UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE ENGENHARIA DE COMPUTAÇÃO

MARISA DO CARMO SILVA

BUSCA HEURÍSTICA SIMBÓLICA PARA PLANEJAMENTO DETERMINÍSTICO

QUIXADÁ

2019

MARISA DO CARMO SILVA

BUSCA HEURÍSTICA SIMBÓLICA PARA PLANEJAMENTO DETERMINÍSTICO

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

Orientadora: Profa. Dra. Maria Viviane de 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)

S581b Silva, Marisa do Carmo.
Busca heurística simbólica para planejamento determinístico / Marisa do Carmo Silva. – 2019.
49 f. : il. color.

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

1. Inteligência artificial. 2. Planejamento automatizado. 3. Heurística. I. Título.

CDD 621.39

MARISA DO CARMO SILVA

BUSCA HEURÍSTICA SIMBÓLICA PARA PLANEJAMENTO DETERMINÍSTICO

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

Aprovada em: __/ __ / __

BANCA EXAMINADORA

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

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

Prof. Dr. Criston Pereira de Souza
Universidade Federal do Ceará (UFC)

Profa. Dra. Leliane Nunes de Barros
Universidade de São Paulo (USP)

Dedico este trabalho à minha família, por todo o apoio e amor que me deram em toda minha vida.

AGRADECIMENTOS

À minha mãe, Terezinha, e ao meu pai, José Simão, pelos sacrifícios que fizeram para que eu pudesse usufruir de uma boa educação, pelo apoio, carinho e amor que sempre me foi dado durante minha vida e por serem meu alicerce. Obrigada por me apoiarem em todas as decisões, acreditarem em mim e se preocuparem tanto comigo.

À minha irmã, Mayara, pelos cuidados, conselhos e por sempre estar ao meu lado quando precisei. Agradeço ao meu irmão, Marcelo, por tudo que me ensinou na vida e por ter me apresentado à todo um universo de coisas que hoje eu sou fã. Agradeço aos meus sobrinhos, Dante e Heitor, por trazerem mais alegria, fofura e luz à minha casa.

À minha família, que sempre me incentivou e me acolheu, em especial à minha prima, Alrilene, que sempre me tratou como uma irmã mais nova. Agradeço aos meus padrinhos, Verbenia Sousa e Ivan Bezerra, por todo apoio, carinho e por sempre me receberem de braços abertos.

À professora Maria Viviane de Menezes e ao professor Paulo de Tarso Guerra Oliveira, por toda orientação, conhecimento e experiências que foram compartilhadas comigo. Obrigada por me apresentarem o mundo da pesquisa, por toda paciência e esforço que tiveram comigo. Admiro vocês como profissionais e pessoas, mais que orientadores, considero vocês amigos e serei sempre grata por tudo que fizeram por mim.

Aos professores Criston Pereira de Souza e Leliane Nunes de Barros pela disponibilidade em participar da banca deste trabalho e pelas excelentes colaborações e sugestões.

Às irmãs que a vida me deu, Daianny e Raquel, por todas as brincadeiras, conversas, aventuras, por estarem comigo em todas as fases da vida, e por, mesmo que distantes, sempre me apoiarem e me ajudarem quando precisei.

A todas as pessoas que eu morei nessa etapa da minha vida e fizeram a saudade de casa ser mais suportável, em especial ao Gustavo, por todos os conselhos, receitas e conversas, e a família Dantas e agregados, por me acolherem e por todo o apoio.

Agradeço à Késsia, por ter me acompanhado na luta diária, por todas as músicas, receitas e risadas. Obrigada por acreditar em mim, me ajudar e me motivar sempre que precisei. Foi incrível trabalhar e morar contigo.

Aos meus amigos de Iguatu, por todos os momentos que vivemos e por todo apoio, especialmente Thiago, Suênia e Maria Rita, e minha mais que cunhada, Heloíza. Por todas as aventuras, risadas e alegrias também. Levarei vocês para sempre.

Agradeço ao grupo “Sem bolsaminion”, Felipe, Iago, Iury, Rafaella e Raynara, por todas as risadas, conversas, fofocas e partidas. Agradeço também a Marianna e Lucas Henrique, pelos momentos que vivemos. Amigos, com vocês compartilhei as felicidades e os sofrimentos dessa jornada, obrigada por me motivarem e me darem forças para continuar, pela paciência e por acreditarem em mim.

A toda a minha turma de Engenharia de Computação, especialmente Assis, Emanuel e Victória, pelos momentos que passamos ao longo desses cinco anos de curso e por todos os trabalhos que desenvolvemos juntos.

Agradeço ao PACCE, por ter sido meu refúgio durante esses anos de graduação, e aos meus amigos da bolsa, por serem a melhor equipe de trabalho do mundo. Especialmente Kassiane e Patrícia, por todos os momentos que vivemos. E aos professores David Sena e Valdemir Queirós, por toda a amizade e orientação que me foram dadas. Foi uma experiência incrível trabalhar com todos vocês.

Ao colégio Ruy Barbosa e todos os funcionários, por terem feito parte na minha formação como pessoa e como profissional. Sempre guardarei o Ruy com carinho no coração.

Agradeço ao campus da UFC Quixadá que foi meu lar durante esses 5 anos. Aos professores e servidores do Campus da UFC Quixadá pelo comprometimento, dedicação e amor com que realizam suas tarefas. Obrigada pelos ensinamentos e apoio que me foram concedidos.

Agradeço a todos que fizeram parte deste ciclo da minha vida e que foram importantes para meu desenvolvimento profissional e pessoal.

“Nunca se esqueça de quem você é, porque é certo que o mundo não se lembrará. Faça disso sua força. Assim, não poderá ser nunca a sua fraqueza. Arme-se com esta lembrança, e ela nunca poderá ser usada para magoá-lo.”

(Tyrion Lannister)

RESUMO

Planejamento Automatizado é uma subárea da Inteligência Artificial que tem como objetivo o estudo do processo deliberativo de escolha de ações para que um agente inteligente possa atingir suas metas. A solução para um problema de planejamento determinístico é denominada *plano*, i.e., uma sequência de ações que leva o agente do estado inicial para um estado que satisfaça a meta. A busca por um plano pode ser feita de duas maneiras: progressiva, a partir do estado inicial, tentando alcançar algum estado que satisfaz a meta e; regressiva, a partir do conjunto de estados que satisfazem a meta, tentando alcançar o estado inicial. No entanto, produzir um plano é computacionalmente difícil e, frequentemente, ocorre o *problema da explosão do espaço de estados*. Na literatura de planejamento automatizado, existem duas abordagens para contornar este problema: (i) a busca heurística e; (ii) a busca simbólica, baseada na representação de estados e ações como Diagramas de Decisão Binária. Este trabalho propõe a implementação e avaliação de uma heurística baseada na busca regressiva simbólica para o algoritmo de busca simbólica A^* na tentativa de produzir planos para problemas de planejamento em domínios determinísticos.

Palavras-chave: Inteligência artificial. Planejamento automatizado. Busca heurística A^*. Diagramas de decisão binária.

ABSTRACT

Automated Planning is a subarea of Artificial Intelligence that aims to study the deliberative process of choosing actions so that an intelligent agent can achieve its goals. The solution to a deterministic planning problem is called *plan*, i.e., a sequence of actions that takes the agent from the initial state to a state that satisfies the goal. The search for a plan can be done in two ways: (i) progressive from the initial state, trying to reach some state that satisfies the goal and; (ii) regressive, from the set of states that satisfy the goal, trying to reach the initial state. However, producing a plane is computationally difficult and the *problem of state space explosion* often occurs. In the automated planning literature, there are two approaches to circumvent this problem: (i) heuristic search and; (ii) the symbolic search, based on the representation of states and actions as Binary Decision Diagrams. This paper proposes the implementation and evaluation of a symbolic regressive search-based heuristic for the A^* symbolic search algorithm in an attempt to produce plans for planning problems in deterministic domains.

Keywords: Artificial intelligence. Automated planning. Heuristic search A^*. Binary decision diagrams.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

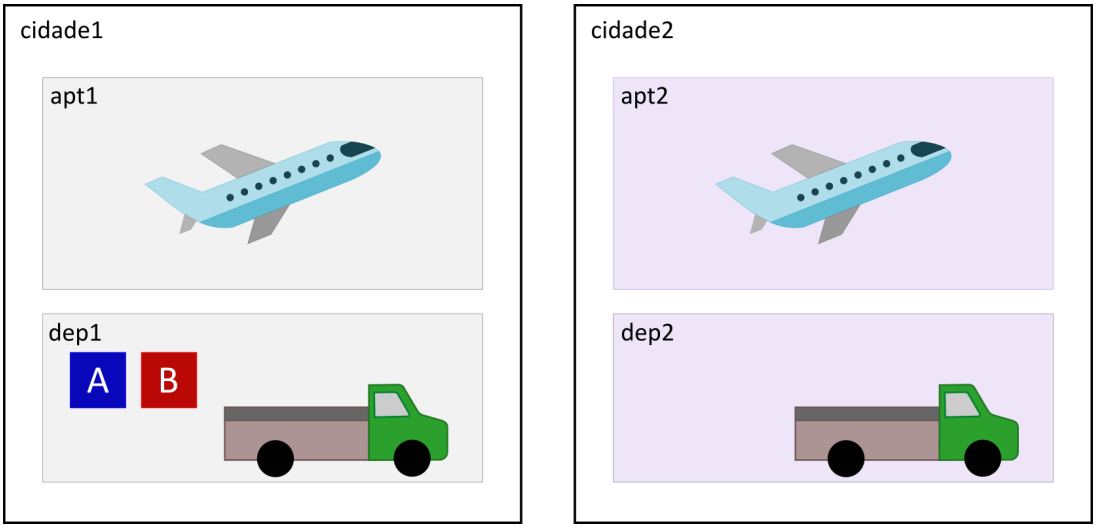
No mundo real constantemente deparamos-nos com a necessidade de realizar o planejamento de nossas ações, uma vez que esta tarefa nos possibilita perceber a realidade e encontrar caminhos para atingirmos, da melhor maneira possível, nossos objetivos.

Em Inteligência Artificial, *Planejamento Automatizado* é o estudo do processo deliberativo de escolha de ações para que um agente inteligente possa atingir suas metas (RUSSELL; NORVING, 2013). Um *problema de planejamento* é dado em termos: (i) do *domínio de planejamento*, i.e., uma descrição de como o ambiente é modificado com as ações do agente, (ii) do *estado inicial*, i.e., da situação inicial que o agente se encontra e; (iii) da *meta de planejamento*, uma propriedade que deve ser satisfeita em algum estado final.

O planejamento clássico supõe que as ações são determinísticas, isto é, não há incerteza sobre os efeitos das ações do agente (PEREIRA, 2007). A solução para um problema de planejamento clássico é denominada *plano*, i.e., uma sequência de ações que leva o agente do estado inicial para um estado que satisfaça a meta.

Exemplo 1.1 *(O Domínio de Logística) Considere um agente no domínio de Logística cuja tarefa é transportar um determinado número de pacotes de um local de origem para um local de destino. Essa tarefa é realizada em uma região contendo n cidades conectadas apenas por transporte aéreo. Cada cidade possui um conjunto de localizações conectadas por rodovias usadas por caminhões. (MENEZES, 2014).*

Figura 1 – Problema de logística com duas cidades



Fonte: Elaborada pela autora.

A Figura 1 ilustra um problema de planejamento de transporte dos pacotes *A* e *B* entre as cidades 1 e 2. Inicialmente, os pacotes *A* e *B* estão no depósito da $cidade_1$. A meta é entregar os pacotes no depósito da $cidade_2$. As ações neste domínio são: *carregar* pacotes dentro de caminhões e de aviões; descarregar pacotes; *transportar* um caminhão entre locais de uma mesma cidade e transportar aviões entre aeroportos de cidades. Um plano para este problema seria a sequência de ações: carregar os pacotes no caminhão do dep_1, transportá-los para o apt_1, descarregá-los do caminhão, carregá-los no avião, transportar o avião da $cidade_1$ para $cidade_2$, descarregar os pacotes no apt_2, carregá-los no caminhão que irá transportar estes para o dep_2. Neste caso, temos que todas as ações têm efeitos certos, ou seja, dependendo da ação escolhida, o próximo estado do problema é conhecido, não havendo imprevistos.

Produzir um plano é computacionalmente difícil. De fato, decidir se existe ou não um plano ou política é um problema PSPACE-Completo (BYLANDER, 1994). Por exemplo, no domínio de Logística, para um problema com 10 aeroportos, 50 cidades e 200 pacotes, temos um espaço de estados de aproximadamente 10^{15} estados. Chamamos este problema de *problema da explosão do espaço de estados* (HUTH; RYAN, 2004).

A busca por uma solução pode ser feita a partir do estado inicial, construindo os estados sucessores, até que um estado meta seja alcançado ou, a partir da meta, construindo os estados predecessores, até que o estado inicial seja alcançado. No entanto, a grande quantidade de estados a serem explorados no processo de busca por uma solução é inapropriada para métodos de busca exaustiva (ou busca sem informação) no espaço de estados. Na literatura de planejamento automatizado, as abordagens para contornar o problema da explosão do espaço de estados são, principalmente, duas:

- Busca com Informação ou Busca Heurística (HOFFMANN, 2001; HELMERT, 2006; RICHTER; WESTPHAL, 2008).
- Busca Simbólica (EDELKAMP; HELMERT, 2001; MENEZES, 2014; TORRALBA *et al.*, 2017).

Os algoritmos de busca com informação utilizam *funções heurísticas* para encontrar soluções de forma mais eficiente do que uma estratégia de busca sem informação (ou busca cega) (RUSSELL; NORVING, 2013). Uma *heurística* é uma estimativa da distância de um dado estado até um estado meta mais próximo (SANTOS, 2018). Em Inteligência Artificial, uma das formas mais conhecidas de busca de melhor escolha é chamada de *busca A** (RUSSELL; NORVING,

2013).

Em diversas edições da Competição Internacional de Planejamento (IPC - *Internacional Planning Competition*) (IPC, 2019), as abordagens de busca heurística obtiveram bons resultados e, na maioria dos anos, foram as abordagens campeãs da competição. Podemos citar como exemplos: o buscador FF (HOFFMANN, 2001) que venceu a IPC 2000; o FAST-DOWNWARD (HELMERT, 2006) que ganhou a IPC 2004; e o LAMA (RICHTER; WESTPHAL, 2008), vencedor da IPC 2008.

A busca simbólica visa contornar o problema de explosão do espaço de estados, representando os estados e transições do domínio *simbolicamente* como conjuntos. Nesta abordagem, estruturas de dados denominadas *Diagramas de Decisão Binária* (BDDs - *Binary Decision Diagrams*) (BRYANT, 1995) são utilizadas para representar os estados e transições codificados como fórmulas proposicionais. Um BDD é uma estrutura similar a uma árvore de decisão, mas que cujo poder de compactação permite a representação de conjuntos de estados com até 10^{20} estados (BURCH *et al.*, 1992).

O planejamento simbólico utilizando BDDs foi abordado primeiramente na área de verificação de modelos (MCMILLAN, 1993), sendo iniciado por (CIMATTI *et al.*, 1997) para o desenvolvimento de planejadores não-determinísticos. Mais recentemente, (TORRALBA *et al.*, 2017) propôs unificar a capacidade de compactação da representação de estados da busca simbólica em algoritmos clássicos de busca heurística, tais como a busca de custo uniforme e a busca simbólica A^*. Esta abordagem propõe a técnica de busca denominada BDDA* (EDELKAMP; REFFEL, 1998) que superou as técnicas de busca estado da arte na maioria dos problemas *benchmarks* da IPC em domínios determinísticos.

Apesar das vantagens que a busca simbólica apresenta sobre busca por estados de maneira explícita e seu sucesso em diferentes competições de inteligência artificial, tais como IPC, a área de busca simbólica ainda é pouco explorada (TORRALBA *et al.*, 2017).

1.1 Objetivos

1.1.1 Objetivo Geral

Propor, desenvolver e avaliar uma heurística baseada na busca regressiva simbólica para o algoritmo de busca simbólica A* para planejamento determinístico.

1.1.2 Objetivos Específicos

- Propor um cálculo heurístico a partir da representação simbólica do problema de planejamento;
- Implementar o algoritmo proposto;
- Comparar o desempenho do algoritmo simbólico de busca exaustiva com o algoritmo simbólico de busca heurística proposto em problemas *benchmarks*.

1.2 Organização

Este trabalho está organizado conforme segue: o Capítulo 2 apresenta os conceitos teóricos relacionados a planejamento clássico, como são realizadas a busca heurística e a busca simbólica, como é possível representar de maneira eficiente os domínios de planejamento e, também, como são gerados os estados sucessores e predecessores; o Capítulo 3 apresenta os trabalhos que possuem relação e foram utilizados como base para o desenvolvimento desta pesquisa e; o Capítulo 4 apresenta a principal contribuição deste trabalho, que é a proposta da busca heurística simbólica para problemas de planejamento determinístico; o Capítulo 5 contém a apresentação dos domínios utilizados para a análise experimental e quais os resultados obtidos nos testes e; por fim, no Capítulo 6 apresentamos as conclusões deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Planejamento Clássico

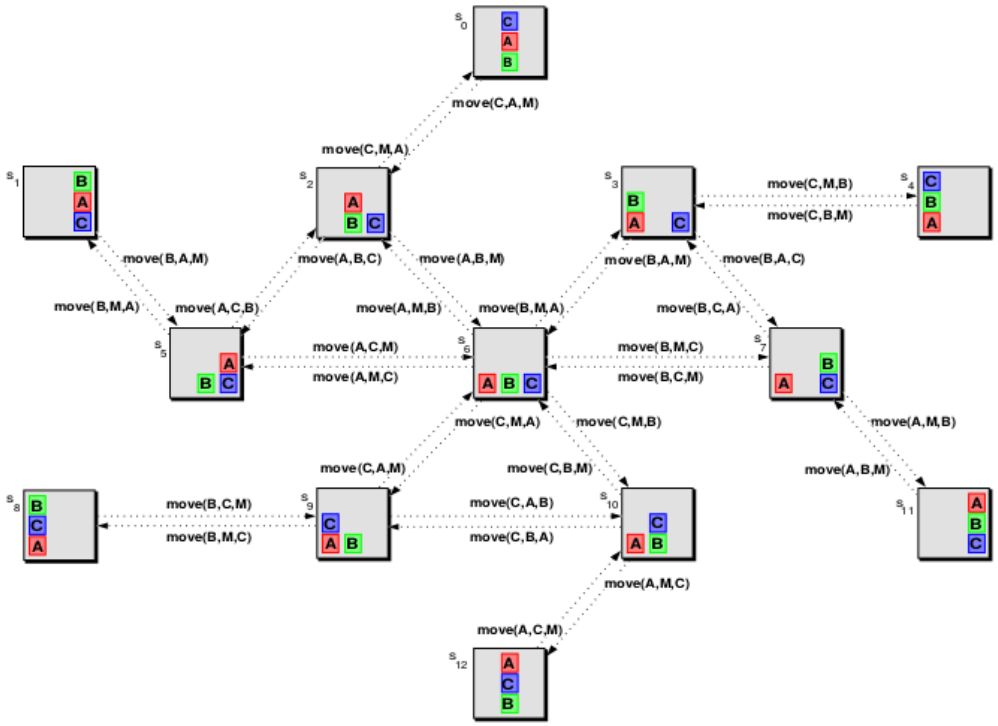
O planejamento clássico supõe que o ambiente evolui de forma *determinística*, ou seja, não há efeitos incertos nas ações executadas pelo agente e o ambiente é alterado por consequência apenas destas ações.

Definição 2.1. (*Domínio de Planejamento*) Um domínio de planejamento clássico é definido por um sistema de transição de estados dado pela tupla $D = \langle S, L, T \rangle$ sobre um conjunto de átomos proposicionais $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, em que:

- $S \neq \emptyset$ é um conjunto finito e enumerado de estados;
- $L \subseteq 2^{\mathbb{P}}$ é uma função de rotulação de estados;
- $T : S \text{ x } A \mapsto S$ é uma função de transição de estados rotulada pelas ações $\mathbb{A}$.

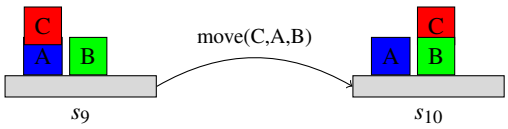
A Figura 2 mostra o sistema de transição de estados representando o domínio do *mundo dos blocos* (NILSSON, 1980). Neste domínio um conjunto de blocos estão dispostos sobre uma mesa. É possível empilhar um bloco livre (que não tenha um outro bloco em cima) sobre outro bloco livre ou sobre a mesa.

Figura 2 – Sistema de transição de estados para o domínio do mundo dos blocos.



Fonte: (PEREIRA, 2007)

Figura 3 – Dois estados s_9 e s_{10} e uma transição rotulada por ação no domínio do mundo dos blocos



Fonte: Elaborada pela autora.

Os estados s_0, s_1, $\cdots$, s_{12} representam as possíveis configurações de três blocos A, B e C sobre uma mesa. Os átomos proposicionais que descrevem um estado no mundo dos blocos são:

- limpo(X): para indicar que o bloco X encontra-se limpo, ou seja, nenhum bloco está sobre ele.
- sobre(X, Y): para indicar que o bloco X encontra-se sobre o bloco Y.
- na-mesa(X): para indicar que o bloco X está sobre a mesa.

Na Figura 3, temos a representação do estado s_9 do domínio do mundo dos blocos. Este estado pode ser representado pelo seguinte conjunto de átomos proposicionais: $L(s_9) = \{$ limpo(B), limpo(C), sobre(C,A), na-mesa(A), na-mesa(B)$\}$. Note que representamos apenas os átomos proposicionais que são verdadeiros no estado e as proposições com valores verdade falso não são representadas (suposição do mundo fechado (RUSSELL; NORVING, 2013)). As transições entre estados são rotuladas com as ações do agente que causam tais transições. Por exemplo, na Figura 3 a ação `move(C,A,B)` causa a transição do estado s_9 para o estado s_{10}.

No entanto, a descrição completa do sistema de transição de estados não é apropriada para a maioria dos domínios reais. Para representar o mundo dos blocos com 10 blocos, por exemplo, seriam necessários 58.941.091 estados (GHALLAB *et al.*, 2004). Assim, a comunidade de planejamento automatizado desenvolveu uma forma de representar os domínios por meio de *linguagens de ações* tais como STRIPS (*Stanford Research Institute Problem Solver*) (FIKES; NILSSON, 1971) e PDDL (*Planning Domain Description Language*) (MCDERMOTT *et al.*, 1998).

Nestas linguagens, uma **ação** (GHALLAB *et al.*, 2004) é dada por meio de *pre-condições* e *efeitos*, que são conjuntos dos átomos proposicionais. As precondições de uma ação a, denotada por *precond*(a), definem o conjunto dos átomos que devem ser verdadeiros em um estado, para que a ação seja executada. Os efeitos definem o resultado desta execução, descrevendo apenas o que é alterado no mundo pela execução da ação; dessa forma, tudo que não é modificado pelas ações não é mencionado nos efeitos. Os efeitos de uma ação a são

divididos em dois conjuntos: efeitos^{+}(a) é o conjunto dos átomos que se tornam verdadeiros com a execução de *a* em um estado, e efeitos^{-}(a) é o conjunto dos átomos que se tornam falsos com a execução de *a* em um estado.

Considere, por exemplo, a ação `move(C,A,B)` que é responsável por mover o *bloco C* que está sobre o *bloco A* para cima do *bloco B*. Esta ação pode ser descrita na linguagem STRIPS, conforme mostrado na Figura 4. As pré-condições desta ação são: $limpo(C), sobre(C,A), limpo(B)$. Os efeitos$^{+}(a)$ são $sobre(C,B), limpo(A)$ e os efeitos$^{-}(a)$ são $sobre(C,A), limpo(B)$.

Figura 4 – A ação STRIPS `move(C,A,B)`.

```
mover(C,A,B)
    precond:  { limpo(C), sobre(C,A), limpo(B) }
    efeitos⁺(a):  {sobre(C, B), limpo(A) }
    efeitos⁻(a):  {sobre(C, A)), limpo(B) }
```

Fonte: Elaborada pela autora.

Definição 2.2. (*Problema de Planejamento*) Dado um domínio de planejamento sobre um conjunto de proposições $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$ (com precondições e efeitos), um problema de planejamento é definido por:

- Um estado inicial $s_0 \in S$
- Uma fórmula proposicional φ representando a meta a ser alcançada. Denotamos por G o conjunto de estados que satisfazem a meta de planejamento.

No caso do *mundo dos blocos*, podemos definir um problema de planejamento em que s_9 é o estado inicial e a meta é ter *o bloco A sobre o bloco B* (i.e., φ = sobre(A,B)). Assim, $G = \{s_0, s_2, s_{11}\}$, isto é, o conjunto de todos os estados em que a meta é verdade.

Um *plano* é a sequência de ações que precisam ser realizadas para que o agente consiga sair do estado inicial e alcance um estado meta. No problema em que s_9 é o estado inicial e a meta é ter *o bloco A sobre o bloco B*, um plano é a sequência de ações *move(C,A,M), move(A,B,M)*, levando o agente de s_9 para o estado s_2 que satisfaz a meta.

2.2 Busca Progressiva por um Plano Solução

A busca progressiva por um plano solução inicia-se a partir de um estado inicial, gerando-se em cada passo, estados sucessores até que um estado meta seja alcançado. É importante notar que o sistema de transição de estados (como por exemplo o da Figura 2) é *gerado sob demanda* por meio das ações a partir de um dado estado inicial.

A geração de estados sucessores é feita por meio da operação de progressão. A partir de um dado estado s, verifica-se que ações podem ser *aplicadas*, isto é, para cada ação a verifica-se se $precond(a) \subseteq L(s)$. O estado sucessor de um estado s por meio de uma ação a é computado da seguinte forma: para cada ação aplicável em s, remove-se os efeitos negativos da ação no estado e adiciona-se os efeitos positivos desta mesma ação no estado s.

Formalmente, a operação de *progressão* de um estado s por uma ação a, computa um estado sucessor s' como a seguir:

$$L(s') = progr^{a}(s) = \begin{cases} (L(s) \setminus efeitos^{-}(a)) \cup efeitos^{+}(a) \; se \; precond(a) \subseteq L(s) \\ \varnothing \quad c.c. \end{cases} \tag{2.1}$$

A partir do estado s_9 podemos verificar de todo o conjunto de ações $\mathbb{A}$ quais são ações aplicáveis em s_9, isto é, para quais ações $a \in \mathbb{A}$ temos $precond(a) \subseteq L(s_9)$. Neste estado podemos aplicar: a ação $move(C, A, M)$ que move o *bloco C* que está sobre o *bloco A* para cima da *mesa*, a ação $move(C, A, B)$ que move o *bloco C* que está sobre o *bloco A* para cima do *bloco B*, e a ação $move(B, M, C)$ que move o *bloco B* que está sobre a *mesa* para cima do *bloco C*. A seguir, observe como a ação `move(C,A,B)`, aplicada em s_9, gera o estado sucessor s_{10}.

$$\begin{aligned} L(s_{10}) =& L(s_9) \setminus \text{efeitos}^{-}(move(C,A,B)) \cup \text{efeitos}^{+}(move(C,A,B)) \\ =& \{limpo(B), limpo(C), sobre(C,A), \text{na-mesa(A)}, \text{na-mesa(B)}\} \setminus \\ & \{sobre(C,A), limpo(B)\} \cup \{sobre(C,B), limpo(A)\} \\ =& \{limpo(C), \text{na-mesa(A)}, \text{na-mesa(B)}\} \cup \{sobre(C,B), limpo(A)\} \\ =& \{limpo(C), \text{na-mesa(A)}, \text{na-mesa(B)}, sobre(C,B), limpo(A)\} \end{aligned}$$

2.3 Busca Regressiva por um Plano Solução

A busca regressiva por um plano solução inicia-se a partir de um estado meta, gerando-se em cada passo, um conjunto de estados predecessores até que o estado inicial seja alcançado. A geração de estados predecessores é feita por meio da operação de regressão. Um conceito muito importante na regressão é a relevância de uma ação para um estado. Para que uma ação a seja relevante em um estado s, ela deve contribuir com as propriedades satisfeitas em s, i.e., no mínimo um dos efeitos da ação deve pertencer ao estado (RUSSELL; NORVING, 2013), e também a ação não deve ter um efeito que negue um literal do estado s.

Assim, dado um estado s, uma ação a é relevante para s se $efeitos^{+}(a) \cap s \neq \varnothing$ e $efeitos^{-}(a) \cap s = \varnothing$. O conjunto de estados predecessores do estado s pela ação a é computado da

seguinte forma: remove-se os efeitos positivos da ação no estado e adiciona-se as precondições, i.e., $(L(s) \subseteq efeitos^{-}(a)) \cup precond(a)$ (GHALLAB *et al.*, 2004). Caso a condição de relevância não seja satisfeita, não existe estado predecessor.

Formalmente, a operação de *regressão* de um estado s por uma ação a, computa um estado predecessor s' como a seguir:

$$L(s') = regr^{a}(s) = \begin{cases} (L(s) \setminus efeitos^{+}(a)) \cup precond(a) \text{ se } & efeitos^{+}(a) \cap L(s) \neq \varnothing \text{ e} \\ & efeitos^{-}(a) \cap L(s) = \varnothing. \\ \varnothing \text{ c.c.} & \end{cases} \quad (2.2)$$

A seguir, observe como a ação `move(C,A,B)`, aplicada em s_{10}, gera o estado predecessor s_9:

$$\begin{aligned} L(s_9) =& L(s_{10}) \setminus \text{efeitos}^{+}(move(C,A,B)) \cup \text{precond}(move(C,A,B)) \\ =& \{limpo(C), \text{na-mesa(A)}, \text{na-mesa(B)}, sobre(C,B), limpo(A\} \setminus \\ & \{sobre(C,B), limpo(A)\} \cup \{limpo(C), sobre(C,A), limpo(B)\} \\ =& \{limpo(C), \text{na-mesa(A)}, \text{na-mesa(B)}\} \cup \{limpo(C), sobre(C,A), limpo(B)\} \\ =& \{limpo(B), limpo(C), sobre(C,A), \text{na-mesa(A)}, \text{na-mesa(B)}\} \end{aligned}$$

Como dito anteriormente, os estados sucessores são gerados a partir do estado inicial até que um estado satisfazendo a meta seja alcançado, e os estados predecessores são gerados a partir da meta até que o estado inicial seja alcançado. No entanto, como na maioria dos problemas reais, o espaço de busca possui proporções astronômicas, tanto a representação individual de cada estado e transição gerada, como o uso de estratégias de busca exaustivas não são apropriadas para obtenção de planos em problemas reais. De fato, decidir se um plano existe é um problema computacionalmente difícil, sendo da classe de complexidade PSPACE-Completo (BYLANDER, 1994).

Na literatura de planejamento, as principais abordagens para lidar com a grande quantidade de estados gerados na busca são duas:

- Busca com Informação ou Busca Heurística (HOFFMANN, 2001; HELMERT, 2006; RICHTER; WESTPHAL, 2008).
- Busca Simbólica (EDELKAMP; HELMERT, 2001; MENEZES, 2014; TORRALBA *et al.*, 2017).

2.3.1 Busca Heurística

Para os casos em que não é possível a expansão completa (exaustiva) do espaço de estados, é necessário utilizar formas "espertas" de expansão dos estados, com alguma forma de guia de escolha de caminhos. As estratégias de *busca heurística* (RUSSELL; NORVING, 2013) conseguem diferenciar se um estado a ser expandido é "mais promissor" que outro. A *busca heurística* tem sido utilizada na maioria dos planejadores estado da arte desde as primeiras edições da Competição Internacional de Planejamento (IPC - *International Planning Competition*) (IPC, 2019).

Uma *heurística* é uma função que estima a distância de um dado estado s até um estado objetivo mais próximo (SANTOS, 2018). Funções heurísticas são uma forma de transmitir um conhecimento adicional do problema ao algoritmo de busca. O desafio de propor heurísticas para planejamento automatizado é que estas devem ser independentes do domínio de aplicação, sendo baseada apenas na estrutura lógica do problema.

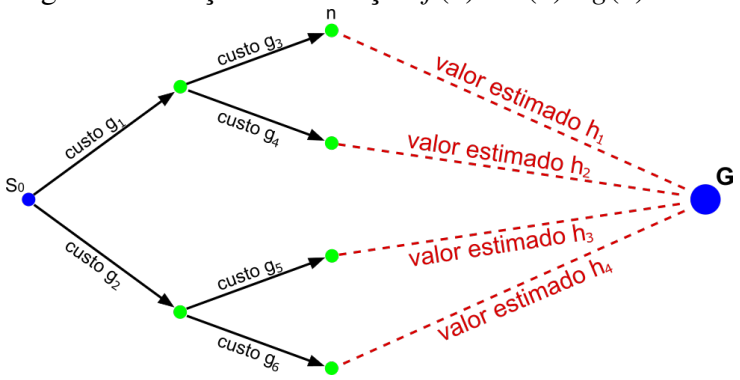
A abordagem geral de busca heurística é denominada *busca de melhor escolha* e considera a expansão de um nó[1] n na árvore de busca com base em uma função de avaliação $f(n)$: o nó com a menor avaliação será expandido primeiro. Uma possível abordagem é a denominada *busca gulosa de melhor escolha* que tenta expandir o nó que está mais próximo do objetivo, isto é, o nó com a menor função de heurística $h(n)$. Desta forma, a função de avaliação é baseada apenas na estimativa heurística, i.e., $f(n) = h(n)$. Já a abordagem mais amplamente conhecida da busca de melhor escolha é chamada *busca* A^* (lê-se busca A estrela). Na busca A*, a função de avaliação $f(n)$ de um nó n combina o valor $h(n)$ de distância de um nó para o objetivo com o valor do custo $g(n)$ de se alcançar o nó n a partir do estado inicial. Assim sendo, em cada passo da busca, o nó com melhor valor de $f(n) = g(n) + h(n)$ é escolhido para ser expandido.

Na Figura 5, por exemplo, o custo para atingir o nó n é dado por $g_1 + g_3$ e a estimativa para alcançar a meta G é representada pelo valor h_1. Dessa forma, a função de avaliação pode ser descrita por: $f(n) = g_1 + g_3 + h_1$.

Desde que a heurística seja admissível, a busca A^* é completa e ótima. Uma heurística admissível é aquela que nunca superestima o custo de atingir o objetivo. Heurísticas admissíveis são otimistas por natureza porque imaginam que o custo de resolver o problema seja menor do que realmente é (RUSSELL; NORVING, 2013).

[1] Um nó é um componente da árvore de busca. Já um estado é uma configuração do mundo, ou seja, os estados não estão em caminhos particulares, podendo um estado estar contido em dois nós diferentes (RUSSELL; NORVING, 2013)

Figura 5 – Função de avaliação $f(n) = h(n) + g(n)$.



Fonte: Adaptada de (RUSSELL; NORVING, 2013).

Em planejamento automatizado, uma abordagem muito utilizada para calcular a estimativa é o relaxamento do problema. Dessa forma, com o intuito de avaliar quão desejável é um nó n, considera-se um problema mais simples que é obtido do problema real, a partir de suposições simplificadoras (GHALLAB *et al.*, 2004). Assim, uma heurística admissível pode ser obtida através do problema relaxado, em que o custo de uma solução ótima para o problema relaxado torna-se a heurística do problema real (RUSSELL; NORVING, 2013). Uma forma muito comum de relaxamento para a busca em planejamento determinístico é desconsiderar os efeitos negativos de uma ação.

2.3.2 Busca Simbólica

A busca simbólica contorna o problema da explosão do espaço de estados, utilizando a representação de conjuntos de estados no lugar de representar estados individuais. Esse tipo de representação é denominada *representação simbólica do espaço de estados* (HUTH; RYAN, 2004). Além disso, este tipo de busca é capaz de raciocinar sobre a representação simbólica, o que chamamos de *raciocínio simbólico*.

Na busca simbólica, estruturas de dados eficientes chamadas *Diagramas de Decisão Binária* (BDDs - *Binary Decision Diagramas*) (BRYANT, 1995) são utilizadas para representação de conjunto de estados e ações e raciocínio sobre tal representação. A busca simbólica tem sido um dos tópicos de interesse na pesquisa em planejamento automatizado, uma vez que planejadores baseados em busca simbólica foram os vencedores das últimas competições internacionais de planejamento (EDELKAMP *et al.*, 2015).

2.3.2.1 Diagrama de Decisão Binária

Diagramas de Decisão Binária (BDDs) permitem uma representação compacta de funções proposicionais as quais só possuem representações exponenciais em outros sistemas tais como tabelas-verdades e formas normais conjuntivas (HUTH; RYAN, 2004).

BDDs são grafos acíclicos, finitos e direcionados com um único nó inicial, em que os *nós internos* são rotulados com átomos proposicionais e os *nós terminais* são rotulados com os valores-verdade 0 (falso) e 1 (verdadeiro). Cada nó interno n possui duas arestas saindo: uma aresta pontilhada, representando que o átomo proposicional correspondente ao nó possui valor 0 e; uma aresta sólida, representando que o átomo proposicional correspondente ao nó possui valor 1. A Figura 6(a) demonstra a árvore de decisão que representa as possíveis valorações para a fórmula proposicional $(p_1 \vee p_2) \wedge (p_2 \vee p_3)$. Nesta fórmula, por exemplo, quando p_1 é 0 (aresta pontilhada saindo de p_1) e p_2 é 0 (aresta pontilhada saindo de p_2, nó mais a direita), a fórmula possui valor verdade 0 (nó terminal zero mais a direita).

No entanto, há muita redundância nestas estruturas e as simplificações são realizadas de forma a torná-las mais compactas:

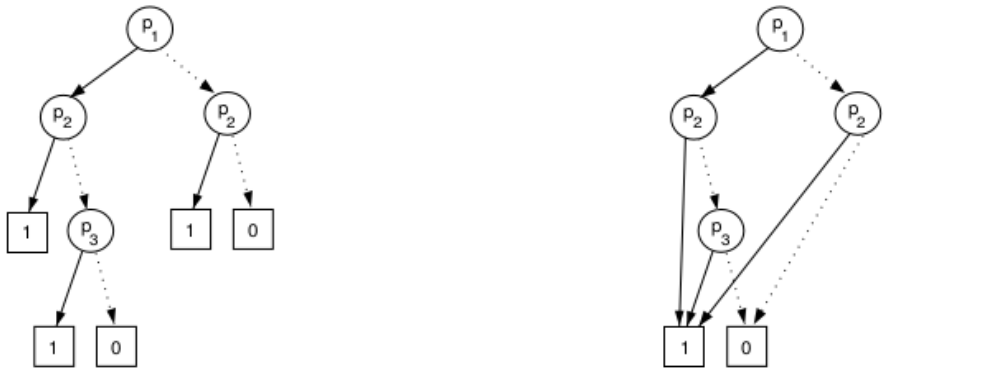
- **R1**: Eliminação de nós terminais redundantes;
- **R2**: Eliminação de testes redundantes;
- **R3**: Eliminação de nós internos redundantes.

Ao realizar todas as possíveis remoções **R1**, **R2** e **R3**, dizemos que o BDD está na forma reduzida, RBDD - *Reduced BDD*. Outra importante característica para uma representação compacta é que haja uma ordenação nos átomos proposicionais dispostos ao longo de qualquer caminho do BDD. Assim, se uma ordem dos átomos for mantida para todos os caminhos do BDD, dizemos que o BDD é ordenado, OBDD - *Ordered BDD*.

Neste trabalho, usaremos os BDDs ordenados e reduzidos - ROBDDs. Para uma dada fórmula proposicional e para uma determinada ordem nas variáveis, os ROBDDs possuem uma forma canônica, isto é, após realizar todas as possíveis simplificações, há uma representação única desta estrutura não importa em que ordem as reduções **R1**, **R2** e **R3** foram realizadas (HUTH; RYAN, 2004). Por simplicidade, no restante deste trabalho chamaremos os ROBDDs apenas de BDDS.

A Figura 6(b) ilustra um BDD para a fórmula proposicional $(p_1 \vee p_2) \wedge (p_2 \vee p_3)$ após aplicadas as possíveis reduções na estrutura da Figura 6(a). Neste caso, apenas a redução R1 (remoção de folhas redundantes) foi aplicada.

Figura 6 – Diagramas de decisão binária.



(a) árvore de decisão binária *(b) diagrama de decisão binária*

Fonte: (PEREIRA, 2007).

2.3.2.2 Representação Simbólica de Estados e Ações

Nesta seção, mostraremos como codificar estados, conjuntos de estados, ações e conjunto de ações de um domínio de planejamento como fórmulas da lógica proposicional.

Codificação de um estado: Um estado $s \in S$ pode ser representado como uma conjunção de todos os átomos proposicionais que estão definidos da função rotulação de s, $L(s)$, e com a conjunção da negação de todos os átomos proposicionais não definidos em $L(s)$. A codificação proposicional de um estado $s \in S$ é denotada pela fórmula (HUTH; RYAN, 2004):

$$\xi(s) = \bigwedge_{p \in L(s)} p \wedge \bigwedge_{q \in \mathbb{P} \setminus L(s)} \neg q \tag{2.3}$$

Considere, por exemplo, o estado s_9 do mundo dos blocos (Figura 3) em que $L(s_9) = \{limpo(B), limpo(C), sobre(C,A), \text{na-mesa(A)}, \text{na-mesa(B)}\}$. Este estado pode ser representado pela fórmula:

$$\begin{aligned} \xi(s_9) = & limpo(B) \wedge limpo(C) \wedge sobre(C,A) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \\ & \neg limpo(A) \wedge \neg \text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \\ & \neg sobre(B,C) \wedge \neg sobre(C,B) \wedge \neg sobre(B,A) \end{aligned} \tag{2.4}$$

Codificação de conjunto de estados: Um conjunto de estados $X \subseteq S$ é representado simbolicamente como uma disjunção das fórmulas proposicionais representando os estados pertencentes a X, dada pela seguinte fórmula:

$$\xi(X) = \bigvee_{s \in X} \xi(s) \tag{2.5}$$

Considere, por exemplo, o conjunto de estados do mundo dos blocos (Figura 3) $X = \{s_9, s_{10}\}$, que pode ser representado pela fórmula.

$$\begin{aligned}\xi(X) = &\{limpo(B) \wedge limpo(C) \wedge sobre(C,A) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \\ &\neg limpo(A) \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \\ &\neg sobre(B,C) \wedge \neg sobre(C,B) \wedge \neg sobre(B,A)\} \vee \\ &\{limpo(A) \wedge limpo(C) \wedge sobre(C,B) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \\ &\neg limpo(B) \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \\ &\neg sobre(B,C) \wedge \neg sobre(C,A) \wedge \neg sobre(B,A)\}\end{aligned}$$

Codificação de ações: Uma ação $a = \langle precond(a), efeitos(a)\rangle$ é representada simbolicamente como sendo um par de fórmulas $a = \langle \xi(precond(a)); \xi(efeitos(a))\rangle$ onde:

- $\xi(precond(a))$ é um literal ou uma conjunção de literais que representam as precondições da ação a:

$$\xi(precond(a)) = \bigwedge_{p \in precond(a)} p \tag{2.6}$$

  e,

- $\xi(efeitos(a))$ é um literal ou uma conjunção de literais que representam os efeitos$^+(a)$ e os efeitos$^-(a)$:

$$\xi(efeitos(a)) = \bigwedge_{q \in efeitos^+(a)} q \quad \wedge \quad \bigwedge_{r \in efeitos^-(a)} \neg r \tag{2.7}$$

Como exemplo, mostraremos a codificação da ação $move(C,A,B)$ representada na Figura 3, sendo representada pela fórmula:

$move(C,A,B) = \langle \xi(precond(move(C,A,B))), \xi(efeitos(move(C,A,B)))\rangle$

Onde,

$\xi(precond(a)) = limpo(C) \wedge sobre(C,A) \wedge limpo(B)$,

$\xi(efeitos(a)) = sobre(C,B) \wedge limpo(A) \wedge \neg sobre(C,A) \wedge \neg limpo(B)$

Adicionalmente é definido para cada ação $a \in \mathbb{A}$ o conjunto $modifica(a)$ contendo todos os átomos proposicionais presentes nos efeitos da ação a. Por exemplo, para a ação $move(C,A,B)$ temos:

$modifica(a) = \{sobre(C,A), limpo(B), sobre(C,B), limpo(A)\}$

2.3.2.3 Raciocínio Simbólico sobre Ações

Nesta seção, mostraremos como, a partir da representação simbólica dos estados e ações, os estados sucessores são gerados.

2.3.2.3.1 Fórmulas Booleanas Quantificadas

Para a realização do raciocínio simbólico, utilizamos operações baseadas na lógica booleana quantificada (QBF - *Quantified Boolean Formulas*) (BUNING *et al.*, 1995), que é uma extensão da lógica proposicional com quantificadores existencial e universal sobre os valores verdades dos átomos proposicionais.

A quantificação existencial dos valores de um átomo proposicional p em uma fórmula φ é definida como a seguir (BUNING *et al.*, 1995):

$$\exists p.\varphi \equiv \varphi[\top/p] \vee \varphi[\bot/p] \tag{2.8}$$

Em que $\varphi[\top/p]$ é a fórmula resultante de se substituir cada ocorrência de p na fórmula φ por $\top$ e $\varphi[\bot/p]$ é a fórmula resultante de se substituir cada ocorrência de p na fórmula φ por $\bot$.

A quantificação universal dos valores de um átomo proposicional p em uma fórmula φ é definida como a seguir (BUNING *et al.*, 1995):

$$\forall p.\varphi \equiv \varphi[\top/p] \wedge \varphi[\bot/p] \tag{2.9}$$

Exemplo 2.1 *Seja $\varphi = (p \wedge q) \vee r$ uma fórmula proposicional sobre o conjunto de proposições $\mathbb{P}$ = {p,q,r}, a quantificação existencial da fórmula φ sobre os valores da proposição p é dada por:*

$$\begin{aligned}
\exists p.\varphi \equiv \quad & \varphi[\top/p] \vee \varphi[\bot/p] \\
\equiv\ & ((\top \wedge q) \vee (\bot \wedge q)) \vee r \\
\equiv\ & (q \vee \bot) \vee r \\
\equiv\ & q \vee r
\end{aligned}$$

A quantificação também pode ser definida para um conjunto de proposições. Dado um conjunto de proposições $\mathbb{P} = \{p_1, p_2, ...p_n\}$, que ocorrem em φ, temos que:

$\exists \mathbb{P}.[\varphi] = \exists p_1.(...\exists p_n.\varphi)$ e $\forall \mathbb{P}.[\varphi] = \forall p_1.(...\forall p_n.\varphi)$

2.3.2.3.2 Progressão Simbólica de Ações Determinísticas

Dado um conjunto de estados X e um conjunto de ações determinísticas A, representadas por fórmulas QBF, (FOURMAN, 2000) propôs uma forma de computar os estados sucessores a partir da especificação de ações com pré-condições e efeitos, uma operação denominada progressão simbólica.

A progressão simbólica do conjunto X por uma ação a devolve um conjunto de estados Y que pode ser alcançando ao executarmos a ação a em estados de X. A progressão de X pela ação a pode ser expressa por (FOURMAN, 2000):

$$\xi(progr(X,a)) = \exists modifica(a).(\xi(X) \wedge \xi(precond(a))) \wedge \xi(efeitos(a)) \tag{2.10}$$

A Equação 2.10 é a versão simbólica da Equação 2.1 para cálculo de estados sucessores. A fórmula $\xi(progr(X,a))$ representa o conjunto Y de estados alcançáveis a partir da aplicação da ação a em estados do conjunto X. Para isso é necessário: (i) verificar a aplicabilidade da ação a no conjunto de estados X por meio da conjunção de $\xi(X)$ com ξ(precond(a)); (ii) eliminar as variáveis presentes nos efeitos da ação a por meio da aplicação da quantificação existencial das variáveis em $modifica(a)$ e; (iii) realizar a conjunção desta fórmula com os efeitos da ação a.

Utilizaremos o exemplo da Figura 3, onde $a = move(C,A,B)$ e o conjunto $\xi(X)$ pode ser descrito pela fórmula 2.4, o primeiro passo é calcular $\xi(X) \wedge \xi(precond(a))$, de tal forma:

$$\begin{aligned}
\xi(progr(X,a)) = \exists modifica(a).[&(limpo(B) \wedge limpo(C) \wedge sobre(C,A) \wedge \\
&\text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg limpo(A) \wedge \neg\text{na-mesa(C)} \wedge \\
&\neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(C,B) \wedge \\
&\neg sobre(B,A)) \wedge limpo(C) \wedge sobre(C,A) \wedge limpo(B)] \wedge \xi(efeitos(a)) \\
= \exists modifica(a).[&(limpo(B) \wedge limpo(C) \wedge sobre(C,A) \wedge \\
&\text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg limpo(A) \wedge \neg\text{na-mesa(C)} \wedge \\
&\neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(C,B) \wedge \\
&\neg sobre(B,A))] \wedge \xi(efeitos(a))
\end{aligned}$$

O segundo passo é realizar a aplicação da quantificação existencial das variáveis em $modifica(a)$:

$$\xi(progr(X,a)) = \exists sobre(C,A).\exists limpo(B).\exists sobre(C,B).\exists limpo(A)[(limpo(B)\wedge$$
$$limpo(C) \wedge sobre(C,A) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg limpo(A)\wedge$$
$$\neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C)\wedge$$
$$\neg sobre(C,B) \wedge \neg sobre(B,A))] \wedge \xi(efeitos(a))$$

Por fim, o terceiro passo é realizar a conjunção com os $efeitos(a)$:

$$\xi(progr(X,a)) = [(limpo(C) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg\text{na-mesa(C)}\wedge$$
$$\neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(B,A))]\wedge$$
$$[sobre(C,B) \wedge limpo(A) \wedge \neg sobre(C,A) \wedge \neg limpo(B)]$$
$$= (limpo(A) \wedge \neg limpo(B) \wedge limpo(C) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)}\wedge$$
$$\neg\text{na-mesa(C)} \wedge sobre(C,B) \wedge \neg sobre(A,C) \wedge \neg sobre(A,B)\wedge$$
$$\neg sobre(B,C) \wedge \neg sobre(B,A) \wedge \neg sobre(C,A)) = \{s_{10}\}$$

2.3.2.3.3 Regressão de Ações Determinísticas

Dado um conjunto de estados X e um conjunto de ações determinísticas A, representadas por fórmulas QBF, (FOURMAN, 2000) propôs uma forma de computar os estados predecessores a partir da especificação de ações com pré-condições e efeitos, uma operação denominada regressão simbólica.

A regressão simbólica computa o conjunto de estados predecessores que leva a estados pertencentes ao conjunto X. A regressão de X pela ação a pode ser expressa por (FOURMAN, 2000):

$$\xi(regr(X,a)) = \xi(precond(a)) \wedge \exists modifica(a).(\xi(efeitos(a)) \wedge \xi(X)) \tag{2.11}$$

A Equação 2.11 é a versão simbólica da Equação 2.2 para cálculo de estados predecessores. A fórmula $\xi(regr(X,a))$ representa o conjunto Y de estados predecessores a partir da aplicação da ação a em estados do conjunto X. Para isso é necessário: (i) eleger o subconjunto X de estados alcançados pelos efeitos de a por meio da conjunção dos efeitos da ação com $\xi(X)$. Se algum efeito de a é relevante para algum estado X, então $\xi(efeitos(a)) \wedge \xi(X) \neq \perp$; (ii) eliminar as variáveis presentes nos efeitos da ação a por meio da aplicação da quantificação existencial das variáveis em $modifica(a)$ e; (iii) realizar a conjunção desta fórmula com ξ(precond(a)).

Utilizaremos o exemplo da Figura 3, onde $a = move(C,A,B)$ e o conjunto $\xi(X)$ pode ser descrito pela fórmula do estado s_{10}:

$$\xi(s_{10}) = \{limpo(A) \wedge limpo(C) \wedge sobre(C,B) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge$$
$$\neg limpo(B) \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge$$
$$\neg sobre(B,C) \wedge \neg sobre(C,A) \wedge \neg sobre(B,A)\}$$

O primeiro passo é calcular $\xi(efeitos(a)) \wedge \xi(X)$, verificando que a é relevante para s_{10}, de tal forma:

$$\xi(regr(X,a)) = \xi(precond(a)) \wedge \exists modifica(a).([sobre(C,B) \wedge limpo(A) \wedge \neg sobre(C,A)$$
$$\wedge \neg limpo(B)] \wedge \{limpo(A) \wedge limpo(C) \wedge sobre(C,B) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)}$$
$$\wedge \neg limpo(B) \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge$$
$$\neg sobre(C,A) \wedge \neg sobre(B,A)\})$$
$$= \xi(precond(a)) \wedge \exists modifica(a).(limpo(A) \wedge limpo(C) \wedge sobre(C,B) \wedge$$
$$\text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg limpo(B) \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge$$
$$\neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(C,A) \wedge \neg sobre(B,A))$$

O segundo passo é realizar a aplicação da quantificação existencial das variáveis em $modifica(a)$:

$$\xi(regr(X,a)) = \xi(precond(a)) \wedge \exists sobre(C,A).\exists limpo(B).\exists sobre(C,B).\exists limpo(A).[(limpo(A)$$
$$\wedge limpo(C) \wedge sobre(C,B) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge \neg limpo(B) \wedge$$
$$\neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(C,A)$$
$$\wedge \neg sobre(B,A))]$$

Por fim, o terceiro passo é realizar a conjunção com os $precond(a)$:

$$\xi(regr(X,a)) = limpo(C) \wedge sobre(C,A) \wedge limpo(B) \wedge [(limpo(C) \wedge \text{na-mesa(A)} \wedge$$
$$\text{na-mesa(B)} \wedge \neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge$$
$$\neg sobre(B,C) \wedge \neg sobre(B,A))]$$
$$= sobre(C,A) \wedge limpo(B) \wedge limpo(C) \wedge \text{na-mesa(A)} \wedge \text{na-mesa(B)} \wedge$$
$$\neg\text{na-mesa(C)} \wedge \neg sobre(A,C) \wedge \neg sobre(A,B) \wedge \neg sobre(B,C) \wedge \neg sobre(B,A)$$

Diferente da progressão que raciocina sobre estados completamente especificados (fazendo-se a suposição do mundo fechado), o resultado obtido da regressão assume que um estado é parcialmente especificado, representando um conjunto de estados, ou seja, s é um estado abstrato.

2.3.3 Busca Heurística Simbólica

A *Busca Heurística Simbólica* é uma abordagem que se propõe a unir as estratégias "espertas" de guia por uma solução no espaço de espaços da busca heurística com o poder de representação compacta da *busca simbólica*. Apesar de ter sido proposta em (EDELKAMP; REFFEL, 1998), utilizada nos planejadores MIPS (EDELKAMP; HELMERT, 2001) e GAMER(EDELKAMP; KISSMANN, 2009), só recentemente este tipo de busca apresentou resultados promissores ao tratar problemas de planejamento com grandes espaços de estados. Isso deveu-se, principalmente, pelo fato de os planejadores que utilizavam essas estratégias serem baseados em operações simbólicas entre conjunto de estados e transições e não de conjunto de estados e ações, como mostrado na seção anterior.

Assim, a busca heurística simbólica tem chamado a atenção de pesquisadores pelo fato de que os planejadores que utilizam tal estratégia terem superado em muitos casos os planejadores estado da arte baseados em busca heurística. No trabalho de (TORRALBA *et al.*, 2017) é apresentada a busca simbólica A* (chamada de BDDA*) para problemas de planejamento determinístico no planejamento denominado cGamer (*constrained* Gamer).

Da mesma forma que na busca A*, a busca BDDA* expande primeiramente os estados que possuem menor função de avaliação $f(n) = g(n) + h(n)$. Todavia, BDDA* agrupa conjunto de estados com os mesmos valores de $g(n)$ e $h(n)$, depois expande todos os estados de uma só vez. A função heurística na busca simbólica é pré-computada antes da busca e representada como uma lista de BDDs. Por fim, a avaliação heurística é realizada com uma conjunção: seja um conjunto de estados S e o BDD h_i (conjunto de estados com valor i), $S \wedge listheur_i$ corresponde ao subconjunto de estados que possuem um valor heurístico igual a i (TORRALBA *et al.*, 2017).

cGamer utiliza o BDDA* com uma versão restrita de simbólicos PDBs (*Pattern databases* - Bancos de Dados de Padrões) (EDELKAMP, 2002), que explora as restrições. Os PDBs mapeiam o problema original para um espaço de estados abstratos menor. (TORRALBA *et al.*, 2017) utiliza uma lista BDDA*, que distingue os valores da lista aberta apenas por seu valor $g(n)$, tendo a vantagem de reduzir o número de conjunções com os BDDs heurísticos.

3 TRABALHOS RELACIONADOS

Nesta seção, são apresentados os trabalhos que contribuíram para o desenvolvimento desta pesquisa.

3.1 PropPlan - Propositional Planning

O planejador PropPlan (FOURMAN, 2000) representa estados e *ações* por meio de Diagramas de Decisão Binária (BDDs). Esta foi a primeira abordagem a realizar a busca simbólica diretamente sobre as ações do domínio de planejamento.

O PropPlan realiza uma busca regressiva no espaço de estados para verificar a existência de um plano para problemas de planejamento determinísticos, construindo uma estrutura em camadas para armazenar os estados alcançáveis em i passos. Depois, realiza uma busca progressiva nos estados alcançáveis para determinar que ação será escolhida para levar o sistema de uma camada i para uma camada j.

A representação e raciocínio simbólico do planejador PropPlan é a mesma que a utilizada neste trabalho, todavia, adicionamos a heurística para guiar a busca.

3.2 MIPS - The Model-Checking Integrated Planning System

O planejador (MIPS - *Model-Checking Integrated Planning System*) (EDELKAMP; HELMERT, 2001) representa estados e *transições* por meio de Diagramas de Decisão Binária (BDDs). O algoritmo é capaz de encontrar planos para problemas de planejamento com ações determinísticas usando a busca bidirecional. O algoritmo também implementa a busca simbólica A^*, denominada $BDDA^*$.

O sistema desenvolvido por (EDELKAMP; HELMERT, 2001) se assemelha ao proposto neste trabalho por utilizar BDDs com método de busca heurística A^* para resolver problemas de planejamento. Entretanto, o MIPS realiza o raciocínio sobre as *transições* do modelo do domínio de planejamento e não diretamente sobre as ações. Neste trabalho será utilizado o raciocínio sobre *ações*, o que permite uma representação mais compacta dos domínios de planejamento.

3.3 MBP - a Model Based Planner

O planejador MBP (*Model Based Planner*) (BERTOLI *et al.*, 2001) representa estados e *transições* por meio de Diagramas de Decisão Binária (BDDs). O MBP é baseado em verificação simbólica de modelos e realiza uma busca regressiva no espaço de estados, sendo capaz de obter soluções para problemas em domínios de planejamento não-determinísticos, i.e., as ações tem efeitos incertos.

O MBP, semelhantemente a este trabalho, realiza a representação simbólica, mas ele não realiza raciocínio sobre ações e nem utiliza heurísticas para tornar a busca mais eficiente. Outra diferença é que o MBP é utilizado para problemas de domínios não-determinísticos, neste trabalho são abordados problemas de domínios determinísticos.

3.4 cGamer - Efficient symbolic search for cost-optimal planning

O planejador de (TORRALBA *et al.*, 2017) representa estados e *ações* como *Diagramas de Decisão Binária* (BDDs). Neste sentido, este planejador é uma evolução do MIPS por realizar o raciocínio diretamente sobre as ações do domínio de planejamento. Assim como o MIPS, ele realiza uma busca bidirecional no espaço de estados e utiliza a heurística BDDA*. Este planejador foi um dos vencedores da competição internacional de planejamento IPC-2014 (*International Planning Competition 2014*), sendo capaz de obter soluções para domínios de planejamento com ações determinísticas com custo associado.

O planejador de (TORRALBA *et al.*, 2017) é bem similar ao proposto neste trabalho: utiliza raciocínio simbólico sobre ações, implementa a busca heurística A*, obtém soluções para problemas de planejamento com ações determinísticas. Entretanto, a heurística é calculada a partir de PDBs, neste trabalho apresentaremos uma maneira diferente de cálculo heurístico.

3.5 Comparativo entre as Abordagens

A Tabela 1 mostra a comparação entre os trabalhos relacionados e esta pesquisa. As principais características consideradas são: o ano de desenvolvimento do trabalho, qual a busca utilizada no planejador, se o domínio é clássico ou não-determinístico, e, por fim, se a forma de raciocínio é por meio de ações ou por meio de transições.

Tabela 1 – Comparativo entre as Abordagens Apresentadas.

Trabalho	**Ano**	**Tipo de Busca**	**Domínio**	**Forma de raciocínio**
PROPPLAN	2000	Simbólica Regressiva	Clássico	Ações
MIPS	2000	Simbólica Bidirecional	Clássico	Transições
MBP	2003	Simbólica Regressiva	Não-determinístico	Transições
CGAMER	2017	Simbólica Bidirecional A* e heurística PDB	Clássico	Ações
Planejador deste trabalho	2019	Simbólica A* e heurística obtida a partir da busca regressiva no problema relaxado	Clássico	Ações

Fonte: Elaborada pela autora.

4 BUSCA HEURÍSTICA SIMBÓLICA PARA PLANEJAMENTO DETERMINÍSTICO

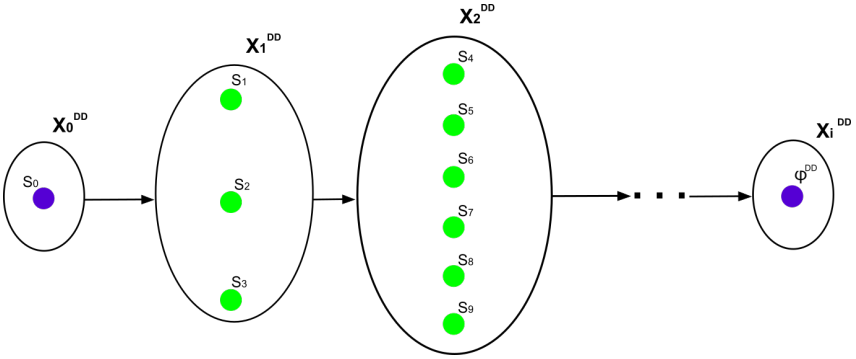
Neste capítulo, iremos propor uma heurística para ser utilizada no algoritmo A* para guiar a busca simbólica por um plano em problemas de planejamento determinístico. Como visto, a busca A* expandirá primeiramente os estados que possuírem menor função de avaliação $f(n) = g(n) + h(n)$, em que $g(n)$ é o custo de alcançar o nó n no processo de busca e $h(n)$ é um valor estimado para se alcançar uma solução a partir de n.

Uma das dificuldades de se utilizar a busca heurística em conjunto com a busca simbólica é que, geralmente, na busca heurística, os valores de $g(n)$ e $h(n)$ são relacionados a nós individuais no processo de busca, porém os estados na busca simbólica são representados como conjuntos de estados em um só Diagrama de Decisão Binária (JENSEN *et al.*, 2008).

O método de busca heurística simbólica implementado foi inspirado no método proposto por (TORRALBA *et al.*, 2017). No entanto, o cálculo do valor heurístico $h(n)$ utilizado neste trabalho é diferente. Realizaremos a busca progressiva e a busca regressiva propostas por (MENEZES, 2014), onde BDDs são utilizados para representar e operar sobre conjuntos de estados e ações do domínio de planejamento, para calcular o custo $g(n)$ e a estimativa $h(n)$.

No trabalho de (MENEZES, 2014), a busca simbólica progressiva tem início no BDD que representa o estado inicial s_0^{DD}, e expande os conjuntos de estados, em cada camada, até encontrar um estado que satisfaça a meta φ. A busca é realizada de maneira exaustiva como apresenta a Figura 7, i.e., todos os estados de uma camada são expandidos.

Figura 7 – Busca simbólica exaustiva no espaço de estados

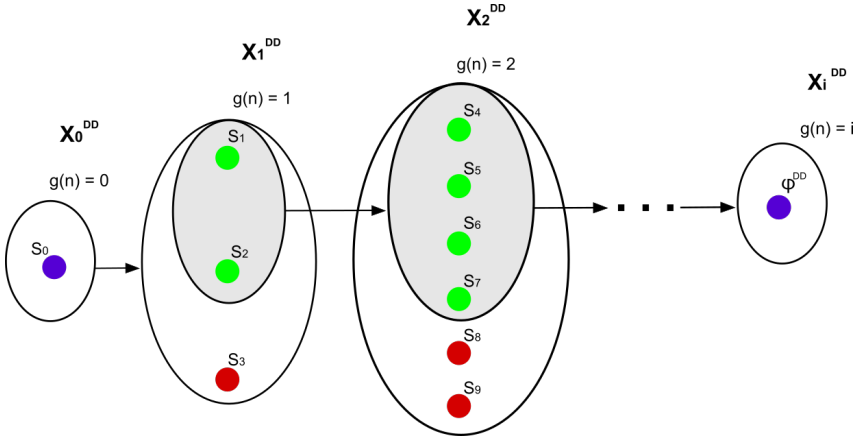


Fonte: Elaborada pela autora.

A busca heurística simbólica implementada neste trabalho, agrupa os estados expandidos em BDDs a partir do valor do custo $g(n)$. A busca também tem início no BDD que representa o estado inicial, denominado de X_0^{DD}, onde o custo $g(n)$ é igual a 0, pois nenhuma

camada foi expandida. Após a expansão, utiliza-se a estimativa $h(n)$ para definir quais estados presentes no BDD possuem melhor função de avaliação para serem expandidos, dessa forma nem todos os estados de uma mesma camada são expandidos. A Figura 8 representa esta abordagem, onde conjuntos de estados com o mesmo custo são expandidos de uma só vez. Realiza-se a expansão até que a meta φ seja alcançada.

Figura 8 – Busca simbólica heurística no espaço de estados



Fonte: Elaborada pela autora.

Neste trabalho, para encontrar a função de avaliação, propõe-se utilizar duas buscas no espaço de estados, que são elas:

1. Busca simbólica regressiva no problema relaxado.
2. Busca simbólica progressiva no problema real.

4.1 Busca simbólica regressiva no problema relaxado

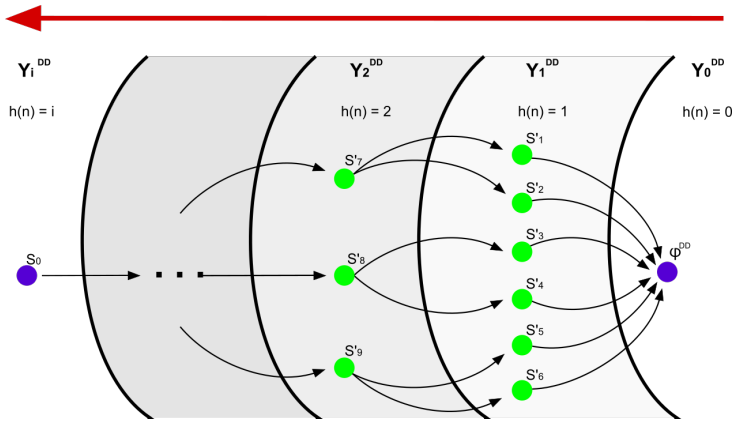
Neste trabalho, propomos um método para cálculo do valor heurístico $h(n)$ que:

- Utiliza um *problema de planejamento relaxado* (GHALLAB *et al.*, 2004), em que ignoramos os efeitos negativos da ação.
- Realiza uma *busca regressiva* a partir da meta no problema relaxado até que um conjunto de estados que contenha o estado inicial seja alcançado.

Esta função heurística é pré-computada antes da busca no problema real. A cada passo i da regressão no problema relaxado, são computados os estados predecessores e armazenados em um BDD denominado Y_i^{DD}, i.e., todos os conjuntos de estados pertencentes à mesma camada são agrupados em um mesmo BDD. Estes BDDs são armazenados em uma lista, ordenada pelo valor heurístico i. A Figura 9 representa como a busca simbólica regressiva foi

implementada. A partir da meta φ^{DD}, que está armazenada no BDD Y_0^{DD}, os estados predecessores são expandidos, ignorando-se os efeitos negativos, até que um conjunto de estados que contenha o estado inicial s_0 seja alcançado. O valor final de $h(n)$ será a estimativa máxima obtida, tornando a heurística admissível, pois o custo para atingir a meta não é superestimado.

Figura 9 – Busca regressiva no problema relaxado para estimar o valor heurístico



Fonte: Elaborada pela autora.

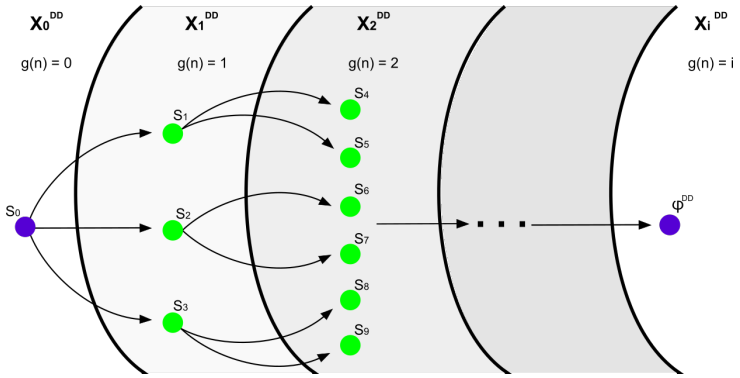
4.2 Busca simbólica progressiva no problema real

A busca simbólica progressiva é realizada no problema real, i.e., os efeitos negativos não são desconsiderados e não há outros relaxamentos. A primeira camada da busca é iniciada do BDD que representa o estado inicial s_0, e nesta camada o valor do custo é igual a 0. A busca avança, expandidos as camadas, ou seja, todos os conjuntos de estados com o mesmo valor $g(n)$ podem ser expandidos de uma só vez. Para cada nova camada o valor de $g(n)$ é incrementado. O custo será calculado até a camada que contenha um estado que satisfaça a meta φ. A Figura 10 demonstra a expansão das camadas.

Durante a busca simbólica progressiva utiliza-se a estimativa para avaliar quais conjuntos de estados serão expandidos. O BDDA* expande os estados em ordem crescente de $f(n) = g(n) + h(n)$, i.e., conjunto de estados com a menor função de avaliação serão expandidos primeiramente.

A função de avaliação é calculada da seguinte forma: dado um conjunto de estados X_i^{DD} e a lista de BDDs obtida no cálculo da estimativa, é realizada uma interseção do conjunto X_i^{DD} com cada BDD da lista, até que o valor obtido seja diferente de $\perp$. Dessa forma, obtém-se os subconjuntos de estados que possuem valor heurístico igual a posição do BDD na lista,

Figura 10 – Busca progressiva no problema real para encontrar o custo



Fonte: Elaborada pela autora.

definindo-se quais conjuntos de estados de uma camada serão expandidos.

5 ANÁLISE EXPERIMENTAL

5.1 Domínio *Benchmarks*

Neste trabalho foi utilizado o *Domínio de Logística* e o *Domínio do Robô de Marte*, provenientes da trilha clássica da Competição Internacional de Planejamento (IPC).

5.1.1 Domínio de Logística

A tarefa no domínio de Logística é transportar um determinado número de pacotes de um local de origem para um local de destino. Essa tarefa é realizada em uma região contendo *n* cidades conectadas apenas por transporte aéreo. Cada cidade possui um conjunto de localizações conectadas por rodovias usadas por caminhões. Os locais de origem e destino podem estar em uma mesma cidade ou em cidades diferentes. Para transportar um pacote dentro de uma mesma cidade, é necessário carregá-lo em um caminhão no local de origem, dirigir o caminhão até o local destino e descarregar o pacote. Para transportar um pacote entre cidades, é necessário levá-lo a um aeroporto da cidade de origem (podendo, para isto, ser necessário utilizar um caminhão), carregar o pacote em um avião e transportá-lo ao aeroporto da cidade de destino. Se o destino final do pacote for o aeroporto, a tarefa da entrega está concluída. Porém, se o destino do pacote for outro, ainda é necessário transportá-lo em um caminhão. Não há restrições sobre o número de pacotes que os caminhões e aviões podem transportar.

Figura 11 – Arquivo PDDL de um problema no domínio de Logística.

```
01.(define (problem logistics-04)
02.  (:domain logistics)
03.  (:objects cit1 cit2 apn1 pos1 pos2 apt1 apt2 tru1 tru2 apn1
04.    obj11 obj12 obj13 obj21 obj21 obj23)
05.   (:init (package obj11) (package obj12) (package obj13)
05.     (package obj21) (package obj22) (package obj23) (truck tru1)
06.    (truck tru2) (city cit1) (city cit2) (location pos1)
07.    (location apt1) (location pos2) (location apt2) (airport apt1)
08.    (airport apt2) (airplane apn1) (at apn1 apt2) (at obj11 pos1)
09.    (at obj12 pos1) (at obj13 pos1) (at tru1 pos1) (at tru2 pos2)
10.    (at obj21 pos2) (at obj22 pos2) (at obj23 pos2)
11.    (in-city pos1 cit1) (in-city apt1 cit1) (in-city pos2 cit2)
12.    (in-city apt2 cit2))
13.  (:goal (and (at obj11 apt1) (at obj23 pos1) (at obj13 apt1)
14.     (at obj21 pos1))))
```

Fonte: Elaborada pela autora.

A Figura 11 exemplifica um problema no domínio de Logística em PDDL. Na linha 03 há a descrição dos objetos deste problema, a saber: duas cidades $cit1$ e $cit2$, dois depósitos $pos1$ e $pos2$, dois aeroportos $apt1$ e $apt2$, dois caminhões $tru1$ e $tru2$, um avião $apn1$ e seis pacotes $obj11$, $obj12$, $obj13$, $obj21$, $obj22$ e $obj23$. Tais objetos são utilizados para instanciar os predicados e as ações do domínio de planejamento. Nas linhas 05-12 há a descrição do estado inicial do problema, onde descreve-se tudo o que é verdadeiro, e o que não é descrito é considerado falso. Nas linhas 13 e 14, há a descrição das metas que devem ser alcançadas.

Na competição para este domínio existem 6 problemas com diferentes números de proposições e ações. Porém, utilizamos nos experimentos versões adaptadas das instâncias do trabalho de (MENEZES, 2014), isto porque estas instâncias já foram traduzidas de PDDL para o formato de entrada aceito pelos algoritmos implementados.

A Tabela 2 apresenta os problemas escolhidos para os experimentos. Na tabela, são apresentados o número de proposições e o número de ações.

Tabela 2 – Diferentes problemas do domínio de Logística

Problema	**Número de Proposições**	**Número de Ações**
Logistics-4	48	78
Logistics-6	48	78
Logistics-8	99	171
Logistics-10	168	308
Logistics-12	168	308
Logistics-14	275	648

Fonte: Elaborada pela autora.

5.1.2 Domínio do Robô de Marte

O domínio do Robô de Marte tem sua modelagem motivada pelas missões executadas pelo MSL (*Mars Science Laboratory*). O domínio do Robô de Marte modela uma tarefa de um ou mais agentes autônomos que realizam exploração planetária. Os robôs navegam entre pontos de apoio no planeta, coletando amostras de solo e rocha para serem analisadas. Cada robô possui um número determinado de câmeras para obter imagens de certos objetos. Dependendo da câmera que o robô possui, é possível obter imagens coloridas, em alta resolução ou baixa resolução. As imagens e os dados resultantes das análises de rocha e solo devem ser transmitidos para a estação espacial que serve de suporte para o robô.

Na competição para este domínio, existem 40 problemas com diferentes números de

proposições e ações. Porém, foram escolhidos apenas 8 problemas para os experimentos.

A Tabela 3 apresenta os problemas escolhidos para os experimentos. Na tabela, são apresentados o número de proposições e o número de ações.

Tabela 3 – Diferentes problemas do domínio de Robô de Marte

Problema	**Número de Proposições**	**Número de Ações**
Rovers-01	35	63
Rovers-02	35	63
Rovers-03	44	76
Rovers-04	50	86
Rovers-05	59	144
Rovers-06	63	178
Rovers-07	68	151
Rovers-08	110	328
...	...	...
Rovers-40	3025	3032

Fonte: Elaborada pela autora.

5.2 Implementação e Resultados

Os algoritmos propostos neste trabalho foram implementados em Java, utilizando-se a biblioteca JavaBdd (WHALEY, 2009) que contém as operações para manipulação dos BDDs. Os testes foram executados em uma máquina virtual Intel(R) Xeon(R) CPU $E5-2640\ v4$ @$2.40GHz$ com $20GB$ de memória RAM dedicados para a máquina virtual Java (JVM). Foram realizados testes em 6 problemas do domínio de logística.

Apesar da busca simbólica contornar o problema da explosão do espaço de estados, a quantidade de memória necessária para armazenar os estados expandidos ainda é um muito grande. O uso da busca heurística em conjunto com a representação simbólica permite que haja podas na árvore de busca. Desta forma, o experimento realizado tem por objetivo constatar se:

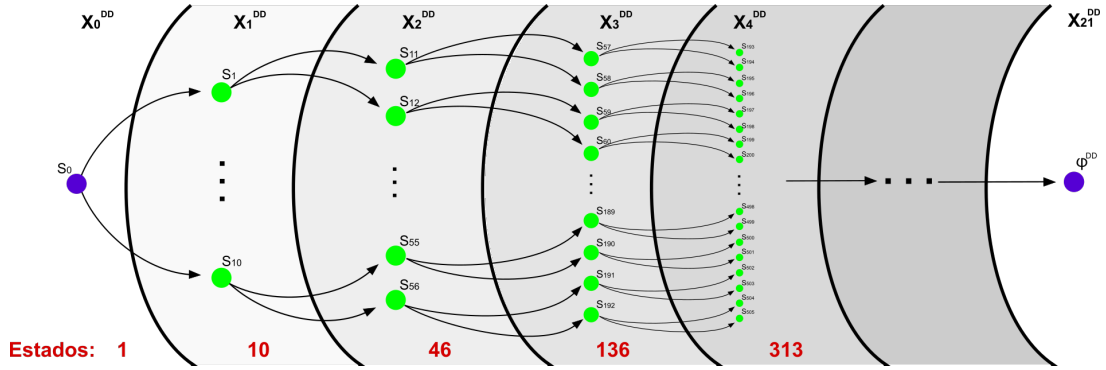
- (i) a busca simbólica heurística proposta neste trabalho realiza podas na árvore de busca em relação à busca simbólica exaustiva e;
- (ii) se os planos encontrados são ótimos, isto é, possuem o mesmo tamanho que os planos encontrados pela busca simbólica exaustiva.

5.2.1 *Problemas do domínio de Logística*

Nesta Subseção, iremos comparar a quantidade de estados expandidos em cada camada pelas buscas simbólicas exaustiva e heurística para uma execução do problema *Logistics-04* citado na Tabela 2. Também iremos abordar todos os experimentos realizados neste domínio.

A Figura 12 ilustra a expansão do espaço de estados em camadas da busca simbólica exaustiva. A expansão tem início no estado inicial, representado pelo BDD X_0^{DD}. O número de estados gerados em cada camada está representado na parte inferior da figura. Na camada seguinte, temos o conjunto dos estados sucessores do estado inicial, agrupados no BDD X_1^{DD}. Nesta camada, 10 estados foram gerados. Na camada seguinte, há a geração de 46 estados. Neste problema, a busca expande 21 camadas até alcançar um estado meta.

Figura 12 – Expansão do espaço de estados em cada camada da busca simbólica exaustiva no problema *Logistics-04* do domínio de Logística.

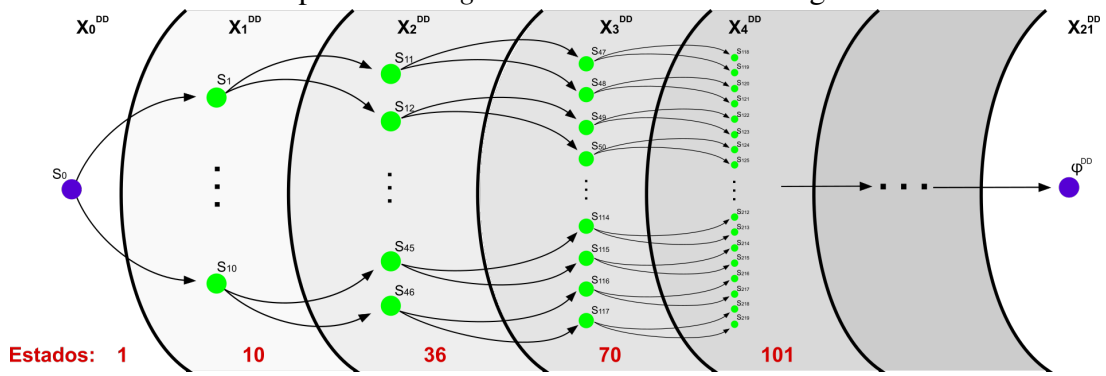


Fonte: Elaborada pela autora.

A Figura 13 mostra a expansão do espaço de estados em camadas da busca heurística simbólica. A expansão também tem início no estado inicial s_0. Na camada seguinte, o número de estados gerados é igual ao da busca simbólica exaustiva, isto pois neste momento não há escolhas de valores de $f(n)$ para ser feita. A partir da camada 1, serão escolhidos apenas os estados com melhores valores de função de avaliação $f(n)$ para serem expandidos. Assim, nas camadas sucessoras, o número de estados gerados começar a ser inferior ao número de estados gerados na busca exaustiva. Observe que na camada 4 da busca simbólica exaustiva (Figura 12) há 313 estados. Porém, nesta mesma camada da busca heurística, o número de estados foi reduzido para 101. É importante destacar que a busca heurística gerou o mesmo número de camadas que a busca exaustiva (21 camadas) para encontrar um plano. Isso se dá pois a heurística proposta é admissível e, assim, a busca A*, caso encontre o plano, garante que este plano seja ótimo.

O problema *Logistics-6* é bem semelhante ao problema *Logistics-4*, como apresen-

Figura 13 – Expansão do espaço de estados em cada camada da busca heurística simbólica no problema *Logistics-04* do domínio de logística.



Fonte: Elaborada pela autora.

tado na Tabela 2, possuindo o mesmo número de proposições e de ações. Neste problema, ambas as buscas conseguem encontrar o plano, expandindo 27 camadas.

Nos demais problemas, tanto a busca simbólica exaustiva quanto a busca heurística simbólica não encontraram uma solução devido aos problemas de memória insuficiente. Analisamos, então, o momento em que cada busca começa a ter problemas com a memória.

No problema *Logistics-8*, a busca exaustiva começar a ter problemas de memória na camada 18, enquanto a busca heurística apresenta problemas ainda no cálculo heurístico, ou seja, na busca regressiva no problema relaxado, mais precisamente na camada 17.

Nos problemas *Logistics-10* e *Logistics-12*, que apresentam o mesmo número de proposições e ações, a busca exaustiva começar a ter problemas de memória na camada 11. Nestes casos, a busca heurística também apresenta problemas ainda no cálculo heurístico, mais precisamente na camada 8.

No problema *Logistics-14*, a busca exaustiva começar a ter problemas de memória na camada 8. Neste caso, a busca heurística também apresenta problemas ainda no cálculo heurístico, mais precisamente na camada 6.

A Tabela 4 apresenta um resumo dos resultados obtidos para o problema do domínio de logística. Para cada tipo de busca são apresentados o tamanho do plano e o tempo de execução. Para os problemas *Logistics-4* e *Logistics-6*, os quais se obteve resultado em ambas as buscas, nota-se que o tamanho do plano é o mesmo. Nos demais problemas, expressa-se com * a camada que as buscas começam a ter problemas de memória. A busca exaustiva tem vantagem em relação ao tempo, obtendo tempos mais curtos que a busca heurística, uma vez que esta última precisa realizar também uma busca regressiva para computação dos valores heurísticos.

Tabela 4 – Resultados obtidos nas buscas simbólicas exaustiva e heurística nos problemas do domínio de logística

Problema	**Busca Exaustiva**		**Busca Heurística**	
	Camadas Expandidas	Tempo	Camadas Expandidas	Tempo
Logistics-4	21	4.396s	21	6.268s
Logistics-6	27	5.340s	27	7.976s
Logistics-8	18*	-	17*	-
Logistics-10	11*	-	8*	-
Logistics-12	11*	-	8*	-
Logistics-14	8*	-	6*	-

Fonte: Elaborada pela autora.

5.2.2 Problemas do domínio do Robô de Marte

Os problemas iniciais, do *Rovers-01* ao *Rovers-04*, tanto a busca simbólica exaustiva quanto a busca heurística simbólica encontram o plano, possuindo o mesmo número de camadas expandidas.

A busca heurística simbólica conseguiu verificar a existência de um plano para o problema *Rovers-05*, com 28 camadas expandidas, o que não foi possível na busca simbólica exaustiva, por começar a ter problemas de memória na cama 11.

No problema *Rovers-06*, ambas as buscas não encontram uma solução devido aos problemas de memória insuficiente. A busca simbólica exaustiva começa a ter problemas na camada 18, enquanto a busca heurística simbólica expande até a camada 28 da busca regressiva.

No problema *Rovers-07*, a busca simbólica exaustiva não conseguiu verificar a existência de um plano, começando a ter problemas de memória na camada 12. Contudo, a busca heurística simbólica consegue encontrar o plano, expandindo 27 camadas.

No problema *Rovers-08*, ambas as buscas não encontram uma solução devido aos problemas de memória insuficiente. A busca simbólica exaustiva começa a ter problemas na camada 8, enquanto a busca heurística simbólica expande até a camada 11 da busca regressiva.

A Tabela 5 apresenta um resumo dos resultados obtidos para o problema do domínio do robô de marte. Para cada tipo de busca são apresentados o tamanho do plano e o tempo de execução. Para os problemas *Rovers-01*, *Rovers-02*, *Rovers-03* e *Rovers-04*, os quais se obteve resultado em ambas as buscas, nota-se que o tamanho do plano é o mesmo. Para os problemas *Rovers-06* e *Rovers-08*, ambas as buscam não conseguiram verificar a existência de um plano, expressa-se com * a camada que as buscas começam a ter problemas de memória. Para os

problemas *Rovers-05* e *Rovers-07*, apresenta-se na busca exaustiva a camada que começam haver problemas de memória e, na busca simbólica apresenta-se o plano encontrado.

Tabela 5 – Resultados obtidos nas buscas simbólicas exaustiva e heurística nos problemas do domínio do robô de marte

Problema	**Busca Exaustiva**		**Busca Heurística**	
	Camadas Expandidas	Tempo	Camadas Expandidas	Tempo
Rovers-01	10	1.540s	10	2.424s
Rovers-02	13	2.972s	13	2.684s
Rovers-03	11	1.980s	11	3.232s
Rovers-04	8	1.576s	8	3.004s
Rovers-05	11*	-	26	22.460s
Rovers-06	18*	-	28*	-
Rovers-07	12*	-	27	2m56.324s
Rovers-08	8*	-	11*	-

Fonte: Elaborada pela autora.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho propôs um método para cálculo do valor heurístico para utilizar na busca simbólica A^* em problemas de planejamento determinístico. Dessa forma o algoritmo proposto realiza:

- Busca simbólica regressiva no problema relaxado para computação dos valores heurísticos;
- Busca simbólica progressiva no problema real para computação dos valores de custo.

Apesar de em (TORRALBA *et al.*, 2017) a busca simbólica A^* também ser utilizada em problemas de planejamento determinístico, o cálculo heurístico é feito de forma diferente do realizado neste trabalho. Em (TORRALBA *et al.*, 2017), utiliza-se a técnica de *Patterns Databases* (PDB) (EDELKAMP, 2002) para cálculo do valor heurístico.

Os experimentos foram realizados em domínios da trilha clássica da Competição Internacional de Planejamento (IPC, 2019): *Domínio de Logística* e *Domínio do Robô de Marte*. A comparação dos algoritmos de *busca simbólica exaustiva* e *busca simbólica heurística* foi realizada considerando como fatores de medida: a quantidade de camadas expandidas para encontrar um plano e o tempo para obtenção de um plano.

Neste tipo de problema, a quantidade de memória necessária para armazenar os estados expandidos na busca é um gargalo. Por um lado, os BDDs são uma forma compacta de representar os estados expandidos em cada camada. No entanto, a quantidade de estados expandidos na busca cega é ainda muito grande. Assim, com os experimentos, mostramos que a busca heurística pode reduzir a quantidade de estados expandidos. Ademais, a heurística proposta é *admissível* e os experimentos confirmaram que o algoritmo, quando encontra uma solução, produz planos ótimos, assim como os produzidos pela busca exaustiva.

Pelos experimentos realizados, constatamos que o ganho de memória conseguido com a utilização da heurística tornou possível encontrar uma solução para alguns dos problemas que a busca exaustiva não encontra.

Como trabalhos futuros podemos destacar: a realização de testes com mais domínios *benchmarks* da trilha clássica; a incorporação de outras técnicas descritas por (TORRALBA *et al.*, 2017) tais como busca bidirecional e cálculo de invariantes. Ainda achamos válido propor a aplicação da busca simbólica heurística para problemas de planejamento com ações não-determinísticas.

REFERÊNCIAS

BERTOLI, P.; CIMATTI, A.; PISTORE, M.; ROVERI, M.; TRAVERSO, P. Mbp: a model based planner. In: **Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information**. [S.l.: s.n.], 2001.

BRYANT, R. E. Binary decision diagrams and beyond: Enabling technologies for formal verification. In: IEEE. **Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on**. [S.l.], 1995. p. 236–243.

BUNING, H. K.; KARPINSKI, M.; FLOGEL, A. Resolution for quantified boolean formulas. In: **Information and computation**. [S.l.]: Elsevier, 1995. v. 117, n. 1, p. 12–18.

BURCH, J. R.; CLARKE, E. M.; MCMILLAN, K. L.; DILL, D. L.; HWANG, L.-J. Symbolic model checking: 10^{20} states and beyond. In: **Information and computation**. [S.l.]: Elsevier, 1992. v. 98, n. 2, p. 142–170.

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

CIMATTI, A.; GIUNCHIGLIA, E.; GIUNCHIGLIA, F.; TRAVERSO, P. Planning via model checking: A decision procedure for ar. In: SPRINGER. **European Conference on Planning**. [S.l.], 1997. p. 130–142.

EDELKAMP, S. Symbolic pattern databases in heuristic search planning. In: **AIPS**. [S.l.: s.n.], 2002. p. 274–283.

EDELKAMP, S.; HELMERT, M. Mips: The model-checking integrated planning system. In: **AI magazine**. [S.l.: s.n.], 2001. v. 22, n. 3, p. 67.

EDELKAMP, S.; KISSMANN, P. Optimal symbolic planning with action costs and preferences. In: **Twenty-First International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2009.

EDELKAMP, S.; KISSMANN, P.; TORRALBA, A. Bdds strike back (in ai planning). In: **Twenty-Ninth AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2015.

EDELKAMP, S.; REFFEL, F. Obdds in heuristic search. In: SPRINGER. **Annual Conference on Artificial Intelligence**. [S.l.], 1998. p. 81–92.

FIKES, R. E.; NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. In: **Artificial intelligence**. [S.l.]: Elsevier, 1971. v. 2, n. 3-4, p. 189–208.

FOURMAN, M. P. Propositional planning. In: **Proceedings of AIPS-00 Workshop on Model-Theoretic Approaches to Planning**. [S.l.: s.n.], 2000. p. 10–17.

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

HELMERT, M. The fast downward planning system. In: **Journal of Artificial Intelligence Research**. [S.l.: s.n.], 2006. v. 26, p. 191–246.

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

HUTH, M.; RYAN, M. **Logic in Computer Science**: Modelling and reasoning about systems. [S.l.]: Cambridge university press, 2004.

IPC. **ICAPS Competitions**.[1998?]. Disponível em: http://icaps-conference.org/index.php/main/competitions. Acesso em: 15 abr. 2019.

JENSEN, R. M.; VELOSO, M. M.; BRYANT, R. E. State-set branching: Leveraging bdds for heuristic search. In: **Artificial Intelligence**. [S.l.]: Elsevier, 2008. v. 172, n. 2-3, p. 103–139.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. The planning domain definition language. In: **Proceedings of the Artificial Intelligence Planning Systems Competition**. [S.l.: s.n.], 1998.

MCMILLAN, K. L. Symbolic model checking. In: **Symbolic Model Checking**. [S.l.]: Springer, 1993. p. 25–60.

MENEZES, M. V. **Mudanças em Problemas de Planejamento sem Solução**. [S.l.]: Universidade de São Paulo, 2014.

NILSSON, N. J. **Principles of artificial intelligence**. [S.l.]: Tioga, Palo Alto, 1980.

PEREIRA, S. L. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. [S.l.]: Universidade de São Paulo, 2007.

RICHTER, S.; WESTPHAL, M. The lama planner using landmark counting in heuristic search. In: CITESEER. **Proceedings of IPC**. [S.l.], 2008.

RUSSELL, S.; NORVING, P. **Inteligência Artificial**. 3th. ed. [S.l.]: Elsevier, 2013.

SANTOS, V. B. **Planejamento baseado em Verificação Simbólica de Modelos**. [S.l.]: Universidade de São Paulo, 2018.

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

WHALEY, J. **JavaBDD-Java binary decision diagram library**.[ca. 2003 - 2007]. Disponível em: http://javabdd.sourceforge.net/. Acesso em: 20 ago. 2009.