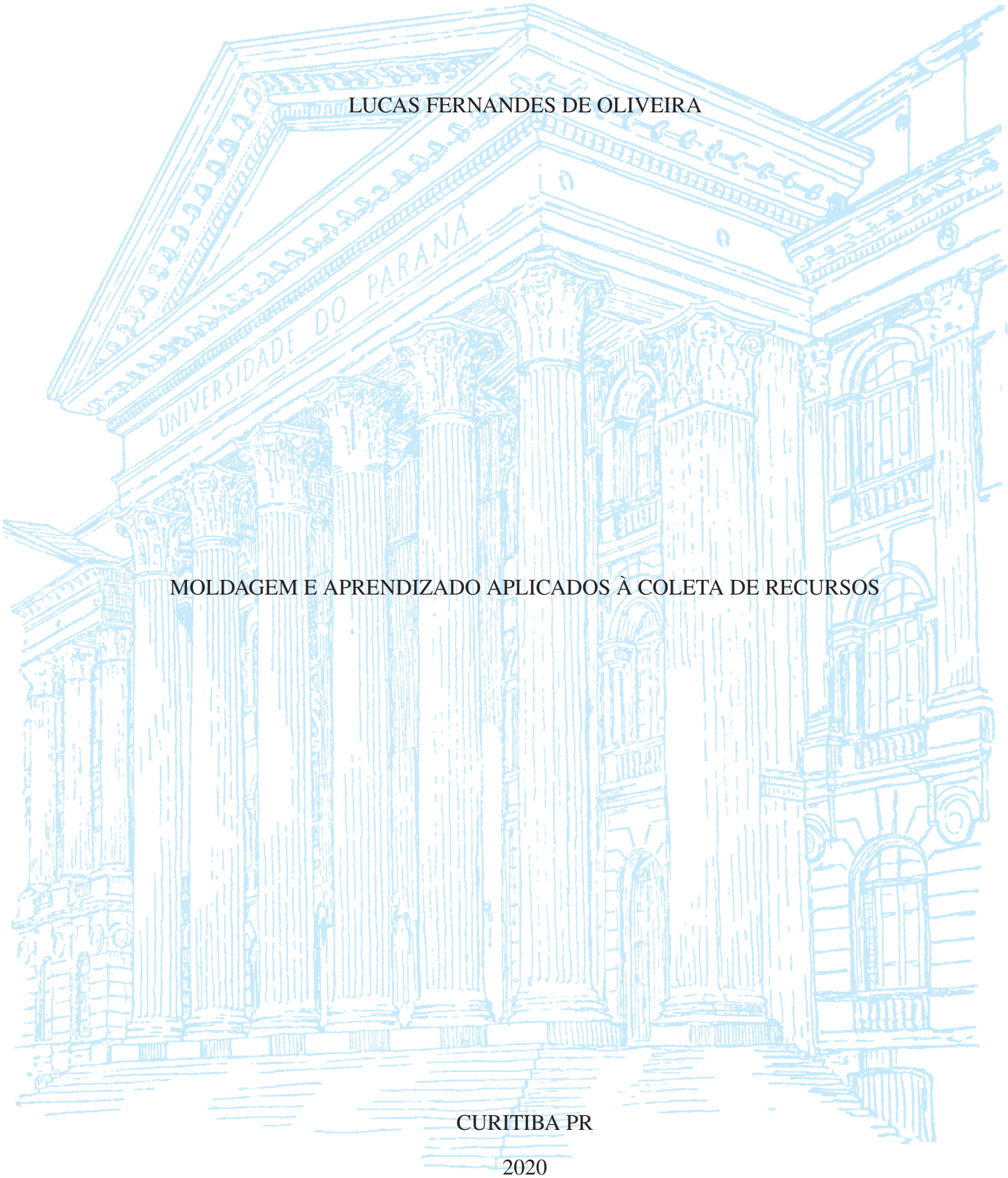
UNIVERSIDADE FEDERAL DO PARANÁ

LUCAS FERNANDES DE OLIVEIRA



MOLDAGEM E APRENDIZADO APLICADOS À COLETA DE RECURSOS

CURITIBA PR

2020

LUCAS FERNANDES DE OLIVEIRA

MOLDAGEM E APRENDIZADO APLICADOS À COLETA DE RECURSOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo J. Spinosa.

CURITIBA PR

2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

O48m Oliveira, Lucas Fernandes de
Moldagem e aprendizado aplicados à coleta de recursos [recurso eletrônico] / Lucas Fernandes de Oliveira. – Curitiba, 2020.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2020.

Orientador: Eduardo Jaques Spinosa.

1. Robótica. 2. Redes neurais (Computação). 3. Computação evolutiva. I. Universidade Federal do Paraná. II. Spinosa, Eduardo Jaques. III. Título.

CDD: 005.13

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **LUCAS FERNANDES DE OLIVEIRA** intitulada: **Moldagem e aprendizado aplicados à coleta de recursos**, sob orientação do Prof. Dr. EDUARDO JAQUES SPINOSA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 09 de Setembro de 2020.

Assinatura Eletrônica
10/09/2020 14:58:51.0
EDUARDO JAQUES SPINOSA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
10/09/2020 14:01:09.0
JOAO ALBERTO FABRO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
10/09/2020 14:08:30.0
EDUARDO TODT
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

A minha família e aos meus amigos.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a minha família por ter tornado essa conquista possível. Vocês me colocaram nesse caminho e me ajudaram a continuar nele nas horas mais difíceis.

Também gostaria de agradecer aos meus amigos, dos que conheço desde o ensino fundamental aos que conheci apenas durante o mestrado. Dos que falavam comigo sobre o mestrado toda semana aos que nem sabiam o que eu estava fazendo. Obrigado pela paciência, pelo incentivo, pelas danças, pelos jogos e pelas risadas. Vocês que me mantiveram a minha sanidade alta o suficiente para que o mestrado fosse concluído.

Gostaria de agradecer ao Centro de computação científica e Software Livre (C3SL) pela bolsa que me sustentou durante a maior parte do mestrado e por permitir a utilização dos recursos do laboratório para a minha pesquisa de mestrado, mesmo que ela não fosse diretamente ligada aos interesses do C3SL.

Gostaria de agradecer ao meu orientador por ter acreditado em algumas das minhas ideias loucas e permitir que uma deles se transformasse nessa pesquisa.

Por fim gostaria de agradecer especialmente a Marcela Ribeiro de Oliveira que esteve ao meu lado durante toda essa jornada. O mestrado não teria acontecido sem você. Não consigo descrever o quão importante foi a sua ajuda e o quão grato eu sou por ela. Obrigado!

RESUMO

Neste trabalho investiga-se a resolução do problema de coleta de recursos utilizando aprendizado por reforço e moldagem. Nesse problema, um agente deve explorar um ambiente desconhecido em busca de recursos e transportar esses recursos até um destino pré-definido. Quando aprendizado por reforço é utilizado, não é necessário escrever um algoritmo que controle as ações do agente, porém, esse tipo de técnica é custosa e demora para convergir. Para mitigar esses problemas em algoritmos de aprendizado existem diferentes técnicas. Neste trabalho pretende-se avaliar a técnica de moldagem. Moldagem consiste em prover ao agente informação privilegiada para guiar o aprendizado por caminhos promissores, reduzindo o esforço gasto ao explorar caminhos improdutivos durante esse processo. Para averiguar a utilidade da moldagem foi realizado um conjunto de experimentos para avaliar diferentes estratégias de moldagem em conjunto com o algoritmo de aprendizado NEAT para a resolução do problema de coleta de recursos. A partir dos resultados dos experimentos concluiu-se que moldagem afeta o treinamento do algoritmo NEAT, no entanto, melhores resultados são obtidos sem a utilização de moldagem.

Palavras-chave: moldagem, aprendizado por reforço, neuroevolução, robótica, redes neurais artificiais, computação evolutiva, coleta de recursos

ABSTRACT

This work researches the resolution of the foraging problem using reinforcement learning and shaping. In this problem, an agent explores a unknown environment searching for resources and delivering these resources to a pre-defined location. When using reinforcement learning to solve this problem, is not required to write an algorithm to control agent actions, however, such approach is expensive and slow to converge. There are some techniques used to mitigate these problems with learning algorithms. In this work, shaping is evaluated. Shaping provide the agent with privileged information used to guide the learning through promising paths, reducing the effort spent exploring unfruitful paths. To investigate the utility of shaping, a set of experiments were realised in order to evaluate different methods of shaping combined with the learning algorithm NEAT to solve the foraging problem. The results obtained through experiments show that shaping affects the performance of NEAT training however better results are obtained without shaping.

Keywords: shaping, reinforcement learning, neuroevolution, robotics, artificial neural networks, evolutionary computing, foraging

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Coleta de recursos é um problema conhecido no contexto de robótica. Nesse problema, um agente (robô) deve explorar um ambiente desconhecido em busca de recursos e transportar esses recursos até um destino pré-definido. Diversas situações como busca e resgate, coleta de alimentos entre outras podem ser reduzidas a esse problema (Winfield, 2009).

Embora o objetivo do problema seja claro, características secundárias afetam diretamente a dificuldade e a forma de resolver esse problema, gerando assim diversas variações do mesmo. Algumas dessas características são: a capacidade sensorial do robô, para onde deslocar o recurso coletado, como o robô interage com os recursos a serem coletados, se os recursos são de um único tipo, entre outras (Winfield, 2009).

Uma das formas de resolver esse problema é utilizando técnicas de aprendizado por reforço. Aprendizado por reforço consiste na utilização de recompensas e punições como estímulo de aprendizado. Essas técnicas devolvem como saída uma política (*policy*), uma função que mapeia para cada estado a ação que o agente deve executar naquele estado.

Apesar de existirem diversos algoritmos de aprendizado por reforço, todos eles compartilham do problema de demora para a convergência do algoritmo. Isso se deve muitas vezes ao tamanho do espaço de busca (número de estados possíveis e número de políticas possíveis).

Para mitigar o problema de convergência lenta, a técnica de moldagem pode ser utilizada. Essa técnica consiste de prover informação privilegiada ao agente, concedendo informações que o agente não conseguiria, ou teria dificuldade de conseguir, durante o aprendizado. A proposta dessa técnica é guiar o aprendizado por um caminho já conhecido como promissor, diminuindo a exploração de caminhos não promissores. O tempo economizado na exploração de caminhos improdutivos pode ser gasto aprimorando o caminho promissor (Kaelbling et al., 1996).

Embora exista uma grande quantidade de trabalhos que estudam o problema de coleta de recursos, durante a realização deste trabalho não foi encontrado um outro trabalho com enfoque na utilização da técnica de moldagem e aprendizado por reforço aplicadas a esse problema. Dessa forma, a proposta deste trabalho está na avaliação de diferentes estratégias de moldagem aplicadas em conjunto com aprendizado por reforço para a resolução do problema de coleta de recursos.

Neste trabalho o algoritmo NEAT (Stanley e Miikkulainen, 2002) foi utilizado como algoritmo de aprendizado e três técnicas de moldagem foram avaliadas: moldagem por alteração de instância, moldagem por alteração de recompensa e moldagem por professor.

A moldagem por alteração de instância consiste em manipular a ordem em que os exemplos são apresentados ao agente, com base no desempenho do agente durante o treinamento. Aprender a resolver problemas em cenários simples pode facilitar a resolução de problemas em cenários mais complexos (Lin, 1991).

A moldagem por alteração de recompensa consiste em manipular a função de recompensa utilizada no algoritmo, isto é, a forma como a recompensa é calculada, com base no desempenho do agente durante o treinamento. Essa técnica pode ser utilizada para enfatizar diferentes fatores durante o aprendizado. Por exemplo, no início do aprendizado pode-se atribuir uma alta recompensa por evitar colisões, mas no final do treinamento apenas recursos coletados devem impactar na recompensa (Kadota et al., 2012).

A última técnica avaliada é a moldagem por professor. Nessa técnica a ação tomada pelo agente (aprendiz) é comparada com um agente externo (professor). Quando a ação do aprendiz é igual a do professor, o aprendiz recebe estímulo positivo, caso contrário, estímulo negativo. No início do aprendizado, seguir a opinião de um especialista reduz o tempo gasto

em exploração. A intensidade da influência do professor sobre o aprendiz pode ser reduzida ao longo do aprendizado (Dorigo e Colombetti, 1994).

O ambiente e os robôs utilizados neste trabalho foram simulados através de um simulador de implementação própria. Foi utilizada uma variação do problema onde existe um enxame de robôs autônomos e homogêneos. O objetivo do problema é coletar a maior quantidade de recursos possível e deslocar esses recursos de volta a posição inicial (ninho). Uma descrição mais detalhada da definição da tarefa está disponível no corpo do texto.

1.1 OBJETIVO GERAL

Averiguar os benefícios da utilização da técnica de moldagem em conjunto com algoritmos de aprendizado por reforço aplicados ao problema de coleta de recursos.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um simulador de robótica simplificado para permitir a análise dos algoritmos de aprendizado em um ambiente controlado.
- Avaliar o desempenho do algoritmo NEAT aplicado ao problema de coleta de recursos.
- Implementar diferentes algoritmos de moldagem.
- Avaliar as diferenças no desempenho entre o NEAT com e sem moldagem no problema de coleta de recursos.
- Avaliar diferentes algoritmos de moldagem aplicados à coleta de recursos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos fundamentais para a compreensão deste trabalho e está dividido em duas seções, coleta de recursos e aprendizado por reforço. Na primeira seção serão apresentados conceitos para fundamentar e compreender a natureza do problema a ser estudado. Já a segunda seção apresenta os conceitos necessários para compreender a solução adotada e suas principais características.

2.1 COLETA DE RECURSOS

Define-se coleta de recursos dentro da área da robótica como o problema de procurar, coletar e transportar recursos até um local determinado. Esse problema é muito estudado no contexto de robótica por integrar diversas tarefas como exploração, navegação, identificação, manipulação e transporte de objetos em um único problema (Winfield, 2009).

Embora essa definição deixe claro um objetivo a ser cumprido, ela é insuficiente para projetar algoritmos para resolver esse problema. Por exemplo, se a localização dos recursos é conhecida pode-se utilizar um algoritmo para traçar uma rota da posição atual até a posição do recurso; se não é conhecida, um algoritmo para pesquisar em ambiente desconhecido deve ser utilizado.

Em outras palavras, para projetar algoritmos para coleta de recursos, além da definição do objetivo, outras características também precisam ser definidas como: quanta informação o robô tem disponível, como o agente interage com o ambiente, como o agente interage com os recursos, entre outras.

A seguir nesta seção serão apresentadas as características consideradas mais relevantes para a compreensão deste trabalho. Essas características serão utilizadas para definir precisamente qual variação do problema de coleta de recursos será estudada neste trabalho.

O objetivo desta seção é apresentar algumas variações do problema e a necessidade de diferentes técnicas para resolver cada uma das varições.

2.1.1 Individual X Enxame

A utilização de um enxame de robôs ao invés de um único robô oferece não apenas possibilidades mas também complicações que não existem em um ambiente com um único robô.

A vantagem mais direta de se utilizar um enxame de robôs está na possibilidade de dividir e paralelizar a resolução dos problemas. Os robôs podem explorar o espaço em paralelo ou assumir diferentes papéis. Enquanto alguns robôs procuram novas fontes de recursos outros se encarregam apenas de transportá-los para local adequado. Além disso, quando se considera um cenário onde os robôs podem falhar, um enxame é mas resistente a falhas do que um único robô (Dudek et al., 1993).

No entanto a gestão de um enxame demanda algoritmos para solução de subproblemas que não existem no cenário de um robô individual como divisão de trabalho, colisão entre robôs do enxame e mecanismos de comunicação entre os robôs (Yang e Liu, 2012).

Algumas variações do problema de coleta com enxame também consideram a possibilidade dos robôs não serem homogêneos, isto é, os robôs podem não possuir o mesmo controlador ou as mesmas características físicas, o que pode impedir a utilização de algoritmos para enxames homogêneos (Wang et al., 2002).

2.1.2 Totalmente X Parcialmente Observável

A quantidade de informação sobre o ambiente que está disponível para o robô impacta diretamente na forma como o problema será resolvido. Em um ambiente completamente observável o robô tem acesso a toda informação do ambiente, inclusive a localização precisa de todos os recursos a serem coletados, seus destinos e a sua própria localização. Nesse caso o problema de coleta de recursos se resume a traçar caminhos entre as localizações, sendo desnecessária a realização de exploração (Dudek et al., 1993).

O cenário mais comum em robótica é de um ambiente parcialmente observável, onde o robô tem conhecimento limitado sobre o ambiente. Usualmente, o robô possui apenas informação sobre o que existe ao seu redor (raio de visão) e eventualmente algum indicativo de distância e/ou direção de locais de interesse. Nesse cenário o robô precisa explorar o ambiente para localizar com precisão os locais de interesse (Dudek et al., 1993).

A capacidade do robô concluir que resolveu o problema também é limitada pelo que ele consegue observar do ambiente. Um robô que deve coletar todos os recursos espalhados em uma arena, mas não tem conhecimento de quantos recursos existem tem mais dificuldade para concluir que resolveu um problema do que um robô que tem o conhecimento de quantos recursos existem, mesmo que não saiba exatamente onde estão localizados (Russell e Norvig, 2010).

2.1.3 Interação com o ambiente

Além de perceber e observar o ambiente, o robô também é capaz de agir nesse ambiente. O que o robô pode fazer no ambiente e como ele pode fazer afetam diretamente a forma como a coleta de recursos pode ser realizada. Se o robô pode apenas empurrar os objetos, ou puxar e empurrar, ou se é necessário mais de um robô para transportar os objetos são exemplos de limitações na interação com o ambiente (Russell e Norvig, 2010).

Assim como alguns algoritmos demandam que algumas informações do ambiente sejam observáveis pelo robô, os algoritmos também são limitados pela capacidade de agir dos robôs. Um algoritmo que foi utilizado por um robô com a habilidade de empurrar e puxar recursos pode não ser utilizável por um robô que apenas empurra.

2.1.4 Local de destino

O local para onde os recursos devem ser levados pode impactar nos algoritmos utilizados. As duas variações principais de local de destino são denominadas: retornar ao local de origem e mover a um terceiro local (Dudek et al., 1993).

Na primeira variação, retornar ao local de origem, o robô deve trazer os recursos a posição onde ele iniciou o problema, ou próximo a ela. Nessa variação aproveita-se que o robô já esteve na posição destino e são utilizados mecanismos para retornar a essa posição anteriormente visitada.

Na segunda variação, mover para uma terceiro local, além de encontrar o recurso, o robô deve traçar uma rota para um local não visitado, provavelmente desconhecido.

2.1.5 Restrições adicionais

As três características previamente apresentadas (local de destino, interação com o ambiente, parte do ambiente observável) estão presentes em todas as variações do problema de coleta de recursos. Também existem características opcionais, isto é, que não estão presentes em todas as variações.

A presença de obstáculos é uma dessas características. Os obstáculos podem obstruir a visão do agente, reduzindo o que ele consegue observar do ambiente. Além disso, a colisão do agente com os obstáculos pode resultar em uma recompensa negativa, ou perda de desempenho, o que faz com que em conjunto com o algoritmo de exploração também seja necessário incorporar um algoritmo de desvio de obstáculos (Shell e Mataric, 2006).

Uma capacidade energética (energia) limitada é outra dessas características. Com energia limitada, o robô deve procurar novas fontes de energia ou estações de recarga, caso contrário para de funcionar (Zedadra et al., 2015b).

No caso de enxames, também existem variações referentes aos mecanismos de comunicação entre os robôs. Se existe ou não comunicação e as limitações para realizar a comunicação são essenciais para a resolução eficiente do problema. Por exemplo, através de mensagens, um robô que já localizou uma fonte de recursos pode informar aos outros; dessa forma os robôs podem parar de explorar e se concentrar em transportar os recursos já encontrados (Lee et al., 2013).

Outro tipo de restrição está na memória do robô. Um robô com muita memória pode memorizar os movimentos feitos e quando encontra um recurso, fazer os movimentos reversos para retornar ao ninho. Já um robô sem memória pode confiar apenas em sua percepção atual (Lima e Oliveira, 2016b).

2.1.6 Conclusão

Embora o problema de coleta de recursos tenha um objetivo claro, para delimitar formalmente o problema e projetar algoritmos eficientes, diversas características adicionais devem ser analisadas. Nesta seção foram apresentadas as características mais comuns e como a presença ou não dessas características modifica os algoritmos que podem ser utilizados para resolver o problema.

2.2 APRENDIZADO POR REFORÇO

Aprendizado por reforço faz parte de um conjunto mais geral de técnicas chamado aprendizado de máquina. Aprendizado de máquina consiste em aprender através de dados e de experiências passadas. O aprendiz é exposto a um conjunto de dados, uma métrica de aprendizado e algum estímulo de aprendizado. Uma forma de classificar e agrupar as diferentes técnicas de aprendizado de máquina é através do estímulo ao qual o aprendiz (agente) é exposto (Alpaydin, 2010).

Aprendizado por reforço é um conjunto de técnicas utilizadas para resolver problemas utilizando um agente que aprende através de punições e recompensas. Em outras palavras, as técnicas de aprendizado por reforço são aquelas nas quais o agente é estimulado através de uma recompensa proporcional à qualidade das decisões tomadas pelo agente (Russell e Norvig, 2010).

Além da diferença no estímulo de aprendizado, as diferentes técnicas de aprendizado de máquina também se diferenciam na aplicação. Aprendizado por reforço é principalmente utilizado para a resolução de problemas que envolvem processos de decisão sequenciais. Processos de decisão sequenciais são aqueles onde as decisões imediatas tem efeito em decisões futuras. Exemplificando, classificação de imagens não é um processo de decisão sequencial pois o agente deve decidir qual a classe de cada imagem e a classe atribuída à imagem atual não interfere na classe que será atribuída à próxima imagem. Já na movimentação de um robô, o movimento feito em um instante de tempo t interfere nas posições que podem ser alcançadas pelo movimento feito no instante de tempo $t + 1$; logo este é um processo de decisão sequencial (Russell e Norvig, 2010).

Muitos problemas em robótica, inclusive o problema de coleta de recursos, podem ser descritos por um processo de decisão sequencial, o que faz com que as técnicas de aprendizado por reforço tenham preferência a outras técnicas de aprendizado nessa área.

Em processos de decisão sequenciais os agentes não são avaliados pela recompensa imediata recebida mas sim pelo resultado atingido ao final de várias ações ou pela recompensa acumulada de todas as ações executadas (Mitchell, 1997).

A resolução de um problema por um agente ocorre através da interação entre o agente e o ambiente de tarefas. O ambiente de tarefas é a representação do problema e também é responsável por enviar o reforço (recompensa) para o agente enquanto o agente observa o estado atual do ambiente (problema) para tomar uma decisão. O agente envia uma ação ao ambiente, resultando em uma troca de estado do ambiente e uma recompensa. Esse processo repete-se por um número determinado de iterações ou até o ambiente atingir um estado final. Exemplificando, em um problema de deslocar um robô de uma origem até o destino, o ambiente contém a localização de todos os elementos, inclusive o robô, origem e destino. As ações do agente alteram a posição do robô no ambiente (troca de estado). Quando o ambiente chegar em um estado onde a posição do robô é igual à posição de destino, o problema está resolvido (Russell e Norvig, 2010).

A figura 2.1 apresenta um esquema da comunicação entre ambiente e agente.

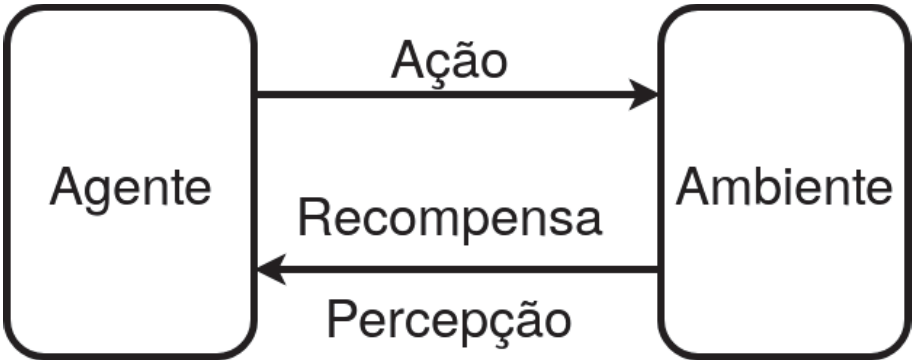


Figura 2.1: Esquema de interação entre agente e ambiente.

A saída de um algoritmo de aprendizado por reforço é uma política, uma função que associa para cada estado do problema uma ação (decisão). Essa função é utiliza pelo agente para tomar decisões. Durante a execução de um algoritmo de aprendizado por reforço, o algoritmo tenta definir qual a melhor ação para cada um dos estados do problema (Russell e Norvig, 2010).

O objetivo de um algoritmo de aprendizado por reforço, através de uma política, é definir qual ação imediata (a curto prazo) gerará mais recompensa acumulada (a longo prazo) (Mitchell, 1997).

2.2.1 Generalização e aproximação

Uma das principais dificuldades que devem ser contornadas para utilizar algoritmos de aprendizado por reforço é o tamanho da representação da política.

Como previamente mencionado, a saída de um algoritmo de aprendizado por reforço é uma função que associa uma ação para cada estado do problema. Uma forma precisa de se representar essa função é utilizando uma tabela onde existe uma posição por estado e o valor da posição é a ação que deve ser tomada. Embora essa representação seja precisa, o tamanho do espaço de estados pode inviabilizar sua utilização, seja por espaço de armazenamento ou eficiência para percorrer a tabela (Alpaydin, 2010).

Uma política deve atribuir uma ação a cada um dos estados do problema. Isso implica que quanto mais estados um problema apresenta, mais tempo é necessário para ajustar as ações de todos os estados. Dessa maneira, o tamanho do espaço de estados também pode inviabilizar o ajuste preciso de todos os estados do problema (Alpaydin, 2010).

Para contornar essas dificuldades pode-se utilizar a técnica de aproximação. Essa técnica consiste de sacrificar a precisão da representação utilizando uma aproximação da função da política. As aproximações, por exemplo redes neurais e aproximadores lineares, podem ser representadas de forma compacta, comparadas ao tamanho da tabela de estados, reduzindo problemas de armazenamento e acesso aos dados (Russell e Norvig, 2010).

Outra vantagem de aproximações é a generalização de estados, isto é, como a aproximação não representa os estados de forma precisa ao realizar ajustes para um estado, estados similares também possuem o seu valor ajustado. Em outras palavras, estados nunca visitados também terão seu valor ajustado em função do valor de estados similares.

2.2.2 Observabilidade parcial

A utilização adequada da política depende da identificação correta do estado do problema, uma vez que a entrada da função é o estado. Quando o ambiente é parcialmente observável, o agente não possui toda a informação necessária para definir qual é o estado atual com precisão. Este não é apenas um problema durante a utilização da política mas também durante a etapa de ajuste, já que não se sabe exatamente qual é o estado atual (Russell e Norvig, 2010).

Apesar dessa característica adicionar complexidade ao problema, existem abordagens para mitigar essa situação, como a utilização estados de crença (*belief-states*) e algoritmos de aprendizado que não necessitam do estado preciso para realizar ajustes na política, como NEAT.

2.2.3 NEAT

O algoritmo *Neuroevolution of augmenting topologies* (NEAT) foi proposto originalmente por Stanley e Miikkulainen (2002) para resolver o problema do pêndulo invertido, considerado um problema clássico de aprendizado por reforço e robótica. Nesse problema, um robô deve manter um pêndulo invertido equilibrado enquanto se desloca para frente e para trás.

Existem dois conceitos principais ligados ao algoritmo NEAT que serão explicados a seguir: redes neurais e algoritmos evolutivos.

Uma rede neural artificial é um modelo de aprendizado formado por partes menores chamadas neurônios artificiais. Essas redes podem ser representadas por um grafo direcionado, onde um nó do grafo representa um neurônio e um arco uma ligação entre dois neurônios. Redes neurais também são chamadas de redes neuronais por se tratarem de uma rede de neurônios (Engelbrecht, 2007).

O modelo mais comum de neurônio artificial é o perceptron. Um perceptron é composto de três elementos: um vetor de pesos w, uma função de ativação f e um viés (*bias*) ou limitante θ. O perceptron recebe como entrada um vetor i de mesmo tamanho que o vetor w e gera como saída um número, resultado da equação 2.1. Em outras palavras, a saída y do perceptron é dada pelo produto escalar do vetor de pesos w com o vetor de entrada i subtraído do viés θ e aplicado na função de ativação f (Engelbrecht, 2007).

$$y = f((w \cdot i) - \theta) \tag{2.1}$$

Em uma rede neural o sinal (valor) de saída de um neurônio é propagado como sinal de entrada para outros neurônios da rede. A definição de quais neurônios recebem quais sinais é

determinada pela topologia da rede, que corresponde ao grafo que representa quais neurônios estão conectados entre si e qual o sentido da conexão. Cada aresta do grafo armazena o peso da conexão que representa.

Os neurônios de uma rede podem ser divididos em 3 categorias: entrada, saída e escondidos. Os neurônios de entrada são aqueles que recebem valores externos à rede, como sinais de entrada; os neurônios de saída são aqueles onde os sinais de saída serão enviados para fora da rede; enquanto os neurônios escondidos são aqueles que tanto sinais de entrada como se saída são utilizados apenas internamente na rede. Para utilizar uma rede neural é necessário apenas conhecer os neurônios de entrada e saída, pois são eles que determinam quais informações a rede precisa para iniciar a calcular e qual o resultado da rede.

Em outras palavras, uma rede neural é um método paramétrico para aproximação de funções onde a topologia e os pesos são os parâmetros utilizados para definir os valores de saída da rede em função dos valores de entrada. O processo de ajustar esses parâmetros é chamado de treinamento da rede (Engelbrecht, 2007).

Um algoritmo evolutivo é um algoritmo de otimização, isto é, um algoritmo utilizado para encontrar a melhor solução dentro de um espaço de possíveis soluções para um problema.

Algoritmos evolutivos são inspirados na teoria da seleção natural, onde o ser vivo mais adaptado tem maior probabilidade de sobreviver e propagar as suas características para as próximas gerações. Sendo inspirados em uma teoria biológica, a nomenclatura utilizada no algoritmo também é inspirada na nomenclatura utilizada na biologia.

Seres vivos representam possíveis soluções para o problema de otimização, também são comumente chamados de indivíduos. A qualidade de uma solução é representada pelo conceito de aptidão, isto é, a probabilidade de passar as suas características para as futuras gerações.

Para realizar a otimização, um algoritmo evolutivo mantém um conjunto de candidatos a solução, chamado de população, e utiliza 3 operações sobre essa população com o intuito de gerar soluções com melhor valor de aptidão (melhores). Essas operações também são inspiradas na teoria da evolução das espécies e são chamadas de: seleção, cruzamento e mutação.

O termo algoritmo evolutivo é usado para descrever uma família de algoritmos e não um algoritmo específico, isto é, cada algoritmo evolutivo implementa as suas operações, representação de candidatos a solução e função de aptidão de uma forma específica, mas compartilham os princípios da seleção natural. Por essa razão, ao invés de descrever genericamente os demais conceitos de algoritmos evolutivos a seguir apresenta-se como o algoritmo NEAT incorpora uma estratégia evolutiva dentro do algoritmo de aprendizado.

O algoritmo NEAT é um algoritmo de aprendizado bioinspirado que utiliza uma rede neural como modelo de aprendizado e técnicas inspiradas em algoritmos evolutivos para realizar o treinamento da rede (Stanley e Miikkulainen, 2002).

Como o NEAT é um algoritmo de aprendizado por reforço, um candidato a solução é uma política, representada por uma rede neural. O algoritmo NEAT mantém durante a sua execução uma população de redes neurais com pesos e topologias bem definidas.

Para definir a aptidão de uma rede, isto é, o quão boa é aquela rede, ela é utilizada em casos de teste e o resultado obtido é utilizado para definir a aptidão. Por exemplo, no problema de deslocar um robô de um ponto a outro, a rede neural seria utilizada como controlador do robô em instâncias desse problema e a aptidão utilizada poderia ser a soma do tempo que o robô precisou para se deslocar da origem até o destino em cada uma das instâncias. Nesse caso, quanto menor a soma melhor a solução.

Uma vez calculada a aptidão de cada indivíduo da população inicia-se o processo de selecionar os indivíduos da próxima geração. Primeiramente utiliza-se a operação de seleção, onde duas redes da população atual serão utilizadas para gerar duas novas redes da próxima

geração. A aptidão atual do indivíduo aumenta a probabilidade de ele ser selecionado. Em outras palavras, quanto maior a aptidão maior a probabilidade de passar as características para a próxima geração.

A operação de cruzamento é aplicada a cada um dos pares formados na etapa de seleção. Nessa operação as características dos pais (indivíduos da população atual) serão recombinadas para criar os filhos (indivíduos da próxima geração). Como os indivíduos são redes neurais, suas características são a topologia e os pesos da rede.

Um dos destaques do algoritmo NEAT está na capacidade de recombinar de forma eficiente a topologia das redes. Para realizar a recombinação as redes são representadas por uma lista de conexões, onde cada conexão possui um identificador único (em toda a população) chamado número de novidade. Os filhos terão todas as arestas dos pais. As arestas comuns entre eles podem ser facilmente identificadas através do número de novidade. Quando a aresta é comum aos dois pais, existe uma combinação dos pesos; quando existe em apenas um deles, o peso permanece (Stanley e Miikkulainen, 2002).

A figura 2.2 é um exemplo de lista de conexões.

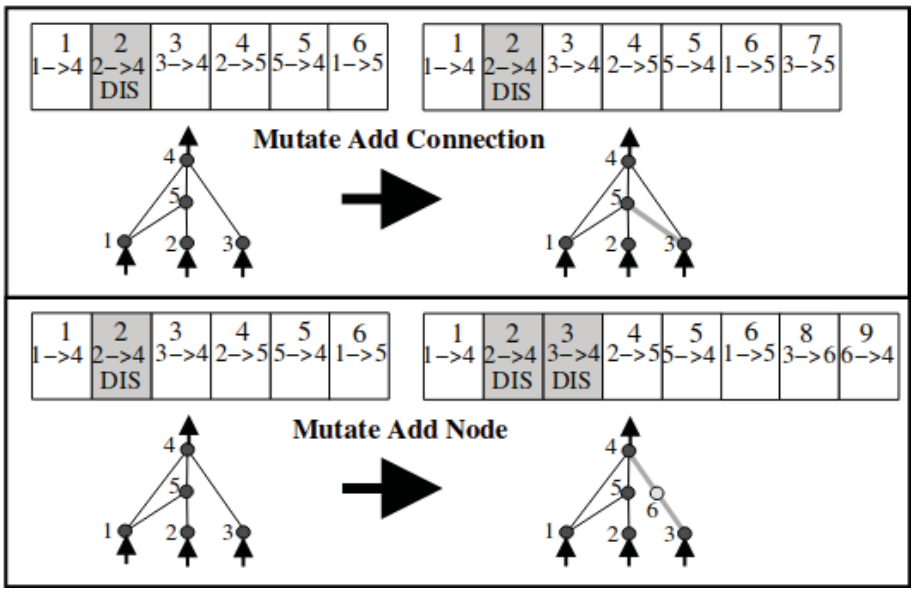


Figura 2.2: Exemplo de lista de conexões (Stanley e Miikkulainen, 2002).

Depois do cruzamento a operação de mutação é aplicada nos novos indivíduos. Essa operação causa alterações nas características do indivíduo que não são geradas em função das características dos pais. O NEAT possui dois tipos de mutação, mutação de topologia e mutação de pesos.

A mutação de topologia adiciona uma nova conexão, remove uma conexão existente ou cria um novo neurônio. A mutação de peso altera o valor do peso de uma conexão. Diferente das operações de seleção e cruzamento, a mutação é aplicada apenas em parte dos indivíduos para população. Um dos parâmetros do algoritmo NEAT é a probabilidade de aplicar mutação no indivíduo. Essa operação é necessária para inserir novidade na população e tentar combinações de características novas (Stanley e Miikkulainen, 2002).

Após a mutação um novo conjunto de indivíduos (redes) é gerado e o processo se repete: avaliação (cálculo da aptidão), seleção, cruzamento e mutação. O processo se repete até um número determinado de iterações ou até que uma das redes da população tenha atingido uma aptidão satisfatória (parâmetro do algoritmo).

Uma característica de destaque do NEAT é a incorporação da propriedade de generalização. Uma vez que o algoritmo utiliza uma rede neural, que é uma aproximação de função, as vantagens da generalização já estão incorporadas ao algoritmo.

Outra característica de destaque é o ajuste orientado aos casos de teste e não aos estados do problema. Ajusta-se a política através do processo iterativo de avaliação, seleção, cruzamento e mutação. Embora os casos de teste passem por estados específicos do problema, o ajuste realizado não é mapeado diretamente a um estado específico. Em outras palavras, não é necessário saber o estado precisamente para realizar ajustes na política, o que é uma grande vantagem em ambientes parcialmente observáveis, que são comuns em robótica.

2.2.4 Moldagem

Previamente mencionou-se que um dos destaques das técnicas de aprendizado por reforço é a capacidade de lidar com processos de decisão sequenciais e que o objetivo de um algoritmo de aprendizado por reforço é encontrar boas decisões de curto prazo para maximizar a recompensa a longo prazo. Uma dificuldade ligada a processos de decisão sequenciais é chamada de recompensa atrasada (*delayed reward*). Recompensa atrasada significa que o benefício de se tomar decisão não vem imediatamente, mas sim atrasado. Os algoritmos de aprendizado por reforço são utilizados especialmente para lidar com essa dificuldade. Entretanto, um dos fatores de grande impacto no tempo de convergência desses algoritmos é o quão atrasada (quão distante na sequência) vem a recompensa. Quanto maiores as sequências de ações e mais distantes estão as recompensas do início da sequência, mais difícil é definir a ação de curto prazo para alcançar o objetivo e mais iterações os algoritmos necessitam para convergir para uma boa política (Alpaydin, 2010).

Embora os algoritmos de aprendizado sozinhos tenham a capacidade de convergir por conta própria, o tempo necessário para a convergência é alto. Para aumentar a eficiência dos algoritmos de aprendizado por reforço pode-se utilizar técnicas complementares. Uma dessas técnicas é a moldagem (Kaelbling et al., 1996).

Moldagem é uma técnica que consiste em prover informação privilegiada ao agente, concedendo informações que o agente não conseguiria, ou teria dificuldade de conseguir, durante o aprendizado. A proposta dessa técnica é guiar o aprendizado por um caminho já conhecido como promissor, diminuindo a exploração de caminhos não promissores. O tempo economizado na exploração de caminhos improdutivos pode ser gasto aprimorando o caminho promissor (Kaelbling et al., 1996).

A moldagem interfere na comunicação entre o agente e o ambiente, alterando por exemplo, a recompensa enviada ao agente. A proposta da técnica de moldagem não é alterar o algoritmo de aprendizado nem o ambiente, mas interferir na comunicação entre eles. Isso permite que a técnica de moldagem possa ser utilizada por qualquer algoritmo de aprendizado por reforço (Dorigo e Colombetti, 1994).

Como exemplo de moldagem, em um problema de deslocar um robô da origem até o destino, o robô receberia uma recompensa positiva apenas ao chegar ao destino. Moldagem poderia ser utilizada, por exemplo, para atribuir grandes recompensas negativas em caso de colisão; e dessa forma o robô aprenderia mais rapidamente a evitar um comportamento prejudicial que, apenas com a recompensa de chegar ao destino, seria difícil de ser inferida como prejudicial. Outro exemplo de moldagem seria ordenar os cenários de teste para treinar primeiramente em cenários onde a distância entre origem e destino é menor, minimizando os efeitos da recompensa atrasada. Ao longo do aprendizado, a distância entre origem e destino poderia ser aumentada em função do desempenho do agente.

Moldagem é um conceito utilizado para definir uma família de técnicas que compartilham as seguintes características: entregar informação privilegiada ao agente e interferir na interação entre agente e ambiente.

O conceito de moldagem é similar ao conceito de heurística no contexto de busca informada. Heurísticas são utilizadas para dar mais informações sobre o domínio do problema para os algoritmos de busca, para aumentar sua eficiência. Como cada problema tem um domínio, cada problema requer uma heurística diferente, mas o propósito permanece. A moldagem atua da mesma forma, todavia, ao invés de informar algoritmos de busca ela informa agentes em algoritmos de aprendizado (Dorigo e Colombetti, 1994).

2.2.5 Considerações finais

Algoritmos de aprendizado por reforço são utilizados para encontrar soluções para processos de decisões sequenciais. Diversos problemas de robótica, inclusive coleta de recursos caracterizam processos de decisão sequenciais. O objetivo desses algoritmos é encontrar tomadores de decisões que escolhem ações no curto prazo com o objetivo de maximizar recompensas a longo prazo. Esses tomadores de decisões são representados por políticas, funções que associam um estado do problema a uma ação a ser tomada.

Embora existam dificuldades como tamanho do espaço de busca e recompensas atrasadas que afetam a eficiência desses algoritmos, também existem técnicas como moldagem que podem ser utilizadas para mitigar esses problemas, permitindo que esses algoritmos possam ser utilizados de forma eficiente.

3 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta a revisão bibliográfica da literatura relacionada a este trabalho. Esta revisão se divide em duas seções: coleta de recursos e moldagem. Na primeira seção são apresentados diversos trabalhos que tentam resolver o problema de coleta de recursos através de diversas técnicas. São apresentadas as semelhanças, diferenças e peculiaridades de cada um desses trabalhos. Já a segunda seção apresenta os principais trabalhos que utilizaram a técnica de moldagem mas não apenas para o problema de coleta de recursos.

3.1 COLETA DE RECURSOS

Uma das dificuldades de se realizar uma revisão de trabalhos sobre coleta de recursos é a quantidade de variações do problema. Com a finalidade de criar uma revisão bibliográfica mais objetiva e relevante para esse estudo, trabalhos cuja variação do problema de recursos difere muito da utilizada neste trabalho foram removidos.

Foram mantidos nesta revisão apenas os trabalhos que satisfizeram os seguintes critérios:

- Todos os experimentos foram realizados em ambientes simulados.
- Foi utilizado um enxame de robôs homogêneos.
- Os recursos, uma vez coletados, devem ser levados a posição de origem do robô.
- Existe um número finito de recursos espalhado no ambiente.
- Não existe reposição de recursos.

A análise dos trabalhos restantes se concentrou em um grupo de características apresentadas a seguir. Para melhor apresentar os trabalhos desta seção será utilizada uma tabela de características. Cada uma das linhas da tabela 3.1 representa um trabalho relacionado e cada uma das colunas contém uma característica utilizada para descrever e classificar os trabalhos.

As seguintes características são utilizadas na tabela: simulação, aprendizado, comunicação, distribuição e obstáculos.

Simulação (SI): qual tipo de simulação foi utilizada. Existem dois valores possíveis: discreta (D) ou contínua (C). Discreta indica que o ambiente simulado foi representado por uma matriz e os elementos da simulação posicionados em células da matriz. Contínua indica que o ambiente simulado foi representado por um plano e os elementos foram posicionados como polígonos em um plano.

Aprendizado (AP): de que forma técnicas de aprendizado foram utilizadas na criação do controlador. Existem dois valores possíveis: sim (S) e não (N). O valor **sim** indica que o controlador foi definido automaticamente por um outro algoritmo, ou seja, foi utilizado um algoritmo de aprendizado. O valor **não** indica que todo o algoritmo controlador e seus parâmetros foram definidos pelo projetista do algoritmo.

Comunicação (CO): de que forma os robôs do enxame se comunicam. Existem três valores possíveis: nenhuma (N), indireta (I) e direta (D): Nenhuma indica que os robôs não conseguem se comunicar de nenhuma forma. Indireta indica que os robôs os robôs não conseguem mandar mensagens entre eles mas podem realizar marcações no ambiente. Direta indica que os robôs podem trocar mensagens entre eles.

Distribuição (DI): de que forma os recursos foram dispostos no mapa do ambiente. Existem três valores possíveis: concentrada (C), distribuída (D) e ambas (A). Concentrada indica que durante os experimentos os recursos estão concentrados em uma ou poucas regiões do mapa e o restante das regiões não contém recursos. Distribuída indica que durante os experimentos os recursos estão espalhados por todo o mapa. Ambas indica que foram realizados experimentos com recursos concentrados e com recursos distribuídos.

Obstáculos (OB): a existência de obstáculos no mapa do ambiente. Existem dois valores possíveis: sim e não. O valor **sim** indica que existem obstáculos, isto é, objetos que não são robôs, recursos ou o ninho que podem interferir nos sensores dos robôs e no deslocamento dos recursos. O valor **não** indica que os únicos elementos no mapa do ambiente são os robôs do enxame, o ninho e os recursos.

Tabela 3.1: Características dos trabalhos da literatura

Nome	AP	SI	CO	DI	OB
Shell e Mataric (2006)	N	C	N	C	S
Lee et al. (2013)	N	C	D	D	S
Ericksen et al. (2017)	S	C	I	A	N
Nogales e de Oliveira (2018)	N	C	D	C	N
Zedadra et al. (2015a)	N	D	I	D	S
Lima e Oliveira (2016b)	N	D	I	D	S
Zhou et al. (2016)	N	C	D	C	S
Afshar e Mahjoob (2008)	S	C	I	D	N
Wei et al. (2017)	S	C	N	D	S
Zedadra et al. (2015b)	N	D	I	D	S
Sugawara et al. (1999)	N	C	I	D	N
Lima e Oliveira (2017)	N	D	I	D	S
Lee e Ahn (2011)	N	C	N	C	N
Dolan-Stern et al. (2018)	N	C	D	A	S
Kadota et al. (2012)	S	C	N	D	S
Yasuda et al. (2013)	S	C	N	D	S
Ohkura et al. (2013)	S	C	N	D	S
Hecker et al. (2015)	N	D	D	C	N
Lima e Oliveira (2016a)	N	D	I	D	S
Yang et al. (2009)	N	C	N	D	S
Hara et al. (2016)	S	C	N	C	N

A seguir, os trabalhos listados na tabela 3.1 são apresentadas em mais detalhes.

Em Shell e Mataric (2006), os autores concentram seu estudo em enxames de robôs de larga escala. Nesse tipo de sistema, onde existem centenas de robôs e o espaço é limitado, a quantidade de robôs gera concorrência em regiões de interesse, como próximo ao ninho e próximo a regiões onde existem recursos. Nesse trabalho, os autores demonstram através de experimentos que nesse cenário a divisão de tarefas se torna mais eficiente por reduzir a concorrência em regiões de interesse. Os robôs, ao invés de realizar a coleta do início ao fim, levam os recursos apenas parte do caminho, o restante é feito por outros robôs.

Em Lee et al. (2013), os autores concentram seu estudo na economia de energia em enxames. Nos experimentos realizados, os robôs apresentam a restrição de limitação energética. As ações do robô consomem energia e quando o nível de energia atinge um limite o robô é obrigado a retornar ao ninho para recarregar, caso contrário ficará desativado. Os autores além

de buscar a maximização de recursos coletados também se preocupam com a eficiência da coleta, utilizando a métrica de energia gasta. Os experimentos são realizados utilizando um algoritmo inspirado em colônia de abelhas que busca cooperação e divisão de tarefas. Os robôs, ao invés de realizar a coleta do início ao fim, levam os recursos apenas parte do caminho, o restante do caminho é feito por outros robôs.

Em Ericksen et al. (2017), os autores comparam a eficiência controladores projetados manualmente e controladores gerados automaticamente. Esse trabalho apresenta como principal problema de controladores projetados manualmente a falta de generalidade, isto é, os controladores manuais são projetados para funcionar apenas em situações e cenários específicos. Os autores comparam dois controladores manuais, *Central Place Foraging Algortihm (CPFA)* e *Distributed Deterministic Spiral Algorithm (DDSA)* com controladores gerados automaticamente pelo algoritmo NEAT e demonstram através de experimentos que os controladores gerados automaticamente obtém desempenho similar aos controladores manuais.

Em Nogales e de Oliveira (2018), os autores concentram seu estudo em um mecanismo para diminuir a concorrência em regiões de interesse. A concentração excessiva de robôs em regiões de interesse como o ninho, regiões de troca de recursos e regiões de coleta de recursos pode prejudicar a desempenho do enxame. Nesse estudo, os autores apresentam um algoritmo para identificar se uma região está congestionada. Caso esteja, o robô abandona a tarefa atual e passa para uma nova tarefa, com o objetivo de não aumentar o congestionamento existente. Os autores comparam a eficiência na tarefa de coleta de recursos considerando diferentes mecanismos de abandono de tarefa e diferentes instâncias do problema.

Em Zedadra et al. (2015a), os autores concentram seu estudo em mecanismos de cooperação, com ênfase na sub tarefa de exploração. Os autores utilizam um algoritmo inspirado no princípio da estigmergia para realizar comunicação indireta entre os robôs do enxame através de marcações no ambiente. Através dessas marcações, os robôs são capazes de identificar regiões que já foram exploradas por outros robôs e também são capazes de se concentrar em regiões que possuem recursos.

Em Lima e Oliveira (2016b), as autoras apresentam um modelo formal para representar tarefas de coleta de recursos e um algoritmo para operar sobre o modelo proposto. O modelo é chamado de *robot probabilistic cellular automata ant memory (RPCAAM)* e representa o ambiente de simulação através de uma matriz de células idênticas. As autoras comparam a eficiência de algoritmos determinísticos e estocásticos e o efeito da utilização de memória de curto prazo, utilizada para evitar células exploradas recentemente.

Em Zhou et al. (2016), os autores apresentam um algoritmo que utiliza o conceito de estabilização do sistema. Os autores definem um estado do sistema como estável, que representa o comportamento desejado para o sistema. Nesse trabalho, o estado estável consiste de os robôs criarem um caminho entre fontes de ninho e percorrerem esse caminho em fila. Os autores apresentam um algoritmo capaz de a partir de qualquer estado do sistema convergir para o estado estável. Utilizando provas formais, os autores demonstram a corretude e convergência do algoritmo e apresentam experimentos para avaliar a eficiência.

Em Afshar e Mahjoob (2008), os autores utilizam lógica *fuzzy* e aprendizado por reforço para induzir a formação de caminhos entre o ninho e a fonte de recursos. O algoritmo proposto pelos autores é inspirado na utilização de feromônio produzido pelas formigas para a formação de caminhos. Esse algoritmo realiza marcações no ambiente para induzir a criação de caminhos. A quantidade de feromônio depositada no ambiente é definida através de lógica *fuzzy*. O comportamento de seguir o caminho é aprendido utilizando o algoritmo de aprendizado por reforço *Q-learning*.

Em Wei et al. (2017), os autores concentram o seu estudo na identificação coletiva de recursos. Nesse estudo existem dois tipos diferentes de recursos, um que gera uma recompensa negativa se levado ao ninho enquanto outro gera uma recompensa positiva. Esses dois tipos de recursos são diferenciados pelo tamanho mas um robô sozinho não consegue mensurar o tamanho de um recurso. Nesse trabalho uma rede neural foi utilizada como controlador, cujos pesos foram otimizados pelo algoritmo evolutivo (μ, λ). A capacidade de diferenciar os diferentes tipos de recursos e transportá-los até o ninho foi demonstrada através de experimentos.

Em Zedadra et al. (2015b), os autores comparam diferentes algoritmos de coleta de recursos considerando a restrição de energia. Além da métrica de recursos coletados, nesse trabalho também é avaliado a eficiência da coleta através da métrica de energia. O algoritmo proposto nesse trabalho é chamado de *Energy aware Cooperative Switching Algorithm for Foraging (EC-SAF)* e utiliza o princípio da estigmergia para induzir a formação de caminhos entre fontes de recursos e o ninho. Segundo os autores a utilização de marcações no ambiente permite exploração e retorno ao ninho de forma eficiente. Os experimentos realizados compararam o algoritmo proposto com outro encontrado no literatura (*Ec-marking*).

Em Sugawara et al. (1999), os autores concentram o seu estudo em avaliar como a quantidade de interações entre os robôs afeta os padrões formados pelo enxame durante a execução da coleta de recursos. O comportamento dos robôs é descrito por uma máquina de estados com 5 estados: vagando, atraído, parado, desviando e retornando. A análise foi realizada através de modelos matemáticos e experimentos. Os autores verificaram que diferentes níveis de interação entre os robôs faz com que o enxame forme padrões diferentes.

Em Lima e Oliveira (2017), as autoras concentram o seu estudo na análise formal no modelo *Cellular Automata Ant (CAA)* utilizando teoria dos grafos. Nesse estudo o comportamento do robô foi modelado por uma máquina de estados finita de 4 estados: procurando, segurando, retornando, depositando. A matriz de células foi representada como um grafo e os robôs tem a capacidade de depositar feromônio. Nesse trabalho é utilizado o "feromônio inverso", isto é, ao invés de simular o comportamento biológico do feromônio, que é atrair os robôs para zonas de maior concentração, o feromônio é utilizado para repelir, ampliando a área de cobertura do enxame. Através de provas formais as autoras demonstram que, para o algoritmo utilizado, que na tarefa de exploração, conforme o número de robôs aumenta, a eficiência na tarefa de exploração também aumenta. Experimentos foram utilizados para avaliar o efeito de diferentes taxas de evaporação de feromônio e diferentes tamanhos de enxame.

Em Lee e Ahn (2011), os autores se concentram na eficiência energética da coleta de recursos. Esse trabalho propõe uma nova máquina de estados para representar o comportamento do robô, composta de diversos estados, cujo benefício é evitar o desperdício de energia. A máquina de estados proposta faz com que os robôs dividam o espaço de busca em diferentes áreas e faz com que cada robô trabalhe apenas em uma determinada área, dessa forma reduzindo a concentração de robôs em uma mesma área de interesse (como o ninho). A tarefa de coleta é feita de forma distribuída, isto é, cada robô transporta um recurso apenas até uma área de transição no limite de sua área de busca. Outro robô assume a tarefa de coleta e esse processo se repete até que o recurso chegue ao ninho.

Em Dolan-Stern et al. (2018), os autores apresentam uma nova máquina de estados inspirada em colônia de abelhas para executar a tarefa de coleta. O algoritmo proposto é comparado com outro algoritmo da literatura, denominado *Deterministic Spiral Search Algorithm (DDSA)*. Os dois algoritmos apresentam desempenho similar. No algoritmo proposto os robôs iniciam uma busca distribuída e guardam a informação de todas as fontes de recursos encontradas. Quando uma fonte se esgota, os robôs já possuem a localização de outras fontes encontradas.

Em Kadota et al. (2012), os autores propõem uma abordagem incremental para um enxame evolutivo. O controlador do robô é uma rede neural otimizada por um algoritmo evolutivo. Durante o treinamento dos robôs a função de aptidão é alterada incrementalmente. A tarefa de coleta é subdivida em 4 sub tarefas, cada uma com a sua própria função de aptidão. Quando o robô executa a sub tarefa com sucesso, a aptidão da sub tarefa é adicionada a aptidão do robô. A abordagem incremental se mostrou mais eficiente que a utilização da função de aptidão não incremental.

Em Yasuda et al. (2013), os autores propõem um método para analisar o comportamento do enxame. Para realizar a análise os autores desenvolvem uma técnica que permite agrupar robôs em sub grupos que realizam comportamentos similares em condições similares. O principal objetivo desse trabalho é avaliar como as tarefas são distribuídas nos sub grupos. Nos experimentos realizados são avaliados diferentes métodos de agrupamento de robôs.

Em Ohkura et al. (2013), os autores propõem a utilização de evolução incremental para induzir a cooperação entre os robôs do enxame. Nos experimentos realizados, os robôs do enxame não são capazes de deslocar os recursos sozinhos. Os autores comparam o método de evolução convencional com o método de evolução incremental e verificam que a evolução incremental apresenta desempenho superior a evolução convencional.

Em Hecker et al. (2015), os autores concentram seu estudo na relação entre a concentração dos recursos e a eficiência da coleta. Segundo os autores, quanto mais concentrados estão os recursos, mais eficiente é o processo de coleta. Os autores também utilizam um mecanismo para avaliar a quantidade de recursos restantes em uma fonte e utilizam essa métrica para determinar quando os robôs devem abandonar a coleta e voltar a exploração. Os experimentos realizados indicam que se parte do grupo retornar a exploração antes do esgotamento total da fonte, a coleta é realizada de forma mais eficiente.

Em Lima e Oliveira (2016a), as autoras concentram seu estudo na utilização de busca tabu para popular a memória dos robôs. Os experimentos avaliam diversas políticas de busca para popular a memória dos robôs. Além da memória interna, os robôs possuem a capacidade de depositar feromônio e utilizam o feromônio invertido para maximizar a área explorada. Os experimentos demonstram que a abordagem proposta apresenta melhor desempenho que controladores determinísticos encontrados na literatura.

Em Yang et al. (2009), os autores concentram seu estudo na divisão de trabalho dentro do problema de coleta. Nos experimentos realizados existe uma demanda por recursos, conforme a demanda aumenta, mais robôs são alocados para realizar a coleta, conforme os recursos são entregues a demanda diminui. O algoritmo proposto é utilizado para alocar dinamicamente a quantidade ideal de robôs para manter a demanda estável.

Em Hara et al. (2016), os autores concentram seu estudo no operador de imitação. Segundo os autores o operador de imitação é uma ferramenta que permite que um robô copie o comportamento de outro. Porém, com o aumento da quantidade de robôs em um enxame, a quantidade de robôs que podem ser copiados aumenta. Nesse trabalho o controlador dos robôs é gerado automaticamente através de programação genética. Experimentos são realizados para avaliar diferentes operadores de imitação.

Além dos trabalhos apresentados na tabela existem dois outros trabalhos precisam ser destacados. Em Just e Moses (2017) e Soares et al. (2018) foi utilizado um método de automação parcial. Nesses trabalhos, parte do controlador foi definido pelo projetista, mas diversos parâmetros foram otimizados através de um algoritmo de otimização. Essa é uma abordagem híbrida entre técnicas de aprendizado e definição manual do controlador.

3.1.1 Análise

As informações da tabela 3.1 são apresentadas de forma sumarizada na tabela 3.2.

Tabela 3.2: Sumarização de características

Característica	Distribuição de trabalhos
Aprendizado	67% Não / 33% Sim
Simulação	67% Contínua / 33% Discreta
Comunicação	38% Nenhuma / 24% Direta / 38% Indireta
Distribuição	29% Concentrada / 62% Distribuída / 9% Ambas
Obstáculos	67% Sim / 33% Não

A tabela 3.2 mostra que a maior parte dos trabalhos não utiliza aprendizado de máquina para gerar os controladores. Vale ressaltar que existem trabalhos recentes que utilizam e não utilizam aprendizado, por exemplo Nogales e de Oliveira (2018) não utiliza enquanto Ericksen et al. (2017) utiliza.

Como previamente apresentado, para gerar um controlador utilizando aprendizado é necessária uma etapa de treinamento, que pode ser custosa. Controladores criados manualmente necessitam apenas de ajuste de parâmetros, sendo assim uma opção de menor custo computacional. Como os resultados obtidos utilizando ou não aprendizado são similares, a opção pela técnica de menor custo pode justificar a preferência por técnicas que não utilizam aprendizado.

Entretanto, outro fator a ser considerado é a variedade de problemas de coleta estudados. Nos trabalhos apresentados nessa revisão, poucos são os que utilizam exatamente a mesma configuração de coleta. Dependendo do que é avaliado em cada estudo, a utilização de aprendizado pode apresentar melhores resultados ou vice-versa.

Em relação aos métodos de simulação utilizados, a representação contínua tem preferência em relação à representação discreta. Enquanto a representação contínua gera simulações mais similares a cenários reais a representação discreta permite simulações mais controladas. Trabalhos que apresentam provas formais normalmente utilizam a simulação discreta enquanto trabalhos que tentam simular cenários reais têm preferência pela simulação contínua.

Em relação ao método de comunicação utilizado pelos robôs, a maior parte dos trabalhos apresenta robôs com algum método de comunicação. Quando não existe comunicação, os robôs mudam seu comportamento apenas quando detectam outro robô. Neste caso os robôs atuam de forma mais individualista, ou seja, o comportamento do robô é determinado apenas pelo que ele mesmo detecta. Quando existe comunicação, ela é utilizada como um meio de coordenação do enxame. Os robôs conseguem informar onde mais trabalho é necessário e conseguem evitar que outro robô repita o trabalho que outro robô já fez.

Outro fator a se considerar sobre comunicação é a capacidade de implementar soluções de comunicação em robôs reais. Troca de mensagens pode ser afetada por interferência enquanto comunicação através de feromônio necessita do feromônio em si e sensores para detectar os mesmos.

Em relação à distribuição dos recursos, a maior parte dos trabalhos distribui os recursos de forma espalhada. Os trabalhos que utilizam os recursos de forma concentrada apresentam as dificuldades adicionadas ao problema quando esse tipo de distribuição é utilizada, por exemplo, a concorrência em zonas de interesse. Trabalhos que utilizam recursos concentrados focam na análise dessas dificuldades adicionais enquanto outros trabalhos apresentam a análise de maneira mais ampla.

Em relação à presença de obstáculos, a maioria dos trabalhos apresenta cenários onde existem obstáculos. Obstáculos adicionam complexidade ao problema e tornam os cenários mais realísticos. Porém mesmo quando não existem obstáculos, os próprios robôs do enxame podem atuar como obstáculos.

3.1.2 Considerações finais

Através da tabela 3.2 podem-se extrair algumas informações sobre o estado atual da literatura. Primeiramente, embora algoritmos de aprendizado sejam utilizados para a geração de controladores para esse problema, essa não é a técnica predominante. A maior parte das simulações utilizam a representação contínua, que é mais próxima de cenários reais. Na maior parte dos trabalhos existe algum tipo de comunicação, indireta na maioria dos casos, embora a quantidade de trabalhos onde o enxame não se comunica seja expressiva. Existe uma preferência em realizar experimentos em cenários onde os recursos estão distribuídos e existem obstáculos.

Ressalta-se que muitos trabalhos de coleta de recursos como um todo foram filtrados e excluídos desta revisão bibliográfica e as estatísticas apresentadas na tabela 3.2 não representam o estado da literatura do problema de coleta de recursos como um todo, apenas apresenta um resumo dos trabalhos considerados mais relevantes para este estudo.

3.2 MOLDAGEM

Nesta seção são apresentados os trabalhos que utilizam a técnica de moldagem. A maior parte dos trabalhos que apresentam o termo moldagem (*shaping*) não são recentes, entretanto também serão apresentados trabalhos que embora não utilizem o termo moldagem se enquadram na definição do termo. Vale ressaltar que os trabalhos apresentados nessa seção se enquadram no contexto de robótica mas não necessariamente na tarefa de coleta de recursos.

Em Lin (1991) o autor utiliza aprendizado por reforço para fazer com que um robô aprenda a realizar 3 tarefas: andar por um corredor, atravessar uma porta e estacionar no carregador. *Q-Learning* é utilizado como algoritmo de aprendizado e para cada um dos comportamentos é utilizada uma função Q diferente. Cada uma das funções Q é representada por uma rede neural distinta. Além das três redes neurais, o robô possui uma máquina de estados que seleciona qual dos comportamentos deve ser utilizado no momento.

Nesse trabalho o autor enfatiza como técnicas de aprendizado por reforço demoram para convergir e apresenta duas técnicas auxiliares que foram utilizadas para acelerar a convergência do algoritmo *Q-Learning*: repetição de experiência (*Experience replay*) e moldagem (*Teaching*).

O *Q-Learning* é um algoritmo iterativo. Segundo esse trabalho, o *Q-learning* tradicional, em cada iteração, gera uma sequência de experiências que são utilizadas para realizar ajustes nessa iteração e depois são descartadas. A repetição de experiência consiste de armazenar a sequência de experiências e utiliza-lá várias vezes em ordem reversa. Segundo esse trabalho, a utilização da sequência em ordem reversa proporciona um ajuste mais rápido dos valores da função Q.

A técnica de moldagem, chamada de *teaching* nesse trabalho, consiste de adicionar experiências artificiais durante o treinamento. Como mencionado previamente, em cada iteração o robô gera um conjunto de experiências. Essas experiências são execuções do atual controlador do robô. Em outras palavras, o robô aprende utilizando a sua própria experiência. Nesse trabalho, além das experiências geradas pelo próprio robô, o autor adiciona experiências criadas ao se controlar o robô manualmente. As experiências artificiais são armazenadas no conjunto de experiências mantido pela técnica de repetição de experiências. Durante a execução do

Q-learning as experiências artificiais são utilizadas da mesma forma que experiências geradas pelo próprio robô.

Nesse trabalho o autor realiza experimentos comparando o tempo de convergência do *Q-learning* tradicional e o assistido por moldagem e repetição de experiência. Nos resultados apresentados, o algoritmo assistido converge mais rapidamente.

Em Dorigo e Colombetti (1994) os autores utilizam aprendizado por reforço para fazer com que um robô aprenda diferentes comportamentos. Esses comportamentos são: Perseguir um objeto, se camuflar no ambiente, fugir de predador e encontrar um objeto escondido. Para cada comportamento foi realizado um conjunto de experimentos separados, isto é, um robô não deve aprender todos os comportamentos ao mesmo tempo. Em cada experimento o robô tenta aprender um comportamento diferente.

Os autores utilizam um conjunto de sistemas classificadores, combinado com um algoritmo genético como modelo de aprendizado. Um sistema classificador é um conjunto de regras. Um sistema classificador é um algoritmo próprio e não deve ser confundido com classificadores do contexto de aprendizado supervisionado como SVM e *naive bayes*. O funcionamento de um sistema classificador, para o contexto desse trabalho, será explicado a seguir.

Quando a entrada do sistema casa com alguma regra existente, o sistema emite uma mensagem correspondente. Nesse caso, a entrada do sistema é a percepção do robô enquanto a mensagem correspondente é a ação que o robô deve tomar. O conjunto de regras é gerado automaticamente por um algoritmo genético.

Para gerar as regras, o sistema de classificadores interage com o ambiente de simulação e recebe reforço do ambiente, utilizado para ajustar o comportamento do robô. O reforço é utilizado para determinar se uma regra deve permanecer ou ser substituída. Para melhorar a eficiência do sistema os autores utilizam um treinador, também chamado de agentes de moldagem.

O treinador é um elemento fora modelo de aprendizado que observa o desempenho do aprendizado. Com base em suas observações, o treinador oferece um reforço adicional (além do reforço obtido pelo ambiente). O reforço adicional faz com que o algoritmo genético tenha maior probabilidade de manter as regras que receberam reforço adicional. Em outras palavras, o treinador molda o sistema classificador ao modificar o reforço recebido pelo sistema.

Nesse trabalho os autores comparam controladores gerados com e sem a utilização de moldagem. Os gerados utilizando moldagem obtiveram melhor desempenho que os gerados sem moldagem. Além disso, os autores ressaltam que o agente de moldagem não precisa ter conhecimento do funcionamento interno do modelo de aprendizado para realizar moldagem.

Em Kaelbling et al. (1996), os autores realizam uma revisão da literatura sobre aprendizado por reforço. Em sua análise, os autores concluem que embora os algoritmos de aprendizado por reforço possam convergir sem o auxílio de técnicas auxiliares, a utilização de tais algoritmos é encorajada. Os autores agrupam diversas técnicas em classes e colocam as técnicas propostas em Lin (1991) (*teaching*) e Dorigo e Colombetti (1994) (*shapping*) na mesma categoria, moldagem.

Em Kadota et al. (2012), os autores utilizam aprendizado por reforço para ensinar um robô a realizar a tarefa de coleta de recursos. Nesse trabalho os autores utilizam um rede neural como controlador cujos pesos são otimizados através de um algoritmo genético. O conceito de aptidão incremental também é utilizado.

Em cada iteração do algoritmo genético, as redes neurais são avaliadas e recebem um valor de aptidão. Esse valor é dado por uma função de aptidão. Segundo os autores, a aptidão tradicional avalia os controladores com base na tarefa final desejada. Essa função é a mesma durante todo o aprendizado. A aptidão incremental avalia os robôs com base em sub tarefas.

A aptidão total do robô é a soma das aptidões das sub tarefas. Se um robô não realiza aquela sub tarefa, essa aptidão é descartada. A aptidão incremental faz com que durante o processo de aprendizado o robô seja avaliado por diferentes funções de aptidão, conduzindo o aprendizado do robô para primeiramente realizar as sub tarefas e considerar a tarefa final como uma combinação das sub tarefas.

Os autores realizaram experimentos comparando a aptidão tradicional com a aptidão incremental. A aptidão incremental obteve melhor desempenho.

A técnica de aptidão incremental também é utilizada em Yasuda et al. (2013) e Ohkura et al. (2013) porém como a técnica é utilizada da mesma forma que em Kadota et al. (2012) e esses dois trabalhos já foram descritos em detalhes na seção anterior, portanto, esses trabalhos não serão discutidos em detalhes nessa seção.

3.2.1 Análise

Nos trabalhos apresentados nessa seção os autores apresentam o tempo de convergência de algoritmos de aprendizado de reforço como um dos principais problemas dessa classe de algoritmos. Para aprender através de reforço, os agentes utilizam as suas próprias experiências, e baseado no reforço recebido, ajustam o seu comportamento.

Como forma de reduzir o tempo de convergência, esse trabalhos utilizam, entre outras técnicas, a moldagem, que consiste de prover informação privilegiada ao agente. Informação privilegiada é informação utilizada durante o aprendizado que não é gerada pelo próprio agente.

Ao aprender utilizando apenas as suas próprias experiências, o agente irá tentar diversos caminhos improdutivos durante o seu aprendizado. Os trabalhos apresentados nessa seção utilizam diferentes técnicas que restringem a quantidade de caminhos explorada pelo agente.

Ao restringir a quantidade de caminhos explorada, os caminhos restantes podem ser averiguados com maior profundidade, resultando a convergência mais rápida para um desses caminhos. A intenção da técnica de moldagem é garantir que os caminhos restantes contenham boas soluções para o problema. Como o agente inicialmente não sabe quais são os caminhos promissores, essa informação vem de uma fonte externa.

Todos os trabalhos apresentados nessa seção realizaram experimentos comparando algoritmos de aprendizado por reforço com e sem a utilização de uma técnica específica de moldagem. Em todos os trabalhos, os algoritmos assistidos por moldagem convergiram mais rápido que o mesmo algoritmo sem a assistência de moldagem.

4 PROPOSTA

Através de análise da literatura sobre o problema de coleta de recursos em robótica detectou-se que existe espaço para a utilização de algoritmos de aprendizado por reforço para a geração automática de controladores de enxames de robôs.

Aprendizado por reforço é uma técnica que permite a geração automática de controladores que tem sido utilizada com sucesso no contexto de coleta de recursos e tem obtido resultados competitivos em ambientes complexos.

Alguns trabalhos na literatura como Ericksen et al. (2017) e Lin (1991) destacam diferentes vantagens na utilização de aprendizado no campo da robótica, dentre elas algoritmos com capacidade de generalização.

Generalização é a capacidade do algoritmo de ser utilizado em diferentes configurações do ambiente e diferentes instâncias desse ambiente. No campo da robótica, os problemas possuem uma grande quantidade de variáveis como capacidade física dos robôs e elementos do ambiente. Controladores projetados para um cenário específico podem não apresentar bom desempenho em cenários diferentes. Algoritmos de aprendizado permitem que os controladores se adaptem.

Embora existam vantagens, algoritmos de aprendizado por reforço possuem um grande custo computacional. Estes algoritmos são iterativos. Em cada iteração o algoritmo realiza ajustes no modelo, no caso de robótica, no controlador do robô, até que o modelo obtenha um desempenho aceitável na tarefa. O custo do algoritmo é medido pela quantidade de iterações necessárias para que o modelo atinja o desempenho desejado. Em outras palavras, para que o modelo atinja um desempenho aceitável, é necessário que o algoritmo execute por muitas iterações.

Isso se deve principalmente a dois fatores, a quantidade de estados e a solução ser representada por uma sequência de ações. Esses dois fatores combinados exigem que o modelo seja ajustado diversas vezes, o que resulta em uma grande quantidade de iterações. O tempo, isto é, quantidade de iterações, necessárias para que o modelo atinja o desempenho desejado é chamado de tempo de convergência.

A proposta deste trabalho é abordar o problema de convergência de algoritmos de aprendizado aplicados a robótica através da técnica de moldagem. Essa técnica foi utilizada com sucesso em robótica, porém não exite um estudo amplo dessa técnica na tarefa de coleta de recursos.

Os objetivos desse trabalho são: desenvolver um simulador de robótica simplificado, avaliar o desempenho do algoritmo NEAT aplicado ao problema de coleta de recursos e implementar e avaliar diferentes algoritmos de moldagem aplicados ao problema de coleta de recursos.

No restante deste capítulo serão definidos: a variação de coleta de recursos que foi utilizada, as características do simulador, as características do robô simulado, a configuração do enxame e os algoritmos de moldagem estudados.

4.1 DESCRIÇÃO DO PROBLEMA E AMBIENTE DE TAREFA

Como mencionado previamente nos capítulos anteriores, embora coleta de recursos tenha um objetivo bem definido, existem diferentes fatores como as capacidades físicas do robô e características do ambiente de simulação que afetam o desempenho dos robôs na resolução do problema. A seguir, será definida qual variação do problema foi utilizada neste estudo, isto é,

quais características estão presentes no ambiente de tarefas e quais as capacidades físicas dos robôs simulados.

4.1.1 Descrição do ambiente de tarefas

As instâncias neste estudo satisfazem as seguintes caraterísticas:

- Foi utilizado um enxame de robôs homogêneos.
- Existe uma região especial no mapa demarcada como ninho. Os robôs devem deslocar os recursos até essa região. Os robôs iniciam a simulação dentro do ninho.
- O ambiente (mapa) é descrito por uma matriz bidimensional.
- O tempo é simulado de forma discreta (iterações).
- A cada iteração da simulação, cada robô pode realizar uma única ação.
- As ações do robô são: mover-se a uma posição adjacente na matriz (cima, baixo, direita ou esquerda) ou ficar na mesma posição.
- Existe um número finito de recursos espalhado no ambiente.
- Um recurso ocupa exatamente uma posição na matriz.
- Um robô ocupa exatamente uma posição na matriz.
- Quando um recurso é coletado (colocado no ninho) ele não volta a ocupar uma posição no mapa, ele é apenas contabilizado como devolvido. Não existe reposição de recursos.
- Cada robô possui um compartimento de carga, com a capacidade suficiente para carregar um único recurso.
- Quando o robô se desloca para uma posição ocupada por um recurso e seu compartimento de carga está vazio, ele coloca esse recurso no compartimento imediatamente.
- Um recurso que é colocado no compartimento de carga é removido do mapa.
- Quando o robô se desloca para uma posição ocupada por um recurso e seu compartimento de carga está cheio, ocorre uma colisão.
- Quando ocorre uma colisão, o robô não consegue mudar de posição, permanecendo na mesma posição que se encontrava anteriormente.
- Quando o robô se desloca para uma posição de ninho e está com o compartimento de carga cheio, ele esvazia o compartimento e um recurso é coletado.
- Podem existir obstáculos fixos. Quando um robô se desloca para uma posição onde existe um obstáculo fixo, ocorre uma colisão.
- Se ao final de uma iteração dois robôs se deslocam para a mesma posição, ocorre uma colisão e eles retornam às posições em que estavam no início da iteração.
- Os robôs possuem apenas visão parcial do ambiente. Eles podem observar apenas posições da matriz a uma distância x, onde x é um parâmetro da instância.
- A simulação termina após um número determinado de iterações.

4.1.2 Simulação e simulador

Os experimentos foram realizados através de simulações. O simulador utilizado é de implementação própria e desenvolvido especialmente para a realização deste trabalho e para simular instâncias com as caraterísticas previamente mencionadas.

O simulador recebe como parâmetro a instância a ser simulada e os parâmetros do enxame. O simulador emite uma saída a cada iteração da simulação. A saída é a percepção de cada robô simulado, isto é, o que cada robô do enxame visualiza do ambiente em que está inserido. Em cada iteração o simulador espera uma ação para cada robô simulado como entrada. Quando o simulador recebe a ação de cada robô ele simula uma iteração e coloca na saída o resultado daquela iteração, isto é, a nova percepção de cada robô. Esse processo se repete até o número limite de iterações. Ao final da simulação o simulador emite um relatório com métricas de resultado da simulação. Essas métricas estão descritas na tabela 4.1.

Tabela 4.1: Relatório da simulação

Métrica	Descrição
Qtd. Rec	Quantidade de recursos coletados
Qtd. Col	Quantidade de colisões
Qtd. Act	Quantidade de ações realizadas
Qtd. Car	Quantidade de recursos colocados no compartimento de carga
Qtd. ActCar	Quantidade de ações realizadas pelo robô carregado
Qtd. ActDes	Quantidade de ações realizadas pelo robô descarregado
Qtd. Expl	Quantidade de posições distintas da matriz visitadas
Qtd. Rev	Quantidade de ações realizadas em posições revisitadas
Dist. Ninho	Distância até o ninho, se o robô está carregado
Dist. Rec	Distância até o recurso mais próximo dentro do raio de visão

O simulador devolve como resultado o relatório de todo o enxame. As métricas apresentadas são dadas soma das métricas individuais de cada robô.

Os seguintes argumentos são utilizados para justificar a implementação da simulação através de mensagens de entrada e saída:

- O simulador se torna independe do enxame, isto é, alterações no código do simulador não interferem no código do algoritmo de aprendizado e vice-versa.
- Permite que o controlador seja implementado em qualquer linguagem, independente da linguagem do simulador.
- O controlador não está restrito às bibliotecas utilizadas para implementar o simulador.

4.1.3 Descrição dos robôs e enxame

O enxame é formado por robôs homogêneos, isto é, todos possuem o mesmo controlador e com as mesmas capacidades físicas. O robô pode realizar uma única ação por iteração da simulação e essas ações são: se deslocar para uma posição adjacente (cima, baixo, direita ou esquerda) ou permanecer parado.

O controlador utilizado é uma rede neural gerada automaticamente pelo algoritmo NEAT. A topologia interna da rede é definida pelo algoritmo NEAT mas os nós de entrada e saída são fixos, já que representam a informação que o robô percebe do ambiente e a ação que irá realizar respectivamente.

A topologia de entrada e saída da rede foi inspirada nos trabalhos de Kadota et al. (2012), Yasuda et al. (2013) e Ohkura et al. (2013). A rede possui as mesmas informações de entrada e saída porém foi adaptada para o cenário discreto. Nesses trabalhos a simulação considera um ambiente contínuo.

A rede possui 9 neurônios de entrada: 1 indicando se o compartimento de carga está vazio ou cheio, 2 utilizados para representar um vetor que indica a direção do ninho, 2 utilizados para representar um vetor que indica a direção do recurso mais próximo e 4 para indicar colisão iminente nas 4 posições adjacentes. A rede possui 5 neurônios de saída, cada um representando uma possível ação. A ação realizada é aquela com o maior valor no respectivo neurônio de saída. Um esquema da rede neural utilizada é apresentado na figura 4.1.

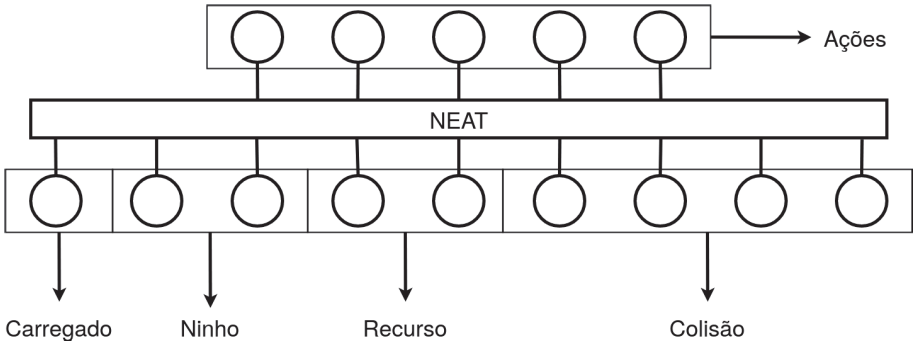


Figura 4.1: Representação da rede neural dos robôs.

4.2 ALGORITMOS DE MOLDAGEM E REGIME DE TREINAMENTO

Este estudo avalia diferentes formas de realizar o treinamento do algoritmo NEAT. Cada uma dessas formas é chamada de regime de treinamento. Quatro regimes diferentes serão avaliados, um regime sem moldagem chamado de regime SM (sem moldagem) e três regimes que utilizam moldagem: regime MP (moldagem por professor), regime MI (moldagem por alteração de instância) e regime MR (moldagem por alteração de recompensa).

4.2.1 Regime sem moldagem (SM)

No regime sem moldagem, o treinamento é realizado utilizando um único conjunto de instâncias. A mesma função de aptidão é utilizada durante todo o treinamento. A função de aptidão utiliza as métricas da tabela 4.1 para calcular a aptidão do enxame. Para referenciar esse regime será utilizada a sigla SM.

4.2.2 Regime moldagem por professor (MP)

O treinamento utilizando moldagem por professor é inspirado em Dorigo e Colombetti (1994). Neste regime, o treinamento é realizado utilizando um único conjunto de instâncias. Uma única função de aptidão é utilizada, porém diferente do treinamento sem moldagem, essa função conta com o reforço adicional do professor.

O professor é uma lógica que compara as ações do robô com as ações de outro controlador chamado de algoritmo do professor. O algoritmo do professor pode ser um outro controlador, uma rede neural já treinada ou outro modelo capaz de apresentar uma ação do robô, dado o estado atual. Neste trabalho o algoritmo do professor utilizado é:

- Se o robô está carregado, move o robô em direção ao ninho (considerando o vetor de entrada)
- Se o robô está não carregado, move o robô em direção ao recurso mais próxima (considerando o vetor de entrada)

Quando o robô executa uma ação, o professor simula qual ação teria sido executada pelo algoritmo do professor. Quando o robô e o professor concordam, o robô recebe reforço positivo, quando discordam ele recebe reforço negativo. A função de aptidão exata será apresentada a seguir na seção de parâmetros. Para referenciar esse regime será utilizada a sigla MP.

4.2.3 Regime moldagem por alteração de instância (MI)

O treinamento utilizando moldagem por alteração de instância é inspirado em Lin (1991). Neste regime, o treinamento é realizado utilizando três conjuntos de instâncias diferentes. Durante o primeiro terço do treinamento, isto é, um terço do total de gerações, o robô treina em um conjunto com instâncias de complexidade simples. Ao iniciar o segundo e o último terços, as instâncias de treinamento são substituídas por instâncias de complexidade maior. Uma única função de aptidão é utilizada durante todo o treinamento. Essa função de aptidão utiliza as métricas geradas no relatório final da simulação para calcular a aptidão do enxame. Para referenciar esse regime será utilizada a sigla MI.

4.2.4 Regime moldagem por alteração de recompensa (MR)

Por fim, o treinamento utilizando moldagem com alteração de função de recompensa é inspirado em Kadota et al. (2012). Neste regime, o treinamento é realizado utilizando um único conjunto de instâncias. Três funções de aptidão diferentes são utilizadas. Quando o treinamento completa o primeiro e segundo terços, a função de aptidão é trocada. As funções de aptidão utilizam as métricas geradas no relatório final da simulação para calcular a aptidão do enxame. Para referenciar esse regime será utilizada a sigla MR.

4.3 CONCLUSÃO

Neste capítulo foram apresentados a proposta e os objetivos deste trabalho. O problema estudado foi explicado em detalhes, assim como as técnicas de moldagem que serão avaliadas neste trabalho. No próximo capítulo serão apresentados os experimentos realizados para avaliar essas técnicas e análises dos resultados obtidos.

5 ANÁLISE EXPERIMENTAL

Para sustentar a proposta deste trabalho foi executado um conjunto de experimentos. O objetivo desses experimentos é avaliar as diferenças no desempenho do algoritmo NEAT causadas por algoritmos de moldagem. Neste capítulo serão apresentados: os materiais e métodos utilizados, resultados obtidos e análise dos resultados.

5.1 MATERIAIS E MÉTODOS

Os experimentos realizados neste trabalho consistem em executar diferentes regimes de treinamento do algoritmo NEAT para avaliar as diferentes formas de gerar controladores de robôs automaticamente.

A biblioteca NEAT-Python, versão 0.92 foi utilizada para gerar os controladores. A mesma configuração da biblioteca foi utilizada para todos os experimentos realizados. O arquivo completo de configuração pode ser encontrado no apêndice A. A função de aptidão não é considerada um parâmetro da biblioteca e será apresentada a seguir.

O simulador utilizado é de implementação própria e está disponível como Software Livre (https://github.com/lucasfo/robot-environment). Os parâmetros utilizados nas simulações serão descritos na seção de parametrização.

Os experimentos foram realizados nas máquinas virtuais do parque computacional do Departamento de Informática da UFPR.

Quatro regimes de treinamento do algoritmo NEAT foram avaliados: regime sem moldagem (SM), regime com moldagem por professor (MP), regime com moldagem por alteração de instância (MI) e regime com moldagem por alteração de recompensa (MR).

Como as condições de treinamento dos regimes são diferentes, os resultados obtidos em cada um dos ambientes de treinamento não são comparáveis entre si de forma justa. Por essa razão, para cada regime, foram utilizados dois ambientes de simulação: um ambiente de treinamento e um ambiente de validação.

No ambiente de treinamento o algoritmo NEAT é executado. Os resultados das simulações são utilizados para definir a aptidão das redes neurais e para criar o conjunto de redes da próxima geração. Os algoritmos de moldagem podem manipular algumas características do ambiente de treinamento como, por exemplo, as instâncias e funções de aptidão utilizadas durante a simulação. Por essa razão, cada regime de treinamento é executado em um ambiente de treinamento diferente.

O ambiente de validação existe para comparar o resultado de cada um dos regimes de treinamento. Para comparar os resultados, as redes de diferentes regimes de treinamento são copiadas para o mesmo ambiente de validação. No ambiente de validação, o algoritmo NEAT não é executado. Neste ambiente, as redes geradas em diferentes regimes executam simulações sob as mesmas condições para que seu desempenho possa ser comparado. Como o ambiente de validação é o mesmo para todos os regimes de treinamento, os resultados obtidos nesse ambiente podem ser comparados entre si. Por essa razão, apenas os resultados obtidos nesse ambiente são reportados nos experimentos.

O algoritmo NEAT é um algoritmo iterativo populacional. Cada uma de suas iterações é chamada de geração. O treinamento do algoritmo NEAT consiste da realização de diversas gerações. Em cada geração, cada uma das redes na população é submetida a uma série de

simulações. O resultado obtido nessas simulações é utilizado para calcular a aptidão e gerar as redes da próxima geração.

Os experimentos realizados avaliam todas as redes geradas em cada uma das gerações do algoritmo NEAT. Durante o treinamento, para cada geração, o seguinte procedimento é realizado:

- O algoritmo NEAT treina por uma geração, no ambiente de treinamento, gerando um conjunto de redes.
- As redes são copiadas do ambiente de treino para o ambiente de validação.
- As redes copiadas executam simulações no ambiente de validação.
- O resultado no ambiente de validação é guardado para ser reportado como resultado dos experimentos.
- O treinamento segue para a próxima geração.

5.2 EXPERIMENTOS PRELIMINARES E FIXAÇÃO DE PARÂMETROS

A quantidade de parâmetros e variáveis envolvidas no treinamento do algoritmo NEAT inviabiliza uma análise rigorosa do efeito de todos os parâmetros no tempo disponível para a realização deste trabalho.

Para mitigar esse problema foi realizado um conjunto de experimentos preliminares com o objetivo de reduzir o escopo dos parâmetros avaliados. Estes experimentos foram utilizados para definir quais parâmetros seriam fixos e quais seriam variáveis nos experimentos. Também foram selecionados os valores possíveis dos parâmetros variáveis.

Os experimentos preliminares e o desenvolvimento do simulador foram realizados simultaneamente. Os resultados preliminares obtidos foram usados para aperfeiçoar o simulador. Diferentes modos de simulação, diferentes capacidades físicas e diferentes métricas usadas em funções de aptidão são algumas das características que foram modificadas e aperfeiçoadas ao longo do desenvolvimento em função dos resultados preliminares.

Algumas caraterísticas que foram avaliadas durante os experimentos preliminares mas foram removidas dos experimentos finais são:

- Utilização de comunicação entre robôs utilizando feromônio.
- Conjuntos de ações possíveis diferentes (sem a possibilidade de se manter parado, com mais opções de movimento)
- Diferentes métodos de entrada de informação (informação bruta, diferentes pré-processamentos)

Essas variações foram rejeitadas durante os experimentos preliminares por adicionar complexidade excessiva. Devido ao tempo e recurso limitado para a realização de experimentos, optou-se por remover essas características ou utilizar características menos complexas.

Além da parametrização, as seguintes conclusões preliminares foram formadas:

- Apenas a quantidade de recursos coletados (principal métrica de desempenho do problema) como função de aptidão é insuficiente para que o algoritmo de treinamento convirja em algumas instâncias.
- Os seguintes parâmetros foram avaliados como os de maior impacto no tempo de convergência e resultado (quantidade de recursos coletados) do algoritmo:

  - As instâncias utilizadas no ambiente de treinamento e no de simulação.
  - A quantidade de instâncias nos conjuntos de treinamento e validação.
  - A quantidade de robôs no enxame.
  - A quantidade de informação usada como entrada da rede neural.
  - A qualidade da informação usada como entrada da rede neural.

Em função dessas conclusões preliminares, para reduzir a quantidade de experimentos realizados e o escopo da análise, foi definido que:

- As funções de aptidão seriam fixas e considerariam diversas métricas retornadas após o fim da simulação.
- Seriam realizados experimentos com diferentes composições dos conjuntos de instâncias de treino e validação.
- A quantidade de robôs no enxame seria fixa.
- A rede receberia uma entrada pré-processada ao invés da percepção completa do robô.

5.3 PARAMETRIZAÇÃO GERAL

Em funções dos resultados preliminares foi escolhido realizar diversos conjuntos de experimentos finais, cada um deles considerando diferentes composições do conjunto de instâncias de treinamento e de validação.

O principal objetivo dos experimentos é avaliar o impacto das diferentes técnicas de moldagem, porém, foi observado que os resultados podem variar dependendo das instâncias utilizadas. Neste capítulo há uma seção dedicada a cada composição do conjunto de instâncias. Nesta seção serão apresentados os parâmetros comuns a todos os experimentos finais.

Os experimentos consistem em executar cada um dos regimes de treinamento pela mesma quantidade de gerações. Em cada geração as redes mantidas na população do algoritmo NEAT passam pelo processo de validação. O objetivo desses experimentos é verificar se existe diferença estatística nos regimes de treinamento avaliados. Para isso, cada um dos regimes foi repetido diversas vezes e submetido ao teste estatístico t de Student.

Uma descrição detalhada dos parâmetros utilizados é apresentada na tabela 5.1. As funções de aptidão referenciam funções que serão descritas em detalhes a seguir.

Tabela 5.1: Configuração dos experimentos (Parâmetros gerais)

Parâmetro	**Valor**
Repetições	30
Gerações	300
Tamanho da população	100
Tamanho do enxame	5
Tamanho da matriz do mapa	20x20
Iterações por simulação	50
Função de aptidão (SM)	f_0
Função de aptidão (MI)	f_0
Função de aptidão (MP)	f_p
Função de treino 1 (MR)	f_1
Função de treino 2 (MR)	f_2
Função de treino 3 (MR)	f_0

Os parâmetros utilizados foram selecionados através de experimentos preliminares. Os regimes: SM, MI e MP utilizam uma única função de aptidão durante todo o treinamento, já o regime MR utiliza três funções diferentes. Vale ressaltar que a função de aptidão utilizada na etapa final do regime MR é a mesma utilizada por todo o regime SM e por todo o regime MI. O regime MP utiliza uma função exclusiva por considerar informação do algoritmo do professor.

As funções de aptidão são descritas em função das métricas retornadas no relatório da simulação e do valor retornado pelo professor. A equação 5.1 descreve a fórmula geral das equações de aptidão:

$$\begin{aligned} f(x) = m_0 * Qtd.Rec + m_1 * Qtd.Col + m_2 * Qtd.Act + m_3 * Qtd.Car + \\ m_4 * Qtd.ActCar + m_5 * Qtd.ActDes + m_6 * Qtd.Expl + \\ m_7 * Qtd.Rev + m_8 * Dist.Ninho + \\ m_9 * Dist.Rec + m_{10} * Rec.Prof \end{aligned} \tag{5.1}$$

Em resumo, cada métrica possui um multiplicador m_i e a função de aptidão é dada pela soma das métricas multiplicadas pelo seu respectivo multiplicador. Utilizando a equação 5.1 é possível descrever as diferentes funções de aptidão apenas informando os multiplicadores.

Os multiplicadores das funções f_0, f_1, f_2 e f_p são descritos pelas tabelas 5.2, 5.3, 5.4 e 5.5 respectivamente.

Os valores utilizados como multiplicadores foram definidos através de experimentos preliminares.

Tabela 5.2: Função de aptidão f_0

Métrica	**Multiplicador**
Qtd. Rec	1000
Qtd. Col	-5
Qtd. Act	0
Qtd. Car	100
Qtd. ActCar	1
Qtd. ActDes	0
Qtd. Expl	10
Qtd. Rev	0
Dist. Ninho	10
Dist. Rec	1
Rec. Prof	0

Tabela 5.3: Função de aptidão f_1

Métrica	**Multiplicador**
Qtd. Rec	1000
Qtd. Col	0
Qtd. Act	0
Qtd. Car	1000
Qtd. ActCar	0
Qtd. ActDes	0
Qtd. Expl	100
Qtd. Rev	0
Dist. Ninho	10
Dist. Rec	1
Rec. Prof	0

Tabela 5.4: Função de aptidão f_2

Métrica	**Multiplicador**
Qtd. Rec	1000
Qtd. Col	-10
Qtd. Act	0
Qtd. Car	1000
Qtd. ActCar	0
Qtd. ActDes	0
Qtd. Expl	0
Qtd. Rev	0
Dist. Ninho	10
Dist. Rec	1
Rec. Prof	0

Tabela 5.5: Função de aptidão f_p

Métrica	**Multiplicador**
Qtd. Rec	1000
Qtd. Col	-10
Qtd. Act	0
Qtd. Car	100
Qtd. ActCar	0
Qtd. ActDes	0
Qtd. Expl	0
Qtd. Rev	0
Dist. Ninho	0
Dist. Rec	0
Rec. Prof	5

5.4 EXPERIMENTOS - MÚLTIPLAS INSTÂNCIAS

O primeiro conjunto de experimentos finais utiliza um conjunto de instâncias para treinamento e validação dos controladores. As instâncias utilizadas estão descritas na tabela 5.6. A descrição completa das instâncias pode ser encontrada no apêndice B.

Tabela 5.6: Configuração dos experimentos (Parâmetros específicos) (Múltiplas instâncias)

Parâmetro	**Valor**
Conjunto de validação	[valid-00, valid-01, valid-02]
Conjunto de treino (SM)	[hard-00, hard-01, hard-02]
Conjunto de treino (MR)	[hard-00, hard-01, hard-02]
Conjunto de treino (MP)	[hard-00, hard-01, hard-02]
Conjunto de treino 1 (MI)	[easy-00, easy-01, easy-02]
Conjunto de treino 2 (MI)	[medium-00, medium-01, medium-02]
Conjunto de treino 3 (MI)	[hard-00, hard-01, hard-02]

O conjunto de validação é o conjunto utilizado no ambiente de validação para comparar o resultado de todos os regimes de treinamento. O conjunto de treino é utilizado pelos regimes SM, MP e MR no ambiente de treino. O regime MI utiliza o conjunto de treino 1 no primeiro terço do treinamento, o conjunto de treino 2 no segundo terço do treinamento e o conjunto 3 no último terço do treinamento. Embora sejam instâncias diferentes, os regimes sempre treinam e são validados utilizando a mesma quantidade de instâncias.

Os parâmetros deste conjunto de experimentos são os parâmetros gerais descritos na tabela 5.1 e os parâmetros específicos descritos na tabela 5.6.

5.4.1 Resultados

Para demonstrar os resultados da simulação serão apresentados gráficos que comparam os regimes de treinamento.

Os gráficos apresentam o desempenho dos regimes ao longo do processo de treinamento. Para cada métrica de desempenho será apresentado um gráfico distinto. Como o algoritmo NEAT é populacional, diversas redes são produzidas por geração. Para representar uma geração foi escolhida a rede que obteve o melhor desempenho naquela geração.

Como nestes experimentos existem diversas instâncias de validação, os gráficos apresentam o valor acumulado de todas as instâncias. Por exemplo, o gráfico que apresenta o total de recursos coletados mostra a soma dos recursos coletados em todas as instâncias.

A Figura 5.1 apresenta a evolução da quantidade de colisões realizadas pelos robôs ao longo das gerações. A Figura 5.2 apresenta a evolução da quantidade de posições inéditas exploradas pelos robôs pela quantidade de gerações. A Figura 5.3 apresenta a evolução da quantidade de recursos carregados pelos robôs pela quantidade de gerações. Os recursos carregados foram encontrados pelos robôs mas não foram necessariamente levados ao ninho. A Figura 5.4 apresenta a evolução da quantidade de recursos efetivamente coletados pelos robôs pela quantidade de gerações. Esses recursos foram encontrados e levados até o ninho. Essa é a principal métrica de avaliação dos regimes de treinamento.

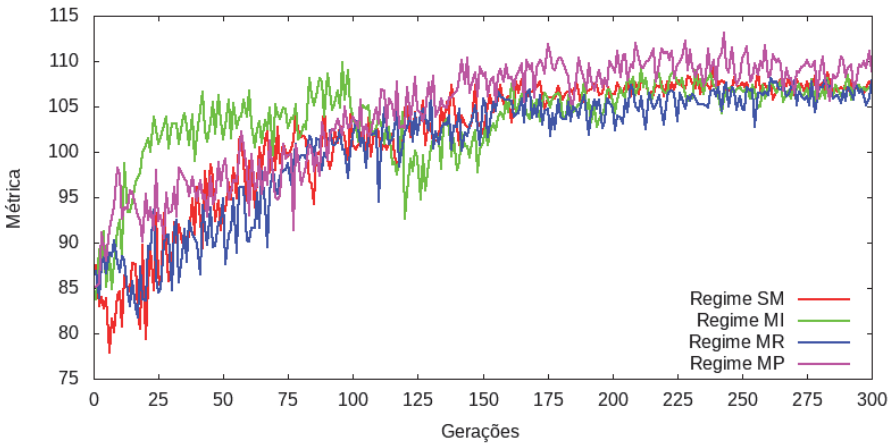


Figura 5.1: Quantidade de colisões por geração (Múltiplas instâncias)

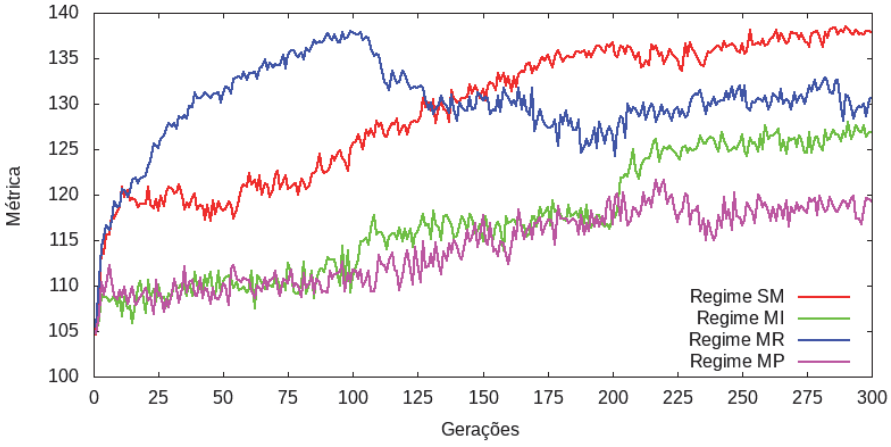


Figura 5.2: Quantidade de posições exploradas por geração (Múltiplas instâncias)

Embora as métricas de exploração, carregamento e colisão sejam interessantes para analisar a evolução do comportamento dos robôs ao longo do tempo, para definir se um regime de treinamento é superior ao outro, é utilizada a métrica de quantidade de recursos coletados.

Os gráficos apresentados a seguir são utilizados para demostrar se existe diferença estatística entre os regimes de treinamento. O teste estatístico t de Student é utilizado para verificar se existe diferença estatística. Esse teste é realizado em cada iteração para que seja possível avaliar em qual parte do treinamento existe diferença estatística e qual regime de treinamento

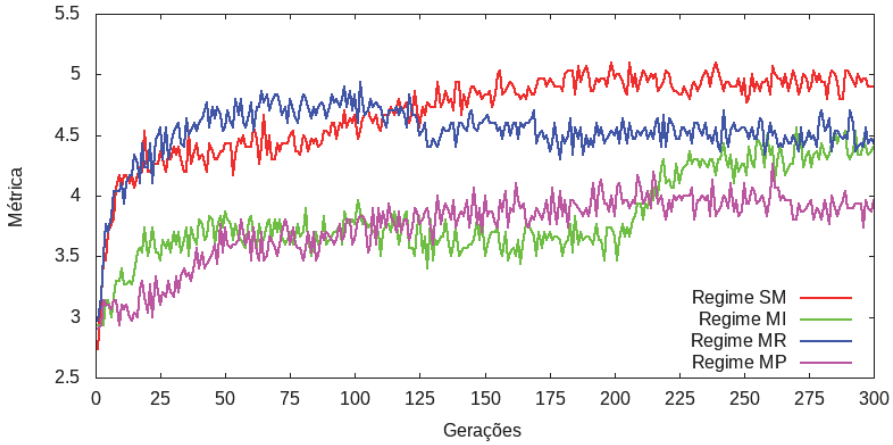


Figura 5.3: Quantidade de recursos carregados por geração (Múltiplas instâncias)

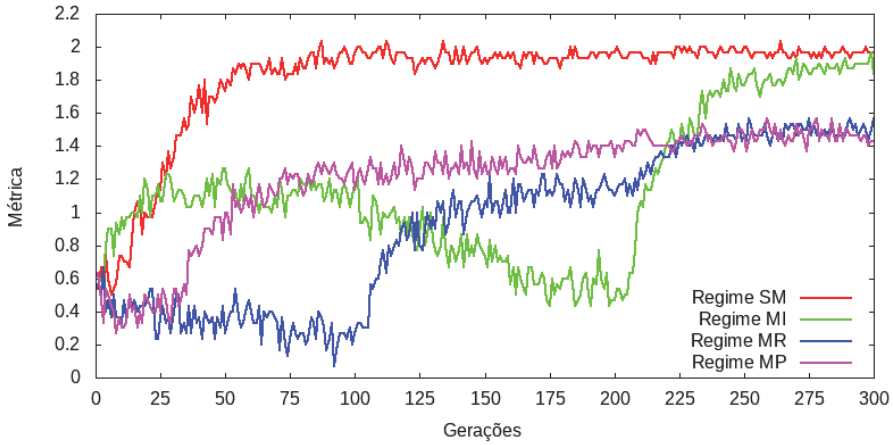


Figura 5.4: Quantidade de recursos coletados por geração (Múltiplas instâncias)

é superior. Como o objetivo desse estudo é avaliar as técnicas de moldagem, os regimes que utilizam moldagem são comparados contra o regime SM.

A Figura 5.5 apresenta a comparação do regime SM com o regime MI. A Figura 5.6 apresenta a comparação do regime SM com o regime MR. A Figura 5.7 apresenta a comparação do regime SM com o regime MP. O desempenho dos regimes é apresentado por curvas. A região sombreada indica que existe diferença estatística entre os regimes nesse intervalo.

5.4.2 Análise

O gráfico que apresenta a quantidade de colisões (Figura 5.1) mostra que conforme a quantidade de gerações aumenta a quantidade de colisões também aumenta. Todos os regimes apresentam a mesma tendência e a diferença entre a quantidade de colisões entre os diferentes regimes é pequena. As curvas neste gráfico apresentam oscilação entre gerações consecutivas. Isso indica que a quantidade de colisões ainda não estabilizou.

O gráfico que apresenta a quantidade de posições exploradas (Figura 5.2) mostra que nas últimas iterações o regime SM explora mais posições. Porém no primeiro terço das gerações, o regime MR explora mais posições que os outros regimes. Isso se deve à função de aptidão utilizada no primeiro terço desse regime, cujo peso para exploração é proporcionalmente maior que nas outras funções de aptidão, levando os robôs nessa etapa do treinamento a explorar mais. Quando a função de aptidão muda, ao entrar no segundo terço no treinamento, a quantidade de

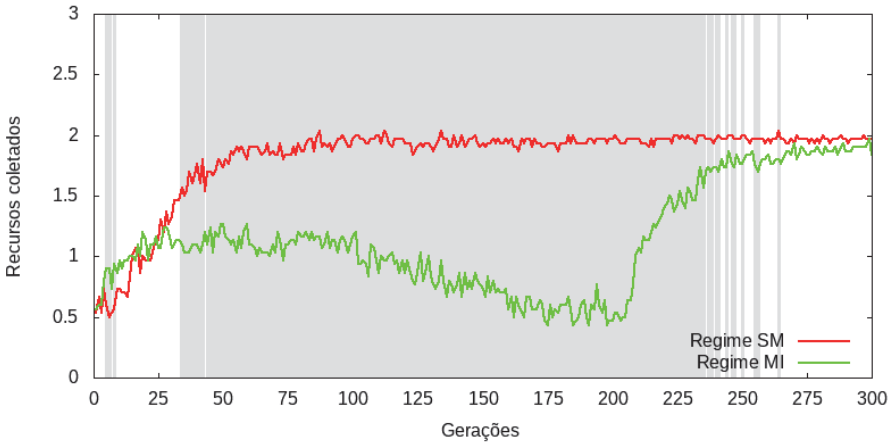


Figura 5.5: Comparação de regimes SM e MI (Múltiplas instâncias) (Regime Instância)

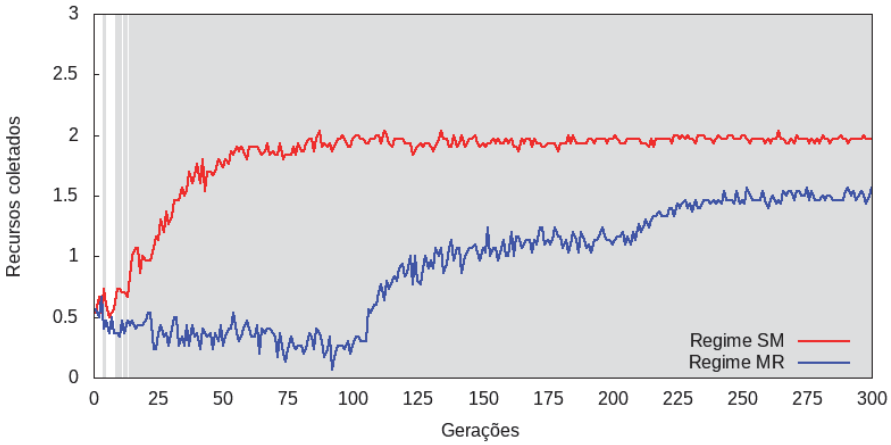


Figura 5.6: Comparação de regimes SM e MR (Múltiplas instâncias) (Regime Recompensa)

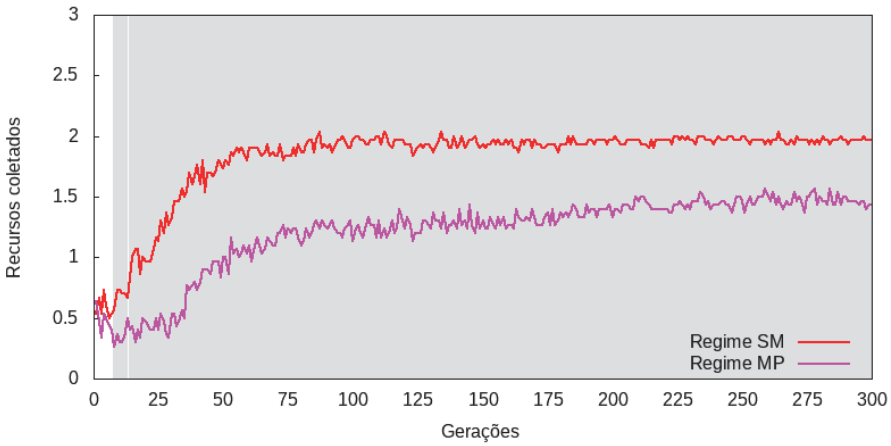


Figura 5.7: Comparação de regimes SM e MP (Múltiplas instâncias) (Regime Professor)

posições exploradas começa a cair e se mantém abaixo do regime SM. Uma mudança brusca no comportamento da curva também pode ser observada no regime MI ao passar do segundo para o terceiro terço do treinamento. O regime MP apresenta uma curva sem mudanças bruscas mas com desempenho inferior aos outros regimes.

O gráfico que apresenta a quantidade de recursos carregados (Figura 5.3) mostra que o regime SM apresenta o melhor desempenho. Porém, assim como no gráfico de posições exploradas, o regime MR obtém melhor desempenho, isto é, carrega mais recursos, no primeiro terço do treinamento, ao mudar para o segundo terço o desempenho do regime MR começa a cair novamente. Uma mudança de curva brusca no regime MI pode ser observada ao se iniciar o último terço do treinamento. O regime MP apresenta uma curva sem mudanças bruscas mas com desempenho inferior aos outros regimes. Vale ressaltar que o gráfico mostra a quantidade coletada em todas as instâncias. Como são três instâncias no conjunto de validação, na média, o enxame de robôs carrega menos de dois recursos por instância.

O gráfico que apresenta a quantidade de recursos coletados (Figura 5.4) mostra que o regime SM apresenta melhor desempenho. O regime SM atinge estabilidade próximo da centésima geração. Embora o regime MR tenha carregado mais recursos no primeiro terço do treinamento, nesse mesmo terço os robôs não conseguem retornar os recursos carregados ao ninho. As mudanças de função de aptidão no inicio do segundo e terceiro terços do treinamento também causam mudanças bruscas na curva desse regime. Quando regime MI chega ao último terço do treinamento, ele apresenta o mesmo comportamento que o regime SM. Vale ressaltar que nesse estágio, a função de aptidão e o conjunto de treino do regime MI são os mesmos que do regime SM. O comportamento da curva no último terço desse regime é similar ao comportamento inicial da curva do regime SM. O regime MP apresenta uma curva sem mudanças bruscas mas com desempenho inferior aos outros regimes. Vale ressaltar que o gráfico mostra a quantidade coletada em todas as instâncias. Como são três instâncias no conjunto de validação, na média, o enxame de robôs carrega menos de um recurso por instância.

A comparação entre o regime SM e o regime MI (Figura 5.5) mostra que nas primeiras iterações, existe diferença estatística e o regime MI apresenta melhor desempenho, porém na maior parte do treinamento o regime SM se mostra superior até que no fim do treinamento não exista diferença estatística entre os dois métodos.

Tanto na comparação dos regimes MR como na comparação do regime MP (Figuras 5.6 e 5.7 respectivamente), o regime SM se mostra superior durante quase todo o treinamento, inclusive no final.

5.5 EXPERIMENTOS - INSTÂNCIA ÚNICA

O segundo conjunto de experimentos finais utiliza um conjunto de instâncias para treinamento e validação dos controladores, cada um com apenas um único elemento. As instâncias utilizadas estão descritas na tabela 5.7. A descrição completa das instâncias pode ser encontrada no apêndice B.

Tabela 5.7: Configuração dos experimentos (Parâmetros específicos) (Instância única)

Parâmetro	**Valor**
Conjunto de validação	[valid-00]
Conjunto de treino (SM)	[hard-00]
Conjunto de treino (MR)	[hard-00]
Conjunto de treino (MP)	[hard-00]
Conjunto de treino 1 (MI)	[easy-00]
Conjunto de treino 2 (MI)	[medium-00]
Conjunto de treino 3 (MI)	[hard-00]

O objetivo e método utilizado durante estes experimentos é o mesmo dos experimentos realizados na seção anterior. A diferença entre os dois conjuntos de experimentos está nos valores atribuídos aos parâmetros.

Os parâmetros deste conjunto de experimentos são os parâmetros gerais descritos na tabela 5.1 e os parâmetros específicos descritos na tabela 5.7.

5.5.1 Resultados

Para demonstrar os resultados da simulação serão apresentados gráficos que comparam os regimes de treinamento.

Os gráficos apresentam o desempenho dos regimes ao longo do processo de treinamento. Para cada métrica de desempenho será apresentado um gráfico distinto. Como o algoritmo NEAT é populacional, diversas redes são produzidas por geração. Para representar uma geração foi escolhida a rede que obteve o melhor desempenho naquela geração.

A Figura 5.8 apresenta a evolução da quantidade de colisões realizadas pelos robôs ao longo das gerações. A Figura 5.9 apresenta a evolução da quantidade de posições inéditas exploradas pelos robôs pela quantidade de gerações. A Figura 5.10 apresenta a evolução da quantidade de recursos carregados pelos robôs pela quantidade de gerações. Os recursos carregados foram encontrados pelos robôs mas não foram necessariamente levados ao ninho. A Figura 5.11 apresenta a evolução da quantidade de recursos efetivamente coletados pelos robôs pela quantidade de gerações. Esses recursos foram encontrados e levados até o ninho. Essa é a principal métrica de avaliação dos regimes de treinamento.

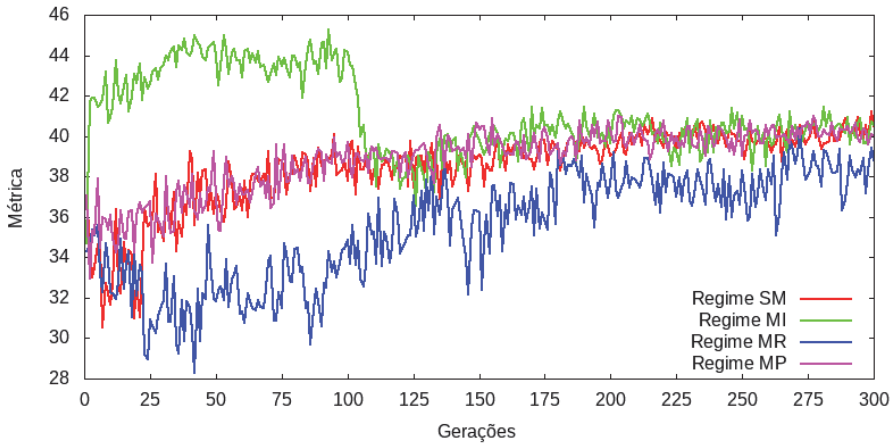


Figura 5.8: Quantidade de colisões por geração (Instância única)

O teste estatístico t de Student é aplicado da mesma forma que na seção anterior sobre a mesma métrica (quantidade de recursos coletados). A Figura 5.12 apresenta a comparação do regime SM com o regime MI. A Figura 5.13 apresenta a comparação do regime SM com o regime MR. A Figura 5.14 apresenta a comparação do regime SM com o regime MP. O desempenho dos regimes é apresentado por curvas. A região sombreada indica que existe diferença estatística entre os regimes nesse intervalo.

5.5.2 Análise

O gráfico que apresenta a quantidade de colisões (Figura 5.8) mostra que a tendência dos regimes é aumentar levemente a quantidade de colisões. O regime MI apresenta uma queda brusca na quantidade de colisões a partir do segundo terço do treinamento. Isso de deve

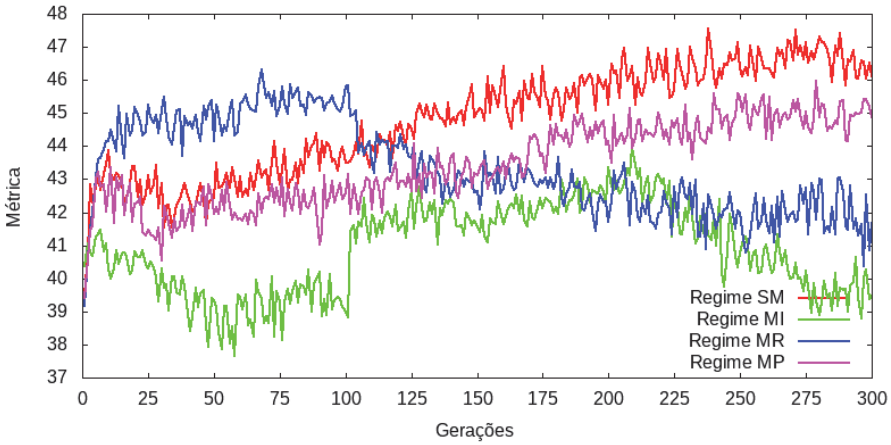


Figura 5.9: Quantidade de posições exploradas por geração (Instância única)

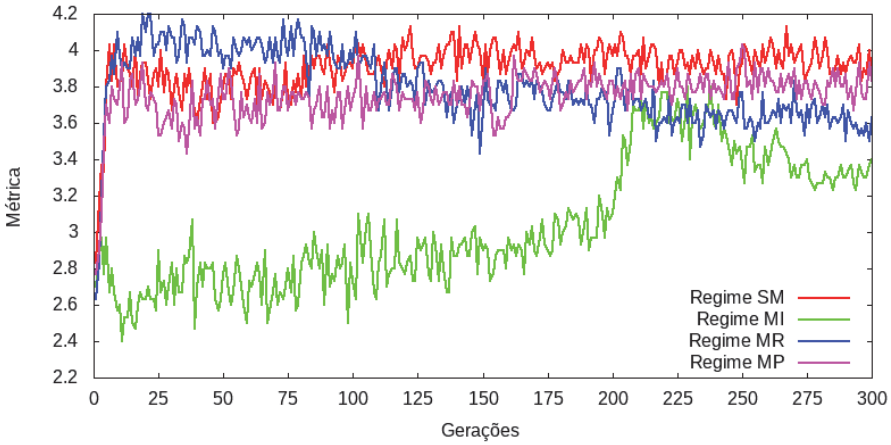


Figura 5.10: Quantidade de recursos carregados por geração (Instância única)

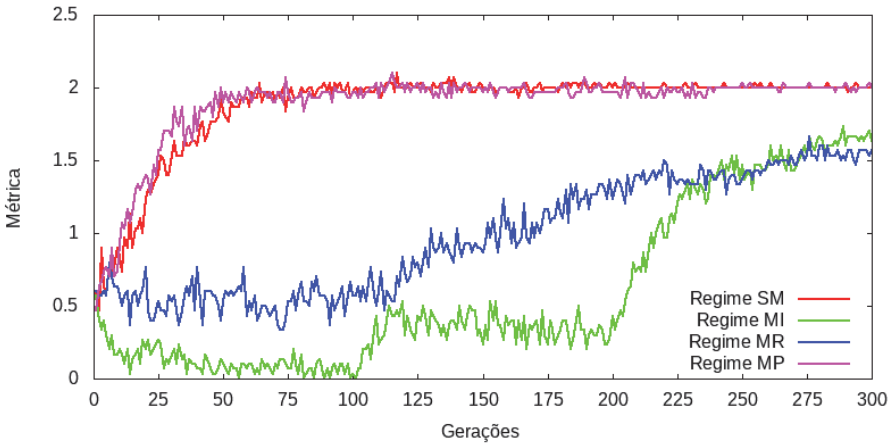


Figura 5.11: Quantidade de recursos coletados por geração (Instância única)

ao fato de a instância utilizada no primeiro terço do treinamento MI não possuir obstáculos fixos. A quantidade de colisões ocorridas durante o treinamento não eram significativas o suficiente para desenvolver o comportamento de evitar colisões durante a validação. O regime MR apresenta a menor quantidade de colisões, contudo apresenta uma grande oscilação entre

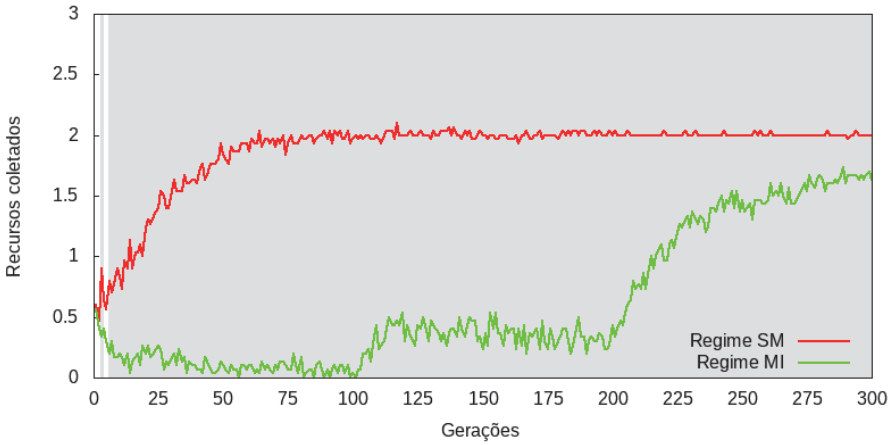


Figura 5.12: Comparação de regimes SM e MI (Única instância) (Regime Instância)

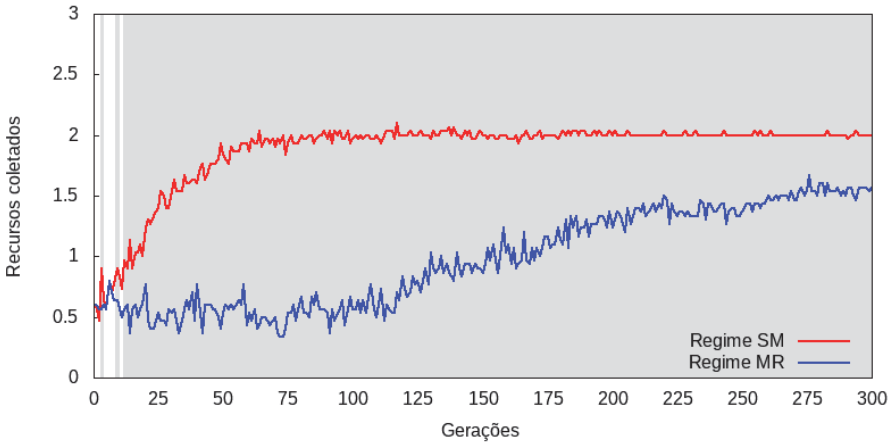


Figura 5.13: Comparação de regimes SM e MR (Única instância) (Regime Recompensa)

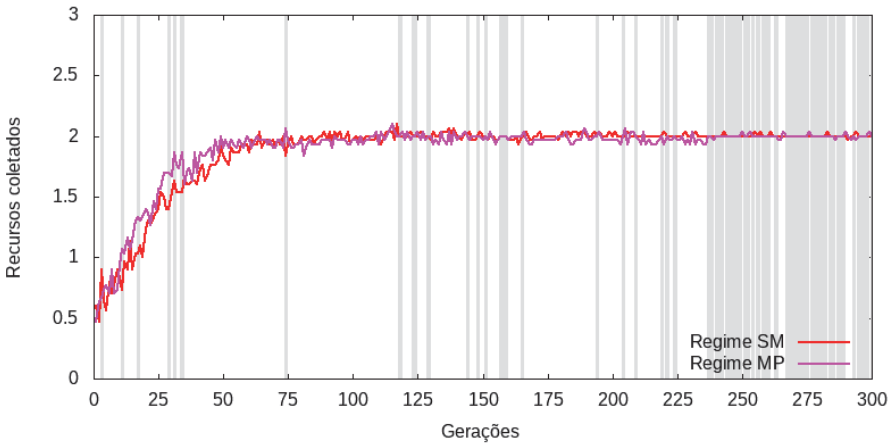


Figura 5.14: Comparação de regimes SM e MP (Única instância) (Regime Professor)

gerações consecutivas, indicando que essa métrica não estabilizou. Para os regimes MR e MP, conforme a quantidade de gerações aumenta o tamanho das oscilações diminui.

O gráfico que apresenta a quantidade de posições exploradas (Figura 5.9) mostra que nas últimas iterações o regime SM explora mais posições. O regime SM e MP apresentam uma tendência crescente durante todo o treinamento. O regime MR, no primeiro terço das gerações

apresenta o melhor desempenho, isto é, explora mais posições. Ao iniciar o segundo terço do treino o regime apresenta uma tendência decrescente. O regime MI apresenta o pior desempenho no primeiro terço das gerações. No início do segundo terço, apresenta uma subida brusca e apresenta uma tendência ascendente. Ao iniciar o terceiro terço, a tendência da curva passa a ser decrescente.

O gráfico que apresenta a quantidade de recursos carregados (Figura 5.10) mostra que o regime SM apresenta o melhor desempenho. O regime SM e o regime MP apresentam uma tendência constante ao longo do treinamento. O regime MR apresenta uma leve queda no terceiro terço do treinamento. O regime MI possui desempenho inferior aos outros regime até o segundo terço do treinamento. No início do último terço esse regime apresenta uma mudança brusca na curva e depois uma leve descida.

O gráfico que apresenta a quantidade de recursos coletados (Figura 5.11) mostra que os regimes SM e MP apresentam um comportamento similar. Os regimes MR e MI apresentam desempenho inferior e mudanças no comportamento da curva no início do segundo e terceiro terços do treinamento. O regime MR apresenta mudanças de comportamento mais suaves enquanto o regime MI apresenta mudanças bruscas.

Tanto na comparação do regime MR como na comparação do regime MI (Figuras 5.13 e 5.12 respectivamente), o regime SM se mostra superior durante quase todo o treinamento, inclusive no final. Durante a maior parte do treinamento existe diferença estatística.

A comparação entre o regime SM e o regime MP (Figura 5.14) mostra que a existência de diferença estatística é intermitente entre gerações consecutivas ou existe apenas em pequenos intervalos. Devido a inexistência de um grande intervalo onde existe diferença estatística, pode-se afirmar que esses regimes são equivalentes.

5.6 EXPERIMENTOS - MESMO TREINO E VALIDAÇÃO

O último conjunto de experimentos finais utiliza um conjunto de instâncias para treinamento e validação dos controladores, cada um com apenas um único elemento. Nestes experimentos, as instâncias utilizadas na validação também são utilizadas durante o treino. As instâncias utilizadas estão descritas na tabela 5.8. A descrição completa das instâncias pode ser encontrada no apêndice B.

Tabela 5.8: Configuração dos experimentos (Parâmetros específicos) (Mesma instância)

Parâmetro	**Valor**
Conjunto de validação	[valid-00]
Conjunto de treino (SM)	[valid-00]
Conjunto de treino (MR)	[valid-00]
Conjunto de treino (MP)	[valid-00]
Conjunto de treino 1 (MI)	[easy-00]
Conjunto de treino 2 (MI)	[medium-00]
Conjunto de treino 3 (MI)	[valid-00]

O objetivo e método utilizado durante estes experimentos é o mesmo dos experimentos realizados na seção anterior. A diferença entre os dois conjuntos de experimentos está nos valores atribuídos aos parâmetros.

Os parâmetros deste conjunto de experimentos são os parâmetros gerais descritos na tabela 5.1 e os parâmetros específicos descritos na tabela 5.8.

5.6.1 Resultados

Para demonstrar os resultados da simulação serão apresentados gráficos que comparam os regimes de treinamento.

Os gráficos apresentam o desempenho dos regimes ao longo do processo de treinamento. Para cada métrica de desempenho será apresentado um gráfico distinto. Como o algoritmo NEAT é populacional, diversas redes são produzidas por geração. Para representar uma geração foi escolhida a rede que obteve o melhor desempenho naquela geração.

A Figura 5.15 apresenta a evolução da quantidade de colisões realizadas pelos robôs ao longo das gerações. A Figura 5.16 apresenta a evolução da quantidade de posições inéditas exploradas pelos robôs pela quantidade de gerações. A Figura 5.17 apresenta a evolução da quantidade de recursos carregados pelos robôs pela quantidade de gerações. Os recursos carregados foram encontrados pelos robôs mas não foram necessariamente levados ao ninho. A Figura 5.18 apresenta a evolução da quantidade de recursos efetivamente coletados pelos robôs pela quantidade de gerações. Esses recursos foram encontrados e levados até o ninho. Essa é a principal métrica de avaliação dos regimes de treinamento.

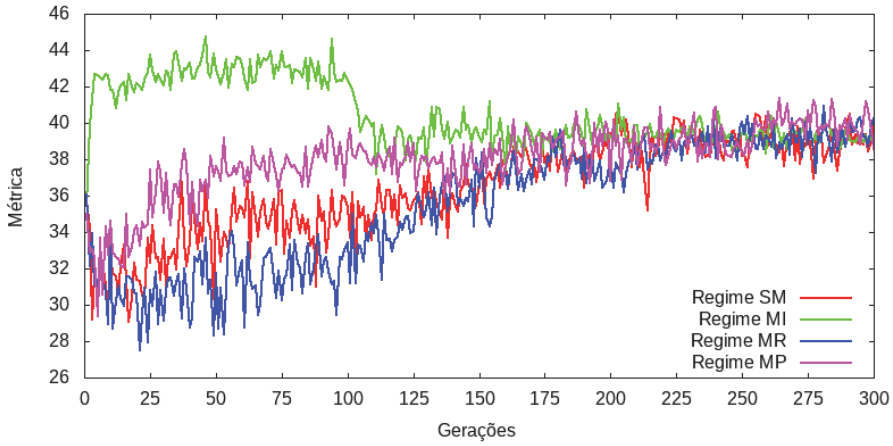


Figura 5.15: Quantidade de colisões por geração (Mesma instância)

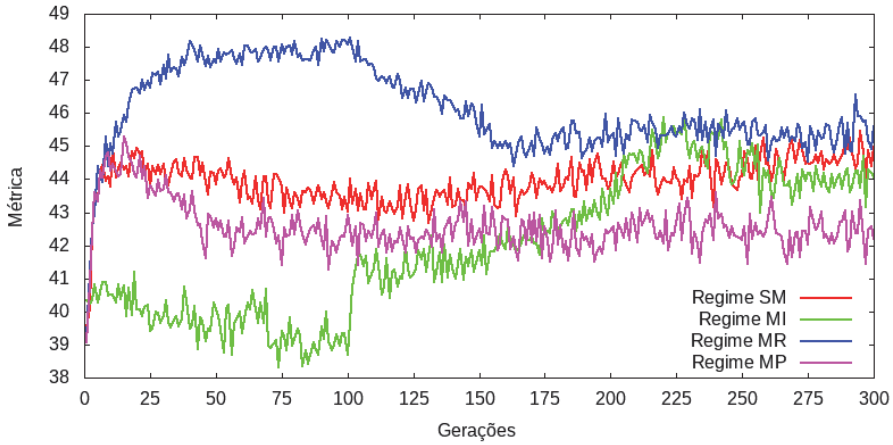


Figura 5.16: Quantidade de posições exploradas por geração (Mesma instância)

O teste estatístico t de Student é aplicado da mesma forma que na seção anterior sobre a mesma métrica (quantidade de recursos coletados). A Figura 5.19 apresenta a comparação do regime SM com o regime MI. A Figura 5.20 apresenta a comparação do regime SM com o regime

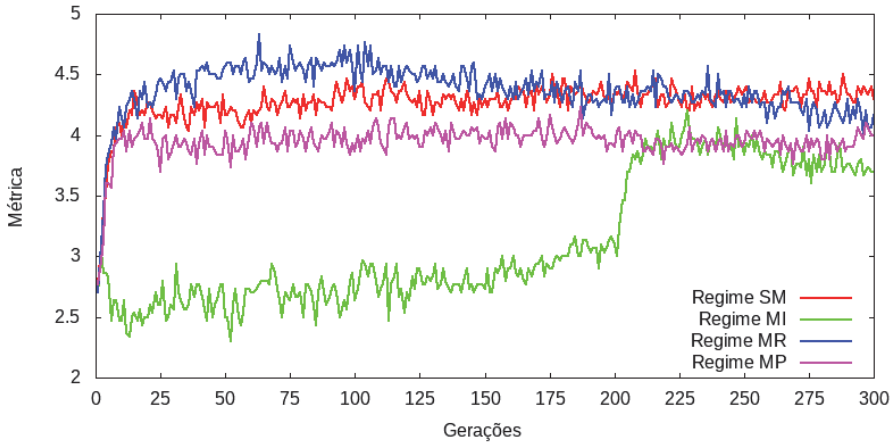


Figura 5.17: Quantidade de recursos carregados por geração (Mesma instância)

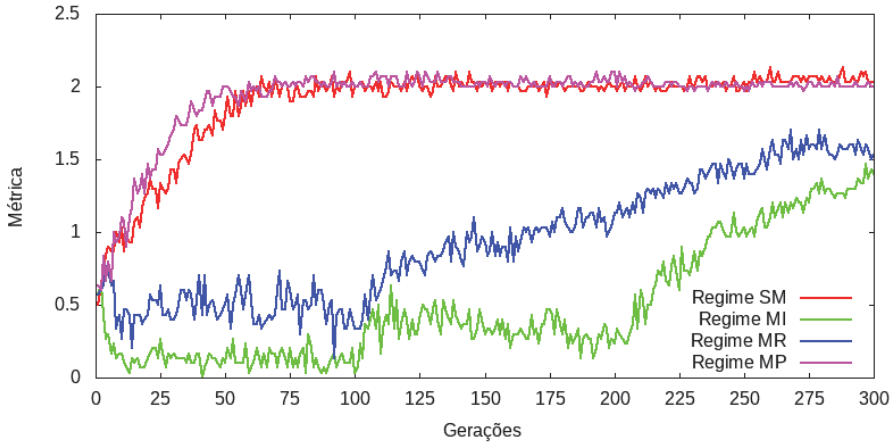


Figura 5.18: Quantidade de recursos coletados por geração (Mesma instância)

MR. A Figura 5.21 apresenta a comparação do regime SM com o regime MP. O desempenho dos regimes é apresentado por curvas. A região sombreada indica que existe diferença estatística entre os regimes nesse intervalo.

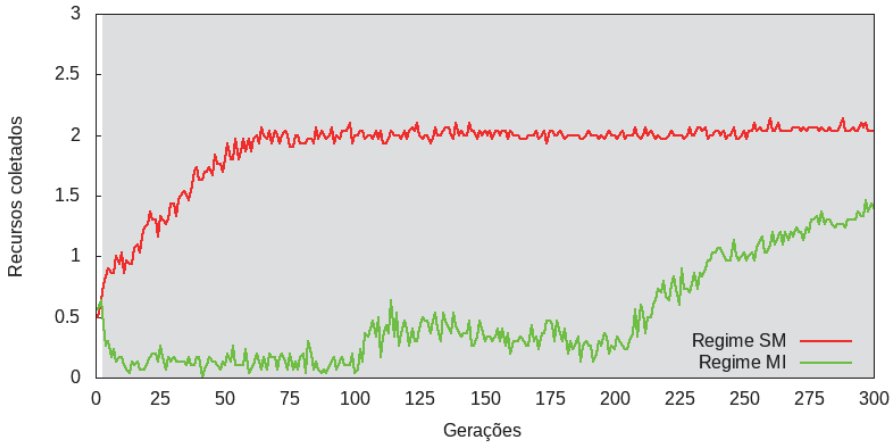


Figura 5.19: Comparação de regimes SM e MI (Mesma instância) (Regime Instância)

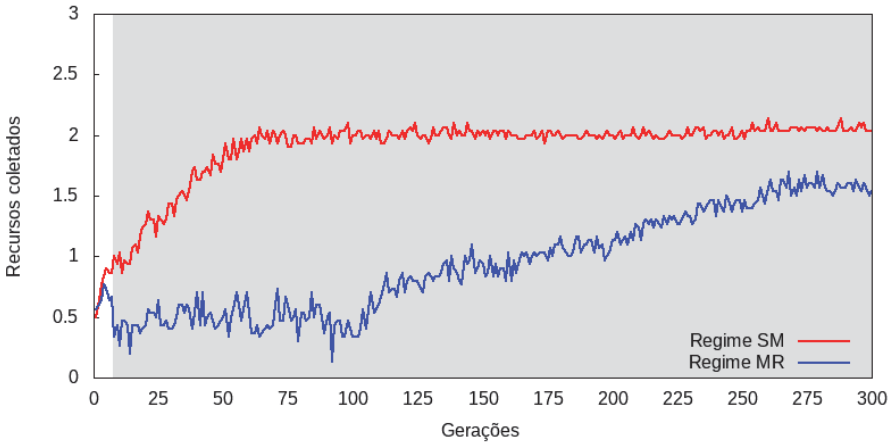


Figura 5.20: Comparação de regimes SM e MR (Mesma instância) (Regime Recompensa)

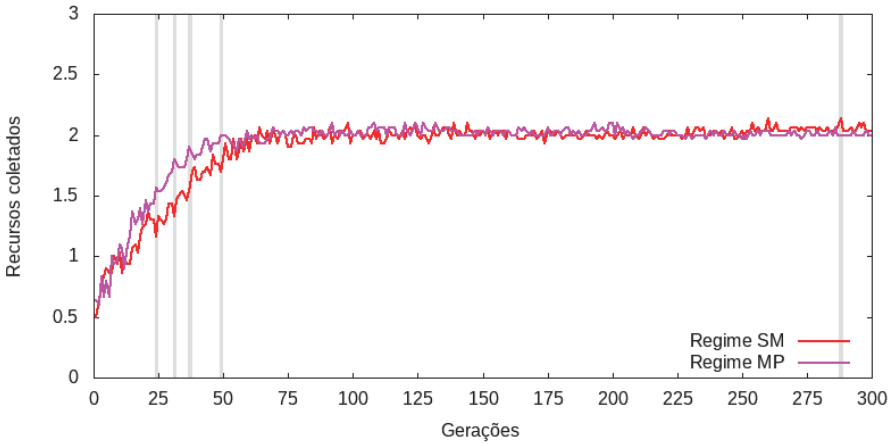


Figura 5.21: Comparação de regimes SM e MP (Mesma instância) (Regime Professor)

5.6.2 Análise

O gráfico que apresenta a quantidade de colisões (Figura 5.15) mostra que conforme a quantidade de gerações aumenta as curvas tendem a apresentar menor oscilação. Todas as curvas apresentam um comportamento similar, com exceção do primeiro terço do regime MI que apresenta uma mudança brusca no início do segundo terço do treinamento. A partir deste momento esse regime apresenta o mesmo comportamento dos outros regimes.

O gráfico que apresenta a quantidade de posições exploradas (Figura 5.16) mostra que o regime MR apresenta melhor desempenho no primeiro terço do treinamento. A partir do segundo terço a quantidade de posições exploradas sofre uma leve queda e apresenta uma tendência constante até o fim do treinamento. O regime MP apresenta uma tendência contante e o regime SM apresenta uma leva subida. O regime MI apresenta mudanças bruscas no comportamento ao iniciar o segundo e terceiro terço do treinamento. O comportamento de todas as curvas a partir do terceiro terço é similar, embora o resultado médio seja diferente.

O gráfico que apresenta a quantidade de recursos carregados (Figura 5.17) mostra que os regimes SM e MP apresentam um comportamento similar e uma tendência constante. O regime SM apresenta uma média maior que o regime MP. O regime MR apresenta o melhor desempenho entre todos os regimes no primeiro terço do treinamento, porém a partir do segundo terço, a curva apresenta uma leve queda e uma tendência contante no terceiro terço. O regime MI apresenta mudanças bruscas na curva no inicio do terceiro terço do treinamento. Todos os

regimes apresentam o mesmo comportamento no último terço do treinamento, todavia, médias distintas.

O gráfico que apresenta a quantidade de recursos coletados (Figura 5.18) mostra que os regimes SM e MP apresentam um comportamento similar. O regime MR apresenta uma tendência constante do primeiro terço do treinamento. A partir do segundo terço o comportamento da curva muda para uma tendência ascendente. O regime MI apresenta uma tendência ascendente apenas a partir do último terço do treinamento.

Tanto na comparação do regime MR como na comparação do regime MI (Figuras 5.20 e 5.19 respectivamente), o regime SM se mostra superior durante quase todo o treinamento, inclusive no final. Durante a maior parte do treinamento existe diferença estatística.

A comparação entre o regime SM e o regime MP (Figura 5.21) mostra que poucas gerações apresentam diferença estatística. Além disso, essa diferença existe apenas por pequenos intervalos. Devido a inexistência de um grande intervalo onde existe diferença estatística, pode-se afirmar que esses regimes são equivalentes.

5.7 ANÁLISE FINAL

Nos três conjuntos de experimentos, o regime de treinamento sem moldagem (SM) apresenta desempenho superior ou equivalente aos outros regimes. Embora para algumas métricas esse regime tenha um desempenho menor nas primeiras iterações, na maioria dos casos, ao final do treinamento esse regime possui o melhor desempenho.

O regime de treinamento com moldagem por professor (MP), assim como o regime SM, utiliza uma única configuração durante todo o treinamento. O comportamento da curva desses dois regimes é, na maioria dos casos, muito similar. Nos experimentos onde existem múltiplas instâncias no conjunto de treinamento e validação, existe diferença estatística entre o regime SM e MP e SM apresenta desempenho superior (figura 5.7). Quando o conjunto de treinamento e validação contém apenas uma instância, os dois regimes não apresentam diferença estatística, considerando a métrica de recursos coletados (figuras 5.14 e 5.21).

O regime de treinamento com moldagem por alteração de recompensa (MR) troca a função de aptidão ao iniciar o segundo e terceiro terço do treinamento. Vale ressaltar que durante o último terço do treinamento, os parâmetros utilizados são exatamente os mesmos dos regimes SM e MI.

Durante essas transições, independente do conjunto de experimentos, na maior parte dessas transições é possível observar uma mudança brusca no comportamento da curva das métricas observadas. Nos três conjuntos de experimentos esse regime obtém um desempenho superior nas métricas de exploração e carregamento no primeiro terço do treinamento. A partir do segundo terço, depois da primeira troca de parâmetros, as curvas dessas métricas passam a exibir o mesmo comportamento dos outros regimes.

Embora esse regime tenha desempenho superior nessas métricas no primeiro terço do treinamento, a métrica de coleta apresenta um resultado constante, passando a apresentar um comportamento crescente apenas a partir do segundo terço do treinamento e, durante a maior parte das gerações, apresenta diferença estatística em relação ao regime SM.

Uma possível explicação para o comportamento desse regime são os multiplicadores utilizados na função de aptidão utilizada no primeiro terço do treinamento (função f_1, tabela 5.3). Essa função atribui grande importância a exploração e ao carregamento. A recompensa por retornar e carregar um recurso são iguais, portanto, do ponto de vista do enxame, são virtualmente equivalentes. Quando a função de aptidão troca, o enxame passa a otimizar outras métricas de maior importância para a função de aptidão.

O regime de treinamento com moldagem por alteração de instância (MI) troca o conjunto de instâncias de treinamento ao iniciar o segundo e terceiro terço do treinamento. Vale ressaltar que durante o último terço do treinamento, os parâmetros utilizados são exatamente os mesmos dos regimes SM e MR.

Durante essas transições, independente do conjunto de experimentos, na maior parte dessas transições é possível observar uma mudança brusca no comportamento da curva das métricas observadas. Embora esse regime apresente mudanças bruscas ao iniciar o segundo terço do treinamento, ele apenas atinge comportamento e desempenho similar aos outros regimes durante o terceiro terço do treinamento.

Na maioria das métricas de avaliação, independente do conjunto de treinamento, esse regime apresenta o pior desempenho, porém no conjunto de experimentos com múltiplas instâncias, ao final do treinamento esse regime é estatisticamente equivalente ao regime SM.

Uma possível explicação para o comportamento desse regime está na composição do conjunto de treinamento. Durante as simulações, cada robô do enxame extrai percepções do ambiente. As percepções extraídas são utilizadas como entradas da rede neural, portanto, utilizadas para decidir qual ação será tomada. Se o conjunto de percepções utilizadas durante o treinamento for muito diferente do conjunto de percepções da validação, o enxame não terá conhecimento o suficiente para tomar boas decisões no ambiente de validação.

Nos experimentos com uma única instância, as instâncias de treinamento utilizadas nos primeiros dois terços da simulação são simplificadas. Dessa forma, oferecem percepções insuficientes para que os robôs treinados com aquelas percepções tenham um bom desempenho no ambiente de validação. Nos experimentos com múltiplas instâncias, os robôs tem acesso a mais instâncias, logo, existem mais dados e mais percepções. Com um conjunto mais rico de informações, eles estão melhor preparados para enfrentar o ambiente de validação.

Vale a pena ressaltar que mesmo que durante o último terço do treinamento os regimes SM, MR e MI utilizem os mesmos parâmetros, os regimes obtêm resultados diferentes na maioria das métricas e conjuntos de treinamento, demonstrando que o histórico de treinamento afeta o resultado final atingido. Mesmo quando a métrica de recursos coletados não apresenta diferença estatística, as outras métricas apresentam valores diferentes, indicando que os regimes geram robôs com comportamento levemente diferente, mesmo quando atingem o mesmo resultado.

Em relação a eficiência do enxame, isto é, a quantidade de recursos coletados, independente do conjunto de experimentos, a quantidade de recursos coletados converge para o número 2. Uma possível explicação para esse fenômeno é a concorrência entre robôs do enxame.

Nos experimentos realizados não foi utilizado nenhum mecanismo de comunicação entre os robôs, além disso os robôs não possuem a habilidade de passar o recurso para outros robôs. Dessa forma, para que um recurso seja coletado, cada robô do enxame deve realizar a tarefa completa.

Outros dois fatos importantes: o controlador utilizado é determinístico, isto é, para uma mesma percepção o robô sempre executa a mesma ação, e quando dois robôs colidem, eles retornam para a posição anterior. Caso a percepção desses robôs não mude, eles repetirão o mesmo movimento que na iteração anterior, pois o controlador é determinístico, fazendo com que os robôs fiquem presos em um laço infinito até que a percepção mude. Como os únicos elementos móveis no ambiente são robôs, a única forma da percepção se modificar é através da ação de um outro robô.

Como os robôs convergem para regiões de interesse, alimento mais próximo e novamente ao ninho, colisões entre robôs são extremamente prováveis. A simulação do ambiente como uma matriz também auxilia na criação dos laços infinitos, pois esta representação permite aos

robôs retornarem a posições exatamente iguais a iteração antes da colisão. Em uma simulação contínua, retornar a mesma posição é menos provável.

A hipótese desta análise é que a ineficiência das técnicas de moldagem no contexto específico deste trabalho se deve principalmente aos seguintes fatores: incapacidade de cooperar e incapacidade de observar propriamente outros robôs do enxame.

Sem a capacidade de se comunicar e coordenar, os robôs não conseguem dividir a tarefa e acabam competindo pelos mesmos recursos. Uma vez que os robôs chegam em um empasse, eles não conseguem se comunicar para sair do mesmo e ficam travados em um ciclo infinito. Além de não poderem se comunicar, os robôs não conseguem transferir recursos para outros robôs do enxame, uma capacidade que poderia resolver alguns empasses entre robôs. Os robôs também não conseguem identificar outros robôs como iguais. O pré-processamento aplicado a percepção do robô faz com que um robô perceba outros robôs apenas como obstáculos. Os robôs ficaram sem ferramentas para resolver seus empasses e a moldagem ficou sem opções para melhorar o desempenho dos robôs.

Embora a moldagem não tenha sido efetiva na configuração de experimentos utilizada neste trabalho acredita-se que concedendo mais opções aos robôs do enxame e uma visão mais clara do ambiente essa técnica possa ser mais eficiente.

6 CONCLUSÃO

Neste trabalho foi apresentado como técnicas de moldagem podem influenciar no desempenho do algoritmo de aprendizado por reforço NEAT, aplicado ao problema de coleta de recursos.

Algoritmos de aprendizado por reforço, como o NEAT, possuem alto custo computacional e não convergem rapidamente. O objetivo das técnicas de moldagem é modificar o regime de treinamento para obter uma convergência mais rápida.

Para avaliar a eficácia e eficiência das técnicas de moldagem, no contexto de coleta de recursos, foi realizada uma série de experimentos com o objetivo de encontrar evidências de que existe diferença estatística na utilização ou não de técnicas de moldagem em conjunto com o algoritmo NEAT. Para a realização desses experimentos foi utilizado um simulador de implementação própria.

Durante os experimentos foram avaliados três diferentes regimes de treinamento. Regime com moldagem por alteração de instância, que modifica as instâncias utilizadas durante o treinamento, regime com moldagem por alteração de recompensa, que modifica a função de recompensa recebida durante o treinamento e o regime com moldagem por professor, que adiciona a função de recompensa informação externa trazida por um professor.

Os resultados dos experimentos demonstraram que as técnicas de moldagem alteram o desempenho do treinamento, porém, o treinamento sem moldagem se mostrou superior ou estatisticamente equivalente ao treinamento assistido de moldagem.

Uma das maiores dificuldades encontradas durante a execução deste trabalho foi a grande quantidade de parâmetros. Primeiramente, o problema de coleta de recursos em si possui uma grande quantidade de variações e, para a realização deste trabalho, foi necessário selecionar uma dentre várias encontradas na literatura. O próprio algoritmo NEAT é um algoritmo que possui diversos parâmetros que precisaram ser ajustados. Por fim, as próprias técnicas de moldagem possuem parâmetros que também precisaram ser ajustados. Foi realizado um conjunto de experimentos preliminares para definir e ajustar esses parâmetros.

Os resultados dos experimentos finais demonstram que mesmo quando não existe diferença estatística entre os regimes pela métrica de recursos coletados, as outras métricas apresentam resultados diferentes, o que indica que os regimes não retornam redes idênticas, mas sim redes similares que cumprem o mesmo objetivo, mas com comportamento levemente diferente.

Além disso, mesmo que diversos regimes utilizassem os mesmos parâmetros, na etapa final do treinamento, eles não convergiam para o mesmo resultado, demonstrando que o histórico do treinamento impacta o resultado final.

Algumas possibilidades de trabalhos futuros são: uma análise com enfoque em outros parâmetros de grande impacto, como a quantidade de robôs; análises que considerem outros algoritmos de aprendizado por reforço; análise de moldagem aplicada a outros problemas; utilização de moldagem com outros propósitos, por exemplo, inserção de diversidade na população; aplicação de moldagem dinâmica, isto é, ao invés de aplicar uma mudança em um ponto determinado do treinamento, analisar as métricas durante o treinamento e aplicar uma alteração em função do resultado das métricas e quantidade de gerações executadas.

REFERÊNCIAS

Afshar, S. e Mahjoob, M. (2008). A simulation of ant formation and foraging using fuzzy logic and reinforcement learning. Em *2008 IEEE Conference on Cybernetics and Intelligent Systems*, páginas 806–811. IEEE.

Alpaydin, E. (2010). *Introduction to machine learning*. MIT press.

Dolan-Stern, N., Scrivnor, K. e Isaacs, J. (2018). Multimodal central place foraging. Em *2018 Second IEEE International Conference on Robotic Computing (IRC)*, páginas 72–78. IEEE.

Dorigo, M. e Colombetti, M. (1994). Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*, 71(2):321–370.

Dudek, G., Jenkin, M., Milios, E. e Wilkes, D. (1993). A taxonomy for swarm robots. Em *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, volume 1, páginas 441–447 vol.1.

Engelbrecht, A. P. (2007). *Computational Intelligence*. John Wiley & Sons Ltd.

Ericksen, J., Moses, M. e Forrest, S. (2017). Automatically evolving a general controller for robot swarms. Em *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, páginas 1–8. IEEE.

Hara, A., Konishi, H., Kushida, J.-i. e Takahama, T. (2016). Efficiency improvement of imitation operator in multi-agent control model based on cartesian genetic programming. Em *2016 IEEE 9th International Workshop on Computational Intelligence and Applications (IWCIA)*, páginas 69–74. IEEE.

Hecker, J. P., Carmichael, J. C. e Moses, M. E. (2015). Exploiting clusters for complete resource collection in biologically-inspired robot swarms. Em *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, páginas 434–440. IEEE.

Just, W. A. e Moses, M. E. (2017). Flexibility through autonomous decision-making in robot swarms. Em *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, páginas 1–8. IEEE.

Kadota, T., Yasuda, T., Matsumura, Y. e Ohkura, K. (2012). An incremental approach to an evolutionary robotic swarm. Em *2012 IEEE/SICE International Symposium on System Integration (SII)*, páginas 458–463. IEEE.

Kaelbling, L. P., Littman, M. L. e Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Lee, J.-H. e Ahn, C. W. (2011). Improving energy efficiency in cooperative foraging swarm robots using behavioral model. Em *2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, páginas 39–44. IEEE.

Lee, J.-H., Ahn, C. W. e An, J. (2013). A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis. Em *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, páginas 489–493. IEEE.

Lima, D. A. e Oliveira, G. M. (2016a). New bio-inspired coordination strategies for multi-agent systems applied to foraging tasks. Em *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, páginas 1–8. IEEE.

Lima, D. A. e Oliveira, G. M. (2016b). A probabilistic cellular automata ant memory model for a swarm of foraging robots. Em *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, páginas 1–6. IEEE.

Lima, D. A. e Oliveira, G. M. (2017). Formal analysis in a cellular automata ant model using swarm intelligence in robotics foraging task. Em *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, páginas 1793–1798. IEEE.

Lin, L. J. (1991). Programming robots using reinforcement learning and teaching. Em *AAAI*, páginas 781–786.

Mitchell, T. M. (1997). *Machine learning*. New York : McGraw-Hill.

Nogales, J. M. e de Oliveira, G. M. B. (2018). Adaptive give-up decisions for a team of robots foraging with task partitioning. Em *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, páginas 445–451. IEEE.

Ohkura, K., Yasuda, T. e Matsumura, Y. (2013). Coordinating the collective behavior of swarm robotics systems based on incremental evolution. Em *2013 IEEE International Conference on Systems, Man, and Cybernetics*, páginas 4024–4029. IEEE.

Russell, S. J. e Norvig, P. (2010). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Shell, D. A. e Mataric, M. J. (2006). On foraging strategies for large-scale multi-robot systems. Em *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 2717–2723. IEEE.

Soares, P. P., de Souza, L. B., Mendonça, M., Palácios, R. H. e de Almeida, J. P. L. S. (2018). Group of robots inspired by swarm robotics exploring unknown environments. Em *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, páginas 1–7. IEEE.

Stanley, K. O. e Miikkulainen, R. (2002). Efficient reinforcement learning through evolving neural network topologies. Em *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, páginas 569–577. Morgan Kaufmann Publishers Inc.

Sugawara, K., Sano, M., Yoshihara, I., Abe, K. e Watanabe, T. (1999). Foraging behaviour of multi-robot system and emergence of swarm intelligence. Em *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, volume 3, páginas 257–262. IEEE.

Wang, S., Wang, S. e Tan, M. (2002). Relation between task-based diversity and efficiency in multi-robot foraging. Em *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002.*, volume 3, páginas 1445–1450. IEEE.

Wei, Y., Yasuda, T. e Ohkura, K. (2017). Collective cognition: A case study of evolutionary swarm robotics in the collective foraging problem with poison. Em *2017 IEEE/SICE International Symposium on System Integration (SII)*, páginas 865–686. IEEE.

Winfield, A. F. (2009). Foraging robots. *Encyclopedia of complexity and systems science*, páginas 3682–3700.

Yang, M. e Liu, Z. (2012). Control of collision avoidance for swarm robots foraging in complex environment. Em *2012 Third International Conference on Intelligent Control and Information Processing*, páginas 525–530.

Yang, Y., Zhou, C. e Tian, Y. (2009). Swarm robots task allocation based on response threshold model. Em *2009 4th International Conference on Autonomous Robots and Agents*, páginas 171–176. IEEE.

Yasuda, T., Ohkura, K., Wada, N. e Matsumura, Y. (2013). Behavior sequence analysis of incrementally evolving robotic swarms in a foraging task. Em *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration*, páginas 790–795. IEEE.

Zedadra, O., Seridi, H., Jouandeau, N. e Fortino, G. (2015a). A distributed foraging algorithm based on artificial potential field. Em *2015 12th International Symposium on Programming and Systems (ISPS)*, páginas 1–6. IEEE.

Zedadra, O., Seridi, H., Jouandeau, N. e Fortino, G. (2015b). Energy expenditure in multi-agent foraging: An empirical analysis. Em *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, páginas 1773–1778. IEEE.

Zhou, G., Bastani, F., Zhu, W. e Yen, I.-L. (2016). A self-stabilizing algorithm for the foraging problem in swarm robotic systems. Em *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, páginas 2907–2912. IEEE.

APÊNDICE A – PARÂMETROS NEAT-PYTHON

Parametrização detalhada da biblioteca NEAT-Python utilizada durante os experimentos. Caso um parâmetro não esteja listado, foi utilizado o valor padrão.

A.1 ARQUIVO DE PARÂMETROS

```
[NEAT]
fitness_criterion     = max
fitness_threshold     = 50000
pop_size              = 100
reset_on_extinction   = True
no_fitness_termination = True

[DefaultGenome]
# node activation options
activation_default      = sigmoid
activation_mutate_rate  = 0.0
activation_options      = sigmoid

# node aggregation options
aggregation_default     = sum
aggregation_mutate_rate = 0.0
aggregation_options     = sum

# node bias options
bias_init_mean          = 0.0
bias_init_stdev         = 2.0
bias_init_type          = normal
bias_max_value          = 30.0
bias_min_value          = -30.0
bias_mutate_power       = 0.5
bias_mutate_rate        = 0.7
bias_replace_rate       = 0.1

# genome compatibility options
compatibility_disjoint_coefficient = 1.0
compatibility_weight_coefficient   = 0.5

# connection add/remove rates
conn_add_prob           = 0.5
conn_delete_prob        = 0.5

# connection enable options
enabled_default         = True
enabled_mutate_rate     = 0.01
```

```
feed_forward            = True
initial_connection      = full

# node add/remove rates
node_add_prob           = 0.2
node_delete_prob        = 0.2

# network parameters
num_hidden              = 0
num_inputs              = 9
num_outputs             = 5

# node response options
response_init_mean      = 1.0
response_init_stdev     = 0.0
response_max_value      = 30.0
response_min_value      = -30.0
response_mutate_power   = 0.0
response_mutate_rate    = 0.0
response_replace_rate   = 0.0

# connection weight options
weight_init_mean        = 0.0
weight_init_stdev       = 2.0
weight_max_value        = 30
weight_min_value        = -30
weight_mutate_power     = 0.5
weight_mutate_rate      = 0.8
weight_replace_rate     = 0.1

[DefaultSpeciesSet]
compatibility_threshold = 3.0

[DefaultStagnation]
species_fitness_func = max
max_stagnation       = 15
species_elitism      = 2

[DefaultReproduction]
elitism            = 2
survival_threshold = 0.2
```

APÊNDICE B – INSTÂNCIAS UTILIZADAS

Instâncias utilizadas na realização dos experimentos.

B.1 DESCRIÇÃO DA INSTÂNCIA

```
<Descrição da tarefa>
<Descrição do robô>
<Dimensões n, m da matriz mapa>
<Descrição do mapa contendo n linhas de tamanho m>
<Quantidade de robôs r>
<Lista Coordenadas i, j dos robôs de tamanho r>
```

B.2 LEGENDA DO MAPA

```
@ - Posição com recurso
# - Posição com obstáculo
. - Posição livre
* - Posição de ninho
```

B.3 INSTÂNCIA *EASY-00*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
....................
....................
....@@@@@@@@@@@@@...
....@.............@...
....@.............@...
....@.............@...
....@...*******...@...
....@...*******...@...
....@...*******...@...
....@...*******...@...
....@...*******...@...
....@.............@...
....@.............@...
....@.............@...
....@@@@@@@@@@@@@...
....................
....................
....................
5
```

```
8 8
8 10
8 11
9 11
11 11
```

B.4 INSTÂNCIA *EASY-01*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
....................
....................
....@@@@@@@@@@@@@...
....@...........@...
....@.@@@@@@@@@.@...
....@.@.......@.@...
....@.@.*****.@.@...
....@.@.*****.@.@...
....@.@.*****.@.@...
....@.@.*****.@.@...
....@.@.*****.@.@...
....@.@@@@@@@@@.@...
....@...........@...
....@@@@@@@@@@@@@...
....................
....................
....................
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.5 INSTÂNCIA *EASY-02*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
....................
....................
.....@.@.@.@.@.@....
```

```
....@..........@...
........@.@.@.@.....
....@.@.......@.@...
........*****.......
....@.@.*****.@.@...
........*****.......
....@.@.*****.@.@...
........*****.......
....@.@.@.@.@.@.@...
....................
....@.@.@.@.@.@.@...
....................
....................
....................
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.6 INSTÂNCIA *MEDIUM-00*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
.....@@@@@@@@@@....
....................
........#####.......
....................
....................
..@@@@@@.....@@@@@@.
....#...******...#...
....#...******...#...
..@.#...******...#.@.
....#...******...#...
....#...******...#...
..@@@@@@.....@@@@@@.
....................
....................
........#####.......
....................
.....@@@@@@@@@@....
....................
5
```

```
8 8
8 10
8 11
9 11
11 11
```

B.7 INSTÂNCIA *MEDIUM-01*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
.....@@@@@@@@@@....
....................
........#####.......
.....@@@@@@@@@@....
......###...###.....
..@@@@#@.....@#@@@@.
....#.#.*****.#.#...
....#...*******...#...
..@.#...*******...#.@.
....#...*******...#...
....#.#.*****.#.#...
..@@@@#@.....@#@@@@.
......###...###.....
.....@@@@@@@@@@....
........#####.......
....................
.....@@@@@@@@@@....
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.8 INSTÂNCIA *MEDIUM-02*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
....................
....................
....................
```

```
.....@@@@...@@@@....
.....@###...###@....
.....@#.......#@....
.....@#.*****.#@....
........*****.......
........*****.......
........*****.......
.....@#.*****.#@....
.....@#.......#@....
.....@###...###@....
.....@@@@...@@@@....
....................
....................
....................
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.9 INSTÂNCIA *HARD-00*

```
1 150 1
1 15 150 1 15
20 20
....................
.@....@#...@...@....
.#....#..@.#.@..#...
...@#...#.#...#.....
...#..........@.....
...@.....#@#....@...
..........###.......
...##.........@...@.
...@#...*****.......
...##...******..#@...
.@#.....*****.......
........*****.......
........******..@#...
...@..............#@
...@....@...##......
..........@@@@......
....#.@...@....#@@..
...............#....
.........@..@..@...@
.....@..............
5
```

```
8 8
8 10
8 11
9 11
11 11
```

B.10 INSTÂNCIA *HARD-01*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
....................
....................
....................
......#@@@@@@#......
......########......
....................
........*****.......
........*****.......
........*****.......
........*****.......
........*****.......
....................
......########......
......#@@@@@@#......
....................
....................
....................
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.11 INSTÂNCIA *HARD-02*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
...@.........@......
...#.........#...#@.
...................#.
```

```
....................
....................
....................
........*****.......
........*****.......
........*****.......
...@#...*****.......
........*****.#@....
....................
....................
....@#..............
..........#.....#...
..........@.....@...
....................
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.12 INSTÂNCIA *VALID-00*

```
1 150 1
1 15 150 1 15
20 20
....................
....@...@...#@....@.
...#..@.#.@..#....#.
.....#...#.#...#@...
.....@..........#...
...@....#@#.....@...
........###.........
.@...@.........##...
.......******...#@...
...@#..*******...##...
.......*******.....#@.
.......*****.........
...#@..*****.........
@#..............@...
......##...@....@...
......@@@@..........
..@@#....@...@.#....
....#...............
@...@..@..@.........
..............@.....
5
```

```
8 8
8 10
8 11
9 11
11 11
```

B.13 INSTÂNCIA *VALID-01*

```
1 150 1
1 15 150 1 15
20 20
....................
....................
.....@@@@@@@@@@....
....................
..@.....#####.....@.
..@...............@.
..@...............@.
..@...............@.
..@.#...*******...#.@.
..@.#...*******...#.@.
..@.#...*******...#.@.
..@.#...*******...#.@.
..@.#...*******...#.@.
..@...............@.
..@...............@.
..@...............@.
..@.....#####.....@.
....................
.....@@@@@@@@@@....
....................
5
8 8
8 10
8 11
9 11
11 11
```

B.14 INSTÂNCIA *VALID-02*

```
1 150 1
1 15 150 1 15
20 20
....................
............@.....
...........#.@.#...
........@....###....
......#.@.#.........
```

```
.......###......#...
...............#....
...............#@@..
.....#.******...#....
......#*******....#...
....@@#*****.........
......#*****.........
.....#.******....#...
...............#....
...............#@@..
...............#....
......###.......#...
.....#.@.#..........
.......@............
....................
5
8 8
8 10
8 11
9 11
11 11
```