UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARIA TASSIANE BARROS DE LIMA

PLANEJAMENTO NÃO DETERMINÍSTICO BASEADO EM REDES NEURAIS ASNET

QUIXADÁ

2019

MARIA TASSIANE BARROS DE LIMA

PLANEJAMENTO NÃO DETERMINÍSTICO BASEADO EM REDES NEURAIS ASNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Maria Viviane Menezes

Coorientador: Prof. Dr. Paulo de Tarso Guerra Oliveira

QUIXADÁ

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L699p Lima, Maria Tassiane Barros de.
Planejamento Não Determinístico baseado em Redes Neurais ASNet / Maria Tassiane Barros de Lima. – 2019.
46 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2019.
Orientação: Profa. Dra. Maria Viviane Menezes.
Coorientação: Prof. Dr. Paulo de Tarso Guerra Oliveira.

1. Inteligência artificial. 2. Planejamento automatizado. 3. Redes neurais (Computação). I. Título.
CDD 004

MARIA TASSIANE BARROS DE LIMA

PLANEJAMENTO NÃO DETERMINÍSTICO BASEADO EM REDES NEURAIS ASNET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: __/__/__

BANCA EXAMINADORA

Profa. Dra. Maria Viviane Menezes (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo de Tarso Guerra Oliveira (Coorientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

À minha família, pelo amor e apoio que sempre me deram.

AGRADECIMENTOS

Ao meu companheiro, Rayson Santos, pelo seu amor, pelo apoio diário, por me ajudar a conquistar meus objetivos e por estar comigo em todos os momentos difíceis na graduação e na vida. Por me entender, me motivar a continuar e estar sempre comigo. Ao meu filho, Ícaro Lucca, por todo amor, carinho e fofura, que tornam meus dias mais felizes.

Á minha mãe, Ivânia Barros, e ao meu pai, José Francisco, por terem me ajudado durante toda minha vida a estudar e ter uma boa formação. Por confiarem nas minhas decisões e me permitirem aprender com elas. Pelo carinho e esforços diários para cuidar da nossa família.

Aos meus irmãos, Tarciano e Tairton, e irmãs, Thais e Tamires, pela amizade e ajuda.

À toda minha família, em especial à minha prima, Patrícia Queiroz, por sua amizade, pelo carinho e ajuda em todas as circunstâncias. Por todas as conversar, risadas e momentos de alegria.

À professora, Maria Viviane de Menezes e ao professor Paulo de Tarso Guerra Oliveira, pela ótima orientação, por compartilharem seu conhecimento, por todos os concelhos e por me ajudarem em vários momentos. Vocês são pessoas admiráveis e sou grata por tudo.

Aos professores Regis Pires Magalhães e Ticiana Linhares Coelho da Silva pela disponibilidade em participar da banca deste trabalho e valorosas contribuições.

À toda minha turma, em especial aos meus amigos Lucas, Bárbara, Joyce e Michel pela amizade, estudos e momentos de descontração.

À todos que fizeram parte dessa minha jornada, em especial a família Dantas e agregados, pela amizade e apoio.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

(José de Alencar)

RESUMO

Planejamento Automatizado é a subárea da Inteligência Artificial que se preocupa com a escolha de ações de um agente inteligente para alcançar uma meta. Um problema de planejamento, em um dado domínio, é descrito por meio de uma situação inicial e de uma meta a ser alcançada. Uma solução para um problema de planejamento é uma sequência de ações que leva o agente do estado inicial a um estado satisfazendo a meta, denominada plano. No planejamento clássico supõe-se que o ambiente de planejamento evolui de forma determinística, ou seja, que não há incerteza sobre os efeitos das ações do agente. No entanto, há situações em que os efeitos das ações do agente são incertos, podemos ter duas situações: as ações com efeitos não determinísticos e ações com efeitos probabilísticos. Para estes tipos de domínios a solução é denominada política. O estado da arte em planejamento automatizado consiste em algoritmos: (i) baseados em busca heurística, satisfazibilidade booleana e grafos de planejamento para problemas com ações determinísticas; (ii) baseados em técnicas formais tais como verificação simbólica de modelos para problemas com ações não determinísticas e; algoritmos baseados em processos de decisão *markovianos* para problemas com ações probabilísticas. Recentemente, Toyer (2017) propôs a utilização de redes neurais artificias na construção de um planejador para problemas com ações probabilísticas. Para isto, definiu uma arquitetura de *rede neural*, denominada *Action Schema Network* (ASNet). Em seguida, Schäfer (2018) utilizou esta arquitetura e algoritmos propostos para obtenção de planos para problemas de *planejamento determinístico*. Este trabalho propõe a utilização das redes neurais ASNets para obter políticas para problemas de planejamento em domínios com ações não determinísticas.

Palavras-chave: Inteligencia artificial. Planejamento automatizado. Neural networks (Computação).

ABSTRACT

Automated Planning is the Artificial Intelligence subarea that is concerned with choosing an intelligent agent's actions to achieve a goal. A planning problem in a given domain is described through an initial situation and a goal to be achieved. One solution to a planning problem is a sequence of actions that takes the agent from the initial state to a state that meets the goal, called the plan. In classical planning it is assumed that the planning environment evolves deterministically, that is, there is no uncertainty about the effects of the agent's actions. However, there are situations where the effects of agent actions are uncertain, we may have two situations: actions with non-deterministic effects and actions with probabilistic effects. For these types of domains the solution is called policy. State of the art in automated planning consists of algorithms: (i) based on heuristic search, Boolean satisfiability, and planning graphs for deterministic action problems; (ii) based on formal techniques such as symbolic model verification for problems with non-deterministic actions and; decision-based *markovian* algorithms for problems with probabilistic actions. Recently, Toyer (2017) proposed the use of artificial neural networks to construct a planner for probabilistic action problems. To this end, he defined a neural network architecture called the Action Schema Network (ASNet). Subsequently, Schäfer (2018) used this architecture and proposed algorithms to obtain plans for deterministic planning problems. This paper proposes the use of ASNets neural networks to obtain policies for planning problems in domains with non-deterministic actions.

Keywords: Artificial intelligence. Automated planning. Neural networks (Computation).

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Planejamento Automatizado (*Automated Planning*) (GHALLAB *et al.*, 2004) é a subárea da Inteligência Artificial que se preocupa com a escolha de ações de um agente inteligente para alcançar uma meta. O agente deve raciocinar sobre um *domínio de planejamento*, que descreve as características do ambiente que o agente irá atuar. Um *problema de planejamento*, em um dado domínio, é descrito por meio de uma situação inicial e de uma meta a ser alcançada. Uma *solução* para um problema de planejamento é uma sequência de ações que leva o agente do estado inicial a um estado satisfazendo a meta, denominada *plano*. Um planejador (Figura 1) é um algoritmo que recebe como entrada a descrição de um problema de planejamento em um dado domínio e devolve, se possível, um plano solução (MENEZES, 2014).

Figura 1 – Esquema de um planejador

Fonte: (PEREIRA, 2007)

Considere um domínio de planejamento denominado *mundo dos blocos* (GHALLAB *et al.*, 2004). Esse domínio consiste em um conjunto de blocos dispostos sobre outros blocos ou sobre uma mesa. Um braço robótico raciocina para formar pilhas de blocos executando as ações *empilhar*, que coloca um bloco que está sobre a mesa em cima de outro bloco, ou *desempilhar*, que retira um bloco do topo de uma pilha e o coloca em cima da mesa. A garra pode pegar apenas um bloco por vez e não consegue pegar um bloco que tenha outro sobre ele. A meta é construir uma ou mais pilhas de blocos, especificada em termos de quais blocos estão no topo de quais blocos (NORVIG; RUSSELL, 2014).

Figura 2 – Estado inicial e um estado meta no domínio do mundo dos blocos

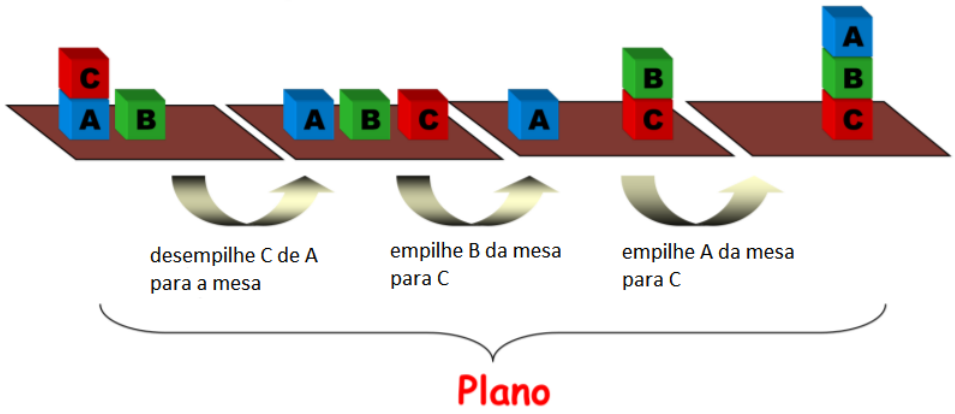
Estado Inicial Meta

Fonte: (PEREIRA, 2007)

A Figura 2 mostra um problema de planejamento no mundo dos blocos em que o estado inicial é que o bloco *A* e *B* estão sobre a mesa e o bloco *C* está sobre *A*. O estado meta é

também exibido no qual o bloco *A* está sobre *B* e *B* está sobre *C*, que, por sua vez, está sobre a mesa. Para o problema da Figura 2 um plano é a sequência de ações *desempilhar(C,A)*, *empilhar (B,C)*, *empilhar (A,B)*, conforme ilustrado na Figura 3.

Figura 3 – Um plano solução para um problema no mundo dos blocos

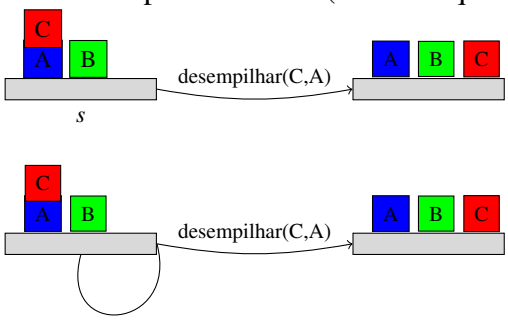


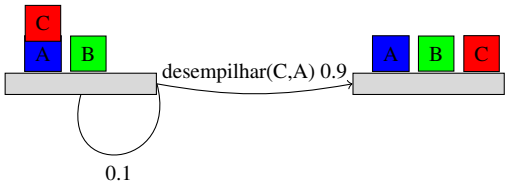
Fonte: (BARROS, 2003)

No planejamento clássico supõe-se que o ambiente de planejamento evolui de forma determinística, ou seja, que não há incerteza sobre os efeitos das ações do agente (PEREIRA, 2007). No entanto, há situações em que os efeitos das ações do agente são incertos, uma área denominada *planejamento sob incerteza* (PEREIRA, 2007; KOLOBOV, 2012; STEINMETZ *et al.*, 2016b; TREVIZAN *et al.*, 2018). Neste contexto, podemos ter duas situações: as ações com *efeitos não determinísticos* (*domínios qualitativo*) e *ações com efeitos probabilísticos* (*domínios quantitativos*). Para problemas de planejamento sob incerteza, a solução é denominada *política*, consistindo num mapeamento de estados em ações (PEREIRA, 2007).

Na Figura 4 (a) há uma representação de uma ação determinística *desempilhar(C,A)* que leva a um próximo estado em que o bloco *C* está sobre a mesa. Já na Figura 4 (b), há

Figura 4 – Representação da ação desempilhar o bloco C de cima do bloco A, em que: (a) a ação é determinística, (b) a ação é não determinística (domínio qualitativo) e (c) a ação é probabilística (domínio quantitativo)





Fonte: Adaptado de (PEREIRA, 2007)

uma ação não determinística que pode levar a dois possíveis estados sucessores, sendo que em uma possível situação futura o bloco *A* está sobre a mesa (modelando o caso em que a garra robótica funciona perfeitamente) e em outra situação nada acontece (considerando que a garra falhou). A ação pode também ter efeitos probabilísticos (Figura 4 (c)), em que há, por exemplo, a probabilidade de 90% de a garra robótica funcionar perfeitamente e uma probabilidade de 10% desta garra falhar.

Entretanto, encontrar um plano ou política é uma tarefa computacionalmente difícil e está na classe de complexidade PSPACE-COMPLETO (BYLANDER, 1994). Isso significa que o uso de representações mais compactas para os domínios podem melhorar a escalabilidade dos algoritmos de planejamento, mas não pode reduzir sua complexidade (PEREIRA, 2007).

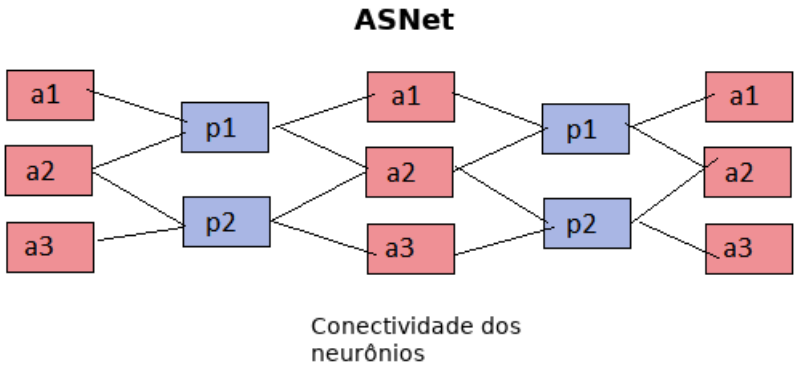
O estado da arte em planejamento automatizado consiste em algoritmos: (i) baseados em busca heurística (HOFFMANN, 2001; RICHTER; WESTPHAL, 2010), satisfazibilidade booleana (`SatPlan`) (KAUTZ; SELMAN, 1998) e grafos de planejamento (`GraphPlan`) (KAMBHAMPATI; NIGENDA, 2000) para problemas com ações determinísticas; (ii) baseados em técnicas formais tais como verificação simbólica de modelos (CIMATTI *et al.*, 2003; TORRALBA *et al.*, 2017) para problemas com ações não determinísticas e; algoritmos baseados em processos de decisão *markovianos* (MDPS) (KELLER; EYERICH, 2012) para problemas com ações probabilísticas.

Mais recentemente, Toyer (2017) propôs a utilização de redes neurais artificias na construção de um planejador para problemas com ações probabilísticas. Para isto, definiu uma arquitetura de *rede neural* (CHOLLET, 2018), denominada *Action Schema Network* (ASNet). Com esta arquitetura, foi proposto um algoritmo que recebe um conjunto de problemas para um certo domínio de *planejamento probabilístico* e as políticas para cada um destes problemas e; o algoritmo é capaz de aprender como obter uma política para um novo problema apresentado. Em seguida, Schäfer (2018) utilizou esta arquitetura e algoritmos propostos para obtenção de planos para problemas de *planejamento determinístico*.

A Figura 5 apresenta um modelo ilustrativo de uma ASNet, que é um grafo cujos nós são átomos proposicionais ou ações e cujas arestas definem como uma ação *a* afeta um átomo

proposicional *p* e como o átomo proposicional *p* influencia na saída da ação *a* (TOYER *et al.*, 2018).

Figura 5 – Arquitetura de uma ASNet em que nós são átomos proposicionais ou ações e as arestas representam as conexões entre os nós



Fonte: (TOYER *et al.*, 2018)

Os experimentos realizados por (TOYER *et al.*, 2018) e (SCHÄFER, 2018) utilizaram os domínios e problemas *benchmarks* da Competição Internacional de Planejamento (IPC - *International Planning Competition*) para avaliar a utilização das ASNets. Nos experimentos, problemas considerados pequenos (e de solução fácil para os planejadores estado da arte) foram utilizados para treinar a rede neural. Em seguida, ao apresentar problemas de tamanho grande (desafiadores até mesmo para os planejadores estado da arte), o algoritmo conseguiu obter uma solução para tais problemas em tempo comparável ao obtido pelos planejadores estado da arte.

Pelas pesquisas realizadas para elaboração deste trabalho, não foram encontradas a utilização de tais redes para obtenção de políticas para problemas de planejamento com *ações não determinísticas*.

1.1 Objetivos

1.1.1 Objetivo Geral

Propor a utilização das redes neurais ASNets para obter políticas para problemas de planejamento em domínios com ações não determinísticas (domínios qualitativos).

1.1.2 Objetivos Específicos

- Propor uma tradução de domínios com ações não-determinísticas para domínios com ações probabilísticas.
- Usar as redes neurais ASNET para solucionar problemas de planejamento não-determinísticos.
- Avaliar as soluções apresentadas pelo algoritmo em um domínio não determinística da Competição Internacional de Planejamento.

1.2 Organização

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos sobre planejamento automatizado e sobre as redes neurais ASNets; o Capítulo 3 apresenta os trabalhos relacionados que ilustram como as redes neurais são utilizadas na área de planejamento automatizado; o Capítulo 4 expõe os resultados e; por fim o Capítulo 5 apresenta as conclusões a respeito deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção apresentamos a fundamentação teórica, necessário ao entendimento deste trabalho. Na Seção 2.1 são introduzidos os conceitos referentes à área de Planejamento Automatizado. Na Seção 2.2 são abordadas aplicações de Aprendizado de Máquina em Planejamento Automatizado. Na Seção 2.3 são apresentados os conceitos de *Deep Learning*. Finalmente, na Seção 2.4 serão apresentados os conceitos de *Action Schema Networks*.

2.1 Planejamento Automatizado

Planejamento é o raciocínio sobre ações para que um agente alcance suas metas. Nesta seção, abordaremos os conceitos fundamentais das áreas planejamento com ações determinísticas, não-determinísticas e probabilísticas.

Um domínio de planejamento clássico ou determinístico pode ser representado formalmente por um sistema de transição de estados em que estados são rotulados por um conjunto de átomos proposicionais $\mathbb{P}$ e as transições são rotuladas por um conjunto de ações $\mathbb{A}$. Na Definição 1, temos a definição formal de um domínio de planejamento clássico. Observe que nesta definição, as ações tem efeitos determinísticos, isto é, uma ação quando aplicada em um estado leva a apenas um estado sucessor.

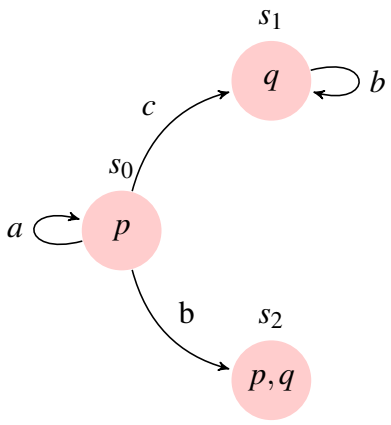
Definição 1 *(Domínio de Planejamento Clássico - Representação por meio de Sistema de Transição de Estados) Dado um conjunto de átomos proposicionais* $\mathbb{P}$ *e um conjunto de ações* $\mathbb{A}$*, um domínio de planejamento determinístico com assinatura* $(\mathbb{P}, \mathbb{A})$ *é definido por uma tupla* $D = \langle S, \mathbb{A}, T \rangle$*, em que (GHALLAB et al., 2004):*

- $S \neq \emptyset$ *é um conjunto finito de estados;*
- $L : S \mapsto 2^{\mathbb{P}}$ *é uma função de rotulação de estados, utilizando a representação utiliza-se a suposição do mundo fechado (closed world assumption)*[1]*;*
- $T : S \times \mathbb{A} \mapsto S$ *é uma função de transição de estados.*

A Figura 6 ilustra um domínio de planejamento clássico representado por meio de um sistema de transição de estados sobre um conjunto de proposições $\mathbb{P} = \{p, q\}$ e em que um conjunto de ações é $\mathbb{A} = \{a, b, c\}$. Os estados no conjunto $S = \{s_0, s_1, s_2\}$ são rotulados por elementos de $\mathbb{P}$, e as transições são rotuladas por elementos de $\mathbb{A}$. Aqui, chamamos atenção

[1] Usando a suposição do mundo fechado, representa-se no rótulo do estado apenas o que é verdadeiro, o que não é representado é considerado falso.

Figura 6 – Sistema de Transição de estados de um domínio de planejamento clássico sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$ e conjunto de ações $\mathbb{A} = \{a, b, c\}$



Fonte: Elaborado pela autora.

para dois fatos: (i) o estado s_0 cujo rótulo $L(s_0) = \{p\}$ possui valores de p verdadeiro e q falso, uma vez que estamos usando a suposição do mundo fechado e; (ii) as ações a, b e c são determinísticas, pois quando aplicadas em um dado estado levam a apenas um estado sucessor.

A representação de domínios de planejamento por meio de sistemas de transição de estados não é escalável, permitindo que se possa representar uma quantidade limitada de estados e transições. Assim, em planejamento usa-se uma *linguagem de ações* por meio da qual é possível representar de forma compacta e implícita o sistema de transição de estados. Nesta linguagem, cada ação é enriquecida de significado deixando de ser apenas um rótulo da transição de estados e passando a representar as condições antes e depois da ação ser executada. Denominamos de *pré-condições* o conjunto de átomos proposicionais que devem ser verdadeiros para que uma ação seja executada em um dado estado, e os *efeitos*, informam como o estado foi modificado após a execução da ação.

Definição 2 *(Domínio de Planejamento Clássico - Representação por meio de Ações) Dado um conjunto de proposições $\mathbb{P}$, o domínio de planejamento clássico pode ser representado por meio de um conjunto de ações $\mathbb{A}$ com pré-condições e efeitos. Cada ação $a \in \mathbb{A}$ é especificada pelo par $\langle precond(a), efeitos(a) \rangle$, sendo (GHALLAB et al., 2004):*

- *$precond(a)$, um conjunto de átomos proposicionais que devem ser verdadeiras no estado em que a ação a é executada;*
- *$efeitos(a)$ é o conjunto do que se é modificado após a execução da ação.*

Exemplo 1 *(Representação de Ação com Pré-condições e Efeitos) A ação a da Figura 6 pode ser representada por meio de suas pré-condições e efeitos, como a seguir:*

- *precond(a)* = $\{p\}$ *e;*
- *efeitos(a)* = $\{\neg q\}$.

A ação *a* (Exemplo 3) é aplicada apenas em estados nos quais *p* tem valor verdadeiro. A execução da ação *a* em um dado estado modifica-o levando a estado sucessor em que *p* é verdade e *q* é falso (*efeitos(a)*). A representação de um domínio de planejamento por meio de ações com pré-condições e efeitos torna possível gerar o sistema de transição de estados correspondente.

Os pesquisadores da área de planejamento adotam a linguagem PDDL (*Planning Domain Description Language*) (MCDERMOTT *et al.*, 1998) como a linguagem padrão para especificação de domínios e problemas de planejamento. Nesta linguagem, o domínio contém predicados e ações representadas por pré-condições e efeitos. Em PDDL, são utilizados *esquemas de ações* que são uma representação com variáveis que posteriormente serão instanciadas por átomos proposicionais. Na literatura, quando uma ação está em sua versão com variáveis, denomina-se *lift action*, operador ou esquema de ações; e quando uma ação está com todas as variáveis instanciadas, denomina-se *ground action*.

A Figura 7 ilustra a representação do domínio do *mundo dos blocos* em PDDL. Na linha 02, temos a descrição do conjunto de predicados que representam as propriedades dos blocos e do ambiente, tais como: *limpo (?x)*, representa que o bloco *x* está limpo; *sobre (?x,?y)* representa que o bloco *x* está sobre o bloco *y*; *na-mesa (?x)* indica que o bloco *x* está sobre a mesa. A partir da linha 03, há a representação do conjunto de ações (*lift actions*): {*empilhar (?x,?y) e desempilhar (?x,?y)*} , sendo que *x* e *y* são variáveis que representam blocos.

Figura 7 – Esquema de ações do domínio do mundo dos blocos

```
01 (define (domain blocksworld)
02   (:predicates (limpo ?x) (sobre ?x ?y) (na-mesa ?x))
03   (:action empilhar
04     :parameters (?x ?y - block)
05     :precondition (limpo ?x) (limpo ?y) (na-mesa ?x)
06     :effect (and (sobre ?x ?y) (not (limpo ?y)) (na-mesa ?x))
07   )
08   (:action desempilhar
09     :parameters (?x ?y - block)
10     :precondition (and (sobre ?y ?y) (limpo ?x))
11     :effect (and (na-mesa ?x) (limpo ?y) (not (sobre ?x,?y)))
12   )
13 )
```

Fonte: (GHALLAB *et al.*, 2004)

Problemas de planejamento são atribuídos a um domínio. Em um arquivo do problema são definidos os objetos concretos, que instanciam os predicados e *esquema de ação* do domínio para proposições e ações, respectivamente. Além disso, são especificados o estado inicial e as metas do agente (GHALLAB *et al.*, 2004).

A Figura 8 mostra a descrição em PDDL de um problema de planejamento no domínio do *mundo dos blocos*. Na linha 02 são descritos os objetos, presentes no domínio, que são os blocos *A* e *B*. Na linha 03 é definido um estado inicial para o problema, em que temos: que os blocos *A* e *B* estão limpos, isto é, não há blocos sobre eles, e que os blocos *A* e *B* estão sobre a mesa. Finalmente, na linha 04 é defina a meta do problema, que é colocar o bloco *A* sobre o bloco *B*.

Figura 8 – Problema de planejamento no domínio do mundo dos blocos

Estado Inicial Meta

```
01 define (problem blocks2)
02    (objects:  A B)
03    (initial:  limpo(A), limpo(B), na-mesa(A), na-mesa(B))
04    (goal:  sobre(A,B))
```

Fonte: (GHALLAB *et al.*, 2004)

Para obter uma solução para o problema de planejamento, o agente (garra robótica) deve observar o estado atual e escolher ações, uma de cada vez, até alcançar um estado de meta. A sequência de ações, aplicada para chegar a tal estado, é chamada de *plano*. Por exemplo, um plano para o problema da Figura 8 é a sequência de ações {*empilhar (A,B)*}.

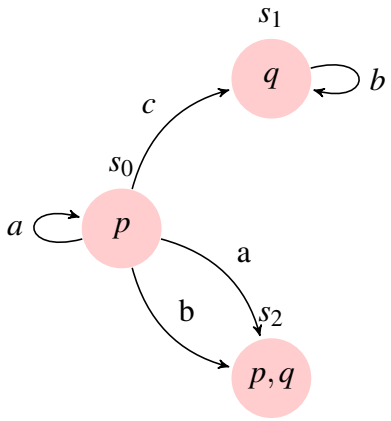
2.1.1 Domínio de Planejamento com ações Não Determinísticas

Em um domínio com ações determinísticas, a cada ação executada a partir do estado inicial é possível saber exatamente qual é o próximo estado do ambiente em que o agente se encontra. Por outro lado, em ambientes não determinísticos, o agente deve estar preparado para lidar com situações em que as ações possam ter efeitos incertos. Um domínio de planejamento não determinístico (*ou qualitativo*) é um modelo que expressa a *incerteza* que o agente tem acerca do comportamento do ambiente. Dado um estado *s* e uma ação *a*, esse modelo fornece ao agente uma *previsão* sobre o próximo estado do ambiente, caso a ação *a* seja executada no estado *s* (PEREIRA, 2007).

Definição 3 *(Domínio de Planejamento Não-Determinístico - Representação por meio de Sistema de Transição de Estados) Dado um conjunto de átomos proposicionais $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, um domínio de planejamento não determinístico (ou qualitativo) com assinatura ($\mathbb{P}$, $\mathbb{A}$) é definido por uma tupla $D = \langle S, L, T \rangle$, em que:*

- *$S \neq \emptyset$ é um conjunto finito de estados;*
- *$L : S \mapsto 2^{\mathbb{P}}$ é uma função de rotulação de estados e;*
- *$T : S \times \mathbb{A} \mapsto 2^{S}$ é uma função de transição de estados.*

Figura 9 – Sistema de Transição de estados de um domínio de planejamento não-determinístico sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$ e conjunto de ações $\mathbb{A} = \{a, b, c\}$



Fonte: Elaborado pela autora.

A Figura 9 ilustra um domínio, em que a ação ação a é não determinística, pois ao executá-la no estado s_0 há dois possíveis estados sucessores: o próprio estado s_0 ou o estado s_2.

Exemplo 2 *(Representação de Ação com pré-condições e efeitos) A ação a da Figura 9 pode ser representada por meio de suas pré-condições e efeitos, como a seguir:*

- *precond(a) = $\{p\}$ e;*
- *efeitos(a) = $\{(\neg q), (q)\}$.*

A Figura 10 ilustra o domínio do mundo dos blocos em `PDDL`, no qual as ações `empilhar` e `desempilhar` possuem efeitos não determinísticos. Para especificar os efeitos não-determinísticos, usa-se a palavra `oneof` para. Por exemplo, a ação *empilhar* um bloco x sobre um bloco y possui dois possíveis efeitos: a garra robótica pode ser bem sucedida na tarefa de empilhar, obtendo como efeito *((sobre ?x ?y) ¬(limpo ?y) (na-mesa ?x))*, ou pode falhar, fazendo com que os blocos permaneçam na mesma configuração. Observe que a linha 08 `(and)` corresponde a situação da falha na garra robótica.

Figura 10 – Esquema de ações do domínio do mundo dos blocos não determinístico

```
01 define (domain nondetblocksworld)
02   (:predicates (limpo ?x) (sobre ?x ?y) (na-mesa ?x))
03   (:action empilhar
04     :parameters (?x ?y - block)
05     :precondition (and (limpo ?x) (na-mesa ?x))
06     :effect
07       (oneof
08         (and)
09         (and (sobre ?x ?y) (not (limpo ?y)) (not (na-mesa ?x))))
10   )
11   (:action desempilhar
12     :parameters (?x ?y - block)
13     :precondition (and (sobre ?x ?y) (limpo ?x))
14     :effect
15       (oneof
16         (and)
17         (and (na-mesa ?x) (limpo ?y) (not (sobre ?x ?y))))
18   )
19
```

Fonte: (GHALLAB *et al.*, 2004)

Em domínios de planejamento não determinísticos, em vez de sequências de ações, planos devem ser *políticas*. Uma *política* é um mapeamento de estados em ações que define um padrão de comportamento para o agente: em cada instante, ele observa o estado corrente do ambiente e executa a ação mais apropriada para esse estado, conforme especificado pela política. Comportando-se dessa forma, o agente deve ser capaz de conduzir a evolução do ambiente, a despeito da ocorrência de eventos exógenos (onde o agente não tem controle), de modo que sua meta possa ser alcançada (PEREIRA, 2007).

Uma política pode ser *fraca*, *forte* ou *forte-cíclica*. Uma política fraca é uma solução em que é possível alcançar um estado meta, mas devido ao não-determinismo das ações esse alcance não é garantido. Uma *política forte* é uma solução que garante o alcance do estado meta, a despeito do não-determinismo das ações. Uma política *forte-cíclica* é uma solução em que se garante alcançar um estado meta, mesmo que esta solução contenha ciclos (PEREIRA, 2007).

A Figura 11 ilustra o problema de planejamento no domínio do mundo dos blocos não determinístico. Considerando s_0 o estado inicial do problema, então a solução para este problema é a política $\pi = \{(s_0, \text{empilhar}(A,B)\}$.

Figura 11 – Problema com 2 blocos no domínio do mundo dos blocos com ações não determinísticas

Estado Inicial Meta

```
define (problem blocks2)
   (domain:  nondetblocksworld)
   (objects:  A B)
   (initial:  limpo(A), limpo(B), na-mesa(A), na-mesa(B))
   (goal:  sobre(A,B))
```

Fonte: (GHALLAB *et al.*, 2004)

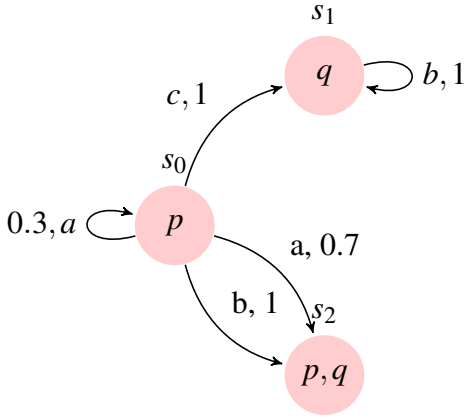
2.1.2 Domínio de Planejamento com Ações Probabilísticas

Um domínio de planejamento com ações probabilísticas supõe que o agente observou a natureza e obteve estatísticas sobre a frequência com que ela escolhe cada uma das ações.

Definição 4 *(Domínio de Planejamento Probabilístico - Representação por meio de Sistema de Transição de Estados) Dado um conjunto de átomos proposicionais $\mathbb{P}$ um conjunto e $\mathbb{A}$, um domínio de planejamento probabilístico (ou quantitativo) com assinatura ($\mathbb{P}$, $\mathbb{A}$) é definido pela seguinte tupla $D = \langle S, L, T \rangle$, em que (PEREIRA, 2007):*

- *$S \neq \varnothing$ é um conjunto finito de estados;*
- *$L : S \mapsto 2^{\mathbb{P}}$ é uma função de rotulação de estados; e*
- *$T : S \times A \times S \mapsto [0,1]$ é uma função de transição de estados probabilística.*

Figura 12 – Sistema de Transição de estados de um domínio de planejamento probabilístico sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$ e conjunto de ações $\mathbb{A} = \{a, b, c\}$



Fonte: Elaborado pela autora.

A Figura 12 ilustra um domínio, em que a ação ação a é probabilística, pois ao

executá-la no estado s_0 há dois possíveis estados sucessores: o próprio estado s_0 ou o estado s_2, em que cada possibilidade tem probabilidades associadas.

Observe que dados dois estados $s, s' \in S$ e uma ação $a \in \mathbb{A}$, a probabilidade de alcançar o estado s', dado que a ação a foi executada no estado s, é denotada por $s \in S$, se existe uma ação $a \in \mathbb{A}$ e um estado $s' \in S$ tal que $T(s,a,s') \neq 0$, então (PEREIRA, 2007):

$$\sum_{s' \in S} T(s,a,s') = 1$$

Exemplo 3 *(Representação de Ação com pré-condições e efeitos) A ação a da Figura 12 pode ser representada por meio de suas pré-condições e efeitos, como a seguir:*

- *precond(a)* = $\{p\}$ *e;*
- *efeitos(a)* = $\{(0.3, \neg q)$, $(0.7, q)\}$.

A Figura 13 ilustra o domínio do mundo dos blocos em `PDDL`, no qual as ações `empilhar` e `desempilhar` possuem efeitos probabilísticos. Para especificar os efeitos probabilísticos, usa-se a palavra `probabilistic` e em cada efeito inclui-se uma probabilidade de ocorrência. Por exemplo, a ação *empilhar* um bloco *x* sobre um bloco *y* possui dois possíveis efeitos: a garra robótica tem uma probabilidade de 0.75 de falhar (linha 07), fazendo com que os blocos permaneçam na mesma configuração, e 0.25 de ser bem sucedida na tarefa de empilhar os blocos (linha 08), obtendo como efeito *((sobre ?x ?y) ¬(limpo ?y) (na-mesa ?x))*.

Tanto problemas de domínios probabilísticos como não determinísticos pertencem ao planejamento sob incerteza, onde a natureza do domínio pode interferir nas ações do agente, modificando seus efeitos e tendo uma política como solução. Sendo que em domínios não determinísticos infere-se que o agente não tem nenhuma ideia sobre como a natureza irá agir. Já em domínios probabilísticos presume-se que o agente observou a natureza e obteve estatísticas sobre a frequência com que ela age, ou seja, uma probabilidade de ocorrer cada um dos efeitos.

2.1.3 Transformação de Domínios Quantitativos em Qualitativos

O trabalho de (PEREIRA, 2007) destaca que a única diferença entre domínios de planejamento sob incerteza quantitativos e qualitativos está no *tipo de função de transição de estados considerada* em cada um deles; sendo que nos domínios quantitativos essa função é probabilística e nos domínios qualitativos essa função é não-determinística. Assim, para

Figura 13 – Esquema de ações do domínio do mundo dos blocos probabilístico

```
01. define (domain prob-blocksworld)
02.   (:predicates (limpo ?x) (sobre ?x ?y) (na-mesa ?x))
03.   (:action empilhar
04.     :parameters (?x ?y - block)
05.     :precondition (and (limpo ?x) (limpo ?x) (na-mesa ?x))
06.     :effect
07.       (probabilistic 0.75 (and)
08.                   0.25 (and (sobre ?x ?y) (not (limpo ?y)) (na-mesa
?y)))
09.   )
10.   (:action desempilhar
11.     :parameters (?x ?y - block)
12.     :precondition (and (sobre ?x ?y) (limpo ?x))
13.     :effect
14.       (probabilistic 0.75 (and (na-mesa ?x) (limpo ?y) (not (sobre
?x ?y)))
15.            0.25 (and))
16.   )
```

Fonte: (GHALLAB *et al.*, 2004)

interpretar um domínio probabilístico como não determinístico, basta definir a função de transição de estados do modelo não-determinístico em termos da função de transição de estados do modelo probabilístico.

Definição 5 *(Transformação de Domínios Quantitativos em Qualitativos) Dado um domínio de planejamento quantitativo* $D = \langle S, L, T \rangle$ *em que* T *é uma função de transição probabilística* $T : S \times A \times S \mapsto [0,1]$*, o domínio de planejamento qualitativo* $D = \langle S, L, T' \rangle$ *correspondente pode ser obtido como a seguir (PEREIRA, 2007):*

$$T'(s,a) = \{s' \in S : T(s,a,s') > 0\},$$

em que T' *é uma função de transição não determinística.*

Exemplo 4 *(Transformando um Domínio Quantitativo em Qualitativo) Seja A um conjunto finito de ações e S um conjuntos finito dos estados, para transformar o exemplo da Figura 12 em não determinístico é preciso aplicar a* ***Definição 5*** *para cada para cada transição* $T(s,a)$*, tal que* $s \in S$ *e* $a \in A$*, teremos:*

$$T'(s_0,a) = \{T(s_0,a,s_0) > 0, T(s_0,a,s_2) > 0\}$$

$$T'(s_0,b) = \{T(s_0,b,s_2) > 0\}$$

$$T'(s_0, c) = \{T(s_0, c, s_1) > 0\}$$

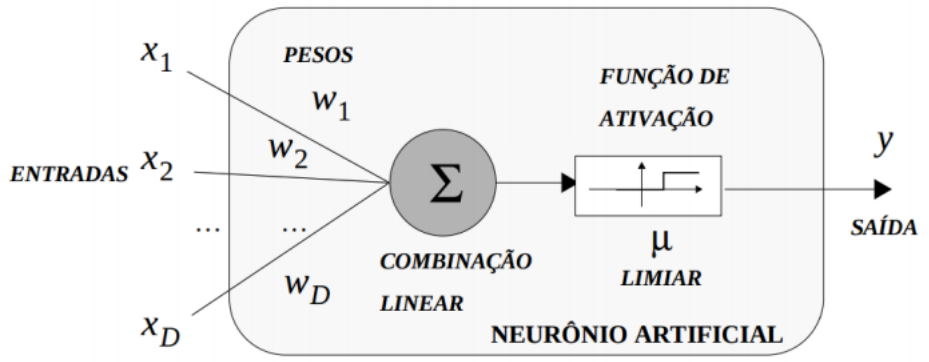
$$T'(s_1, b) = \{T(s_1, b, s_1) > 0\}$$

2.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são inspiradas na hipótese de que a atividade mental consiste basicamente na propagação de informações em redes de neurônios (NORVIG; RUSSELL, 2014). A Figura 14 apresenta o modelo de neurônio artificial proposto por (MCCULLOCH; PITTS, 1943), que é utilizado como modelo básico de um neurônio artificial.

Um neurônio artificial recebe um conjunto de valores de entrada, representado na Figura 14 por $x_1, x_2, ..., x_D$ e produz um valor de saída y. A entrada é representada por um vetor de valores $X = [x_1\ x_2\ ...\ x_D]^T$ em $\mathbb{R}^D$. A saída y irá indicar se o neurônio está ativo ou não. A ativação do neurônio é dada por uma função de ativação f, cuja entrada é uma combinação linear entre o vetor de entrada e um vetor de pesos $W = [w_1\ w_2\ ...\ w_D]$, que associa cada entrada com um fator de importância. A função de ativação realiza a propagação de informação quando a combinação de valores de suas entradas excede um dado limiar μ (RAUBER, 2005).

Figura 14 – Modelo de um neurônio



Fonte: (RAUBER, 2005)

A função de ativação desempenha um papel importante no treinamento de RNAs. Atualmente, a função de ativação mais utilizada, pela sua simplicidade e eficácia, é a *Unidade Linear Retificada (ReLU)*, definida como $f(x) = max(x, 0)$ (RAMACHANDRAN *et al.*, 2017).

Quando o propósito de uma RNA é realizar a tarefa de classificação, é comum selecionar para f a função denominada *softmax*, que permite interpretar os valores da camada de

saída como probabilidades posteriores. O valor produzido pela função *softmax* para o *i*-ésima neurônio da camada de saída é dada pela equação (BEZERRA, 2016):

$$f(a_i^{(L+1)}) = \frac{e^{a_i^{(L+1)}}}{\sum_{j=1}^{C} e^{a_j^{(L+1)}}}$$

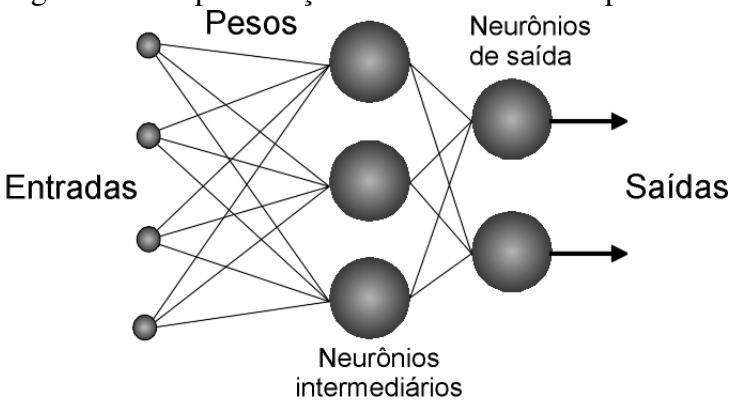
As saídas das unidades a i, $1 \leq i \leq C$, podem ser interpretadas como valores de probabilidade, visto que pertencem ao intervalo $[0,1]$ e sua soma é igual a 1.

A estrutura de neurônios também pode incluir uma polarização (ou bias) de entrada. Esta variável é incluída na função de ativação, com o intuito de aumentar o grau de liberdade desta função e, consequentemente, a capacidade de aproximação da rede (TALON, 2018).

O potencial e flexibilidade do cálculo baseado em RNA vêm da criação de conjuntos de neurônios que estão interligados entre si. Um neurônio da rede recebe um sinal nas suas entradas, processa esse sinal e emite um novo sinal de saída, que por sua vez é recebido por outros neurônios (RAUBER, 2005).

Em uma RNA os neurônios são comumente organizados em camadas. A Figura 15 apresenta uma rede neural com 3 camadas. Os neurônios na camada de entrada relacionam-se com os neurônios da camada intermediária, e cada neurônio desta camada intermediária comunica-se com os neurônios da camada de saída (CRESPO; HIDALGA, 2017).

Figura 15 – Representação de uma rede neural padrão



Fonte: (TAFNER, 1998)

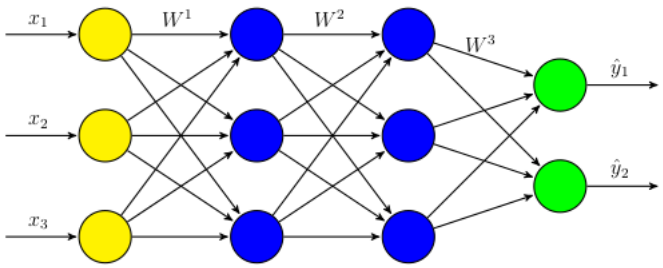
O modo como as camadas estão conectadas define a arquitetura de uma RNA.Essa arquitetura influencia diretamente nos problemas que uma RNA pode resolver. As seções a seguir apresentam duas arquiteturas de RNAs.

2.2.1 Multi-Layer Perceptron

Uma das arquiteturas mais simples de RNA é denominada *multi-layer perceptron* (MLP) (ROSENBLATT, 1958), que consiste em uma RNA com múltiplas camadas intermediarias e os neurônios de cada camadas são conectados com todos neurônios da camada sucessiva. A Figura 16 ilustra um MLP.

A implementação desta rede é realizada por meio de operações com matrizes. A entrada da rede são vetores de valores numéricos, por exemplo, na Figura 16 $X = [x_1 \quad x_2 \quad x_3]$. As conexões entre as camadas intermediárias são representadas por matrizes de pesos. Na Figura 16, por exemplo, W^1 e W^2 são matrizes de dimensão 3 por 3, representando a conexão de cada neurônio da camada anterior e seu grau de importância. A matriz W_3, por sua vez, possui dimensão 3 x 2, representando as conexões para a camada de saída. A saída é o vetor $\hat{y}$, formado pelos valores $\hat{y_1}$ e $\hat{y_2}$.[2]

Figura 16 – Arquitetura perceptron multicamadas com 4 camadas: uma camada de entrada, duas camadas intermediárias e uma camada de saída



Fonte: (SCHÄFER, 2018)

O valor de ativação de uma camada l é representado por um vetor h^l, resultante da aplicação da função de ativação f sobre o produto da matriz de pesos W^l correspondente a camada e o valor de ativação da camada anterior, somado ao vetor b^l (*bias*):

$$h^l = f(W^l \dot{h}^{l-1} + b^l)$$

O processo de aprendizado de uma MLP se dá por algoritmos que atualizam o vetor de pesos para produzir a saída esperada. Em um algoritmo de aprendizado supervisionado, por exemplo, o *treino* de uma rede é feito com um conjunto de dados rotulados. Depois é feito testes utilizando dados não rotulados e gera-se uma saída $\hat{y}$, que é comparada com a saída esperada y.

[2] O vetor $\hat{y}$ representa o vetor de valores preditos pela rede.

Utiliza-se métricas para avaliar o erro, que podem ser vistas como uma distância estimada da predição da rede até as respostas corretas. Tais métricas podem ser usadas tanto para avaliar a qualidade da rede como também para melhorar os parâmetros da rede.

O algoritmo de aprendizado, como por exemplo o *backpropagation* (HECHT-NIELSEN, 1992) , usa esse valor de erro para atualizar as matrizes de pesos nas camadas intermediárias.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (LECUN *et al.*, 1989) são RNAs que diferem das redes MLP por utilizar uma conectividade limitada entre neurônios, no lugar de conectar cada neurônio na camada de entrada com todos os neurônios da camada sucessiva (EL-SAYED *et al.*, 2013).

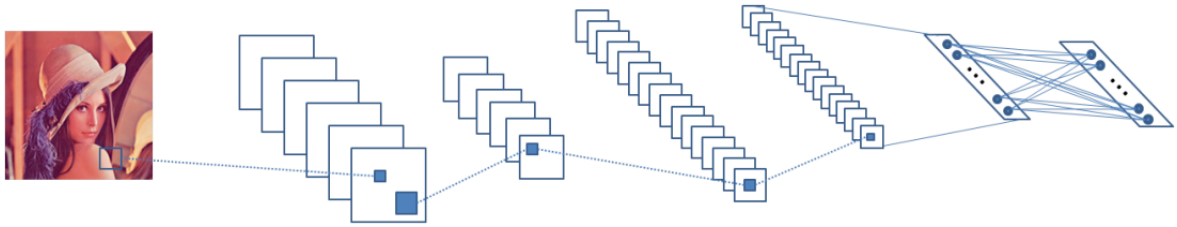
Em geral, CNNs são compostas por camadas *convolucionais*, camadas de subamostragem e camadas completamente conectadas (RASCHKA, 2015). As camadas convolucionais são responsáveis por aplicar filtros para extrair características da entrada. A camada de subamostragem (também conhecidas como camadas de *pooling*) reduz a dimensionalidade de um mapa de características gerando um novo mapa, com uma espécie de resumo. As camadas totalmente conectadas são essencialmente um MLP, onde cada unidade de entrada está conectada a cada unidade de saída (BEZERRA, 2016).

Existe uma diferença fundamental entre uma camada totalmente conectada e uma camada de convolução, as camadas densas aprendem padrões globais de acordo com as entradas (por exemplo para uma imagem, padrões envolvendo todos os *pixels*), enquanto camadas de convolução aprendem padrões locais: no caso de imagens, padrões encontrados em pequenas partes das entradas (CHOLLET, 2018).

Com isso permitimos a diminuição da quantidade de conexões intermediárias, evitando que cada neurônio em uma camada seja conectada com todos os neurônios da camada que a sucede. Assim, significa que os neurônios da rede não influenciam necessariamente todos os nós da camada sucessiva. Entretanto, ao avançar na rede o número de nós afetados aumenta. Essa forma de interagir é eficiente para aprender dependências complexas, reduzindo a quantidade de operações (SCHÄFER, 2018).

A Figura 17 apresenta um exemplo esquemático de uma CNN na qual, após a camada de entrada (que corresponde aos *pixels* da imagem), temos uma camada de convolução composta

Figura 17 – Arquitetura típica de uma CNN



Fonte: (BEZERRA, 2016)

de 6 mapas de características (representados como planos na figura), seguida de uma camada de subamostragem, completando o primeiro estágio. Essa figura ainda ilustra mais uma camada de convolução e outra de subamostragem, com um segundo estágio do processo antes das duas camadas completamente conectadas (BEZERRA, 2016).

CNNs são bastante utilizados e funcionam particularmente bem para problemas de visão computacional devido à sua capacidade de operar convolucionalmente, extraindo características de partes especificas de uma imagem e permitindo uma representação de maneira eficiente dos dados (CHOLLET, 2018).

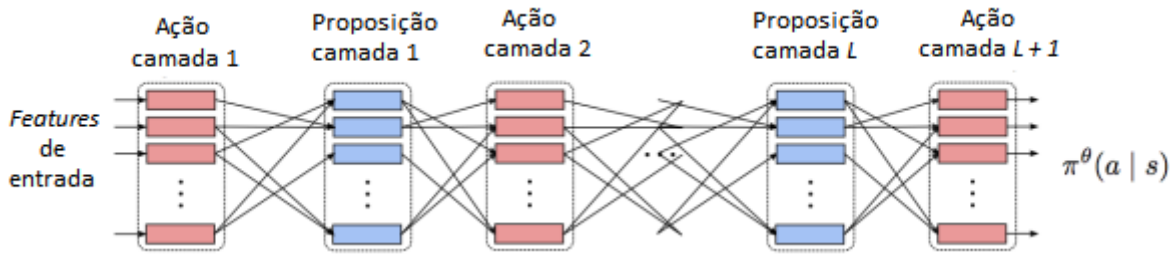
2.3 *Action Schema Networks*

Em (TOYER, 2017; TOYER *et al.*, 2018; SCHÄFER, 2018), propõem o uso de redes neurais para aprender a realizar planejamento automatizado, às quais denominam de Redes de Esquemas de Ações (*Action Schema Networks* - ASNets). Assim como em CNN, ASNets utilizam camadas intermediárias, e os neurônios de uma camada podem não ser completamente conectados com os neurônios da camada sucessiva. Além disso, são capazes de aprender como extrair uma solução a partir da descrição de problemas de planejamento e suas respectivas soluções (TOYER *et al.*, 2018).

ASNets são compostas por dois tipo de camadas, de ação e de proposição, dispostas de maneira alternada, havendo relações entre módulos contidos nas camadas. A Figura 18 ilustra a estrutura de uma ASNet, onde as entradas e saídas são alimentadas em uma camada de ação e os módulos em uma camada são conectados apenas aos módulos que possuem relação. As camadas de ação e proposição serão explicados a seguir.

Estas redes foram originalmente propostas para domínios de planejamento probabilísticos e se propõem a calcular uma política π_θ, gerando uma probabilidade $\pi_\theta(a|s)$ para escolher a ação a em um dado estado s para cada ação $a \in \mathbb{A}$. A saída é feita por uma camada de

Figura 18 – ASNet com L camadas de proposições e $L+1$ camadas de ações



Fonte: (TOYER, 2017)

ação, que depende dos parâmetros de pesos da rede, θ. Uma estratégia que pode ser adotada para escolha de ações, de acordo com uma política, é escolher a ação com maior probabilidade de ocorrer em π_θ (TOYER, 2017).

2.3.1 Camada de Ações

A camada de ação é composta por módulos que representam ações do domínio de planejamento. Um módulo de ação para $a \in \mathbb{A}$ na camada l recebe um vetor de entrada u_a^l e produz como saída uma valor ϕ_a^l dado por:

$$\phi_a^l = f(W_a^l \,.\, u_a^l + b_a^l)$$

onde d_h é o tamanho (fixo) de uma camada intermediária, e d_a^l o tamanho das entradas de um módulo de ação, $u_a^l \in \mathbb{R}^{d_a^l}$ é o vetor de entrada para a camada l, $W_a^l \in \mathbb{R}^{d_h \; x \; d_a^l}$ é a matriz de peso para este módulo de ação, e $b_l^a \in \mathbb{R}^{d_h}$ corresponde a *bias*. A função f é uma função de ativação não linear, como por exemplo ReLU na seção 2.2.

Dado um conjunto de átomos proposicionais $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$. O vetor de u_a^l corresponde ao vetor de entrada obtido pela camada de ação l. Esse vetor contém apenas valores que correspondem as proposições $p_1, \cdots, p_M$, que estão relacionadas com a ação a. Essa relação é formalmente definida como $R = \{(a,p) \in \mathbb{A} \times \mathbb{P} \mid p \in (a) \cup efeitos_a^+ \cup efeitos_a^-\}$, denotando por $R(a,p)$ se $(a,p) \in R$. O vetor u_a^l é dado por:

$$u_a^l = \begin{bmatrix} \psi_1^{l-1} \\ \vdots \\ \psi_M^{l-1} \end{bmatrix}$$

Cada módulo de proposição $\psi_i^{l-1} \in \mathbb{R}^{d_h}$ tem como tamanho de representação d_h. A

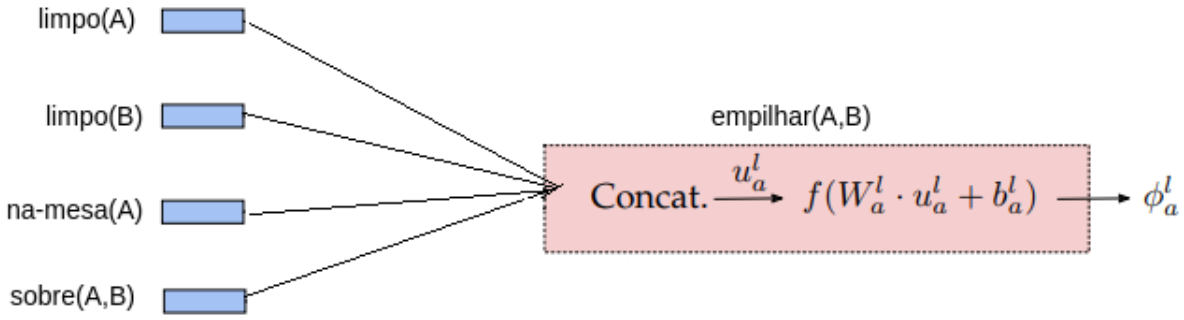
conectividade esparsa de ASNets com base na definição de relacionamentos é inspirada na ideia de filtros (*kernels*) de CNNs, que conectam pontos de dados com base na vizinhança espacial.

Figura 19 – Ação *empilhar(A,B)* do domínio do mundo dos blocos

```
action:  empilhar(A,B)
  precondition:  {limpo(A), limpo(B), na-mesa(A)}
  effects+:  {sobre(A,B)}
  effects−:{limpo(B), na-mesa(A)}
```

Fonte: Adaptado de (GHALLAB *et al.*, 2004).

Figura 20 – Ilustração do módulo de ação para *empilhar(A,B)* do *mundo dos blocos*



Fonte: Adaptado de (TOYER, 2017)

A Figura 19 ilustra o PDDL da ação *empilhar(A,B)* com as suas precondições, efeitos positivos e negativos. Em ASNet é construído um vetor (u_a^l) a partir da concatenação de todas as proposições que se relacionam com a ação. Para a ação *empilhar(A,B)* as proposições relacionadas são {*limpo(A),limpo(B),na-mesa(A),sobre(A,B)*}. O vetor pode ser visualizado na Figura 20, em seguida u_a^l passa pela função f, gerando a ação ϕ_a^l.

2.3.2 Camada de Entrada

Na camada de entrada os módulos recebem um vetor binário de entrada u_a^1. Este vetor inclui valores verdadeiros para proposições no estado s, valores indicando quais proposições aparecem na meta e um valor indicando se uma ação é aplicável em s.

$$u_a^1 = \left[v^T \; g^T \; m \right]^T$$

Supondo que as proposições relacionadas com a ação a_i sejam $p_1, \cdots, p_M$. Nesse

caso, $v \in \{0,1\}^M$ é o vetor de valores verdade para as proposições $p_1, \cdots, p_M$. Teremos $v_i = 1$ se proposição p_i é verdadeira no estado s e $v_i = 0$ caso contrário. Da mesma forma, $g \in \{0,1\}^M$ indicando se cada proposição p_i aparece na meta do problema $(g_i = 1)$ ou não $(g_i = 0)$. Finalmente, o valor da máscara $m \in \{0,1\}$ indicando se a é aplicável no estado s.

Por exemplo, para $\mathbb{A}$ = *{empilhar (A,B), empilhar (B,A), desempilhar (A,B), desempilhar (B,A)}* e $\mathbb{P}$ = *{limpo (A), limpo (B), na-mesa (A), na-mesa(B), sobre (A,B), sobre (B,A)}*. De acordo com o problema do *mundo dos blocos* (Figura 8) teremos que o vetor $m = [1100]$, no qual as ações *{empilhar (A,B), empilhar (B,A)}* seriam aplicáveis e as ações *{desempilhar (A,B), desempilhar (B,A)}* não seriam aplicáveis. Por fim, o vetor $u^1_{a_1}$ para a ação *empilhar(A,B)* seria o seguinte:

$$u^1_{a_1} = \Big[[1110]^T \ [0001]^T \ 1 \Big]^T$$

2.3.3 Camada de Saída

Na camada de saída a rede gera uma probabilidade $\pi_\theta(a|s)$ para cada ação a. Esta camada calcula uma distribuição de probabilidade sobre todas as ações aplicáveis, assegurando que $\sum_{a \in \mathbb{A}} \pi_\theta(a|s) = 1$ e que $\pi_\theta(a|s) = 0$ se as precondições de a não são satisfeitas em s. Para garantir que as ações que não são aplicáveis nunca sejam selecionadas, e garantir que as probabilidades de ação sejam normalizadas para 1, as saídas passam pela função de ativação *softmax* com as máscara m_i, garantindo que $\pi_\theta(a|s) = 0$ se a não é aplicável em s.

A função de ativação *softmax* que calcula a probabilidade π_i de ocorrência da ação a_i entre todas as ações $\mathbb{A} = \{a_1, \ldots, a_N\}$:

$$\pi_i = \frac{m_i \cdot exp(\phi^{L+1}_{a_i})}{\sum_{j=1}^{N} m_j \cdot exp(\phi^{L+1}_{a_j})}$$

2.3.4 Camada de Proposição

A camada de proposição é semelhante as camadas de ação. Especificamente, um módulo de proposição para proposição $p \in \mathbb{P}$ na *l-ésima* camada de proposição irá calcular uma representação:

$$\psi^l_p = f(W^l_p \,.\, v^l_p + b^l_p)$$

onde $v_a^l \in \mathbb{R}^{d_p^l}$ é o vetor de entrada, $W_p^l \in \mathbb{R}^{d_h \; x \; d_p^l}$ é a matriz de peso aprendido para este módulo de proposição, e $b_p^l \in \mathbb{R}^{d_h}$ corresponde a *bias*. A função f é a mesma não linear usada nos módulos de ação.

A principal diferença entre os módulos de proposição e ação, e que também torna a entrada v_p^l nas camadas de proposição um pouco mais complicadas, é que o número de ações relacionadas a uma proposição podem variar.

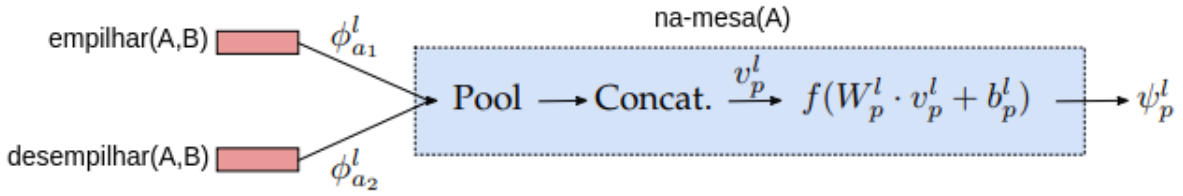
Para construir o vetor v_p^l, primeiro é encontrado o predicado para a proposição $p \in \mathbb{P}$, em seguida, enumerar todos os esquemas de ação $A_1, \cdots, A_L \in \mathbb{A}$ que possui ralação com p. O vetor v_p^l pode ser definido como:

$$v_p^l = \begin{bmatrix} pool(\{\phi_p^{l\,T} \mid op(a) = A_1 \wedge R(a,p)\}) \\ \vdots \\ pool(\{\phi_p^{l\,T} \mid op(a) = A_L \wedge R(a,p)\}) \end{bmatrix}$$

onde $op(a) \in \mathbb{A}$ representa o esquema de ação para a ação a, e *pool* é uma função de *pool* que combina vários vetores de ações. Em ASNets é suposto que a função de *pool* calcula o *max pool* (isto é, mantém apenas a maior entrada).

A Figura 21 ilustra o módulo de proposição para *na-mesa(a)*, que está relacionada com as ações *empilhar(A,B)* e *desempilhar(A,B)*. Que passam por uma função de *pool*, em seguida por uma concatenação e é aplicada uma função f gerando a saída ψ_p^l.

Figura 21 – Ilustração do módulo de proposição para *na-mesa(a)* do *mundo dos blocos*



Fonte: Adaptado de (TOYER, 2017)

3 TRABALHOS RELACIONADOS

Aprendizado de Máquina vêm sendo utilizado na área de *Planejamento Automatizado* para aprender automaticamente: (i) um esquema de ações do domínio de planejamento; (ii) novas heurísticas para tornar mais eficiente a busca por uma solução e, mais recentemente; (iii) como obter uma solução para um problema de planejamento.

Planejadores raciocinam sobre esquemas de ações corretos e completos que modelam as transições entre estados do mundo. No entanto, construir um novo esquema de ações é uma tarefa difícil e demorada. Uma abordagem alternativa é usar aprendizado de máquina no processo de construção de um novo esquema de ações. Em 2005, ocorreu a primeira Competição de Engenharia de Conhecimento para Planejamento (ICKEPS - *International Competition on Knowledge Engineering for Planning and Schedule*) na qual abordagens de aquisição automática de esquemas de ações começaram a ser apresentadas. Destaque para os trabalhos de (WALSH; LITTMAN, 2008) e (CRESSWELL *et al.*, 2009) que aprendem esquemas de ações com pré-condições e efeitos para domínios com ações determinísticas e; o trabalho de (JIMÉNEZ *et al.*, 2008) que aprende um esquema de ações para domínios com ações probabilísticas.

Planejadores estado da arte são em sua grande maioria baseado em busca heurística (RICHTER; WESTPHAL, 2010; TORRALBA *et al.*, 2017; STEINMETZ *et al.*, 2016a). Desta forma, algoritmos de aprendizado de máquina estão sendo utilizados para aprender heurísticas automaticamente. A partir de 2008, foi introduzida uma trilha de planejadores baseados em aprendizado de máquina na Competição Internacional de Planejamento (IPC - *International Planning Competition*). Destaque para os trabalhos de (YOON *et al.*, 2006; XU *et al.*, 2007; BOHNENBERGER, 2016) que são capazes de aprender heurísticas a partir da análise da estrutura de domínios, problemas e planos.

Mais recentemente, (TOYER *et al.*, 2018) propuseram um algoritmo que recebe um conjunto de problemas para um certo domínio de *planejamento probabilístico* e os planos soluções para cada um destes problemas. Com isso, o algoritmo é capaz de aprender como planejar para um problema novo. A abordagem apresenta uma arquitetura de rede neural, denominada *Action Schema Network* (ASNet) que é um grafo cujos nós podem ser átomos proposicionais e ações e cujas arestas definem como um átomo proposicional influencia no efeito de uma ação. O trabalho de (SCHÄFER, 2018) utilizou desta abordagem para obtenção de planos para problemas de *planejamento determinístico*. Bem como em (TOYER *et al.*, 2018) e (SCHÄFER, 2018), este trabalho utilizará ASNet, mas para domínios com ações não

determinísticas.

O estado da arte para obtenção de políticas para planejamento não-determinístico é o planejador PRP (*Planning for Relevance Policies*) (MUISE *et al.*, 2012). Este planejador realiza um processo de conversão de ações não-determinísticas em determinísticas (*determinização de ações*) e usa um planejador estado-da-arte clássico para obtenção de soluções. Este algoritmo não utiliza algoritmos de aprendizado de máquina para obtenção de soluções.

Tabela 1 – Comparativo entre as abordagens apresentadas.

Trabalho(s)	**Usa Aprendizado de Máquina? Como?**	**Tipo de Domínio**
(WALSH; LITTMAN, 2008; CRESSWELL *et al.*, 2009)	Sim, para aprender esquemas de ações.	Determinístico
(JIMÉNEZ *et al.*, 2008)	Sim, para aprender esquemas de ações.	Probabilístico.
(YOON *et al.*, 2006; XU *et al.*, 2007; BOHNENBERGER, 2016)	Sim, para aprender heurísticas.	Determinístico
(TOYER *et al.*, 2018)	Sim, para obter políticas.	Probabilístico
(SCHÄFER, 2018)	Sim, para obter planos.	Determinístico
PRP (MUISE *et al.*, 2012)	Não.	Não-determinístico
Este trabalho	Sim, para obter políticas	Não-determinístico

Fonte: Elaborado pela autora.

.

4 RESULTADOS

Como visto no Capítulo 2, as ASNets raciocinam sobre domínios com ações probabilísticas. No entanto, o foco deste trabalho é a obtenção de soluções de problemas de planejamento não-determinísticos.

Desta forma, neste trabalho são apresentadas duas contribuições: (i) a proposta de transformação das ações não-determinísticas em probabilísticas e; (ii) o uso das ASNets na obtenção de políticas para esta classe de problemas de planejamento.

4.1 Transformação de domínios Qualitativos para Quantitativos

Como as ASNets são capazes de raciocinar sobre os domínios com ações probabilísticas, propomos, nesta seção, um método de transformação de cada ação $a \in \mathbb{A}$ de um domínio não-determinístico (domínio qualitativo) em uma ação probabilística correspondente (domínio quantitativo).

Transformar uma classe de problemas em outra é uma tarefa efetuada por alguns trabalhos da literatura de planejamento. O planejador PRP (MUISE *et al.*, 2012), por exemplo, obtém soluções para planejamento não-determinístico realizando a "*determinização*" das ações não-determinísticas, isto é, transformando as ações não-determinísticas em determinísticas e usando planejadores clássicos estado-da-arte para obtenção de soluções. Outro planejador que realiza transformações de uma classe de problemas em outro é a proposto por (PEREIRA, 2007). Este planejador realiza a transformação de ações probabilísticas em não-determinísticas a fim de utilizar algoritmos de *planejamento baseado em verificação de modelos* para obter políticas.

Como observado em Pereira (2007), a diferença entre domínios qualitativos e quantitativos está no tipo de função de transição de estados; nos domínios quantitativos a função de transição é probabilística e nos qualitativos a função de transição é não-determinística. Assim, para interpretar um domínio não determinístico como probabilístico, basta definir a função de transição de estados do modelo probabilístico em termos da função de transição de estados do modelo não determinístico.

Definição 6 *(Transformação de Domínios Qualitativos em Quantitativos) Dado um domínio de planejamento qualitativo $D = \langle S, L, T \rangle$ em que T é uma função de transição não-determinística $T : S \times \mathbb{A} \mapsto 2^S$, o domínio de planejamento qualitativo $D = \langle S, L, T' \rangle$ correspondente pode ser*

obtido como a seguir. Seja $X = \{s_1, ..., s_n\}$, $X \subseteq S$, *o conjunto de estados tal que* $T(s,a) = X$, *para cada* $s \in S$ *temos que:*

$$T'(s,a,s') = \frac{1}{n}$$

em que T' *é uma função de transição probabilística.*

Exemplo 5 *(Transformando um Domínio Qualitativo em Quantitativo) Seja* $D =< S, L, T >$ *com assinatura* $\mathbb{P}, \mathbb{A}$ *o domínio não-determinístico (Figura 9). Será mostrado aqui como transformar tal domínio em um domínio* $D' =< S, L, T' >$ *probabilístico correspondente. Aplicando a Definição 6, cada transição* $T(s,a)$ *do domínio D será transformada em uma transição de* T' *da seguinte maneira:*

$$T(s_0, a, s_0) = \frac{1}{2} = 0.5$$

$$T(s_0, a, s_2) = \frac{1}{2} = 0.5$$

$$T(s_0, b, s_2) = 1$$

$$T(s_0, c, s_1) = 1$$

$$T(s_1, b, s_1) = 1$$

4.2 Usando redes neurais ASNET para solucionar problemas de planejamento não determinísticos

Nesta seção, mostraremos como domínios não-determinísticos na linguagem PDDL podem ser convertidos em domínios probabilístico nesta mesma linguagem. A transformação de domínios PDDL não-determinísticos em probabilísticos segue o proposto na Definição 6, uma vez que a linguagem de ações é a forma compacta, porém, correspondente das transições de um domínio de planejamento.

4.2.1 Transformação dos Arquivos de Entrada

Como entrada, tendo um domínio PDDL com ações não-determinísticas, como a a ação `empilhar` do domínio do mundo dos blocos mostrada na Figura 22. Os efeitos não-determinísticos da ação são dadas pelo operador `oneof` que determina que um dos efeitos pode ocorrer. No caso da ação não-determinística `empilhar` podem ocorrer as seguintes situações: não acontecer nada, efeito representado por `(and)` com nenhum literal associado; ou o agente ser bem sucedido na execução da ação de empilhar blocos, efeito representado pela conjunção de literais `(and (sobre ?x ?y) (limpo ?y) (na-mesa ?x))`. A palavra `oneof` do domínio não-determinístico será substituída pela palavra `probabilistic` e cada efeito não determinístico terá uma probabilidade associada, conforme calculado pela Definição 6.

A Figura 23 ilustra a ação *empilhar* do domínio do mundo dos blocos, após ser transformada em uma ação probabilística. Neste caso, há a probabilidade de 0.5 de não ocorrer nada e 0.5 do agente ser bem sucedido na execução da ação de empilhar blocos.

Figura 22 – Esquema da ação empilhar do domínio do mundo dos blocos não determinístico

```
(:action empilhar
    :parameters (?x ?y - block)
  :precondition (and (limpo ?x) (limpo ?y) (na-mesa ?x))
  :effect
    (oneof
      (and)
      (and (sobre ?x ?y) (limpo ?y) (na-mesa ?x)))
```

Fonte: (GHALLAB *et al.*, 2004)

Figura 23 – Esquema da ação empilhar do domínio do mundo dos blocos transformado para probabilístico

```
(:action empilhar
  :parameters (?x ?y - block)
  :precondition (and (limpo ?x) (limpo ?y) (na-mesa ?x))
  :effect
      (probabilistic (0.5 (and))
      (0.5 (and) (and (sobre ?x ?y) limpo(?y) (na-mesa ?x))))
```

Fonte: Elaborado pela autora.

É importante ressaltar que apenas é preciso fazer a conversão no arquivo em que há a descrição das ações, isto é, o arquivo do domínio. Todos os problemas, que descrevem objetos, estado inicial e meta, relacionados ao domínio ficam inalterados.

4.2.2 Experimentos

Para os experimentos, utilizamos os problemas do domínio não-determinístico do mundo dos blocos da IPC 2008 (*International Planning Competition*)[1] e os problemas gerados por (TOYER *et al.*, 2018)[2]. Os experimentos foram executados em uma máquina com processador Intel Core *i5-3470 CPU @3.20GHz x 4* e 8GB de memória RAM.

O objetivo deste experimento é verificar o desempenho das redes ASNETS na tentativa de solucionar problemas de um domínio não-determinístico, convertidos para probabilístico. No total foram utilizados 150 problemas do mundo dos blocos com diferentes quantidades e configurações de blocos. Destas instâncias, o menor problema possui 4 blocos e o maior problema possui 40 blocos. Os problemas foram separados em: 119 problemas para treinamento da rede e 31 problemas para testes. A arquitetura da rede foi configurada com uma camada de entrada, 16 camadas intermediárias e uma camada de saída. Após configurar a arquitetura, foi realizado o treinamento supervisionado da rede com 70 épocas. Para cada um dos 31 problemas teste, 100 tentativas (*trials*) de execução da política foram executadas. Em cada *trial*, uma execução de uma ação corresponde a seleção feita pela natureza de que efeito não-determinístico da ação deve ocorrer.

A Tabela 2 apresenta os resultados obtidos pelas AsNets. A tabela contém: o número de blocos de cada problema; se foi possível encontrar uma solução; a quantidade de *trials* da política que levaram a obtenção da meta (total de 100 trials para cada problema); o custo da melhor solução encontrada e o tempo de execução. Os experimentos mostram que as ASNETS conseguiram encontrar uma solução para 6 dos 31 problemas testados. Estas soluções obtidas foram para problemas com até 25 blocos. A partir deste tamanho de problema, as ASNETS não obtiveram sucesso na elaboração de soluções. Nos 6 problemas solucionados, a rede encontrou 1 ou mais soluções, tendo um caso em que obteve 23 soluções para um problema.

A rede não obteve um bom desempenho no domínio não determinístico apresentado, solucionando apenas 6 dos 31 problemas apresentados. Observa-se que isto pode ter ocorrido devido a interpretação feita neste trabalho de que os efeitos das ações não determinísticas possuem a mesma probabilidade de ocorrer. Visto que, com ações probabilísticas, em que se tem a obtenção de dados estatísticos sobre os efeitos das ações, pode haver probabilidades diferentes e estas são convertidas nos pesos que a rede atribui para a ocorrência dos efeitos. Neste tipo de

[1] Problemas disponíveis em <http://icaps-conference.org/ipc2008/deterministic>
[2] Problemas disponíveis em <https://github.com/qxcv/asnets/tree/master/problems>

Tabela 2 – Desempenho das ASNETs em 31 problemas teste do domínio do mundos dos blocos não determinístico

ASNET				
Blocos	**Solução**	**Menor Custo**	**Menor Tempo**	**Nº Soluções**
5	Não	—	—	-
15	Sim	79	4.15s	23
15	Não	—	—	-
15	Não	—	—	-
15	Sim	79	4.06s	11
15	Sim	163	8.48s	3
20	Não	—	—	-
20	Não	—	—	-
20	Não	—	—	-
20	Sim	272	26.98s	1
20	Não	—	—	-
25	Não	—	—	-
25	Não	—	—	-
25	Sim	212	37.77s	1
25	Não	—	—	-
25	Sim	290	50.61s	2
30	Não	—	—	-
30	Não	—	—	-
30	Não	—	—	-
30	Não	—	—	-
30	Não	—	—	-
35	Não	—	—	-
35	Não	—	—	-
35	Não	—	—	-
35	Não	—	—	-
35	Não	—	—	-
40	Não	—	—	-
40	Não	—	—	-
40	Não	—	—	-
40	Não	—	—	-
40	Não	—	—	-

Fonte: Elaborado pela autora.

situação, como apresentado por Toyer *et al.* (2018), tais redes obtém um resultado bem superior.

Para melhorar tal resultado pode-se testar outros parâmetros de configuração e propor uma arquitetura de rede que seja mais apropriada para lidar com ações não determinísticas.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi realizada uma avaliação do uso das ASNETS para obtenção de soluções para domínios de planejamento não determinísticos. Os testes foram realizados com problemas do domínio do mundo dos blocos não determinístico.

As ASNETS foram criadas para solucionar problemas probabilísticos, mas também já foi testado seu uso para problemas determinísticos, como mostrado por (TOYER *et al.*, 2018; SCHÄFER, 2018). Para utilizar as ASNETS para solucionar problemas de domínios não determinísticos foi proposta uma transformação de ações não determinísticas em probabilísticas, atribuindo probabilidade iguais para cada efeito de uma ação.

A rede conseguiu encontrar uma solução para 6 dos 31 problemas testados. Estas soluções obtidas foram para problemas com até 25 blocos. A partir deste tamanho de problema, as ASNETS não obtiveram sucesso na elaboração de soluções. Porém, a rede mostra-se promissora para solucionar problemas de domínios não determinísticos, considerando que mesmo sem estar preparada para este tipo de domínio ela obteve soluções para parte dos problemas apresentados.

Como trabalhos futuros podemos citar: modificar o algoritmo utilizado nas ASNETS para ser aplicável em domínios não determinístico; utilizar outra técnica de transformação de um domínio não determinístico para probabilístico e; testar o uso das ASNETS para outros domínios não determinísticos.

REFERÊNCIAS

BARROS, L. N. de. **Planejamento em Inteligência Artificial**. São Paulo: IME-USP, 2003.

BEZERRA, E. Introdução à aprendizagem profunda. **Simpósio Brasileiro de Banco de Dados–SBBD2016**, Salvador, 2016.

BOHNENBERGER, C. **A case study exploring the potential of deep learning in ai planning**. Tese (Doutorado) — Saarland University, 2016.

BYLANDER, T. The computational complexity of propositional strips planning. **Artificial Intelligence**, Elsevier. [S.l], v. 69, n. 1-2, p. 165–204, 1994.

CHOLLET, F. **Deep Learning with Python**. [S.l.]: MITP-Verlags GmbH & Co. KG, 2018.

CIMATTI, A.; PISTORE, M.; ROVERI, M.; TRAVERSO, P. Weak, strong, and strong cyclic planning via symbolic model checking. **Artificial Intelligence**, Elsevier. [S.l], v. 147, n. 1-2, p. 35–84, 2003.

CRESPO, N.; HIDALGA, S. A. de L. **Reconhecimento de tumores cerebrais utilizando redes neurais convolucionais**. Rio Grande do Sul: Universidade Federal do Pampa, 2017.

CRESSWELL, S.; MCCLUSKEY, T. L.; WEST, M. Acquisition of object-centred domain models from planning examples. In: **Nineteenth International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2009.

EL-SAYED, M. A.; ESTAITIA, Y. A.; KHAFAGY, M. A. Automated edge detection using convolutional neural network. **Editorial Preface**, Citeseer. [S.l], v. 4, n. 10, 2013.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: theory and practice**. [S.l.]: Elsevier, 2004.

HECHT-NIELSEN, R. Theory of the backpropagation neural network. In: **Neural networks for perception**. [S.l.]: Elsevier, 1992. p. 65–93.

HOFFMANN, J. Ff: The fast-forward planning system. **AI magazine**, [S.l], v. 22, n. 3, p. 57–57, 2001.

JIMÉNEZ, S.; FERNÁNDEZ, F.; BORRAJO, D. The pela architecture: integrating planning and learning to improve execution. In: **National Conference on Artificial Intelligence (AAAI'2008)**. [S.l.: s.n.], 2008.

KAMBHAMPATI, S.; NIGENDA, R. S. Distance-based goal-ordering heuristics for graphplan. In: **AIPS**. [S.l.: s.n.], 2000. p. 315–322.

KAUTZ, H.; SELMAN, B. Blackbox: A new approach to the application of theorem proving to problem solving. In: **AIPS98 Workshop on Planning as Combinatorial Search**. [S.l.: s.n.], 1998. v. 58260, p. 58–60.

KELLER, T.; EYERICH, P. Prost: Probabilistic planning based on uct. In: **Twenty-Second International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2012.

KOLOBOV, A. Planning with markov decision processes: An ai perspective. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, Morgan & Claypool Publishers. [S.l], v. 6, n. 1, p. 1–210, 2012.

LECUN, Y. *et al.* Generalization and network design strategies. In: **Connectionism in perspective**. [S.l.]: Citeseer, 1989. v. 19.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer. [S.l], v. 5, n. 4, p. 115–133, 1943.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. **PDDL-the planning domain definition language**. [S.l], 1998.

MENEZES, M. V. **Mudanças em Problemas de Planejamento sem Solução**. Tese (Doutorado) — UNIVERSIDADE DE SAO PAULO, 2014.

MUISE, C. J.; MCILRAITH, S. A.; BECK, C. Improved non-deterministic planning by exploiting state relevance. In: **Twenty-Second International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2012.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial, 3a Edição**. [S.l.]: Elsevier Brasil, 2014.

PEREIRA, S. d. L. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. Tese (Doutorado) — Universidade de São Paulo, 2007.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **arXiv preprint arXiv:1710.05941**, [S.l], 2017.

RASCHKA, S. **Python machine learning**. [S.l.]: Packt Publishing Ltd, 2015.

RAUBER, T. W. **Redes neurais artificiais**. Universidade Federal do Espírito Santo, 2005.

RICHTER, S.; WESTPHAL, M. The lama planner: Guiding cost-based anytime planning with landmarks. **Journal of Artificial Intelligence Research**, [S.l], v. 39, p. 127–177, 2010.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association. [S.l], v. 65, n. 6, p. 386, 1958.

SCHÄFER, L. **Domain-Dependent Policy Learning using Neural Networks in Classical Planning**. Tese (Doutorado) — Saarland University, 2018.

STEINMETZ, M.; HOFFMANN, J.; BUFFET, O. Goal probability analysis in probabilistic planning: Exploring and enhancing the state of the art. **Journal of Artificial Intelligence Research**, [S.l], v. 57, p. 229–271, 2016.

STEINMETZ, M.; HOFFMANN, J.; BUFFET, O. Revisiting goal probability analysis in probabilistic planning. In: **Twenty-Sixth International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2016.

TAFNER, M. A. Redes neurais artificiais: aprendizado e plasticidade. **Revista Cérebro e Mente**, Universidade Estadual de Campinas. [S.l], 1998.

TALON. **Deep Lerning Book**. [S.l.]: Data Science Academy, 2018. Disponível em: http://deeplearningbook.com.br. Acesso em: 24 maio, 2019.

TORRALBA, Á.; ALCÁZAR, V.; KISSMANN, P.; EDELKAMP, S. Efficient symbolic search for cost-optimal planning. **Artificial Intelligence**, Elsevier. [S.l], v. 242, p. 52–79, 2017.

TOYER, S. Generalised policies for probabilistic planning with deep learning. **Research and development, honours thesis, Research School of Computer Science**, Australian National University, 2017.

TOYER, S.; TREVIZAN, F.; THIÉBAUX, S.; XIE, L. Action schema networks: Generalised policies with deep learning. In: **AAAI Conference on Artificial Intelligence (AAAI)**. [S.l.: s.n.], 2018.

TREVIZAN, F. W.; THIÉBAUX, S.; HASLUM, P. Operator counting heuristics for probabilistic planning. In: **IJCAI**. [S.l.: s.n.], 2018. p. 5384–5388.

WALSH, T. J.; LITTMAN, M. L. Efficient learning of action schemas and web-service descriptions. In: **AAAI**. [S.l.: s.n.], 2008. v. 8, p. 714–719.

XU, Y.; FERN, A.; YOON, S. W. Discriminative learning of beam-search heuristics for planning. In: **IJCAI**. [S.l.: s.n.], 2007. p. 2041–2046.

YOON, S. W.; FERN, A.; GIVAN, R. Learning heuristic functions from relaxed plans. In: **ICAPS**. [S.l.: s.n.], 2006. v. 2, p. 3.