperação coletiva favorece a qualidade das soluções.

A análise das variantes revelou dois grupos de comportamento distintos: um agrupamento rápido, formado por PSO-UCB e PSO-DUCB, com redução de aproximadamente 15% no tempo médio em relação ao MULTI-EG, e um agrupamento intermediário, estatisticamente equivalente em tempo, composto pelas demais variantes. Em termos de qualidade, o PSO-DUCB destacou-se como a variante mais robusta, com ganhos de

20% a 48% na qualidade mediana das instâncias das famílias R e RC em relação ao MULTI-EG, ao mesmo tempo em que integra o agrupamento de menor tempo de execução. Esses resultados confirmam que a incorporação de mecanismos de desconto temporal ao *Pool* de soluções não estacionário é especialmente benéfica, permitindo ao sistema redirecionar os agentes para soluções mais recentes e potencialmente mais promissoras, e estabelecem o PSO-DUCB como referência para as extensões multiobjetivo apresentadas no Capítulo 6.

Capítulo 6

Aprendizado por Reforço Multiobjetivo no *Framework* AMAM

Este capítulo apresenta a extensão de aprendizado por reforço multiobjetivo das contribuições incorporadas ao *Framework* AMAM, desenvolvida em dois níveis complementares. O capítulo está organizado da seguinte forma: a Seção 6.1 propõe uma extensão multiobjetivo do mecanismo de aprendizado por reforço aplicado ao *Pool* de cooperação, substituindo as recompensas escalares por vetores de recompensa e adaptando as políticas de seleção do *Multi-armed Bandit* para operar sobre estimativas vetoriais de qualidade segundo critérios de dominância de Pareto; a Seção 6.2 complementa essa proposta, com a reestruturação do próprio *Pool* de soluções com base em dominância de Pareto e *crowding distance*, oferecendo suporte estrutural ao aprendizado multiobjetivo e garantindo que o conjunto de soluções compartilhadas seja mantido, segundo critérios simultâneos de qualidade e diversidade.

6.1 *Multi-armed Bandit* Multiobjetivo

Nas Seções 5.1 e 5.2 o aprendizado sobre a coleta de soluções no *Pool* de cooperação é escalar: cada solução mantém um único valor q , atualizado por uma recompensa numérica que sintetiza, em um único número, toda a contribuição da solução ao processo de busca. Esse modelo unidimensional, no entanto, limita o aproveitamento das informações disponíveis a cada interação cooperativa, pois não distingue se uma solução contribuiu para a melhoria da função objetivo, para a diversidade estrutural do sistema ou para a redução do número de rotas.

Esta seção propõe uma extensão multiobjetivo do mecanismo de aprendizado por reforço aplicado ao *Pool*. A coleta de soluções é modelada como um problema de decisão multiobjetivo em um cenário *single-state*, no qual o estado é único (a cooperação ocorre sobre o mesmo contexto compartilhado pelos agentes), cada solução armazenada corresponde a uma ação (ou braço) disponível e a recompensa observada após sua utilização passa a ser vetorial. A decisão de coleta passa, assim, a considerar a dominância de Pareto entre vetores de qualidade associados às soluções. A Seção 6.2 complementa esta proposta com a reestruturação do próprio *Pool* segundo critérios multiobjetivo de manutenção, baseados em dominância de Pareto e *crowding distance*, oferecendo suporte estrutural ao aprendizado aqui proposto. Dessa forma, o problema

objeto de solução (no caso, para fins de teste, instâncias do PRVJT) permanece sendo resolvido pelos métodos locais dos agentes na forma originalmente prevista, mas a dimensão social do *Framework* AMAM passa a aprender em função de múltiplos critérios simultaneamente.

6.1.1 Vetor de recompensa social

Ao longo desta seção, a notação “ a ” designa uma solução do *Pool*, que é o braço do *Multi-armed Bandit* nesta instanciação: coletar a solução a é a ação do agente, e $\mathcal{K}(t)$ é o conjunto das soluções disponíveis no *Pool* no instante t , ou seja, o conjunto de braços. A convenção é, portanto, a mesma adotada na Seção 5.2.

Considere um vetor de recompensa social $\mathbf{r}(a)$, associado a cada utilização da solução a do *Pool*, composto por três componentes, na forma:

$$\mathbf{r}(a) = [r_{fo} \ r_{shi} \ r_{rotas}], \quad (6.1)$$

em que r_{fo} representa a melhora normalizada da função objetivo; r_{shi} a melhora associada ao critério de similaridade estrutural; e r_{rotas} é a melhora associada ao número de rotas da solução resultante. A inclusão do número de rotas como terceiro critério reconhece sua relevância prática no PRVJT, pois soluções com menor quantidade de rotas tendem a ser preferíveis em aplicações reais, ainda que apresentem valor de função objetivo levemente superior.

Para a função objetivo, em problemas de minimização, a melhora é calculada pela diferença relativa entre o valor anterior à coleta e o valor obtido após a aplicação da busca local sobre a solução coletada, na forma:

$$r_{fo} = \max \left\{ 0, \frac{f_{anterior} - f_{atual}}{\max(|f_{anterior}|, 1)} \right\}. \quad (6.2)$$

O mesmo princípio é adotado para o número de rotas e para o termo de diversidade, ambos tratados como critérios de minimização, ou seja:

$$r_{rotas} = \max \left\{ 0, \frac{K_{anterior} - K_{atual}}{\max(K_{anterior}, 1)} \right\}, \quad (6.3)$$

$$r_{shi} = \max \left\{ 0, \frac{shi_{anterior} - shi_{atual}}{\max(shi_{anterior}, 1)} \right\}, \quad (6.4)$$

em que K é o número de rotas e shi é a medida de similaridade estrutural da solução em relação ao conjunto do *Pool*. A medida segue o princípio da função de compartilhamento da Expressão (2.1) da Seção 2.1.1, com duas adaptações: agrega-se pelo *máximo*, e não pelo somatório, e a normalização usa uma constante fixa. Formalmente:

$$shi(a) = \max_{b \in P, b \neq a} \sigma(a, b), \quad \sigma(a, b) = \begin{cases} 0, & \text{se } \lambda_{ab} \leq pr \\ 1 - \frac{\lambda_{ab}}{100}, & \text{caso contrário} \end{cases} \quad (6.5)$$

em que λ_{ab} é a distância entre as soluções a e b , medida pelo número de arcos distintos, e pr é o raio de nicho. O valor de shi é tratado como critério de *minimização*

na relação de dominância, assim como a função objetivo e o número de rotas. Os subscritos *anterior* e *atual* referem-se, respectivamente, ao valor associado à solução coletada do *Pool* e ao valor estimado para a solução obtida após a busca local. A recompensa é positiva apenas quando a coleta reduz a similaridade, e cresce proporcionalmente a essa redução. Todos os três componentes do vetor de recompensas dado pela Expressão (6.1) são limitados ao intervalo $[0,1]$, o que mantém escalas comparáveis entre objetivos e preserva as condições normalmente assumidas pelos algoritmos multiobjetivo baseados no procedimento *Multi-armed Bandit*.

6.1.2 Estrutura vetorial associada às soluções

Cada solução $a \in \mathcal{K}(t)$ do *Pool* é estendida para armazenar, além das informações estruturais *fo* e *shi* utilizadas na manutenção descrita na Seção 6.2, dados específicos do aprendizado multiobjetivo: a soma acumulada das recompensas vetoriais observadas, um vetor de valores acumulados $\mathbf{Q}(a)$, o histórico de recompensas vetoriais e o número de interações $N(a)$ associadas à solução. Com isso, cada solução passa a funcionar como um braço com memória vetorial própria, permitindo o cálculo de médias vetoriais, estimativas descontadas e atualizações *stateless* do tipo *Q-learning* sobre vetores de qualidade.

A seleção da próxima solução a ser coletada opera em duas etapas. Primeiro, constrói-se o conjunto não dominado $A_0 \subseteq \mathcal{K}(t)$ dentre as soluções candidatas, definido como:

$$A_0 = \{ a \in \mathcal{K}(t) \mid \nexists b \in \mathcal{K}(t) \text{ tal que } \mathbf{S}(b) \succ \mathbf{S}(a) \}, \quad (6.6)$$

em que $\mathbf{S}(a)$ é o vetor de escore atribuído à solução a pela política em uso e o símbolo $\succ$ representa a relação de dominância estrita de Pareto. Note que A_0 é um conjunto de soluções candidatas, equivalentemente de ações de coleta, e não de vetores de escore: o vetor $\mathbf{S}(\cdot)$ comparece apenas no critério que decide quais soluções candidatas entram no conjunto. Em seguida, a solução efetivamente coletada é escolhida de forma aleatória uniforme em A_0 . Essa estratégia de seleção, alinhada às formulações clássicas de *Pareto-UCB* (Drugan e Nowé, 2013) e do *Pareto Q-learning* (Van Moffaert e Nowé, 2014), evita a necessidade de escalarização explícita e preserva a multiplicidade de compromissos entre objetivos durante o processo de aprendizagem.

Cabe explicitar a correspondência com a formulação geral apresentada na Seção 2.3.6, da qual esta é uma instanciiação. O conjunto A_0 da Expressão (6.6) é o conjunto $\mathcal{K}^*(t)$ de braços não dominados definido na Seção 2.3.6, e o vetor de escore $\mathbf{S}(a)$ desempenha o papel do vetor de índices $\mathbf{U}(a,t)$ da Expressão (2.19). A média vetorial $\bar{\mathbf{r}}(a)$, apresentada adiante, é a estimativa $\bar{\mathbf{Q}}(a,t)$ da Expressão (2.17). A segunda etapa, ou seja, a escolha aleatória uniforme dentro do conjunto não dominado, é idêntica nas duas formulações.

Quatro adaptações distinguem a instanciiação da formulação geral, todas decorrentes das particularidades do *Pool*:

- (i) a dimensão do vetor de recompensa é fixada em $d = 3$, correspondendo aos critérios de função objetivo, similaridade estrutural e número de rotas da Expressão (6.1);
- (ii) o conjunto de braços não é fixo. Enquanto na Seção 2.3.6 considera-se um

conjunto $\mathcal{K}$ de K braços constante ao longo do horizonte, aqui o conjunto $\mathcal{K}(t)$ é o conteúdo do *Pool* no instante t , que muda a cada inserção e a cada remoção;

(iii) em consequência, o bônus de diversificação não pode reproduzir o termo

$$\sqrt{2 \ln(t \cdot d\sqrt[4]{K}) / N(a,t)}$$

do *Pareto-UCB1*, cujo fator $d\sqrt[4]{K}$ pressupõe K constante e é o que sustenta o limite logarítmico de arrependimento, demonstrado por Drugan e Nowé (2013). Adota-se, em seu lugar, o bônus da Expressão (6.8), com constante c calibrada empiricamente, e o total de coletas N_{tot} no papel do índice t , uma vez que vários agentes coletam em paralelo e não há uma rodada única por instante;

(iv) a formulação geral pressupõe braços estacionários, com vetores de médias fixos. O *Pool* é não estacionário pelas razões descritas na Seção 5.2, o que motiva as variantes com desconto, com janela deslizante e com atualização do tipo *Q-learning* descritas a seguir.

Essas adaptações afastam a instanciação das garantias teóricas do *Pareto-UCB1*, que valem para o cenário estacionário com conjunto fixo de braços. A avaliação da abordagem é, por isso, empírica, conforme a Seção 6.1.6.

6.1.3 Variantes da Política de Seleção Multiobjetivo

A política π de seleção sobre o *Pool* multiobjetivo é mantida como um componente genérico: o que difere entre as variantes é a forma de calcular o vetor de escore $\mathbf{S}(a)$ atribuído a cada solução candidata. São propostas quatro variantes, identificadas pelo prefixo MO (Multiobjetivo) e descritas a seguir.

MO-PSO-UCB

Estende o *Upper Confidence Bound* clássico, descrito na Seção 2.3.2.1, para o caso vetorial. A média vetorial das recompensas observadas para a solução a é combinada com um bônus escalar de diversificação, somado a todos os componentes do vetor:

$$\mathbf{S}_{ucb}(a) = \bar{\mathbf{r}}(a) + b(a) \mathbf{1} \quad (6.7)$$

$$b(a) = c \sqrt{\frac{2 \ln N_{\text{tot}}}{N(a)}} \quad (6.8)$$

em que

$$\bar{\mathbf{r}}(a) = \frac{1}{N(a)} \sum_{m=1}^{N(a)} \mathbf{r}_m(a) \quad (6.9)$$

é a média vetorial das recompensas observadas, $b(a)$ é o bônus escalar de diversificação, $N_{\text{tot}} = \sum_a N(a)$ é o número total de coletas, c é a constante de diversificação e $\mathbf{1}$ é o vetor unitário em $\mathbb{R}^3$. Soluções ainda não exploradas recebem vetor com componentes infinitos, garantindo diversificação inicial.

MO-PSO-DPUCB

Adapta o *Discounted UCB*, descrito na Seção 2.3.2.1, ao caso multiobjetivo. O escore é construído a partir de um histórico ponderado por um fator de desconto $\gamma \in (0,1]$, atribuindo maior peso às observações recentes:

$$\bar{\mathbf{r}}_\gamma(a) = \frac{\sum_{m=1}^{N(a)} \gamma^{t-t_m} \mathbf{r}_m(a)}{\sum_{m=1}^{N(a)} \gamma^{t-t_m}}, \quad (6.10)$$

em que t_m é o instante em que a m -ésima recompensa associada à solução a foi observada. O termo de diversificação segue o mesmo princípio do MO-PSO-UCB, mas aplicado sobre a contagem ponderada da solução. Essa variante é particularmente adequada para ambientes em que a relevância relativa das soluções do *Pool* se altera ao longo da execução.

MO-PSO-SWUCB

Adapta o *Sliding-Window UCB*, descrito na Seção 2.3.2.1, ao caso multiobjetivo. Apenas as recompensas observadas dentro de uma janela de tamanho W são consideradas no cálculo da média vetorial:

$$\bar{\mathbf{r}}_W(a) = \frac{1}{N_W(a)} \sum_{m: t_m \geq t-W+1} \mathbf{r}_m(a), \quad (6.11)$$

em que $N_W(a)$ representa o número de coletas da solução a ocorridas dentro da janela. Diferentemente do MO-PSO-DPUCB, que atenua progressivamente o peso de observações antigas, o MO-PSO-SWUCB descarta completamente o histórico anterior à janela, sendo especialmente apropriado para ambientes com mudanças abruptas na composição ou na qualidade relativa das soluções disponíveis no *Pool*.

MO-PSO-mPQL *stateless*

Versão *stateless* do *Pareto Q-learning* (Van Moffaert e Nowé, 2014) adequada ao cenário de cooperação modelado neste trabalho. Como a interação no *Pool* ocorre em um único estado, não há necessidade de manter conjuntos de vetores associados às transições entre estados, e cada solução armazena apenas um vetor $\mathbf{Q}(a)$. A atualização segue uma combinação convexa entre o vetor atual e a recompensa observada:

$$\mathbf{Q}(a) \leftarrow \alpha \mathbf{r}(a) + (1 - \alpha) \mathbf{Q}(a), \quad (6.12)$$

em que $\alpha \in (0,1]$ é a taxa de aprendizado. A política de seleção opera sobre a frente de Pareto formada pelos vetores $\mathbf{Q}(a)$ das soluções candidatas, conforme a Equação (6.6).

6.1.4 Algoritmo *Multi-armed Bandit* Multiobjetivo

O Algoritmo 10 apresenta a estrutura geral do procedimento de aprendizado multiobjetivo sobre o *Pool* de soluções da AMAM. O algoritmo recebe como entrada o *Pool* de soluções $\mathcal{K}$, a política de seleção π , definida entre as quatro variantes da Seção 6.1.3, e os respectivos parâmetros de controle: o coeficiente de exploração C , o fator de desconto γ , o tamanho da janela deslizante W e a taxa de aprendizado α . Na etapa de inicialização (linha 1), a memória vetorial $\mathbf{Q}(a)$ e o contador de coletas $N(a)$ de cada solução são zerados. O laço principal (linha 2) permanece ativo enquanto o critério de parada do *framework* não é atingido, e cada iteração é disparada por uma requisição de coleta de um agente (linha 3), o que caracteriza o funcionamento assíncrono e orientado a eventos do mecanismo. Atendida a requisição, o vetor de escore $\mathbf{S}(a)$ é calculado para cada solução ativa do *Pool* de acordo com a política configurada (linhas 4 e 5), combinando a estimativa de qualidade acumulada com o termo de incerteza próprio de cada variante. Sobre esses escores, constrói-se a frente de Pareto A_0 conforme a Equação (6.6) (linha 7), restringindo a escolha às soluções não dominadas. A seleção entre elas é uniforme (linha 8), pois as soluções da frente são incomparáveis entre si sob a relação de dominância, e o sorteio equiprovável evita a introdução de viés em favor de qualquer objetivo. A solução escolhida a^* é, então, devolvida ao agente requisitante (linha 9), que sobre ela aplica o procedimento de busca local, enquanto o algoritmo aguarda o retorno da solução resultante (linha 10). Concluída a busca, o vetor de recompensa $\mathbf{r}(a^*)$ é calculado conforme a Equação (6.1) (linha 11), medindo a melhora obtida em cada objetivo, e a memória vetorial da solução é atualizada segundo a regra específica da variante em uso (linha 12): média incremental no MO-PSO-UCB, acumulação descontada por γ no MO-PSO-DPUCB, estatísticas restritas à janela W no MO-PSO-SWUCB e mistura convexa com taxa α no MO-PSO-mPQL, conforme a Equação (6.12). Por fim, o contador de coletas da solução é incrementado (linha 13), informação utilizada pelos termos de exploração das políticas na iteração seguinte.

Algoritmo 10 *Multi-armed Bandit* Multiobjetivo no *Pool* da AMAM

Entrada: *Pool* de soluções $\mathcal{K}$, política $\pi \in \{\text{MO-PSO-UCB}, \text{MO-PSO-DPUCB}, \text{MO-PSO-SWUCB}, \text{MO-PSO-mPQL}\}$, parâmetros C, γ, W, α

- 1: Inicializar $\mathbf{Q}(a) \leftarrow \mathbf{0}$, $N(a) \leftarrow 0$ para todo $a \in \mathcal{K}$
- 2: **enquanto** critério de parada do *framework* não atingido **faça**
- 3: Aguardar requisição de coleta por um agente
- 4: **para todo** soluções $a \in \mathcal{K}(t)$ **faça**
- 5: Calcular vetor de escore $\mathbf{S}(a)$ conforme π
- 6: **fim para**
- 7: Construir frente de Pareto A_0 conforme a Equação (6.6)
- 8: Selecionar $a^* \sim \mathcal{U}(A_0)$
- 9: Devolver a^* ao agente requisitante
- 10: Aguardar retorno da solução resultante após busca local
- 11: Calcular vetor de recompensa $\mathbf{r}(a^*)$ conforme a Equação (6.1)
- 12: Atualizar memória vetorial de a^* conforme π
- 13: $N(a^*) \leftarrow N(a^*) + 1$
- 14: **fim enquanto**

6.1.5 Definição da Experimentação

A configuração experimental segue a mesma estrutura adotada na Seção 5.2.7: o cenário ILS_MIX com 20 agentes, as 24 instâncias advindas de Solomon (1987) descritas na Seção 3.3 e 30 execuções independentes por instância. As métricas de

avaliação são o *gap* percentual em relação ao melhor valor conhecido e o tempo médio de execução em segundos, calculados conforme a Equação (3.2). A análise estatística segue a mesma abordagem não paramétrica adotada nos demais experimentos desta tese, justificada na Seção 3.3.

Quatro variantes do algoritmo *Multi-armed Bandit* Multiobjetivo são avaliadas: MO-PSO-UCB, MO-PSO-DPUCB, MO-PSO-SWUCB e MO-PSO-mPQL, descritas na Seção 6.1.3. Para permitir uma comparação direta entre o aprendizado multiobjetivo e o melhor resultado obtido com aprendizado multiagente, inclui-se também o cenário PSO-DUCB, identificado como a variante mais robusta do MAMaB-PSO na Seção 5.2.9. O cenário de referência para análise comparativa é o MULTI-EG (*multi_epsilon_greedy*), ou seja, o cenário sem componente de aprendizado adaptativo. A Tabela 6.1 resume os cenários avaliados.

Tabela 6.1: Cenários experimentais do MaB Multiobjetivo. Todos os cenários utilizam a mesma composição de agentes, o ILS_MIX com 20 agentes, descrita na Seção 5.2.7.

Cenário	Política
MULTI-EG	<i>multi_epsilon_greedy</i> (referência escalar)
PSO-DUCB	<i>pso_ducb</i> (melhor MAMaB-PSO)
MO-PSO-UCB	<i>pso_morl_pucb</i>
MO-PSO-DPUCB	<i>pso_morl_dpucb</i>
MO-PSO-SWUCB	<i>pso_morl_swpucb</i>
MO-PSO-mPQL	<i>pso_morl_mpql</i>

Tabela 6.2: Parâmetros das políticas MaB Multiobjetivo.

Parâmetro	Símbolo	Valor	Políticas
Constante de diversificação UCB	c	1,0	MO-PSO-UCB, MO-PSO-DPUCB, MO-PSO-SWUCB
Fator de desconto temporal	γ	0,77	MO-PSO-DPUCB
Tamanho da janela deslizante	W	100	MO-PSO-SWUCB
Taxa de aprendizado vetorial	α	0,9	MO-PSO-mPQL

Os parâmetros específicos de cada variante estão descritos na Tabela 6.2. Os parâmetros c , γ e W seguem os mesmos valores adotados nas políticas MAMaB-PSO equivalentes, garantindo a possibilidade de comparação entre os experimentos. O parâmetro α do MO-PSO-mPQL controla a taxa de aprendizado da atualização vetorial da Equação (6.12): o valor 0,9 atribui peso elevado à recompensa mais recente, conferindo alta adaptabilidade ao ambiente não estacionário do *Pool*.

6.1.6 Resultados Computacionais

Esta seção apresenta os resultados obtidos pelas quatro variantes do algoritmo *Multi-armed Bandit* Multiobjetivo em comparação com as referências MULTI-EG e PSO-DUCB, conforme a configuração experimental definida na Seção 6.1.5. A análise é organizada em quatro perspectivas complementares:

- (i) a qualidade das soluções, medida pelo *gap* percentual em relação ao melhor valor conhecido, examinada por instância, por bloco de instâncias e pela melhor solução encontrada entre as execuções independentes;
- (ii) o custo computacional, avaliado pelo tempo médio de execução;
- (iii) a evolução temporal da qualidade das soluções ao longo da busca, para um subconjunto de instâncias representativas; e
- (iv) o perfil de desempenho de [Dolan e Moré \(2002\)](#), que sintetiza a robustez relativa dos métodos sobre todo o conjunto de instâncias.

Em cada perspectiva, as diferenças observadas são submetidas à análise estatística não paramétrica descrita na Seção 3.3.

6.1.6.1 Análise do *gap* percentual

A Tabela 6.3 apresenta a média do *gap* percentual por instância para as seis configurações avaliadas.

As instâncias do grupo C apresentam *gaps* próximos de zero em todas as configurações. Nas instâncias mais desafiadoras, ou seja, dos grupos R e RC, as variantes MO superam tanto o MULTI-EG quanto o PSO-DUCB: em R106, o MO-PSO-DPUCB reduz o *gap* para 2,10% contra 6,60% do MULTI-EG e 4,20% do PSO-DUCB; em RC102, o MO-PSO-UCB obtém 8,48% contra 11,04% (MULTI-EG) e 9,81% (PSO-DUCB); em RC103, o MO-PSO-DPUCB alcança 1,87% frente a 3,67% (MULTI-EG) e 2,90% (PSO-DUCB). Em termos de média global, o MO-PSO-DPUCB lidera

Tabela 6.3: Média do *gap* (%) por instância e configuração. Os valores após $\pm$ são o desvio-padrão das 30 execuções de cada instância. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-DUCB	MO-PSO-UCB	MO-PSO-DPUCB	MO-PSO-SWUCB	MO-PSO-mPQL
C101	0,01 $\pm$ 0,02	0,00 $\pm$ 0,01	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,01	0,00 $\pm$ 0,00
C103	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C107	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C109	0,01 $\pm$ 0,04	0,01 $\pm$ 0,04	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,01 $\pm$ 0,03	0,00 $\pm$ 0,00
C204	0,12 $\pm$ 0,09	0,10 $\pm$ 0,08	0,11 $\pm$ 0,06	0,11 $\pm$ 0,06	0,11 $\pm$ 0,08	0,12 $\pm$ 0,07
C205	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C206	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
C208	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00	0,00 $\pm$ 0,00
R102	0,59 $\pm$ 1,62	0,93 $\pm$ 2,01	0,79 $\pm$ 1,82	0,44 $\pm$ 1,34	0,44 $\pm$ 1,34	0,45 $\pm$ 1,33
R104	9,93 $\pm$ 0,10	9,97 $\pm$ 0,11	9,96 $\pm$ 0,08	9,97 $\pm$ 0,09	9,99 $\pm$ 0,08	9,99 $\pm$ 0,08
R106	6,60 $\pm$ 2,56	4,20 $\pm$ 3,71	3,96 $\pm$ 3,70	2,10 $\pm$ 3,09	2,57 $\pm$ 3,39	3,27 $\pm$ 3,60
R112	10,03 $\pm$ 0,14	10,07 $\pm$ 0,09	10,08 $\pm$ 0,10	10,08 $\pm$ 0,10	10,08 $\pm$ 0,11	10,08 $\pm$ 0,10
R201	0,23 $\pm$ 0,17	0,26 $\pm$ 0,13	0,16 $\pm$ 0,13	0,20 $\pm$ 0,13	0,15 $\pm$ 0,14	0,17 $\pm$ 0,14
R208	0,54 $\pm$ 0,33	0,51 $\pm$ 0,32	0,47 $\pm$ 0,16	0,49 $\pm$ 0,26	0,45 $\pm$ 0,19	0,55 $\pm$ 0,25
R209	0,86 $\pm$ 0,37	0,71 $\pm$ 0,28	0,86 $\pm$ 0,35	0,70 $\pm$ 0,26	0,78 $\pm$ 0,30	0,85 $\pm$ 0,35
R211	31,72 $\pm$ 0,50	31,95 $\pm$ 0,49	31,85 $\pm$ 0,38	31,98 $\pm$ 0,37	31,79 $\pm$ 0,51	31,74 $\pm$ 0,38
RC102	11,04 $\pm$ 3,61	9,81 $\pm$ 3,50	8,48 $\pm$ 2,67	8,91 $\pm$ 3,04	8,93 $\pm$ 3,04	9,21 $\pm$ 3,19
RC103	3,67 $\pm$ 4,04	2,90 $\pm$ 3,74	3,22 $\pm$ 3,82	1,87 $\pm$ 3,05	2,39 $\pm$ 3,51	2,21 $\pm$ 3,25
RC105	7,45 $\pm$ 2,33	9,01 $\pm$ 3,28	7,03 $\pm$ 1,69	6,57 $\pm$ 0,15	6,82 $\pm$ 1,24	7,27 $\pm$ 2,03
RC108	8,81 $\pm$ 1,60	8,54 $\pm$ 2,23	8,88 $\pm$ 1,55	8,88 $\pm$ 1,56	8,91 $\pm$ 1,59	8,87 $\pm$ 1,60
RC201	0,73 $\pm$ 0,39	0,84 $\pm$ 0,41	0,85 $\pm$ 0,28	0,76 $\pm$ 0,27	0,84 $\pm$ 0,25	0,82 $\pm$ 0,26
RC202	18,92 $\pm$ 0,60	18,83 $\pm$ 0,59	18,88 $\pm$ 0,44	19,02 $\pm$ 0,51	18,87 $\pm$ 0,45	18,95 $\pm$ 0,51
RC206	1,59 $\pm$ 0,80	1,64 $\pm$ 0,81	1,55 $\pm$ 0,82	1,54 $\pm$ 0,57	1,30 $\pm$ 0,54	1,31 $\pm$ 0,49
RC208	1,38 $\pm$ 0,56	1,25 $\pm$ 0,60	1,43 $\pm$ 0,56	1,45 $\pm$ 0,50	1,34 $\pm$ 0,44	1,35 $\pm$ 0,51
Média	4,76 $\pm$ 7,60	4,65 $\pm$ 7,64	4,52 $\pm$ 7,53	4,38 $\pm$ 7,57	4,41 $\pm$ 7,54	4,47 $\pm$ 7,56

Tabela 6.4: Média do *gap* (%) por bloco de instâncias e configuração. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito. Os valores após $\pm$ são o desvio-padrão calculado sobre todas as execuções das instâncias do bloco e, portanto, refletem também a heterogeneidade entre instâncias, não apenas a variação entre execuções.

Bloco	MULTI-EG	PSO-DUCB	MO-PSO-UCB	MO-PSO-DPUCB	MO-PSO-SWUCB	MO-PSO-mPQL
C	0,02 $\pm$ 0,05	0,01 $\pm$ 0,05	0,01 $\pm$ 0,04	0,01 $\pm$ 0,04	0,02 $\pm$ 0,05	0,01 $\pm$ 0,05
R	7,56 $\pm$ 10,03	7,33 $\pm$ 10,20	7,27 $\pm$ 10,19	7,00 $\pm$ 10,32	7,03 $\pm$ 10,25	7,14 $\pm$ 10,19
RC	6,70 $\pm$ 6,24	6,60 $\pm$ 6,25	6,29 $\pm$ 5,96	6,12 $\pm$ 6,06	6,18 $\pm$ 6,05	6,25 $\pm$ 6,13
Média	4,76 $\pm$ 7,60	4,65 $\pm$ 7,64	4,52 $\pm$ 7,53	4,38 $\pm$ 7,57	4,41 $\pm$ 7,54	4,47 $\pm$ 7,56

com 4,38%, seguido pelo MO-PSO-SWUCB (4,41%), MO-PSO-mPQL (4,47%), MO-PSO-UCB (4,52%) e PSO-DUCB (4,65%), todos superiores ao MULTI-EG (4,76%). O teste de Friedman, pareado por instância, não revelou diferença estatisticamente significativa ($p = 0,35$), e nenhum dos 15 pares apresentou diferença no teste de Wilcoxon *signed-rank* com correção de Bonferroni, confirmando equivalência estatística na qualidade final das soluções. A equivalência se mantém quando a análise é feita por bloco, inclusive nos conjuntos mais difíceis: C ($p = 0,158$), R (0,461) e RC (0,776).

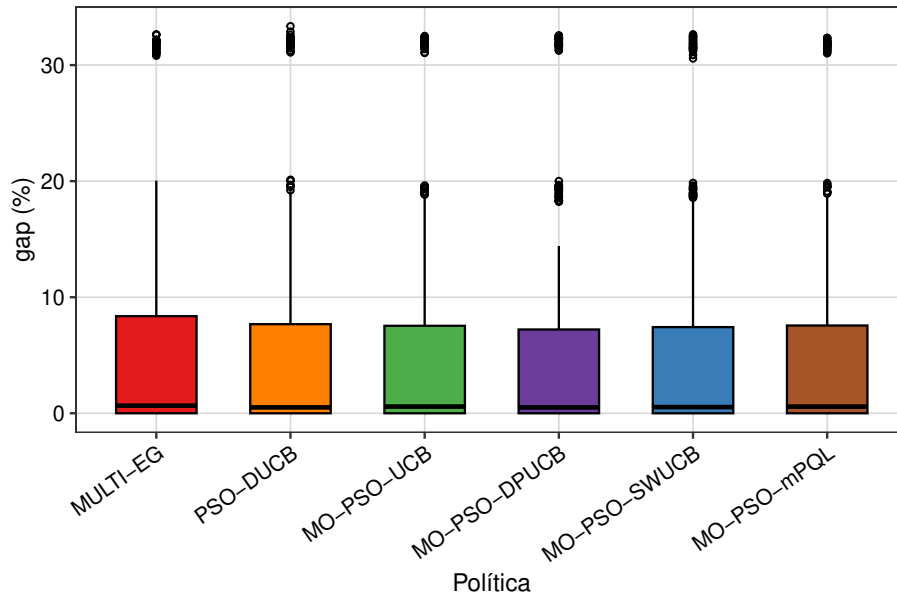


Figura 6.1: Distribuição do *gap* (%) por configuração ao longo das 24 instâncias avaliadas.

A Figura 6.1 apresenta os *boxplots* do *gap* percentual das seis configurações. As distribuições estão praticamente sobrepostas: todas apresentam mediana próxima de zero, caixa que se estende até cerca de 8% e a mesma nuvem de *outliers* em torno de 31%, correspondente à instância R211. A única distinção perceptível é o bigode superior do MO-PSO-DPUCB, que termina próximo de 14% contra aproximadamente 20% das demais configurações, coerente com o menor *gap* médio obtido por essa variante. O aspecto geral da figura confirma visualmente a equivalência estatística apontada pelo teste de Friedman e mostra que a dispersão do *gap* é governada pela instância, e não pela política adotada.

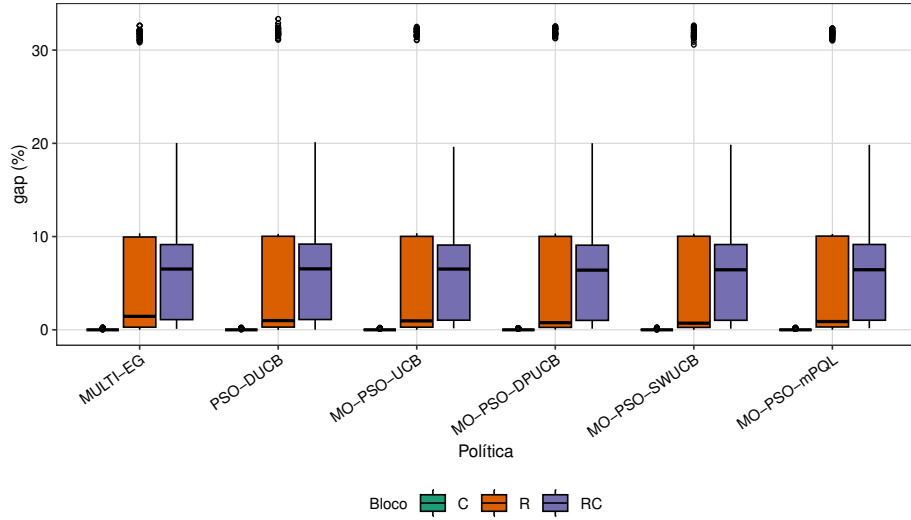


Figura 6.2: Distribuição do *gap* (%) por configuração e bloco de instâncias (C, R e RC).

A Figura 6.2 separa a mesma distribuição por bloco e deixa claro a fonte da dispersão de valores. No bloco C, as caixas colapsam sobre o zero em todas as seis configurações, refletindo os *gaps* nulos da Tabela 6.4. Toda a variabilidade concentra-se nos blocos R e RC, de amplitude semelhante, com mediana baixa em R e próxima de 6% em RC, além dos *outliers* de R211. O padrão repete-se sem alteração visível nas seis configurações, o que reforça que é o bloco de instâncias, e não a política, que determina a qualidade final alcançada.

A Tabela 6.4 confirma o padrão observado. Nos blocos R e RC, todas as variantes MO superam tanto o MULTI-EG quanto o PSO-DUCB, com o MO-PSO-DPUCB obtendo os menores *gaps* médios em ambos (7,00% e 6,12%, respectivamente), contra 7,33% e 6,60% do PSO-DUCB.

Para avaliar o potencial de qualidade de cada configuração, a Tabela 6.5 apresenta o menor *gap* obtido em cada instância, ou seja, a melhor solução encontrada entre as 30 execuções. A linha *Vitórias* contabiliza em quantas instâncias cada configuração atinge o menor *gap* (empates incluídos), contagem também resumida na Figura 6.3.

Diferentemente da análise do *gap* médio (Tabela 6.3), na qual as variantes multiobjetivo lideram, na melhor solução o MULTI-EG (17 vitórias) e o PSO-DUCB (16) atingem o menor *gap* no maior número de instâncias. Esse contraste sugere que as variantes MO são mais *consistentes* (melhor desempenho médio), ao passo que as referências escalar e multiagente, embora mais sujeitas à variabilidade, alcançam ocasionalmente as melhores soluções individuais. As diferenças, contudo, permanecem marginais, reforçando a equivalência estatística entre os métodos.

6.1.6.2 Análise do tempo de execução

A Tabela 6.6 apresenta a média do tempo de execução por instância para as seis configurações avaliadas. A Tabela 6.7, além disso, agrega esses valores por bloco. Os valores em negrito indicam o menor tempo.

O PSO-DUCB foi a configuração mais eficiente, com a menor média global (58,01 s) e o menor tempo em 14 das 24 instâncias. As variantes multiobjetivo apresentaram

Tabela 6.5: Menor *gap* (%) por instância e configuração (melhor solução entre as 30 execuções). A linha *Vitórias* indica em quantas instâncias cada configuração atinge o menor *gap* (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-DUCB	MO-PSO-UCB	MO-PSO-DPUCB	MO-PSO-SWUCB	MO-PSO-mPQL
C101	0,00	0,00	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00	0,00	0,00
C109	0,00	0,00	0,00	0,00	0,00	0,00
C204	0,00	0,00	0,00	0,00	0,00	0,00
C205	0,00	0,00	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00	0,00	0,00
R102	0,01	0,00	0,01	0,05	0,02	0,03
R104	9,79	9,79	9,86	9,82	9,84	9,82
R106	0,10	0,09	0,13	0,11	0,00	0,11
R112	9,82	9,92	9,89	9,83	9,82	9,89
R201	0,01	0,02	0,02	0,02	0,02	0,02
R208	0,03	0,03	0,14	0,14	0,13	0,16
R209	0,03	0,18	0,37	0,20	0,20	0,34
R211	30,81	31,09	31,05	31,25	30,57	31,01
RC102	6,81	6,96	6,96	7,04	7,07	6,90
RC103	0,10	0,12	0,16	0,12	0,12	0,19
RC105	6,32	6,31	6,38	6,34	6,30	6,40
RC108	0,34	0,34	0,68	0,62	0,47	0,40
RC201	0,21	0,10	0,21	0,12	0,32	0,34
RC202	18,22	18,22	18,22	18,22	18,22	18,22
RC206	0,37	0,18	0,25	0,18	0,45	0,18
RC208	0,40	0,01	0,32	0,28	0,56	0,17
Vitórias	17	16	9	10	13	10

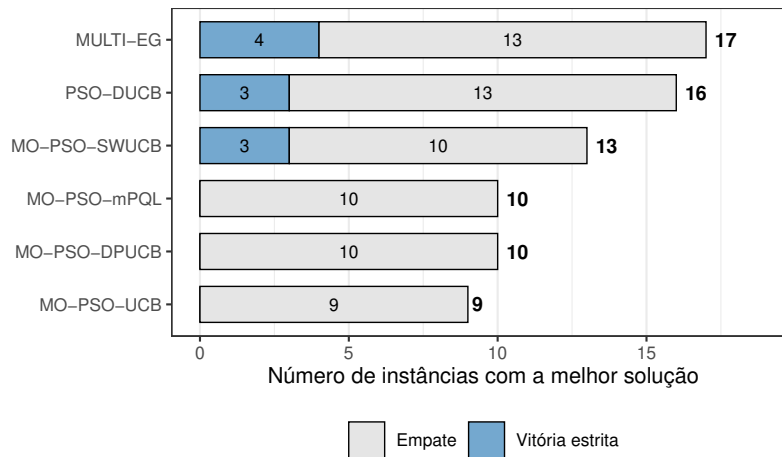


Figura 6.3: Número de instâncias em que cada configuração encontra a melhor solução (menor *gap*), separando vitórias estritas (único mínimo) de empates.

médias intermediárias e próximas entre si (≈ 61 s), enquanto o MULTI-EG (67,77 s) foi o mais lento na média, penalizado sobretudo pelas instâncias C205 (137,93 s) e C206 (109,23 s), nas quais as demais configurações concluíram em ≈ 50 –60 s.

Tabela 6.6: Média do tempo de execução (s) por instância e configuração.

Instância	MULTI-EG	PSO-DUCB	MO-PSO-UCB	MO-PSO-DPUCB	MO-PSO-SWUCB	MO-PSO-mPQL
C101	74,20	56,23	53,60	53,70	53,33	53,60
C103	50,33	45,67	47,47	48,63	48,33	47,53
C107	63,40	49,27	47,37	47,40	48,13	47,43
C109	69,50	62,87	64,03	64,67	64,03	63,53
C204	91,40	89,90	91,83	92,50	94,97	92,33
C205	137,93	50,70	50,70	50,83	50,67	50,20
C206	109,23	59,00	58,83	59,77	58,43	59,40
C208	94,33	62,63	62,70	63,03	63,50	63,20
R102	44,27	45,83	59,07	58,80	56,97	57,20
R104	41,00	42,37	48,37	47,80	48,50	47,40
R106	43,63	40,80	44,80	45,63	44,57	44,23
R112	34,33	33,93	41,70	41,93	42,10	42,33
R201	64,23	61,07	63,20	62,70	62,10	61,93
R208	109,93	110,70	111,53	110,40	113,40	109,00
R209	66,73	64,87	68,23	69,07	69,63	68,73
R211	83,33	83,77	86,30	85,73	85,97	87,00
RC102	50,97	47,07	50,77	50,50	50,53	50,70
RC103	42,30	41,63	45,77	46,20	46,03	44,93
RC105	45,10	45,20	51,60	51,33	51,60	52,20
RC108	41,17	42,50	44,00	44,00	43,47	43,50
RC201	60,00	54,27	56,00	56,70	57,07	57,07
RC202	64,73	64,23	66,63	66,33	66,90	66,10
RC206	64,93	59,23	69,63	68,73	69,20	68,77
RC208	79,47	78,47	80,67	81,60	79,87	79,57
Média	67,77	58,01	61,03	61,17	61,22	60,74

Tabela 6.7: Média do tempo de execução (s) por bloco de instâncias e configuração.

Bloco	MULTI-EG	PSO-DUCB	MO-PSO-UCB	MO-PSO-DPUCB	MO-PSO-SWUCB	MO-PSO-mPQL
C	86,29	59,53	59,57	60,07	60,17	59,65
R	60,93	60,42	65,40	65,26	65,40	64,73
RC	56,08	54,08	58,13	58,17	58,08	57,85
Média	67,77	58,01	61,03	61,17	61,22	60,74

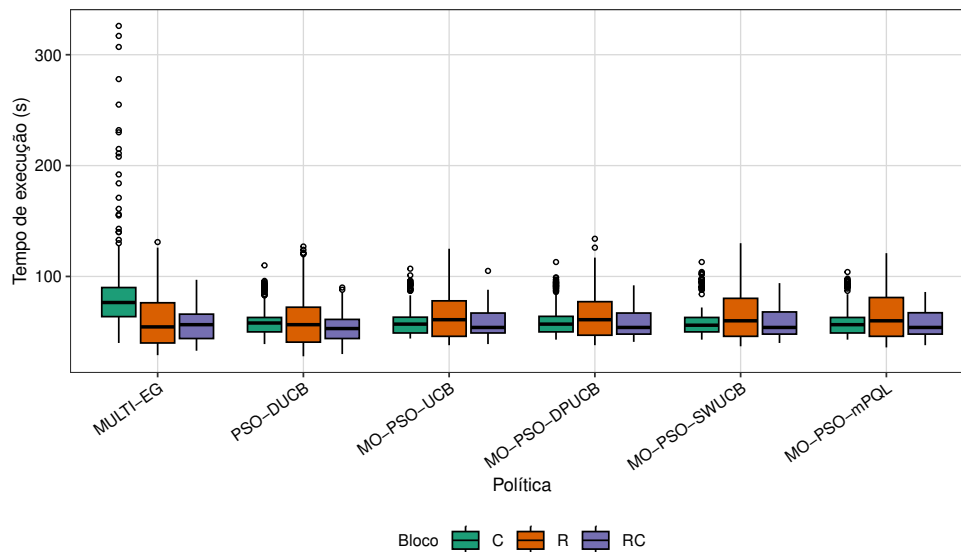


Figura 6.5: Distribuição do tempo de execução (s) por configuração e bloco de instâncias (C, R e RC).

A Tabela 6.7 revela padrão crítico: PSO-DUCB é significativamente mais rápido

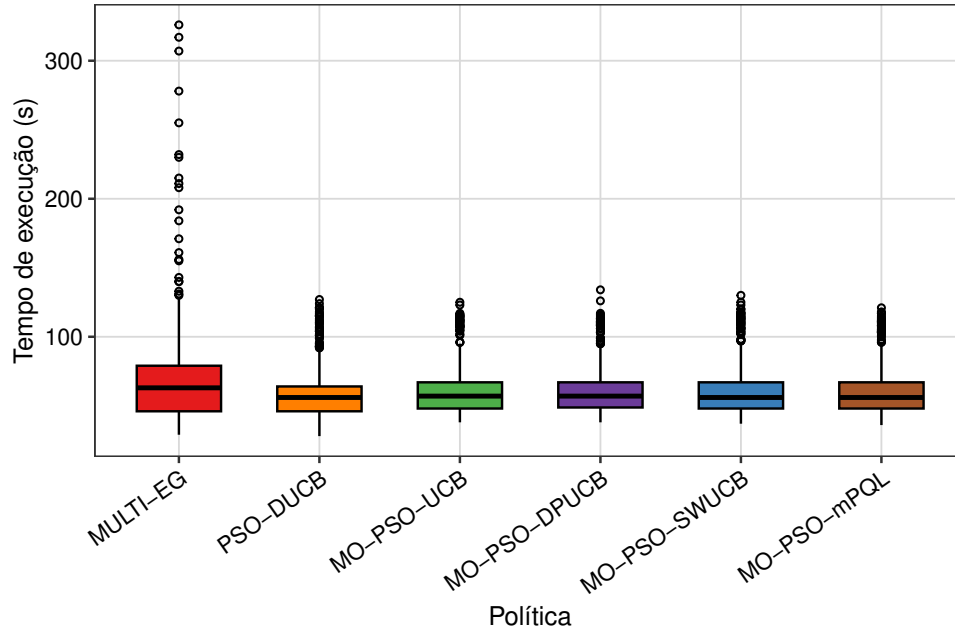


Figura 6.4: Distribuição do tempo de execução (s) por configuração ao longo das 24 instâncias avaliadas.

que todas as variantes MO em todos os blocos. No bloco C, tanto PSO-DUCB (59,53s) quanto as variantes MO (≈ 60 s) são muito mais rápidos que MULTI-EG (86,29s). Nos blocos R e RC, PSO-DUCB permanece competitivo (60,42s e 54,08s), enquanto as variantes MO são consistentemente mais lentas (64–65s em R e 57–58s em RC). O teste de Friedman, pareado por instância, confirmou diferença significativa ($p = 1,07 \times 10^{-4}$), com 5 dos 15 pares significativos no teste de Wilcoxon *signed-rank* com correção de Bonferroni. Por bloco, o teste de Friedman acusa diferença em C ($p = 0,0069$), R (0,0025) e RC (0,0020), mas nenhum par isolado permanece significativo após a correção dentro de um bloco de 8 instâncias. O teste *post-hoc* de Wilcoxon com correção de Bonferroni mostrou que PSO-DUCB é significativamente mais rápido que MULTI-EG ($p < 0,001$) e que todas as variantes MO ($p < 0,005$), enquanto entre as variantes MO não há diferença significativa.

6.1.6.3 Análise da evolução da qualidade das soluções

A Figura 6.6 apresenta a evolução temporal do *gap* para seis instâncias representativas. As estratégias destacadas são o PSO-DUCB (melhor variante MAMaB-PSO) e o MO-PSO-DPUCB (melhor variante MO); as demais variantes são exibidas em cinza para contexto. As instâncias foram selecionadas com base no maior contraste de desempenho entre as políticas.

Na instância C109, todos os métodos convergem rapidamente para *gap* zero, confirmando a facilidade de solução das instâncias do grupo C. Em R106, o contraste é expressivo: o MO-PSO-DPUCB atinge *gap* próximo de 2%, enquanto o PSO-DUCB estagna em torno de 4,2% e o MULTI-EG em 6,6%, demonstrando clara vantagem da abordagem multiobjetivo nessa instância desafiadora. Comportamento similar é observado em R209 e RC102, na qual MO-PSO-DPUCB supera PSO-DUCB em velocidade e qualidade final de convergência. Nas instâncias RC103 e C204, a vantagem

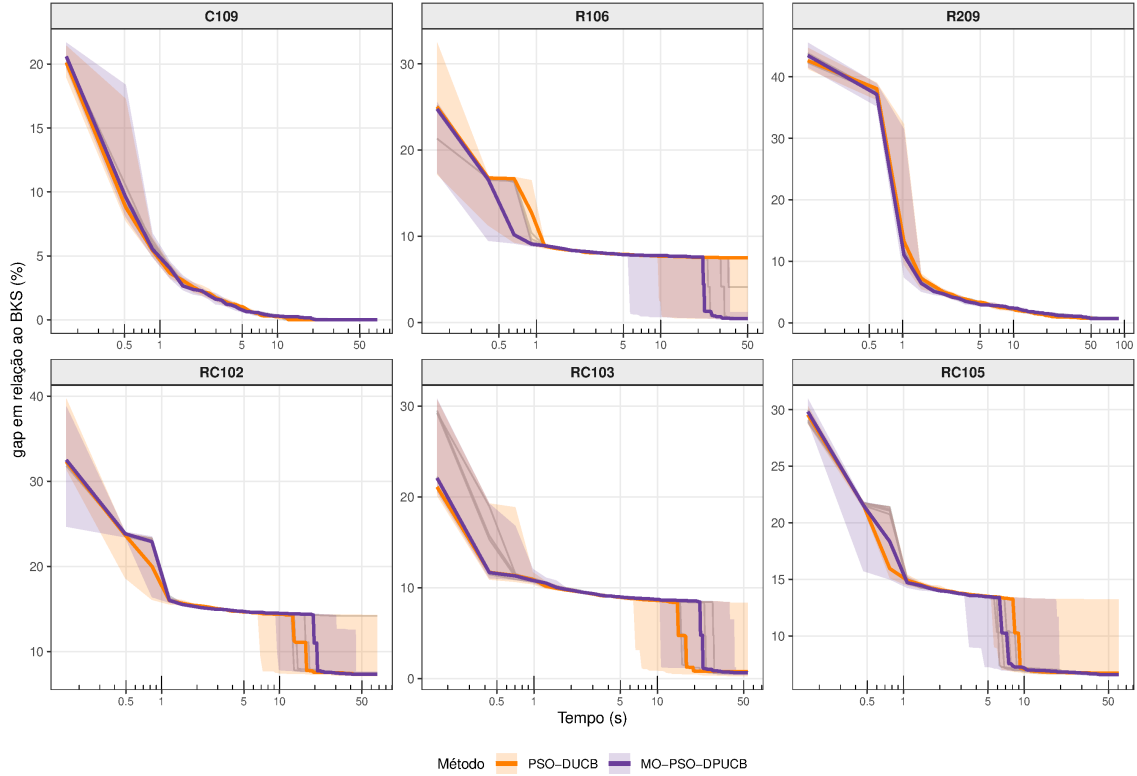


Figura 6.6: Evolução temporal do *gap* em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem ao PSO-DUCB (laranja, melhor multiagente) e ao MO-PSO-DPUCB (roxo, melhor multiobjeto); as demais variantes MO são mostradas em cinza. As faixas sombreadas indicam o IQR entre as 30 execuções independentes. O eixo x está em escala logarítmica.

do MO-PSO-DPUCB sobre PSO-DUCB é consistente ao longo da execução, embora ambos superem o MULTI-EG.

6.1.6.4 Perfil de desempenho comparativo

A Figura 6.7 apresenta o perfil de desempenho proposto por Dolan e Moré (2002) para os seis métodos avaliados, incluindo PSO-DUCB para comparação direta entre as abordagens multiobjetivo e multiagente.

O perfil revela diferenças críticas de robustez entre as abordagens. Em $\tau = 1,01$, PSO-DUCB e MO-PSO-SWUCB lideram com 66,7% de cobertura, demonstrando competitividade no melhor desempenho. Em $\tau = 1,05$, ambas atingem 75,0%. O critério mais expressivo é o limiar τ necessário para cobrir 90% das instâncias: MO-PSO-SWUCB exige apenas $\tau = 1,219$, seguido por MO-PSO-DPUCB ($\tau = 1,247$), PSO-DUCB e MO-PSO-UCB ($\tau = 1,356$) e MO-PSO-mPQL ($\tau = 1,356$). Isso demonstra que, embora MO-PSO-DPUCB seja superior em qualidade média e robustez, PSO-DUCB oferece desempenho equivalente com vantagem crítica em tempo de execução (40% mais rápido que variantes MO em blocos R e RC). O MULTI-EG apresenta o pior resultado ($\tau = 1,794$). Em termos de vitórias diretas, PSO-DUCB lidera com 13/24 instâncias, igualado por MO-PSO-DPUCB e MO-PSO-SWUCB (12/24).

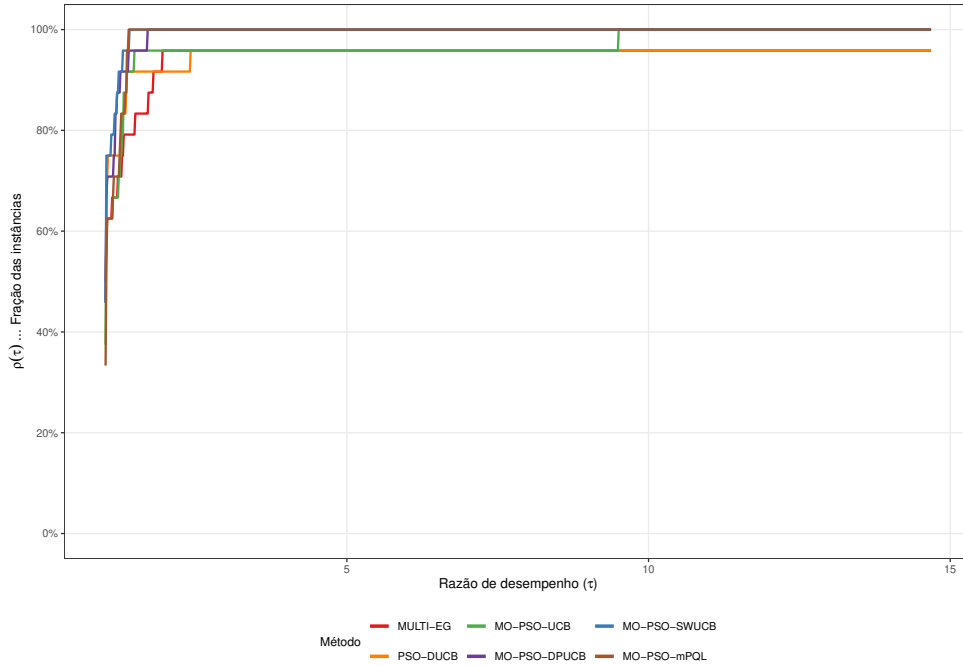


Figura 6.7: Perfil de desempenho (Dolan e Moré, 2002) dos seis métodos avaliados nas 24 instâncias de Solomon. O eixo x representa a razão de desempenho τ ; o eixo y indica a fração das instâncias em que o método possui razão $\leq \tau$.

6.1.7 Considerações Parciais

Esta subseção se refere aos resultados devidos ao algoritmo *Multi-armed Bandit* Multiobjetivo e, portanto, finaliza a presente Seção. Os resultados da Seção 6.1 mostram que a substituição das recompensas escalares por vetores de recompensa compostos por função objetivo, diversidade estrutural e número de rotas, combinada à seleção por dominância de Pareto, confere ao mecanismo de aprendizado do *Pool* uma capacidade de discriminação mais refinada entre as soluções candidatas. O MO-PSO-DPUCB emergiu como a variante de melhor desempenho geral, alcançando a menor média de *gap* em todo o conjunto de instâncias (4,38% contra 4,76% do MULTI-EG e 4,65% do PSO-DUCB), com reduções expressivas nas instâncias mais desafiadoras dos blocos R e RC: em R106, o *gap* caiu de 6,60% para 2,10%, representando uma redução de 68% em relação à referência escalar.

A análise temporal revelou uma compensação importante: as variantes multiobjetivo incorrem em custo computacional adicional de aproximadamente 5% em relação ao PSO-DUCB (≈ 61 s contra 58 s), reflexo do processamento do vetor de recompensa e da construção da frente de Pareto a cada coleta. Esse compromisso entre riqueza representacional e custo computacional constitui a principal limitação da abordagem proposta nesta seção, motivando diretamente a reestruturação do próprio *Pool* apresentada na Seção 6.2, que busca reduzir esse custo por meio de uma organização multiobjetivo mais eficiente das soluções armazenadas.

Em síntese, os resultados confirmam que as variantes MO produzem soluções de qualidade estatisticamente equivalente ao MULTI-EG e ao PSO-DUCB ($p = 0,35$ no teste de Friedman) enquanto apresentam maior robustez em cenários adversos. O MO-PSO-SWUCB destaca-se como a variante mais robusta em cobertura de instân-

cias, enquanto MO-PSO-DPUCB oferece melhor equilíbrio entre qualidade de solução e tempo de execução. Comparativamente, PSO-DUCB oferece alternativa rápida com qualidade equivalente, mas sem a exploração multiobjetivo da estrutura do *Pool*. Esses resultados respondem ao objetivo específico (viii) da Seção 1.3.2, demonstrando tanto a viabilidade quanto a complementaridade do aprendizado por reforço multiobjetivo incorporado ao *Framework* AMAM.

6.2 *Pool* de Soluções Multiobjetivo

A Seção 6.1 propôs a extensão do mecanismo de aprendizado por reforço para recompensas vetoriais, modelando a coleta de soluções como um problema de decisão multiobjetivo. Para que esse aprendizado seja plenamente aproveitado, o próprio *Pool* de soluções precisa ser organizado segundo critérios multiobjetivo: não basta manter soluções de boa qualidade se a estrutura de armazenamento não garante diversidade. Esta seção propõe a reestruturação do *Pool* com base em dominância de Pareto e *crowding distance*, complementando o aprendizado vetorial proposto na seção anterior.

6.2.1 Descrição do *Pool* Multiobjetivo

Em métodos de busca populacional, é essencial que, além da qualidade dos valores da função objetivo, o conjunto de soluções apresente diversidade. Dessa forma, torna-se necessário que o conjunto de soluções pertencentes ao *Pool* seja diverso, garantindo o elitismo com a permanência da solução com o melhor valor de função objetivo. Na versão original do AMAM (Silva et al., 2019), apresentada na Seção 2.1.1.3, a inserção de soluções ocorre exclusivamente com base no valor da função objetivo, isto é, uma solução é inserida no *Pool* caso apresente desempenho superior ao pior valor presente no instante de inserção t . A definição da solução a ser removida do *Pool*, por sua vez, baseia-se na diversidade em relação às demais soluções armazenadas.

A dominância de Pareto (Deb, 2001) é um critério utilizado em problemas de otimização multiobjetivo para comparar soluções com base em múltiplos objetivos simultaneamente, sem a necessidade de agregá-los em uma única função objetivo. Diz-se que uma solução domina outra quando apresenta desempenho igual ou superior em todos os objetivos e estritamente superior em pelo menos um deles, estabelecendo uma relação de ordem parcial entre as soluções. A partir dessa relação, é possível identificar o conjunto de soluções não dominadas, denominado fronteira de Pareto (Deb, 2001), que representa os melhores compromissos possíveis entre os objetivos considerados (Deb, 2001). Entretanto, em algoritmos evolutivos multiobjetivo, a dominância por si só não garante diversidade adequada entre as soluções. Para lidar com essa limitação, a métrica *crowding distance* é empregada como uma medida de densidade local que estima o grau de aglomeração das soluções ao redor de cada indivíduo na fronteira de Pareto, calculando a distância normalizada entre soluções vizinhas para cada objetivo, o que favorece a preservação da diversidade ao longo da fronteira (Deb et al., 2002).

Nesta tese, propõe-se um mecanismo biobjetivo para manutenção do *Pool* de soluções, no qual cada solução é avaliada simultaneamente por seu valor de função

objetivo (fo) e por sua medida de similaridade estrutural (shi), definida na Expressão (6.5) da Seção 6.1.2, sendo ambos critérios de minimização. As soluções são inseridas continuamente no *Pool* até que um número predefinido de inserções seja atingido, ou até que o tamanho máximo do *Pool* seja excedido. Nesse momento, executa-se um procedimento de ordenação por dominância de Pareto para identificar as frentes não dominadas, seguindo o procedimento *fast non-dominated sorting* proposto por Deb et al. (2002) para o algoritmo NSGA-II. Em seguida, calcula-se a métrica *crowding distance*, também introduzida por Deb et al. (2002), para cada frente, com o objetivo de preservar a diversidade entre as soluções sobreviventes. O processo de seleção mantém inicialmente todas as soluções da primeira frente de Pareto. Caso essa quantidade seja inferior ao número mínimo estabelecido para o *Pool*, soluções das frentes subsequentes são adicionadas, em ordem decrescente de valores de *crowding distance*, até que o tamanho mínimo seja satisfeito. As soluções remanescentes são removidas do *Pool* ativo e armazenadas em uma estrutura auxiliar de descarte. Os Algoritmos 11 e 12 ilustram o processo de inserção e manutenção das soluções no *Pool* ao longo da execução do método.

Algoritmo 11 Pool Multiobjetivo

```

1: Pool                                ▷ Conjunto de soluções ativas
2: pool_descartado                     ▷ Conjunto de soluções removidas
3: contador                             ▷ Contador de inserções desde a última filtragem
4: num_insercoes                         ▷ Limiar de inserções para acionar a filtragem
5: num_minimo                             ▷ Número mínimo de soluções no Pool
6: tamanho_max                           ▷ Limiar de tamanho que também aciona a filtragem
7: função INSERIRNOPOOL(solucao)
8:   Inserir solucao em Pool
9:   contador ← contador + 1
10:  se contador = num_insercoes ou |Pool| > tamanho_max então
11:    Calcular o valor de similaridade shi para cada solução
12:    Construir as frentes de Pareto com base em fo e shi
13:    Calcular a crowding distance em cada frente
14:    Ordenar cada frente em ordem decrescente de crowding distance
15:    novo_pool ← primeira frente de Pareto
16:    Remover a primeira frente da lista de frentes
17:    enquanto |novo_pool| < num_minimo e ainda existem frentes faça
18:      Inserir a próxima solução da frente corrente em novo_pool
19:      Remover a solução inserida da frente corrente
20:      se a frente corrente estiver vazia então
21:        Avançar para a próxima frente
22:      fim se
23:    fim enquanto
24:    para cada solução remanescente nas frentes não selecionadas faça
25:      Mover a solução para pool_descartado
26:    fim para
27:    Pool ← novo_pool
28:    contador ← 0
29:  fim se
30: fim função

```

O Algoritmo 11 descreve o mecanismo de gerenciamento do *Pool* de soluções utilizado na dimensão de cooperação do *framework*. As linhas 1 a 6 definem as estruturas de armazenamento e os parâmetros de controle: o conjunto de soluções ativas, o conjunto auxiliar de soluções descartadas, o contador de inserções desde a última filtragem, o limiar *num_insercoes*, o número mínimo de soluções *num_minimo* e o limiar de tamanho *tamanho_max*.

Convém esclarecer o papel do parâmetro *tamanho_max*, que não é um valor teto. O *Pool* não é truncado quando o atinge: o parâmetro apenas antecipa o acionamento

da filtragem, em paralelo ao contador de inserções. Quem de fato determina o tamanho do *Pool* após cada filtragem é a cardinalidade da primeira frente de Pareto, preservada integralmente, complementada até *num_minimo* quando insuficiente. O comportamento difere, nesse ponto, do *Pool* monobjetivo da Seção 5.1, em que o tamanho máximo é respeitado e a inserção em um *Pool* cheio implica a substituição de uma solução existente.

Assim, a cada chamada, a nova solução é inserida e o contador é incrementado (linhas 8 e 9). A filtragem é acionada quando o contador atinge o limiar de inserções ou quando o tamanho do *Pool* ultrapassa *tamanho_max* (linha 10). Acionada a filtragem, calcula-se o valor de similaridade *shi* de cada solução (linha 11), conforme a Expressão (6.5), e constroem-se as frentes de Pareto sobre os pares (fo, shi) , ambos critérios de minimização (linha 12). A *crowding distance* é então calculada em cada frente e as frentes são ordenadas em ordem decrescente desse valor (linhas 13 e 14), de modo que as soluções em regiões menos povoadas do espaço de objetivos tenham prioridade na sobrevivência.

A seleção ocorre em três etapas. A primeira frente é preservada integralmente e passa a compor o novo *Pool* (linhas 15 e 16). Se ela sozinha não alcança *num_minimo*, o laço das linhas 17 a 21 completa o conjunto com soluções das frentes subsequentes, tomadas na ordem de *crowding distance* já estabelecida e avançando de frente em frente à medida que cada uma se esgota. Por fim, as soluções que restaram nas frentes não selecionadas são transferidas para o conjunto de descarte (linhas 24 e 25), o novo conjunto substitui o *Pool* ativo e o contador é zerado (linhas 27 e 28).

As soluções descartadas não são eliminadas da execução: permanecem registradas na estrutura auxiliar com o instante em que saíram do *Pool*, o que permite reconstruir, ao final, o histórico de ocupação e de rotatividade do conjunto cooperativo.

Algoritmo 12 Interação Agente – *Pool* de soluções

```

1:  $s_0 \leftarrow$  construir uma solução inicial
2: InserirNoPool( $s_0$ )
3:  $s \leftarrow$  busca_local( $s_0$ )
4:  $melhor \leftarrow s$ 
5: InserirNoPool( $melhor$ )
6: enquanto critério de parada não for satisfeito faça
7:    $p \leftarrow$  perturbacao( $s$ )                                 $\triangleright$  um dos operadores coleta uma solução do Pool
8:    $s' \leftarrow$  busca_local( $p$ )
9:    $melhorou \leftarrow$  melhoraObjetivo( $s', melhor$ )
10:   $aceita \leftarrow$  viavel( $s'$ ) e criterioAceitacao( $s', melhor$ )
11:  se  $s'$  foi coletada do Pool então
12:    se  $melhorou$  então
13:      Registrar melhoria do Pool
14:    fim se
15:    AtualizarPool( $id(s'), agente, melhorou, s'$ )
16:  fim se
17:  se  $aceita$  então
18:     $melhor \leftarrow s'$ 
19:    InserirNoPool( $melhor$ )
20:  fim se
21:   $s \leftarrow melhor$ 
22: fim enquanto
  
```

O Algoritmo 12 descreve o comportamento de um agente que interage com o *Pool* de soluções ao longo do processo iterativo de busca. Inicialmente, o agente constrói uma solução viável e a insere no *Pool* (linhas 1 e 2). Em seguida, aplica-se um procedimento de busca local para intensificação (linha 3), definindo a melhor solução

corrente do agente (linha 4) e realizando uma nova inserção no *Pool* (linha 5). São, portanto, duas contribuições ao conjunto cooperativo antes mesmo do início do ciclo iterativo: a solução construída e a solução refinada pela busca local.

Durante o ciclo iterativo (linha 6), o agente gera uma solução perturbada a partir da solução corrente (linha 7). É nessa linha que ocorre a leitura do *Pool*: a coleta de uma solução cooperada não é um passo à parte, e sim *um dos operadores de perturbação* disponíveis ao agente, aplicado segundo as regras de acesso da Seção 5.1. Quando esse operador é sorteado, a solução perturbada passa a ser a solução coletada, marcada como proveniente do *Pool*; quando qualquer outro operador é sorteado, a perturbação é puramente local e nada é lido do conjunto cooperativo. Essa é a razão pela qual a verificação da linha 11 é necessária: nem toda iteração envolve cooperação.

Sobre a solução perturbada executa-se novamente a busca local (linha 8). A solução resultante é então avaliada sob dois critérios: (i) verificação de melhoria em relação à melhor solução corrente do agente (linha 9) e (ii) a conjunção entre a viabilidade da solução e o critério de aceitação do método (linha 10). Essa separação é importante pois a atualização do aprendizado do *Pool* depende da existência de melhora objetiva, enquanto a substituição da solução corrente depende do critério de aceitação. As duas condições são independentes: uma solução pode melhorar a função objetivo e ainda assim ser recusada por inviabilidade, e pode ser aceita pelo critério do método sem ter melhorado o valor de referência.

Quando a solução utilizada na iteração foi obtida a partir do *Pool* (linha 11), o mecanismo de atualização do *Pool* é acionado. Caso tenha havido melhora objetiva, o contador de melhorias é incrementado (linha 13); em seguida, o *Pool* é atualizado com o identificador da solução, o agente responsável, a informação de melhora e a solução encontrada (linha 15). Essa etapa corresponde, na implementação, à atualização do valor associado ao braço selecionado. Note que ela ocorre independentemente de a solução ter sido aceita: o aprendizado registra a qualidade do que a solução coletada produziu, e não o destino que o agente deu ao resultado.

Por fim, se a solução gerada for aceita (linha 17), ela passa a representar a nova melhor solução corrente do agente (linha 18) e é inserida no *Pool* (linha 19). A solução corrente é então realinhada com a melhor solução conhecida ao término de toda iteração (linha 21), tenha havido aceitação ou não, de modo que a perturbação da iteração seguinte parte sempre do melhor ponto já alcançado pelo agente.

6.2.2 Definição da Experimentação

A avaliação do *Pool* multiobjetivo segue a mesma configuração experimental adotada nas Seções 5.2.7 e 6.1.5: o cenário ILS_MIX com 20 agentes, as 24 instâncias de Solomon descritas na Seção 3.3 e 30 execuções independentes por instância. As métricas de avaliação são o *gap* percentual em relação ao melhor valor conhecido e o tempo médio de execução em segundos, calculados conforme a Equação (3.2), e a análise estatística mantém a abordagem não paramétrica adotada nos demais experimentos desta tese, justificada na Seção 3.3.

O objetivo deste experimento é isolar o efeito da reestruturação do *Pool* proposta na Seção 6.2.1. Para isso, adota-se uma comparação *cross-pool*: as políticas de seleção são executadas sobre o novo *Pool* multiobjetivo, mantido por dominância de Pareto e *crowding distance* (`pool_multi`), e confrontadas diretamente com as mesmas políticas

executadas sobre o *Pool* de diversidade original (`pool_diversity`), usado nas seções anteriores. Dessa forma, cada par de cenários difere exclusivamente na estrutura de manutenção do *Pool*, permitindo atribuir eventuais diferenças de desempenho à organização multiobjetivo das soluções armazenadas.

Os parâmetros de manutenção do *Pool* multiobjetivo, utilizados no Algoritmo 11, são apresentados na Tabela 6.8.

Tabela 6.8: Parâmetros de manutenção do *Pool* multiobjetivo.

Parâmetro	Variável	Valor
Tamanho máximo do <i>Pool</i>	<i>max_size</i>	100
Número mínimo de soluções	<i>number_minimum</i>	10
Limiar de inserções para filtragem	<i>number_insertion</i>	15
Raio de nicho (diversidade <i>shi</i>)	<i>radius_niche</i>	10

O limiar *number_insertion* determina de quantas em quantas inserções o procedimento de filtragem multiobjetivo é acionado: a cada 15 inserções (ou sempre que o tamanho máximo é excedido) executa-se a ordenação por dominância de Pareto seguida do cálculo da *crowding distance*, mantendo no *Pool* entre *number_minimum* e *max_size* soluções. O raio de nicho é empregado no cálculo da medida de diversidade estrutural (*shi*). Esses valores são idênticos aos adotados no *Pool* de diversidade original, de modo que a comparação *cross-pool* isola exclusivamente o efeito da estrutura de manutenção das soluções.

São avaliadas quatro políticas executadas sobre o *Pool* multiobjetivo, quais sejam: a melhor variante multiagente (PM-PSO-DUCB) e as três variantes multiobjetivo (PM-MO-PSO-DPUCB, PM-MO-PSO-SWUCB e PM-MO-PSO-mPQL), identificadas pelo prefixo PM (*Pool* Multiobjetivo). Como referências, incluem-se o cenário escalar MULTI-EG (`multi_epsilon_greedy`) e as mesmas políticas executadas sobre o *Pool* de diversidade: PSO-DUCB (melhor multiagente, Seção 5.2.9), MO-PSO-DPUCB (melhor multiobjetivo, Seção 6.1.6), MO-PSO-SWUCB e MO-PSO-mPQL. A Tabela 6.9 resume os nove cenários comparados. Os parâmetros das políticas de seleção são idênticos aos das Seções 5.2 e 6.1, de modo que apenas o mecanismo de manutenção do *Pool* (Algoritmo 11) varia entre as abordagens.

Tabela 6.9: Cenários experimentais do *Pool* Multiobjetivo. O prefixo PM indica execução sobre o *Pool* multiobjetivo (`pool_multi`); os demais cenários operam sobre o *Pool* de diversidade (`pool_diversity`).

Cenário	Estrutura do <i>Pool</i>	Política de seleção
MULTI-EG	diversidade	<code>multi_epsilon_greedy</code> (referência escalar)
PSO-DUCB	diversidade	<code>pso_ducb</code> (melhor multiagente)
MO-PSO-DPUCB	diversidade	<code>pso_morl_dpucb</code> (melhor multiobjetivo)
MO-PSO-SWUCB	diversidade	<code>pso_morl_swpuqb</code>
MO-PSO-mPQL	diversidade	<code>pso_morl_mpql</code>
PM-PSO-DUCB	multiobjetivo	<code>pso_ducb</code>
PM-MO-PSO-DPUCB	multiobjetivo	<code>pso_morl_dpucb</code>
PM-MO-PSO-SWUCB	multiobjetivo	<code>pso_morl_swpuqb</code>
PM-MO-PSO-mPQL	multiobjetivo	<code>pso_morl_mpql</code>

6.2.3 Resultados Computacionais

Esta seção apresenta os resultados da comparação *cross-pool* definida na Seção 6.2.2, confrontando os nove cenários da Tabela 6.9: as políticas executadas sobre o *Pool* multiobjetivo (prefixo PM), suas versões equivalentes sobre o *Pool* de diversidade original e a referência escalar MULTI-EG. A análise é organizada em quatro perspectivas complementares:

- (i) a qualidade das soluções, medida pelo *gap* percentual em relação ao melhor valor conhecido, examinada por instância, por bloco de instâncias e pela melhor solução encontrada entre as execuções independentes;
- (ii) o custo computacional, avaliado pelo tempo médio de execução por instância e por bloco;
- (iii) a evolução temporal da qualidade das soluções ao longo da busca, com destaque para os pares *cross-pool* de maior interesse; e
- (iv) o perfil de desempenho de Dolan e Moré (2002), que sintetiza a robustez relativa dos cenários sobre todo o conjunto de instâncias.

Em cada perspectiva, as diferenças observadas são submetidas à análise estatística não paramétrica descrita na Seção 3.3, de modo a verificar se a reestruturação multiobjetivo do *Pool* afeta a qualidade das soluções, o tempo de execução, ou ambos.

6.2.3.1 Análise do *gap* percentual

A Tabela 6.10 apresenta a média do *gap* percentual por instância para os nove cenários avaliados.

Tabela 6.10: Média do *gap* (%) por instância e cenário. Cenários PM operam sobre o *Pool* multiobjetivo. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-DUCB	PM-PSO	MO-PSO-DP	PM-MO-PSO-DP	MO-PSO-SW	PM-MO-PSO-SW	MO-PSO-mPQL	PM-MO-PSO-mPQL
C101	0,01	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C109	0,01	0,01	0,01	0,00	0,02	0,01	0,02	0,00	0,00
C204	0,12	0,10	0,10	0,11	0,13	0,11	0,12	0,12	0,09
C205	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R102	0,59	0,93	1,48	0,44	1,12	0,44	0,76	0,45	0,41
R104	9,93	9,97	9,94	9,97	9,92	9,99	9,95	9,99	9,92
R106	6,60	4,20	4,89	2,10	5,62	2,57	4,12	3,27	4,41
R112	10,03	10,07	9,97	10,08	10,05	10,08	10,01	10,08	10,02
R201	0,23	0,26	0,21	0,20	0,17	0,15	0,22	0,17	0,20
R208	0,54	0,51	0,50	0,49	0,51	0,45	0,50	0,55	0,52
R209	0,86	0,71	0,80	0,70	0,84	0,78	0,64	0,85	0,76
R211	31,72	31,95	31,88	31,98	31,83	31,79	31,60	31,74	31,64
RC102	11,04	9,81	8,11	8,91	9,56	8,93	10,27	9,21	10,30
RC103	3,67	2,90	3,70	1,87	3,08	2,39	4,44	2,21	3,35
RC105	7,45	9,01	7,66	6,57	8,49	6,82	8,60	7,27	9,25
RC108	8,81	8,54	8,53	8,88	8,52	8,91	7,95	8,87	9,14
RC201	0,73	0,84	0,70	0,76	0,64	0,84	0,77	0,82	0,76
RC202	18,92	18,83	18,92	19,02	19,02	18,87	18,74	18,95	18,84
RC206	1,59	1,64	1,49	1,54	1,10	1,30	2,36	1,31	1,22
RC208	1,38	1,25	1,02	1,45	1,29	1,34	1,16	1,35	1,17
Média	4,76	4,65	4,58	4,38	4,66	4,41	4,68	4,47	4,67

As instâncias do grupo C mantêm *gaps* próximos de zero em todos os cenários, confirmando sua facilidade de solução. Nas instâncias mais desafiadoras, as políticas executadas sobre o *Pool* multiobjetivo produzem os melhores resultados individuais em diversos casos: em RC102, o PM-PSO-DUCB alcança o menor *gap* entre os nove cenários (8,11%, contra 11,04% do MULTI-EG e 9,81% do PSO-DUCB no *Pool* de diversidade); em RC108, o PM-MO-PSO-SWUCB lidera com 7,95%; em RC206 e RC201, o PM-MO-PSO-DPUCB obtém os menores valores (1,10% e 0,64%); em RC208, o PM-PSO-DUCB reduz o *gap* para 1,02%; e em R102 e C204, o PM-MO-PSO-mPQL atinge os melhores resultados (0,41% e 0,09%). Em termos de média global, o MO-PSO-DPUCB sobre o *Pool* de diversidade permanece como menor *gap* médio (4,38%), mas o PM-PSO-DUCB (4,58%) supera todas as demais referências multiagente e escalar, inclusive o próprio PSO-DUCB no *Pool* de diversidade (4,65%). O teste de Friedman, pareado por instância, não revelou diferença estatisticamente significativa entre os nove cenários ($p = 0,47$), e nenhum dos 36 pares apresentou diferença no teste de Wilcoxon *signed-rank* com correção de Bonferroni; por bloco, o resultado se repete em C ($p = 0,278$), R (0,225) e RC (0,786), confirmando que a reestruturação multiobjetivo do *Pool* preserva a qualidade final das soluções.

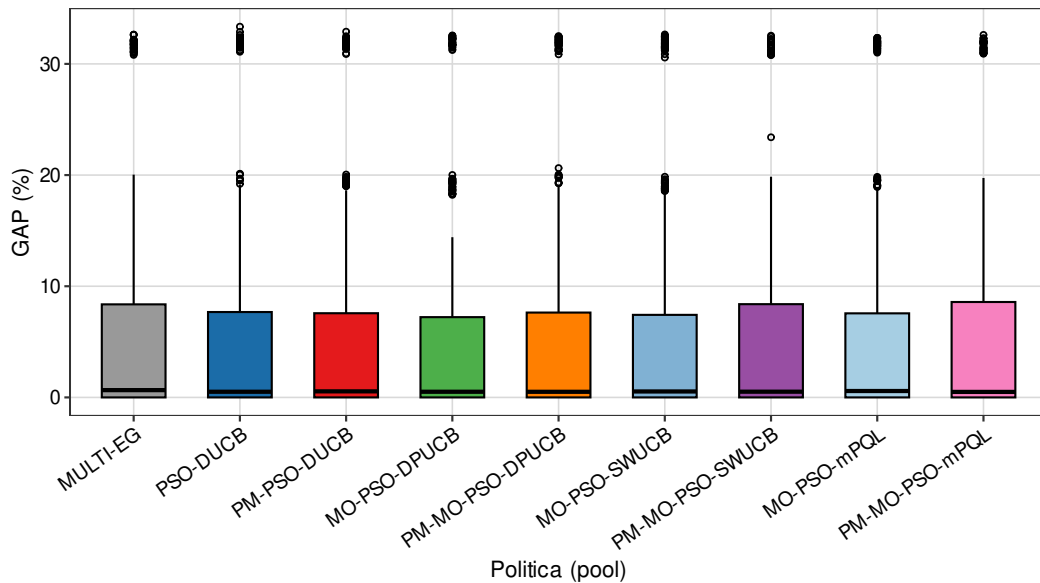


Figura 6.8: Distribuição do *gap* (%) por cenário ao longo das 24 instâncias avaliadas. Cenários PM operam sobre o *Pool* multiobjetivo.

A Figura 6.8 apresenta os *boxplots* do *gap* percentual dos nove cenários, dispostos de modo que cada cenário PM fique ao lado de sua contraparte no *Pool* de diversidade. As distribuições são visualmente indistinguíveis: mediana próxima de zero, caixa até cerca de 8%, bigode superior em torno de 20% e a mesma nuvem de *outliers* próxima de 31% da instância R211. Nenhum cenário PM apresenta deslocamento perceptível em relação ao seu par, registrando-se apenas um ponto isolado próximo de 23% no PM-MO-PSO-SWUCB. A Figura 6.8 corrobora graficamente a conclusão de que a reestruturação multiobjetivo do *Pool* preserva a qualidade final das soluções.

A Figura 6.9 desdobra a distribuição por bloco e reproduz, para os nove cenários, o mesmo comportamento verificado no *Pool* de diversidade. O bloco C reduz-se a uma linha sobre o zero em todos os cenários, enquanto os blocos R e RC concentram

Tabela 6.11: Média do *gap* (%) por bloco de instâncias e cenário. Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Bloco	MULTI-EG	PSO-DUCB	PM-PSO	MO-PSO-DP	PM-MO-PSO-DP	MO-PSO-SW	PM-MO-PSO-SW	MO-PSO-mPQL	PM-MO-PSO-mPQL
C	0,02	0,01	0,01	0,01	0,02	0,02	0,02	0,01	0,01
R	7,56	7,33	7,46	7,00	7,51	7,03	7,23	7,14	7,24
RC	6,70	6,60	6,27	6,12	6,46	6,18	6,79	6,25	6,75
Média	4,76	4,65	4,58	4,38	4,66	4,41	4,68	4,47	4,67

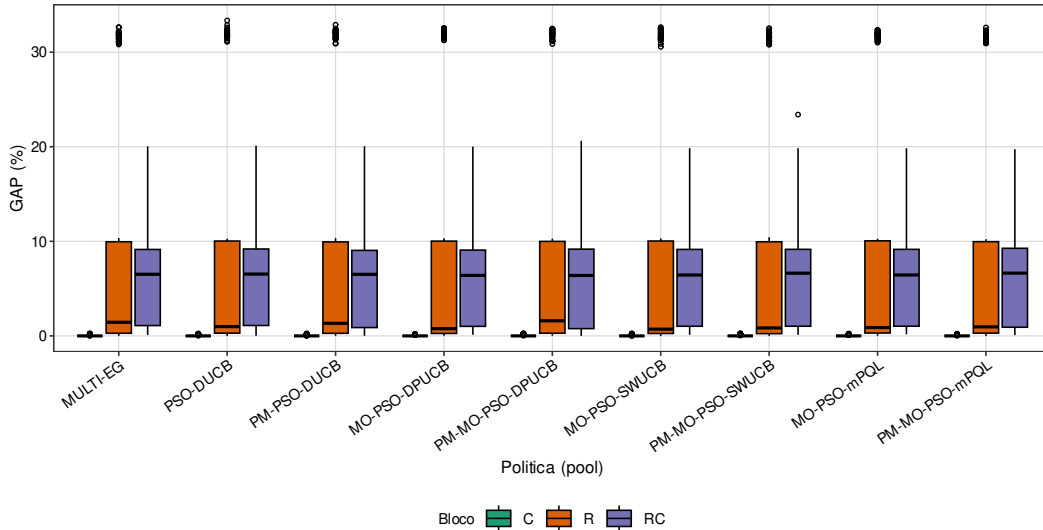


Figura 6.9: Distribuição do *gap* (%) por cenário e bloco de instâncias (C, R e RC).

a totalidade da dispersão, com mediana baixa em R e em torno de 6% em RC. A comparação par a par entre cada cenário PM e sua versão correspondente não revela deslocamento sistemático em nenhum dos três blocos, confirmando que o efeito do *Pool* multiobjetivo não se manifesta na qualidade das soluções, mas no tempo de execução, analisado na sequência.

A Tabela 6.11 detalha o comportamento por bloco. No bloco RC, o PM-PSO-DUCB (6,27%) é o segundo melhor cenário geral, à frente de todas as variantes do *Pool* de diversidade exceto o MO-PSO-DPUCB (6,12%) e superando o PSO-DUCB original (6,60%). No bloco R, as variantes MO sobre o *Pool* de diversidade mantêm leve vantagem média, mas as diferenças permanecem dentro da faixa de equivalência estatística observada globalmente.

Para avaliar o potencial de qualidade de cada cenário, a Tabela 6.12 apresenta o menor *gap* obtido em cada instância, ou seja, a melhor solução encontrada entre as 30 execuções. A linha *Vitórias* contabiliza em quantas instâncias cada cenário atinge o menor *gap* (empates incluídos), contagem também resumida na Figura 6.10.

Em termos da melhor solução, o PM-PSO-DUCB (15 vitórias) e o PM-MO-PSO-mPQL (14) atingem o menor *gap* no maior número de instâncias, confirmando que os cenários executados sobre o *Pool* multiobjetivo (prefixo PM) tendem a alcançar as melhores soluções. Observa-se, contudo, que o PM-MO-PSO-mPQL obtém muitas vitórias por empate, sendo penalizado por um resultado isolado de pior desempenho na instância RC108 (8,94%).

Tabela 6.12: Menor *gap* (%) por instância e cenário (melhor solução entre as 30 execuções). A linha *Vitórias* indica em quantas instâncias cada cenário atinge o menor *gap* (empates incluídos). Por se tratar de um problema de minimização, os melhores valores são os menores e estão destacados em negrito.

Instância	MULTI-EG	PSO-DUCB	PM-PSO	MO-PSO-DP	PM-MO-PSO-DP	MO-PSO-SW	PM-MO-PSO-SW	MO-PSO-mPQL	PM-MO-PSO-mPQL
C101	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C103	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C107	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C109	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C204	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C205	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C206	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C208	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R102	0,01	0,00	0,02	0,05	0,02	0,02	0,01	0,03	0,00
R104	9,79	9,79	9,77	9,82	9,80	9,84	9,79	9,82	9,78
R106	0,10	0,09	0,10	0,11	0,09	0,00	0,01	0,11	0,00
R112	9,82	9,92	9,78	9,83	9,91	9,82	9,80	9,89	9,87
R201	0,01	0,02	0,01	0,02	0,02	0,02	0,02	0,02	0,02
R208	0,03	0,03	0,03	0,14	0,03	0,13	0,09	0,16	0,03
R209	0,03	0,18	0,03	0,20	0,18	0,20	0,03	0,34	0,03
R211	30,81	31,09	30,89	31,25	30,86	30,57	30,80	31,01	30,92
RC102	6,81	6,96	6,85	7,04	7,05	7,07	6,95	6,90	6,84
RC103	0,10	0,12	0,11	0,12	0,09	0,12	0,11	0,19	0,08
RC105	6,32	6,31	6,31	6,34	6,24	6,30	6,25	6,40	6,34
RC108	0,34	0,34	0,10	0,62	0,02	0,47	0,24	0,40	8,94
RC201	0,21	0,10	0,32	0,12	0,21	0,32	0,22	0,34	0,21
RC202	18,22	18,22	18,22	18,22	18,22	18,22	18,22	18,22	18,22
RC206	0,37	0,18	0,36	0,18	0,00	0,45	0,50	0,18	0,18
RC208	0,40	0,01	0,01	0,28	0,05	0,56	0,11	0,17	0,46
Vitórias	13	13	15	9	13	11	10	9	14

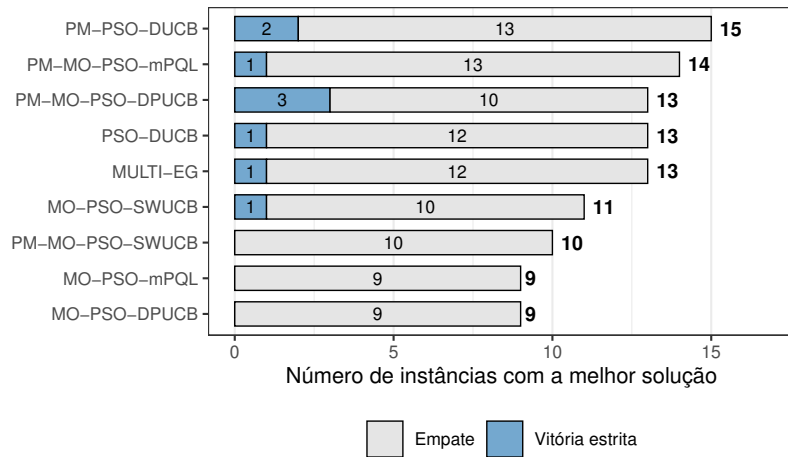


Figura 6.10: Número de instâncias em que cada cenário encontra a melhor solução (menor *gap*), separando vitórias estritas (único mínimo) de empates.

6.2.3.2 Análise do tempo de execução

A Tabela 6.13 apresenta a média do tempo de execução por instância para os nove cenários avaliados, e a Tabela 6.14 agrega esses valores por bloco. Os valores em negrito indicam o menor tempo.

Os cenários executados sobre o *Pool* multiobjetivo destacam-se também em eficiência: o PM-MO-PSO-DPUCB obteve a menor média global (57,70 s), seguido de perto pelo PSO-DUCB (58,01 s) e pelo PM-MO-PSO-SWUCB (58,04 s). Considerando o menor tempo por instância, os cenários PM lideram em 15 das 24 instâncias (PM-MO-PSO-SWUCB em 7 e PM-MO-PSO-DPUCB em 6). As exceções são o PM-MO-PSO-mPQL (70,94 s), o mais lento do conjunto, e o MULTI-EG (67,77 s), penalizado pelas instâncias C205 e C206.

A Tabela 6.14 revela o principal benefício da reestruturação do *Pool*: a manuten-

ção multiobjetivo reduz expressivamente o tempo de execução das variantes MO nos blocos mais difíceis. No bloco R, o PM-MO-PSO-DPUCB (59,87 s) e o PM-MO-PSO-SWUCB (59,79 s) são cerca de 8% mais rápidos que suas versões no *Pool* de diversidade (MO-PSO-DPUCB, 65,26 s, e MO-PSO-SWUCB, 65,40 s). No bloco RC, a redução é ainda maior: o PM-MO-PSO-DPUCB (53,09 s) e o PM-MO-PSO-SWUCB (53,15 s) superam em cerca de 9% as versões originais (58,17 s e 58,08 s). Esse ganho decorre da filtragem por dominância de Pareto e *crowding distance*, que mantém o *Pool* mais compacto e diverso, acelerando a etapa de seleção das soluções. Na média global, o PM-MO-PSO-DPUCB torna-se o cenário mais rápido (57,70 s), à frente do PSO-DUCB (58,01 s). A exceção é o PM-MO-PSO-mPQL, cuja média (70,94 s) é a maior do experimento, penalizada sobretudo pelo bloco C (80,15 s): a atualização vetorial *stateless* combinada à manutenção por dominância eleva o custo de processamento nas instâncias agrupadas. O teste de Friedman, pareado por instância, confirmou diferença altamente significativa entre os tempos ($p < 2 \times 10^{-16}$), com 16 dos 36 pares significativos; por bloco, o teste de Friedman acusa diferença em C ($p = 7,3 \times 10^{-7}$), R ($3,2 \times 10^{-4}$) e RC ($2,1 \times 10^{-7}$), embora nenhum par isolado sobreviva à correção de Bonferroni dentro de um bloco de 8 instâncias, e o *post-hoc* de Wilcoxon não acusou diferença significativa entre PM-MO-PSO-DPUCB, PM-MO-PSO-SWUCB e PSO-DUCB, que formam o agrupamento mais rápido.

A Figura 6.11 apresenta os *boxplots* do tempo de execução dos nove cenários e torna visível a dispersão que as médias resumem. Dois cenários destacam-se nos extremos: o MULTI-EG, cuja caixa é acompanhada de uma longa cauda de *outliers* que ultrapassa 300 s, e o PM-MO-PSO-mPQL, que apresenta a caixa mais alta do conjunto, deslocada para cima em relação a todas as demais. Os sete cenários restantes formam um grupo compacto, com caixas estreitas e valores extremos contidos em torno de 130 s, no qual o PM-MO-PSO-DPUCB exibe a caixa mais baixa. Ao contrário do que ocorre com o *gap*, o tempo de execução discrimina claramente os cenários, o que é coerente com a forte significância apontada pelo teste de Friedman.

A Figura 6.12 separa os tempos por bloco e localiza a origem dos dois extremos.

Tabela 6.13: Média do tempo de execução (s) por instância e cenário. Cenários PM operam sobre o *Pool* multiobjetivo.

Instância	MULTI-EG	PSO-DUCB	PM-PSO	MO-PSO-DP	PM-MO-PSO-DP	MO-PSO-SW	PM-MO-PSO-SW	MO-PSO-mPQL	PM-MO-PSO-mPQL
C101	74,20	56,23	63,53	53,70	57,43	53,33	57,30	53,60	71,37
C103	50,33	45,67	47,17	48,63	43,60	48,33	43,67	47,53	72,23
C107	63,40	49,27	59,37	47,40	48,70	48,13	48,73	47,43	83,73
C109	69,50	62,87	66,10	64,67	60,43	64,03	62,00	63,53	97,73
C204	91,40	89,90	88,27	92,50	86,27	94,97	89,10	92,33	97,00
C205	137,93	50,70	60,90	50,83	53,53	50,67	56,90	50,20	64,40
C206	109,23	59,00	71,73	59,77	64,47	58,43	63,63	59,40	76,97
C208	94,33	62,63	72,60	63,03	66,63	63,50	68,20	63,20	77,80
R102	44,27	45,83	53,03	58,80	49,97	56,97	49,47	57,20	49,63
R104	41,00	42,37	41,43	47,80	41,87	48,50	42,27	47,40	51,83
R106	43,63	40,80	43,70	45,63	41,17	44,57	40,13	44,23	48,33
R112	34,33	33,93	34,33	41,93	34,23	42,10	34,83	42,33	39,50
R201	64,23	61,07	62,83	62,70	59,80	62,10	58,87	61,93	76,17
R208	109,93	110,70	104,93	110,40	107,07	113,40	106,77	109,00	114,37
R209	66,73	64,87	64,57	69,07	65,53	69,63	66,87	68,73	70,93
R211	83,33	83,77	81,60	85,73	79,30	85,97	79,13	87,00	83,77
RC102	50,97	47,07	47,07	50,50	47,50	50,53	46,00	50,70	59,57
RC103	42,30	41,63	42,27	46,20	39,20	46,03	41,93	44,93	51,00
RC105	45,10	45,20	46,83	51,33	44,83	51,60	46,43	52,20	61,00
RC108	41,17	42,50	42,10	44,00	40,10	43,47	40,03	43,50	56,03
RC201	60,00	54,27	55,07	56,70	52,67	57,07	52,20	57,07	69,07
RC202	64,73	64,23	63,57	66,33	63,73	66,90	61,77	66,10	70,00
RC206	64,93	59,23	61,90	68,73	59,23	69,20	59,33	68,77	79,87
RC208	79,47	78,47	79,60	81,60	77,43	79,87	77,47	79,57	80,30
Média	67,77	58,01	60,60	61,17	57,70	61,22	58,04	60,74	70,94

Tabela 6.14: Média do tempo de execução (s) por bloco de instâncias e cenário.

Bloco	MULTI-EG	PSO-DUCB	PM-PSO	MO-PSO-DP	PM-MO-PSO-DP	MO-PSO-SW	PM-MO-PSO-SW	MO-PSO-mPQL	PM-MO-PSO-mPQL
C	86,29	59,53	66,21	60,07	60,13	60,17	61,19	59,65	80,15
R	60,93	60,42	60,80	65,26	59,87	65,40	59,79	64,73	66,82
RC	56,08	54,08	54,80	58,17	53,09	58,08	53,15	57,85	65,85
Média	67,77	58,01	60,60	61,17	57,70	61,22	58,04	60,74	70,94

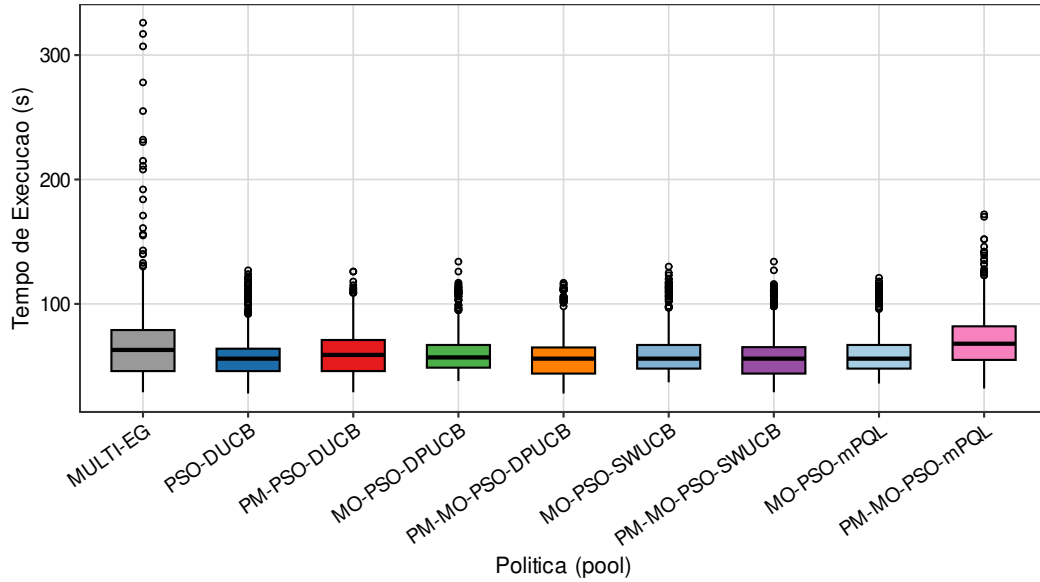


Figura 6.11: Distribuição do tempo de execução (s) por cenário ao longo das 24 instâncias avaliadas.

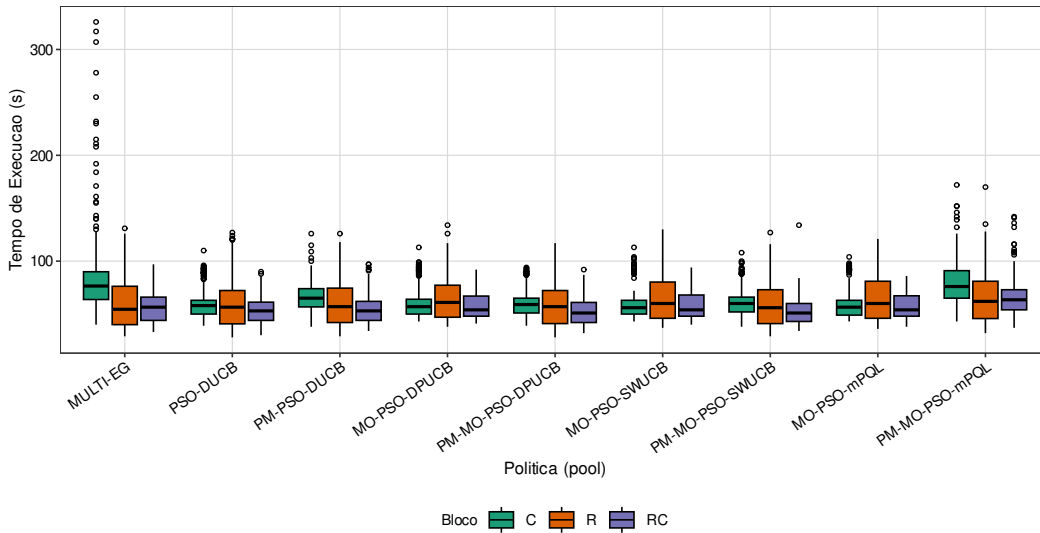


Figura 6.12: Distribuição do tempo de execução (s) por cenário e bloco de instâncias (C, R e RC).

A cauda do MULTI-EG e a caixa elevada do PM-MO-PSO-mPQL pertencem ambas ao bloco C, o mesmo grupo em que a Tabela 6.14 registra 86,29s e 80,15s, respectivamente. Nos blocos R e RC, a comparação par a par evidencia o efeito da reestruturação do *Pool*: as caixas do PM-MO-PSO-DPUCB e do PM-MO-PSO-SWUCB situam-se

visivelmente abaixo das caixas do MO-PSO-DPUCB e do MO-PSO-SWUCB, o que corresponde às reduções de cerca de 8% e 9% apontadas anteriormente. O ganho de eficiência do *Pool* multiobjetivo concentra-se, assim, nos blocos mais difíceis.

6.2.3.3 Análise da evolução da qualidade das soluções

A Figura 6.13 apresenta a evolução temporal do *gap* para seis instâncias representativas. As curvas destacadas correspondem aos dois pares *cross-pool* de maior interesse, PSO-DUCB *versus* PM-PSO-DUCB e MO-PSO-DPUCB *versus* PM-MO-PSO-DPUCB, permitindo isolar visualmente o efeito do *Pool* multiobjetivo sobre cada política; as demais variantes são exibidas em cinza para contexto.

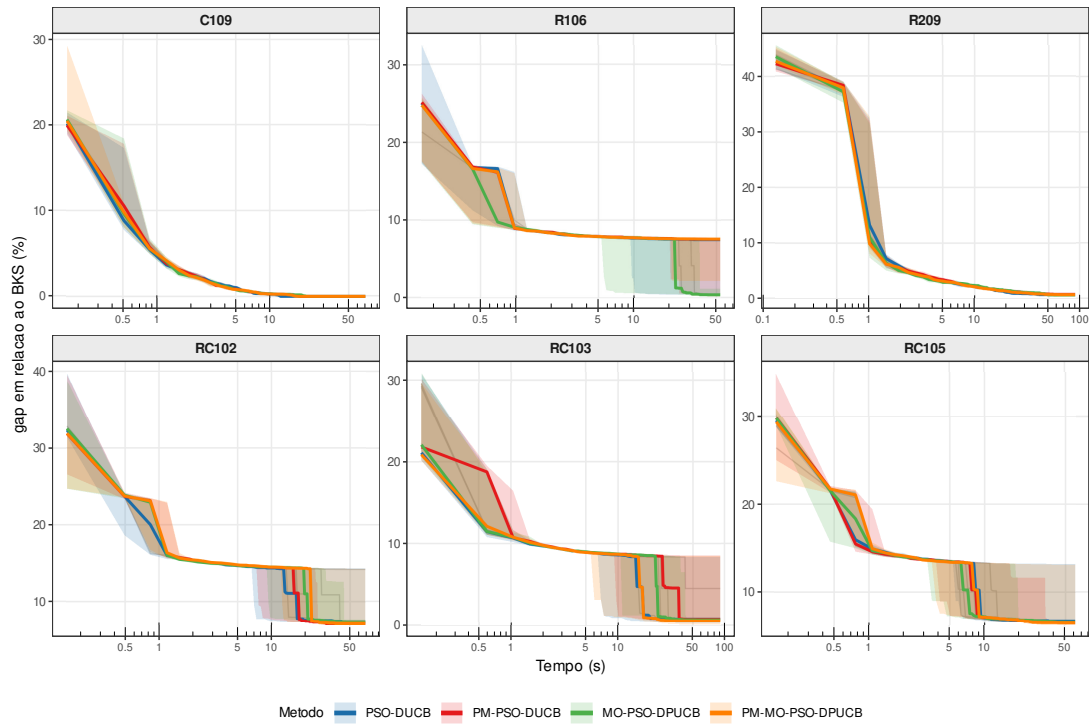


Figura 6.13: Evolução temporal do *gap* em relação ao melhor valor conhecido (%) para seis instâncias representativas. As curvas destacadas correspondem aos pares PSO-DUCB/PM-PSO-DUCB e MO-PSO-DPUCB/PM-MO-PSO-DPUCB; as demais variantes são mostradas em cinza. As faixas sombreadas indicam o IQR entre as 30 execuções independentes. O eixo x está em escala logarítmica.

Na instância C109, todos os métodos convergem rapidamente para *gap* nulo. Em RC102, o PM-PSO-DUCB apresenta a convergência mais favorável, estabilizando em valor inferior ao do PSO-DUCB sobre o *Pool* de diversidade, em consonância com a vantagem observada na Tabela 6.10. Em R106 e RC105, o MO-PSO-DPUCB sobre o *Pool* de diversidade preserva a melhor trajetória, evidenciando que, nessas instâncias, a vantagem do aprendizado vetorial não depende da reestruturação do *Pool*. Nas instâncias R209 e RC103, as trajetórias dos pares *cross-pool* permanecem próximas ao longo de toda a execução, reforçando a equivalência estatística de qualidade entre as estruturas de *Pool*.

6.2.3.4 Perfil de desempenho comparativo

A Figura 6.14 apresenta o perfil de desempenho proposto por Dolan e Moré (2002) para os nove cenários avaliados.

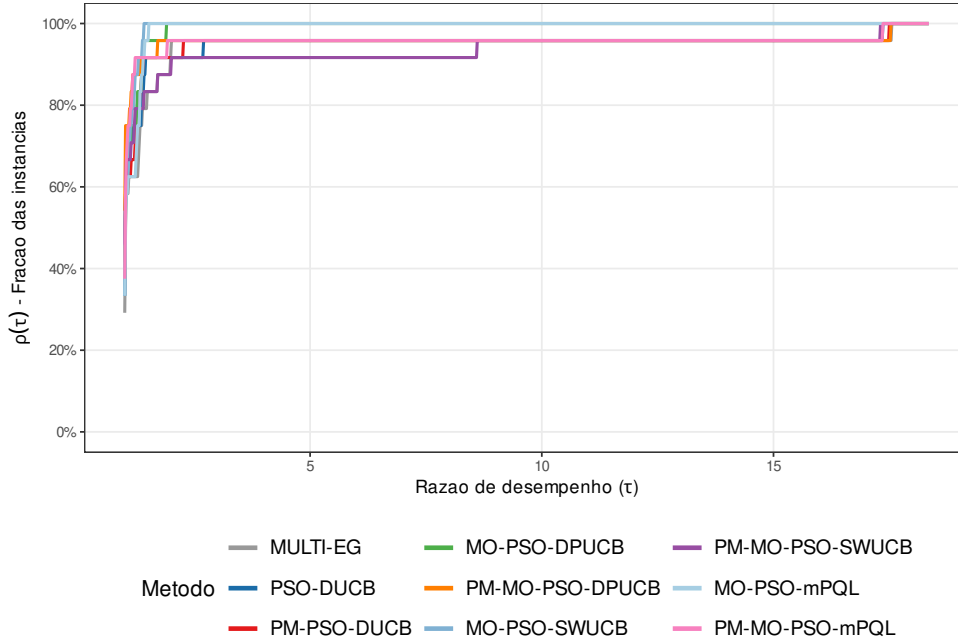


Figura 6.14: Perfil de desempenho (Dolan e Moré, 2002) dos nove cenários avaliados nas 24 instâncias de Solomon. O eixo x representa a razão de desempenho τ ; o eixo y indica a fração das instâncias em que o cenário possui razão $\leq \tau$.

O perfil evidencia o protagonismo das políticas executadas sobre o *Pool* multiobjetivo. Em $\tau = 1,01$, o PM-MO-PSO-DPUCB lidera com 62,5% de cobertura, seguido pelo PM-MO-PSO-SWUCB (58,3%); em $\tau = 1,05$, o PM-MO-PSO-DPUCB mantém a liderança com 75,0%. Em termos de vitórias diretas (menor *gap* por instância), os cenários PM dominam: o PM-MO-PSO-DPUCB lidera com 13/24 instâncias, seguido pelo PM-MO-PSO-SWUCB (11/24) e pelo PM-PSO-DUCB (10/24), contra 9/24 do MO-PSO-DPUCB no *Pool* de diversidade. Quanto ao limiar τ necessário para cobrir 90% das instâncias, o PM-MO-PSO-mPQL apresenta a maior robustez ($\tau = 1,226$), seguido pelo MO-PSO-SWUCB ($\tau = 1,295$) e pelo PM-MO-PSO-DPUCB ($\tau = 1,347$); o MULTI-EG ($\tau = 1,990$) e o PM-MO-PSO-SWUCB ($\tau = 2,007$) apresentam os piores valores, este último penalizado por instâncias isoladas de pior desempenho (RC206 e RC103).

6.2.4 Considerações Parciais

Esta subseção encerra a Seção 6.2, dedicada à reestruturação multiobjetivo do *Pool* de soluções, e não a Seção 6.1, cujas considerações parciais estão na Seção 6.1.7. Em síntese, os resultados confirmam que a reestruturação multiobjetivo do *Pool*, baseada em dominância de Pareto e *crowding distance*, preserva a qualidade final das soluções (equivalência estatística, $p = 0,47$ no teste de Friedman) e, simultaneamente, produz o maior número de vitórias diretas (PM-MO-PSO-DPUCB, 13/24) e reduz em 8–9% o tempo de execução das variantes MO nos blocos R e RC, eliminando

a penalidade de tempo dessas políticas observada sobre o *Pool* de diversidade na Seção 6.1.6. O PM-MO-PSO-DPUCB configura-se como o cenário de melhor equilíbrio entre qualidade, robustez e tempo, ao passo que o PM-MO-PSO-mPQL oferece a maior robustez de cobertura ao custo de maior tempo de processamento. Esses resultados respondem ao objetivo específico (ix) da Seção 1.3.2, demonstrando que organizar o *Pool* de cooperação segundo critérios multiobjetivo oferece suporte estrutural efetivo ao aprendizado vetorial proposto na Seção 6.1.

Os resultados da Seção 6.2 fecham o ciclo das contribuições da abordagem multiobjetivo proposta neste capítulo, demonstrando que a reestruturação do *Pool* por dominância de Pareto e *crowding distance* não apenas preserva a qualidade obtida com o aprendizado vetorial, mas também elimina a principal limitação computacional identificada na Seção 6.1. A análise *cross-pool* mostrou que os cenários com *Pool* multiobjetivo (prefixo PM) dominam a contagem de vitórias diretas: o PM-MO-PSO-DPUCB liderou com 13/24 instâncias no perfil de desempenho, contra 9/24 da versão equivalente sobre o *Pool* de diversidade. A redução de 8–9% no tempo de execução nos blocos R e RC, decorrente de um *Pool* mais compacto e estruturalmente diverso após cada filtragem multiobjetivo, aproximou o custo computacional das variantes MO ao nível do PSO-DUCB: o PM-MO-PSO-DPUCB tornou-se o cenário de menor tempo médio global (57,70 s), à frente do próprio PSO-DUCB (58,01 s). Tomados em conjunto, os resultados dos Capítulos 5 e 6 traçam uma trajetória coerente de evolução: do aprendizado escalar no *Pool* individual (Seção 5.1), passando pela cooperação multiagente com inspiração PSO (Seção 5.2), até o aprendizado vetorial com suporte estrutural multiobjetivo (Seções 6.1 e 6.2). Em cada etapa, a qualidade das soluções foi preservada ou aprimorada, e o custo computacional foi progressivamente reduzido, confirmando a viabilidade e a complementaridade das contribuições propostas para o *Framework* AMAM.

Capítulo 7

Considerações Finais e Direções Futuras

Este capítulo apresenta as considerações finais e as direções futuras deste trabalho. Está organizado da seguinte forma: a Seção 7.1 aborda as considerações finais obtidas com os experimentos realizados; a Seção 7.2 lista as publicações derivadas desta pesquisa; e a proposta de continuidade é apresentada na Seção 7.3.

7.1 Considerações Finais

O objetivo geral desta tese consistiu em estudar, implementar e adicionar técnicas de aprendizado por reforço, nas dimensões mono-agente, multiagente e multiobjetivo, na estrutura de um *framework* multiagente para a solução de problemas de otimização combinatória. O *framework* utilizado foi o AMAM. Para atingir esse objetivo, partiu-se de uma contextualização dos principais temas abordados, seguida da realização de testes computacionais voltados a compreender e validar as ferramentas contidas no *framework*, processo que se mostrou indispensável para a posterior implementação das técnicas de aprendizado propostas.

Com o propósito de alcançar o objetivo principal, o trabalho foi organizado em duas etapas complementares: (i) consolidação do *Framework* AMAM quanto à modelagem de recompensas, escalabilidade, cooperação e diversidade; e (ii) desenvolvimento e implementação de técnicas de aprendizado por reforço sobre o *Pool* de cooperação, em três dimensões progressivas: mono-agente (MaB *Pool*), multiagente (MAMaB-PSO) e multiobjetivo (*Pool* multiobjetivo e MaB multiobjetivo). As principais conclusões obtidas em cada uma dessas etapas são apresentadas a seguir.

- (i) Modelagem de recompensas: esta experimentação direcionou suas atenções em avaliar o *Framework* AMAM quanto à modelagem de recompensas do seu algoritmo de aprendizagem por reforço. Para isso, utilizou-se de uma aplicação, desenvolvida a partir do *Framework* AMAM, para a solução do PRVJT. As métricas desta proposta foram o valor da função objetivo e tempo de execução.

Testes computacionais foram realizados e observa-se que as modelagens de recompensas não apresentaram diferenças significativas entre a utilização das modelagens de recompensas e a modelagem de Silva et al. (2019), em que a recompensa é fortemente ligada ao problema. Portanto, as modelagens de recompen-

sas se mostram úteis, no contexto de *frameworks*, uma vez que não são ligadas a características específicas do problema. Indica-se a utilização de modelagens de recompensas baseadas em diferenças, uma vez que obtiveram os menores *gap*(%) e mostraram-se mais estáveis, em especial a modelagem *rd_lilo* (recompensa por diferença r_linha por r_local). Adicionalmente, indica-se a utilização da estrutura com oito agentes, visto que obteve os menores *gap*(%) e foi a única estrutura a melhorar as soluções das modelagens de recompensas em relação às melhores soluções obtidas a partir da modelagem de recompensas apresentada em [Silva et al. \(2019\)](#).

- (ii) Escalabilidade: este experimento avaliou a escalabilidade do *Framework AMAM*. O *Framework AMAM* é uma estrutura genérica e flexível, cuja principal força está na facilidade de incluir agentes meta-heurísticos com base no uso de conceitos relacionados a sistemas multiagente. O experimento realizado envolveu a criação de dez cenários, com diferentes números de agentes, que replicaram o agente ILS_ALS. As métricas usadas para avaliar a escalabilidade foram a qualidade das soluções obtidas, avaliada a partir dos valores da função objetivo e o tempo de execução.

Os resultados obtidos foram analisados usando testes estatísticos não paramétricos para compreender o comportamento dos dados obtidos. Essa análise demonstrou a escalabilidade do *Framework AMAM* em termos de crescimento no número de agentes na forma de melhor desempenho em relação à qualidade das soluções. Por outro lado, a análise estatística mostrou que, na maioria dos casos, a inclusão de novos agentes resultou em um aumento no tempo de execução. No entanto, esse aumento não é significativo o suficiente para comprometer o ganho de qualidade encontrado na função objetivo.

Uma vez que a melhoria no valor da função objetivo e a tendência de aumento do tempo de computação a partir da inserção de agentes foram verificadas, esta experimentação aplicou uma técnica de regressão linear sob as medidas estatísticas de média, mediana e melhores valores obtidos para determinar um modelo de ajuste linear que melhor representasse a inserção de agentes em relação ao valor da função objetivo e ao tempo de execução. A regressão linear para o valor da função objetivo apresentou bons ajustes lineares, alcançando uma boa função linear para representar a inclusão de agentes em relação ao valor da função objetivo. Por outro lado, a regressão linear para o tempo de execução não apresentou bons ajustes lineares. O melhor ajuste foi obtido a partir dos melhores valores para o tempo de execução e, portanto, não representou um bom comportamento do tempo de execução em relação à inserção de agentes.

Finalmente, o sistema de agente único foi comparado ao sistema multiagente. Os resultados medianos do sistema multiagente foram menores em todos os cenários, e testes não paramétricos encontraram uma diferença significativa. O tempo de execução foi ligeiramente maior ao usar sistemas multiagente, mas esse fato não invalida o uso dessa configuração, uma vez que a diferença encontrada foi pequena. É importante destacar que o tempo de execução dos sistemas multiagente foi menor em alguns casos em relação ao tempo de execução do sistema de agente único. Portanto, a análise dos resultados obtidos levou à

conclusão que o *Framework AMAM* é escalável e tem uma resposta robusta em relação ao aumento de agentes em sua estrutura.

- (iii) *Cooperação*: este experimento concentrou-se na avaliação da dimensão da cooperação no *Framework AMAM*. O objetivo da avaliação da cooperação no *framework* é validar as ferramentas de cooperação contidas em sua estrutura, uma vez que essa validação ainda não havia sido realizada para esta ferramenta. Para isso, esta experimentação utilizou uma aplicação desenvolvida com base no *Framework AMAM* para resolver o PRVJT. As métricas usadas para avaliar esta proposta foram o valor da função objetivo, juntamente com o tempo de execução. A validação foi realizada por meio de testes estatísticos não paramétricos.

No que diz respeito à cooperação, a partir dos testes computacionais realizados, verifica-se a eficiência e as dificuldades das ferramentas de cooperação no processo de busca do *Framework AMAM* na forma em que ele está apresentado neste trabalho. A ferramenta de cooperação melhorou significativamente os valores da função objetivo obtidos nos experimentos. O tempo de execução foi menor quando a cooperação não foi empregada, devido ao fato de que, nas execuções sem cooperação, os agentes evoluíram menos. Isso ocorreu pois o critério de parada do algoritmo era o número de iterações sem melhora. Assim, os testes computacionais e os testes estatísticos validaram a cooperação no *Framework AMAM*.

- (iv) *Diversidade*: com o propósito de avaliar a diversidade dos agentes presentes no *Framework AMAM*, este experimento envolveu a instanciação de agentes minimamente distintos no *framework*. O objetivo era avaliar a heterogeneidade dos agentes e entender os resultados e o comportamento da estrutura em relação à coordenação e à comunicação entre os agentes. Da mesma forma que nos testes anteriores, esses testes buscaram resolver o PRVJT por meio da instanciação de agentes a partir do *Framework AMAM*. As métricas de avaliação incluíram os valores da função objetivo e o tempo de execução obtidos.

Os testes computacionais demonstraram que a diversidade, mesmo que pequena, resultou em melhores desempenhos para a estrutura. Em relação aos valores da função objetivo, os experimentos com diversidade melhoraram os resultados obtidos em [Silva et al. \(2019\)](#). O cenário ILSFI obteve os melhores resultados entre os cenários testados. Além dos valores da função objetivo, os cenários também melhoraram o tempo de execução. Todos os cenários apresentaram tempos menores em comparação com a execução do *framework* de [Silva et al. \(2019\)](#), sendo que o cenário ILSRI alcançou os melhores resultados em termos de tempo de execução. Pode-se, portanto, constatar a importância e a eficácia da diversidade em sistemas computacionais e de busca. A diversidade de agentes no *Framework AMAM* resultou em melhores desempenhos nos testes realizados, resultados esses que foram validados pelos testes estatísticos executados neste trabalho.

- (v) *Aprendizado mono-agente na cooperação (Multi-armed Bandit Pool)*: este trabalho desenvolveu um algoritmo de aprendizado por reforço baseado no *Multi-*

armed Bandit, com o objetivo de controlar a estrutura de cooperação do *Framework* AMAM. A implementação visou reduzir os passos desnecessários ou ineficientes durante o processo de cooperação. Foram introduzidos dois modos de coleta (**qmax** e **softmax**) e dois critérios de exclusão de soluções do *Pool* (SHI e Q). A avaliação foi realizada com base em valor da função objetivo, tempo de execução e testes estatísticos não paramétricos sobre 30 execuções por instância.

Observou-se que o cenário em que o aprendizado foi incorporado ao *framework* alcançou melhores resultados que a execução do algoritmo de Silva et al. (2019). Também se constatou que o uso do *Multi-armed Bandit Pool* foi superior em vários dos testes em relação ao cenário ILSMI, tanto em termos de valor da função objetivo quanto de tempo de execução. Os métodos de aprendizado, portanto, além de reduzirem o tempo de execução, podem melhorar as soluções obtidas em determinadas situações, e os testes estatísticos validaram essas premissas.

- (vi) Aprendizado multiagente com cooperação inspirada em PSO (MAMaB-PSO): com o objetivo de estender o MaB *Pool* ao cenário em que múltiplos agentes aprendem simultaneamente sobre o mesmo *Pool*, foi proposto o algoritmo *Multi-Agent Multi-armed Bandit with PSO-Inspired Cooperation* (MAMaB-PSO). O algoritmo incorpora os componentes cognitivo e social do *Particle Swarm Optimization* à estimativa de valor das soluções, mantendo a política de seleção como componente genérico e permitindo a instanciação de seis variantes (**psu_epsilon_greedy**, **psu_ucb**, **psu_ducb**, **psu_swucb**, **psu_frmab** e **psu_framab**).

Os experimentos realizados sobre 24 instâncias de Solomon mostraram que a incorporação dos mecanismos PSO ao MaB multiagente não compromete a qualidade das soluções (equivalência estatística confirmada pelo teste de Friedman, pareado por instância, com $p = 0,81$) e introduz ganhos computacionais relevantes nas variantes PSO-UCB e PSO-DUCB, que se configuram como as variantes Pareto-dominantes do experimento. A análise da evolução temporal evidenciou ainda que a política UCB com componente social PSO produz convergência mais estável que a referência sem PSO, especialmente em instâncias dos blocos R e RC.

- (vii) Aprendizado multiobjetivo no *Pool* (*Multi-armed Bandit* multiobjetivo): foi proposta uma versão multiobjetivo do MaB *Pool*, na qual o valor escalar q é substituído por um vetor de recompensa $[r_{fo}, r_{shi}, r_{rotas}]$ que captura, simultaneamente, qualidade da função objetivo, diversidade estrutural e número de rotas da solução resultante. A seleção das soluções passa a operar em duas etapas (construção da frente de Pareto seguida de escolha uniforme entre as soluções não dominadas) e são propostas quatro variantes da política: MO-PSO-UCB, MO-PSO-DPUCB, MO-PSO-SWUCB e uma versão *stateless* do *Pareto Q-learning* (MO-PSO-mPQL). Essa formulação alinha o nível de aprendizado sobre a coleta de soluções ao nível estrutural do *Pool*, ambos operando agora sob critérios multiobjetivo.

A avaliação sobre as 24 instâncias de Solomon mostrou que as variantes vetoriais produzem qualidade estatisticamente equivalente às referências escalar (MULTI-EG) e multiagente (PSO-DUCB) ($p = 0,975$), com o MO-PSO-DPUCB alcançando o menor *gap* médio (4,38%) e o MO-PSO-SWUCB a maior robustez no

perfil de desempenho. Esse comportamento evidencia a complementaridade entre as abordagens: o aprendizado vetorial favorece a qualidade e a robustez em instâncias adversas dos blocos R e RC, ao passo que a variante multiagente PSO-DUCB preserva vantagem em tempo de execução.

- (viii) *Pool* de soluções multiobjetivo: foi proposta uma extensão da estrutura do *Pool* para tratar simultaneamente os critérios de função objetivo (*fo*) e de diversidade estrutural (*shi*) por meio de dominância de Pareto e *crowding distance*. A manutenção do conjunto ativo passa, com isso, a equilibrar de forma explícita qualidade e diversidade, mantendo na estrutura de cooperação apenas soluções pertencentes às frentes de Pareto mais informativas, com soluções remanescentes descartadas para uma estrutura auxiliar.

Os experimentos sobre as 24 instâncias de Solomon, organizados como uma comparação *cross-pool* entre as políticas executadas sobre o novo *Pool* multiobjetivo e sobre o *Pool* de diversidade original, confirmaram que a reorganização multiobjetivo preserva a qualidade final das soluções (equivalência estatística pelo teste de Friedman, pareado por instância, $p = 0,47$) e, simultaneamente, reduz em 8–9% o tempo de execução das variantes vetoriais nos blocos R e RC, eliminando a penalidade computacional dessas políticas observada sobre o *Pool* de diversidade. A variante MO-PSO-DPUCB executada sobre o novo *Pool* obteve o maior número de vitórias diretas (13 das 24 instâncias) e o menor tempo médio global (57,70 s), enquanto a versão *stateless* MO-PSO-mPQL sobre a mesma estrutura apresentou a maior robustez de cobertura no perfil de desempenho.

Em conjunto, os experimentos computacionais realizados ao longo da tese permitem afirmar que a inclusão de modelagem de recompensas, a validação das dimensões de escalabilidade, cooperação e diversidade, e a incorporação de mecanismos de aprendizado por reforço escalar, multiagente e multiobjetivo se mostraram eficazes no aprimoramento do *Framework* AMAM como ferramenta para a resolução de problemas de otimização combinatória.

Os resultados obtidos consolidam o *Framework* AMAM como plataforma para a solução de problemas de otimização que permite a hibridização de meta-heurísticas com técnicas de aprendizado de máquina, agora estendida ao regime multiobjetivo por meio de aprendizado vetorial sobre o *Pool* de cooperação.

7.2 Publicações derivadas desta Pesquisa

As experimentações realizadas neste trabalho resultaram, até o momento da defesa da tese em submissões de artigos para eventos e revistas. Esta seção lista as publicações originadas desta pesquisa.

7.2.1 Artigo Apresentado em Evento

- (i) Título: *An analysis of reward shaping for reinforcement learning in a multi-agent framework for combinatorial optimization*

- Autores: Jardell Fillipe da Silva, Maria Amélia Lopes Silva e Sérgio Ricardo de Souza
- Evento: XIX Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2022)
- Local: Campinas, SP, Brasil
- Período: 28 de novembro a 01 de dezembro de 2022
- Páginas: 130–141
- DOI: <https://doi.org/10.5753/eniac.2022.227627>

7.2.2 Artigo Publicado em Revista

(i) Título: *A scalability analysis of a Multi-agent framework for solving combinatorial optimization via Metaheuristics*

- Autores: Jardell Fillipe da Silva, Maria Amélia Lopes Silva, Sérgio Ricardo de Souza e Marcone Jamilson Freitas Souza
- Revista: *Engineering Applications of Artificial Intelligence*
- Volume: 142, Número do artigo: 109738, fevereiro de 2025
- DOI: <https://doi.org/10.1016/j.engappai.2024.109738>

(ii) Título: *An application of Multi-armed Bandit Learning on a Multi-agent Framework for Metaheuristics*

- Autores: Jardell Fillipe da Silva, Maria Amélia Lopes Silva e Sérgio Ricardo de Souza
- Revista: *Memetic Computing*
- Volume: 18, Número do artigo: 48, agosto de 2026
- DOI: <https://doi.org/10.1007/s12293-026-00529-y>

7.2.3 Artigo Submetido a Revista

(i) Título: *MAMaB-PSO: A Multi-Agent Reinforcement Learning Multi-armed Bandit Algorithm with PSO-Inspired Cooperation*

- Autores: Jardell Fillipe da Silva, Maria Amélia Lopes Silva e Sérgio Ricardo de Souza
- Submetido em junho de 2026.

7.3 Proposta de Continuidade

A tese consolidou o *Framework* AMAM como ambiente para a incorporação de técnicas de aprendizado por reforço, nas dimensões mono-agente, multiagente e multi-objetivo, aplicadas ao gerenciamento do *Pool* de cooperação. As experimentações realizadas, no entanto, evidenciaram caminhos de aprofundamento que extrapolam o

escopo deste trabalho e podem orientar pesquisas futuras. Propõem-se, a seguir, as principais direções de continuidade:

- (i) Aplicação a outros problemas de otimização combinatória: para uma avaliação mais ampla da flexibilidade do *framework* e dos algoritmos aqui propostos, pretende-se aplicar a estrutura do *Framework* AMAM e seus mecanismos de aprendizado a outros problemas de otimização combinatória disponíveis na literatura, como problemas de *scheduling* e de localização de facilidades. Essa extensão permitirá investigar a generalidade das contribuições propostas em domínios além do roteamento de veículos.
- (ii) Diversidade de agentes via meta-heurísticas heterogêneas: [Marques e Júnior \(2023\)](#) destacam que a maioria dos métodos de aprendizagem por reforço emprega agentes homogêneos, enquanto a investigação sobre heterogeneidade se concentra em conjuntos de agentes com diferentes habilidades. Propõe-se, então, desenvolver agentes capazes de implementar diferentes meta-heurísticas para resolver o PRVJT e outros problemas dentro do *Framework* AMAM, incluindo métodos de busca local (VNS e GRASP) e métodos populacionais (algoritmos genéticos), e analisar o desempenho do *framework* ao empregar essas meta-heurísticas em uma mesma execução.
- (iii) Agentes baseados em aprendizado profundo: no estudo de [Nazari et al. \(2018\)](#), é elaborada uma estrutura para solucionar o Problema de Roteamento de Veículos na qual um modelo político único é treinado para resolver o problema e encontra soluções quase ótimas para uma ampla gama de instâncias. Até o momento, todos os agentes implementados no *Framework* AMAM são instanciados na forma de meta-heurísticas. Propõe-se, portanto, instanciar agentes de aprendizado por reforço puro, inclusive baseados em aprendizado profundo, na AMAM, fazendo-os cooperar, via *Pool*, com os agentes meta-heurísticos já implementados.
- (iv) Extensão do aprendizado multiobjetivo: as contribuições em aprendizado multiobjetivo apresentadas no Capítulo 6 mantêm o PRVJT em sua formulação mono-objetivo, deslocando a multiobjetividade para o nível do *Pool* e do aprendizado vetorial. Uma direção natural de continuidade consiste em estender essa multiobjetividade também à formulação do problema, tratando explicitamente, durante a busca dos agentes, critérios como número de veículos, distância total, balanceamento de rotas e tempo de espera. A integração desse cenário com as variantes vetoriais propostas (MO-PSO-UCB, MO-PSO-DPUCB, MO-PSO-SWUCB e MO-PSO-mPQL) abre espaço para investigar a interação entre não estacionariedade do *Pool* e dominância de Pareto sob múltiplos critérios.
- (v) Aprendizado por reforço multiagente avançado: propõe-se aprofundar o estudo de algoritmos de aprendizado por reforço multiagente além do MAMaB-PSO, incluindo algoritmos cooperativos com comunicação explícita, mecanismos de *credit assignment* entre agentes e técnicas de aprendizado profundo multiagente. Revisões recentes sobre MARL ([Buşoniu et al., 2010](#); [Canese et al., 2021](#); [Zhu et al., 2022](#)) apontam direções promissoras para enriquecer a dimensão social da AMAM e ampliar seu potencial de aplicação em cenários multiagente.

-
- (vi) Representação do processo de busca via redes de trajetória: propõe-se representar o comportamento dos agentes durante o processo de busca por meio de técnicas de redes de trajetória (Ochoa et al., 2021), com vistas a compreender de forma estruturada como diferentes variantes de aprendizado (escalar, multi-agente, multiobjetivo) percorrem o espaço de soluções e interagem com o *Pool* de cooperação.

Referências Bibliográficas

Agogino, Adrian K e Tumer, Kagan. (2008). Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, v. 17, n. 2, p. 320–338.

Auer, P. Finite-time analysis of the multiarmed bandit problem, (2002).

Balaji, P. G. e Srinivasan, D. (2010). An introduction to multi-agent systems. *Innovations in Multi-Agent Systems and Applications - 1*, p. 1–27. Springer. doi: 10.1007/978-3-642-14435-6_1.

Barroso, Luiz André; Hölzle, Urs e Ranganathan, Parthasarathy. (2019). *The data-center as a computer: Designing warehouse-scale machines*. Springer Nature.

Bishop, Christopher M e Nasrabadi, Nasser M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Blum, Christian e Roli, Andrea. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308. doi: 10.1145/937503.937505. URL <https://doi.org/10.1145/937503.937505>.

Buşoniu, Lucian; Babuška, Robert e De Schutter, Bart. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 38, n. 2, p. 156–172. ISSN 1558-2442. doi: 10.1109/TSMCC.2007.913919.

Buşoniu, Lucian; Babuška, Robert e De Schutter, Bart. (2010). *Multi-agent Reinforcement Learning: An Overview*, p. 183–221. Springer Berlin Heidelberg, Berlin, Heidelberg. doi: 10.1007/978-3-642-14435-6_7. URL https://doi.org/10.1007/978-3-642-14435-6_7.

Canese, Lorenzo; Cardarilli, Gian Carlo; Di Nunzio, Luca; Fazzolari, Rocco; Giardino, Daniele; Re, Marco e Spanò, Sergio. (2021). Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, v. 11, n. 11. ISSN 2076-3417. doi: 10.3390/app11114948. URL <https://www.mdpi.com/2076-3417/11/11/4948>.

Chang, William; Jafarnia-Jahromi, Mehdi e Jain, Rahul. Online learning for cooperative multi-player multi-armed bandits, (2021). arXiv:2109.03818.

Chowdhury, Sadat e Sklar, Elizabeth I. (2014). Exploring interaction, diversity and efficiency of biologically inspired evolutionary multiagent systems. *Proceedings of the*

8th International Conference on Bioinspired Information and Communications Technologies, BICT'14, p. 63–66, Brussels, BEL. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 9781631900532. doi: 10.4108/icst.bict.2014.257922. URL <https://doi.org/10.4108/icst.bict.2014.257922>.

Cuevas, Erik; Zaldívar, Daniel e Pérez-Cisneros, Marco. (2024). *Exploring the Potential of Agent Systems for Metaheuristics*, p. 11–74. Springer Nature Switzerland, Cham. doi: 10.1007/978-3-031-45561-2_2. URL https://doi.org/10.1007/978-3-031-45561-2_2.

da Costa Oliveira, Artur Leandro; Britto, André e Gusmão, Renê. (2023). Machine learning enhancing metaheuristics: a systematic review. *Soft Computing*, v. 27, n. 21, p. 15971–15998.

de Oliveira, Thiago BF; Bazzan, Ana LC; da Silva, Bruno C e Grunitzki, Ricardo. jul(2018). Comparing multi-armed bandit algorithms and q-learning for multiagent action selection: a case study in route choice. *2018 International Joint Conference on Neural Networks (IJCNN)*, p. 1–8. IEEE, jul(2018). doi: 10.1109/IJCNN.2018.8489655. ISSN: 2161-4407.

Deb, K.; Pratap, A.; Agarwal, S. e Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197.

Deb, Kalyanmoy. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester.

Deters, Ralph. (2001). Scalability and information agents. *ACM SIGAPP Applied Computing Review*, v. 9, n. 3, p. 13–20. doi: 10.1145/570132.570135. URL <https://doi.org/10.1145/570132.570135>.

Devlin, Sam; Yliniemi, Logan; Kudenko, Daniel e Tumer, Kagan. (2014). Potential-based difference rewards for multiagent reinforcement learning. Lomuscio, Alessio; Scerri, Paul; Bazzan, Ana e Huhns, Michael, editors, *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France*, p. 165–172, (2014).

Dolan, Elizabeth D. e Moré, Jorge J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, v. 91, n. 2, p. 201–213. doi: 10.1007/s101070100263.

Dorigo, M.; Maniezzo, V. e Colorni, A. Feb(1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 26, n. 1, p. 29–41. ISSN 1041-0492. doi: 10.1109/3477.484436.

Drugan, Madalina M. e Nowé, Ann. (2013). Designing multi-objective multi-armed bandits algorithms: A study. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, p. 1–8. IEEE, (2013). doi: 10.1109/IJCNN.2013.6707036.

- Esmaeili, Ahmad; Mozayani, Nasser; Motlagh, Mohammad Reza Jahed e Matson, Eric T. (2016). The impact of diversity on performance of holonic multi-agent systems. *Engineering Applications of Artificial Intelligence*, v. 55, p. 186–201. doi: <https://doi.org/10.1016/j.engappai.2016.06.011>. URL <https://www.sciencedirect.com/science/article/pii/S095219761630121X>.
- Feo, Thomas A e Resende, Mauricio GC. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133.
- Fernandes, F. C. sep(2009). Arquitetura multiagente metaheurísticos cooperativos para resolução de problemas de otimização combinatória. Dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, Brasil.
- Ferreira, Alexandre Silvestre; Gonçalves, Richard Aderbal e Pozo, Aurora. (2017). A multi-armed bandit selection strategy for hyper-heuristics. *2017 IEEE Congress on Evolutionary Computation (CEC)*, p. 525–532, (2017). doi: 10.1109/CEC.2017.7969356.
- Foerster, Jakob; Assael, Ioannis Alexandros; de Freitas, Nando e Whiteson, Shimon. (2016). Learning to communicate with deep multi-agent reinforcement learning. Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I. e Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., (2016). URL https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbe4-Paper.pdf.
- Garivier, Aurélien e Moulines, Eric. (2011). On upper-confidence bound policies for switching bandit problems. *Algorithmic Learning Theory*, p. 174–188, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Gaspar-Cunha, Antônio; Takahashi, Ricardo e Antunes, Carlos Henggeler. (2012). *Manual de Computação Evolutiva e Metaheurística*. Imprensa da Universidade de Coimbra e Editora da Universidade Federal de Minas Gerais, Coimbra, Portugal.
- Glover, Fred. (1990). Tabu search: A tutorial. *Interfaces*, v. 20, n. 4, p. 74–94.
- Golberg, David E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley, Reading, MA, EUA.
- Grzes, Marek. May(2017). Reward shaping in episodic reinforcement learning. *Sixteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, p. 565–573. ACM, May(2017). URL <https://kar.kent.ac.uk/60614/>.
- Hayes, Conor F.; Rădulescu, Roxana; Bargiacchi, Eugenio; Källström, Johan; Macfarlane, Matthew; Reymond, Mathieu; Verstraeten, Timothy; Zintgraf, Luisa M.; Dazeley, Richard; Heintz, Fredrik; Howley, Enda; Irissappane, Athirai A.; Mannion, Patrick; Nowé, Ann; Ramos, Gabriel; Restelli, Marcello; Vamplew, Peter e Roijers, Diederik M. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, v. 36, n. 1, p. 26.

- Huanca-Anquise, Candy A; Bazzan, Ana Lúcia Cetertich e Tavares, Anderson R. (2023). Multi-objective, multi-armed bandits: Algorithms for repeated games and application to route choice. *Revista de Informática Teórica e Aplicada*, v. 30, n. 1, p. 11–23.
- Iqbal, Shariq e Sha, Fei. 09–15 Jun(2019). Actor-attention-critic for multi-agent reinforcement learning. Chaudhuri, Kamalika e Salakhutdinov, Ruslan, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, p. 2961–2970. PMLR, 09–15 Jun(2019). URL <https://proceedings.mlr.press/v97/iqbal19a.html>.
- Juhasz, Z. e Paul, P. May(2002). Scalability analysis of the contract net protocol. *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, p. 346–346, May(2002). doi: 10.1109/CCGRID.2002.1017157.
- Kamdar, Renuka; Paliwal, Priyanka e Kumar, Yogendra. (2018). A state of art review on various aspects of multi-agent system. *Journal of Circuits, Systems and Computers*, v. 27, n. 11, p. 1830006. doi: 10.1142/S0218126618300064. URL <https://doi.org/10.1142/S0218126618300064>.
- Karimi-Mamaghan, Maryam; Mohammadi, Mehrdad; Meyer, Patrick; Karimi-Mamaghan, Amir Mohammad e Talbi, El-Ghazali. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, v. 296, n. 2, p. 393–422. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2021.04.032>. URL <https://www.sciencedirect.com/science/article/pii/S0377221721003623>.
- Kennedy, J. e Eberhart, R. Nov(1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, p. 1942–1948 vol.4, Nov(1995). doi: 10.1109/ICNN.1995.488968.
- Kirkpatrick, S.; Gelatt, C. D. e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680. doi: 10.1126/science.220.4598.671. URL <https://www.science.org/doi/abs/10.1126/science.220.4598.671>.
- Kocsis, Levente e Szepesvári, Csaba. (2006). Discounted UCB. *2nd PASCAL Challenges Workshop*, volume 2, p. 51–134, (2006).
- Korba, Larry e Song, Ronggong. (2002). Modeling and simulating the scalability of a multi-agent application system. *National Research Council of Canada*, v. .
- Lagos, Felipe e Pereira, Jordi. (2023). Multi-armed bandit-based hyper-heuristics for combinatorial optimization problems. *European Journal of Operational Research*, v. 312, n. 1, p. 70–91. doi: <https://doi.org/10.1016/j.ejor.2023.06.016>. URL <https://www.sciencedirect.com/science/article/pii/S0377221723004678>.
- Landgren, Peter; Srivastava, Vaibhav e Leonard, Naomi Ehrich. (2021). Distributed cooperative decision making in multi-agent multi-armed bandits. *Automatica*, v. 125, p. 109445. doi: 10.1016/j.automatica.2020.109445.

- Lee, Lyndon C; Nwana, Hyacinth S; Ndumu, Divine T e De Wilde, Philippe. (1998). The stability, scalability and performance of multi-agent systems. *BT Technology Journal*, v. 16, n. 3, p. 94–103.
- Leibo, Joel Z.; Zambaldi, Vinicius; Lanctot, Marc; Marecki, Janusz e Graepel, Thore. (2017). Multi-agent reinforcement learning in sequential social dilemmas. *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS '17*, p. 464–473, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. doi: 10.5555/3091125.3091194.
- Li, Ke; Fialho, Alvaro; Kwong, Sam e Zhang, Qingfu. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, v. 18, n. 1, p. 114–130. doi: 10.1109/TEVC.2013.2239648.
- Li, Xiaodong; Epitropakis, Michael G.; Deb, Kalyanmoy e Engelbrecht, Andries. Aug(2017). Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, v. 21, n. 4, p. 518–538. ISSN 1941-0026. doi: 10.1109/TEVC.2016.2638437.
- Liu, Keqin e Zhao, Qing. (2010). Distributed learning in multi-armed bandit with multiple players. *IEEE Transactions on Signal Processing*, v. 58, n. 11, p. 5667–5681. doi: 10.1109/TSP.2010.2062509.
- Lorig, Fabian; Dammenhayn, Nils; Müller, David-Johannes e Timm, Ingo J. (2015). Measuring and comparing scalability of agent-based simulation frameworks. Müller, Jörg P.; Ketter, Wolf; Kaminka, Gal; Wagner, Gerd e Bulling, Nils, editors, *Multiagent System Technologies*, p. 42–60, Cham. Springer International Publishing. ISBN 978-3-319-27343-3.
- Lourenço, Helena R; Martin, Olivier e Stützle, Thomas. (2001). A beginner's introduction to iterated local search. *Proceedings of the 4th Metaheuristics International Conf. (MIC 2001)*, volume 2, p. 1–6, (2001).
- Lowe, Ryan; Wu, Yi; Tamar, Aviv; Harb, Jean; Abbeel, Pieter e Mordatch, Igor. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Guyon, I.; Luxburg, U. Von; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S. e Garnett, R., editors, *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, p. 6382–6393, Red Hook, NY, USA. Curran Associates Inc. doi: 10.5555/3295222.3295385.
- Mahajan, Aditya e Teneketzis, Demosthenis. (2008). *Multi-Armed Bandit Problems*, p. 121–151. Springer US, Boston, MA. ISBN 978-0-387-49819-5. doi: 10.1007/978-0-387-49819-5_6. URL https://doi.org/10.1007/978-0-387-49819-5_6.
- Mann, H. B. e Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, v. 18, n. 1, p. 50–60. ISSN 00034851. URL <http://www.jstor.org/stable/2236101>.

- Marques, Rodrigo e Júnior, Zenilton Patrocínio. (2023). Impact of heterogeneity on multi-agent reinforcement learning. *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional*, p. 1048–1062, Porto Alegre, RS, Brasil. SBC. doi: 10.5753/eniac.2023.234582. URL <https://sol.sbc.org.br/index.php/eniac/article/view/25763>.
- Meidani, Kazem; Mirjalili, Seyedali e Barati Farimani, Amir. (2022). MAB-OS: Multi-armed bandits metaheuristic optimizer selection. *Applied Soft Computing*, v. 128, p. 109452. doi: <https://doi.org/10.1016/j.asoc.2022.109452>. URL <https://www.sciencedirect.com/science/article/pii/S1568494622005671>.
- Michael, Maged; Moreira, Jose E.; Shiloach, Doron e Wisniewski, Robert W. March(2007). Scale-up $\times$ scale-out: A case study using nutch/lucene. *2007 IEEE International Parallel and Distributed Processing Symposium*, p. 1–8, March(2007). doi: 10.1109/IPDPS.2007.370631.
- Mladenović, Nenad e Hansen, Pierre. (1997). Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097–1100.
- Montgomery, Douglas C e Runger, George C. (2010). *Applied statistics and probability for engineers*. John Wiley & Sons.
- Nazari, Mohammadreza; Oroojlooy, Afshin; Takáč, Martin e Snyder, Lawrence V. (2018). Reinforcement learning for solving the vehicle routing problem. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 9861–9871, Red Hook, NY, USA. Curran Associates Inc. doi: 10.5555/3327546.3327651.
- Ochoa, Gabriela; Malan, Katherine M. e Blum, Christian. (2021). Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, v. 109, p. 107492. ISSN 1568-4946. doi: <https://doi.org/10.1016/j.asoc.2021.107492>. URL <https://www.sciencedirect.com/science/article/pii/S1568494621004154>.
- Parejo, José Antonio; Ruiz-Cortés, Antonio; Lozano, Sebastián e Fernandez, Pablo. (2012). Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing*, v. 16, n. 3, p. 527–561.
- Rana, Omer F. e Stout, Kate. (2000). What is scalability in multi-agent systems? *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, p. 56–63, New York, NY, USA. Association for Computing Machinery. ISBN 1581132301. doi: 10.1145/336595.337033. URL <https://doi.org/10.1145/336595.337033>.
- Robbins, Herbert. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, v. 58, n. 5, p. 527–535.
- Rodrigues, Henrique Donâncio Nunes. *Avaliação de escalabilidade e desempenho da camada de transporte de mensagens em plataformas multiagente*. PhD thesis, Universidade de São Paulo, (2019). URL <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-07102019-150704/>.

- Rojiers, Diederik M.; Vamplew, Peter; Whiteson, Shimon e Dazeley, Richard. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, v. 48, p. 67–113.
- Silva, Jardell; Silva, Maria e Souza, Sérgio. (2022). An analysis of reward shaping for reinforcement learning in a multi-agent framework for combinatorial optimization. *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*, p. 130–141, Porto Alegre, RS, Brasil. SBC. doi: 10.5753/eniac.2022.227627. URL <https://sol.sbc.org.br/index.php/eniac/article/view/22776>.
- Silva, M. A. L.; de Souza, S. R.; Souza, M. J. F. e de Oliveira, S. M. Nov(2015). A multi-agent metaheuristic optimization framework with cooperation. *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, p. 104–109, Natal, Brazil. doi: 10.1109/BRACIS.2015.64.
- Silva, Maria Amélia Lopes. (2007). Modelagem de uma arquitetura multiagente para a solução, via metaheurísticas, de problemas de otimização combinatória. Dissertação de mestrado, Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, Brazil.
- Silva, Maria Amélia Lopes. (2019). AMAM: *Framework* multiagente para otimização usando metaheurísticas. Tese de doutorado, Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, Brazil.
- Silva, Maria Amélia Lopes; da Silva, Jardell Fillipe; de Souza, Sérgio Ricardo e Souza, Marcone Jamilson Freitas. (2025). A scalability analysis of a multi-agent framework for solving combinatorial optimization via metaheuristics. *Engineering Applications of Artificial Intelligence*, v. 142, p. 109738. ISSN 0952-1976. doi: 10.1016/j.engappai.2024.109738.
- Silva, Maria Amélia Lopes; de Souza, Sérgio Ricardo; Souza, Marcone Jamilson Freitas e Bazzan, Ana Lúcia C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications*, v. 131, p. 148–171.
- Silva, Maria Amélia Lopes; de Souza, Sérgio Ricardo; Souza, Marcone Jamilson Freitas e de França Filho, Moacir Felizardo. (2018). Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied Soft Computing*, v. 71, p. 433–459. doi: <https://doi.org/10.1016/j.asoc.2018.06.050>. URL <http://www.sciencedirect.com/science/article/pii/S1568494618303867>.
- Silver, David; Singh, Satinder; Precup, Doina e Sutton, Richard S. (2021). Reward is enough. *Artificial Intelligence*, v. 299, p. 103535.
- Slivkins, Aleksandrs. (2019). Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, v. 12, n. 1-2, p. 1–286. ISSN 1935-8237. doi: 10.1561/22000000068. URL <http://dx.doi.org/10.1561/22000000068>.

- Smith, Reid G. e Davis, Randall. (1981). Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 11, n. 1, p. 61–70. doi: 10.1109/TSMC.1981.4308579.
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 2, n. 35, p. 254–264.
- Sutton, Richard S e Barto, Andrew G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Talbi, El-Ghazali. (2021). Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys*, v. 54, n. 6. doi: 10.1145/3459664. URL <https://doi.org/10.1145/3459664>.
- Tampuu, Ardi; Matiisen, Tanel; Kodelja, Dorian; Kuzovkin, Ilya; Korjus, Kristjan; Aru, Juhan; Aru, Jaan e Vicente, Raul. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, v. 12, n. 4, p. e0172395.
- Tan, Ming. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proceedings of the tenth international conference on machine learning*, p. 330–337, (1993).
- Toth, Paolo e Vigo, Daniele. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.
- Turner, Phillip J. e Jennings, Nicholas R. (2001). Improving the scalability of multi-agent systems. Wagner, Tom e Rana, Omer F., editors, *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, p. 246–262, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Van Moffaert, Kristof e Nowé, Ann. (2014). Multi-objective reinforcement learning using sets of Pareto dominating policies. *Journal of Machine Learning Research*, v. 15, n. 1, p. 3483–3512.
- Wang, Juan e De Wilde, Philippe. (2007). Scalability of evolving networks of trading agents. *Intelligent Decision Technologies*, v. 1, n. 4, p. 197–204. ISSN 1872-4981. doi: 10.3233/IDT-2007-1403. URL <https://content.iospress.com/articles/intelligent-decision-technologies/idt00016>.
- Watkins, Christopher JCH e Dayan, Peter. (1992). Q-learning. *Machine learning*, v. 8, n. 3, p. 279–292.
- Wilcoxon, Frank. (1992). *Individual Comparisons by Ranking Methods*, p. 196–202. Springer New York, New York, NY. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_16. URL https://doi.org/10.1007/978-1-4612-4380-9_16.
- Wong, K. Y. Michael; Lim, S. W. e Gao, Zhuo. Jun(2005). Effects of diversity on multiagent systems: Minority games. *Physical Review E*, v. 71, p. 066103. doi: 10.1103/PhysRevE.71.066103. URL <https://link.aps.org/doi/10.1103/PhysRevE.71.066103>.

Wooldridge, Michael. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Zhang, Shuyan; Yang, Tailong; Liang, Jing e Yue, Caitong. (2023). A novel adaptive bandit-based selection hyper-heuristic for multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 53, n. 12, p. 7693–7706. doi: 10.1109/TSMC.2023.3299982.

Zhu, Changxi; Dastani, Mehdi e Wang, Shihan. A survey of multi-agent reinforcement learning with communication, (2022).