Analysis of Planning Performance Module Theories for a New Domain Using Resources and Time

Danielle de Fátima Ivanchechen[1], **Marcos Alexandre Castilho**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brazil

danielle.ivanchechen@gmail.com, marcos@inf.ufpr.br

Abstract. *As the search for plans becomes more relevant to applications in the real world, it increases the demands for expression in the modeling language. In particular, there is a new planning formalism for the use of new theories to increase the power of modeling. This paper shows how the Planning Modules Theory (PMT) fulfills this role by analyzing its performance compared to the MetricFF planner. Also, a new domain is presented in which time and resources are used simultaneously , which is capable of making the PMT able to solve problems that are outside the existing domains.*

Resumo. *À medida que a pesquisa de planejamento se torna mais relevante para aplicações no mundo real, aumentam as demandas de poder expressivo na linguagem de modelagem. Em particular, existe um novo formalismo de planejamento em que se utiliza noções de teorias para aumentar o poder da modelagem. Este trabalho mostra como o Planejamento Módulo Teorias (PMT) cumpre esse papel, analisando o seu desempenho em comparação com o planejador MetricFF. Também, apresenta-se um novo domínio em que utiliza-se tempo e recursos simultaneamente, o qual mostra como o PMT pode resolver problemas que vão além dos domínios existentes.*

1. Introdução

Há muitos anos o estudo de planejamento tem sido um dos principais objetivos de pesquisa em Inteligência Artificial (IA). Russell e Norvig [Russell and Norvig 2003] citam que um dos principais motivos do interesse pela área é o fato dela combinar dois conceitos em IA: a busca e a lógica. Assim, um planejador, que é um sistema implementado para encontrar um plano de ações, pode ser visto tanto como um algoritmo que busca por uma solução quanto um algoritmo que de maneira construtiva prova a existência (ou não) de uma solução.

Pode-se modelar o problema de planejamento em alguma linguagem formal, como descrito por Weld [Weld 1999]:

1. uma descrição do estado inicial do mundo;
2. uma descrição do objetivo do agente (isto é, qual comportamento é desejado);
3. uma descrição das possíveis ações que podem ser realizadas.

De acordo com a modelagem do problema, os planejadores dividem-se em duas categorias de planejamento: clássico e não clássico.

No planejamento clássico, o domínio do problema apresenta algumas restrições, como conjunto finito de estados, ambiente completamente observável, ações determinísticas, estados estáticos, objetivos restritos, planos sequenciais, ações sem duração e planejamento offline [Ghallab et al. 2004]. Assim, ao se modelar um problema de planejamento clássico, muitas vezes é preciso abandonar os detalhes, como recursos, tempo, mundo não observável, entre outros fatores. Isso faz-se necessário devido a complexidade de tempo para se encontrar uma solução.

Ainda que o planejamento clássico resolva uma grande variedade de problemas ele ainda é limitado e não é realista. Para ir além do domínio proposicional e para que os estudos possam avançar no sentido de resolver problemas reais surge a necessidade de adicionar noções de tempo, quantidade, espaço, entre outros. Com isso, as restrições do planejamento clássico são violadas, possibilitando aos modelos o tratamento de outros problemas em que esses fatores são fundamentais. Ao violar essas restrições o planejamento é chamado de não clássico.

Até o momento, para cada problema de planejamento não clássico deve ter um algoritmo específico para tratá-lo, ou seja, um algoritmo com sua própria sintaxe e integração para tratar o problema em questão. Assim, resolver um problema que use tempo e recurso simultaneamente não é possível com o uso de um único planejador.

Em 2012, Gregory e colegas [Gregory et al. 2012] propuseram um novo formalismo para trabalhar com planejamento não clássico, no qual permite-se que o planejamento seja ampliado em um caminho modular, de modo que o planejador trate de forma geral os problemas de planejamento. Esse formalismo é chamado de Planejamento Módulo Teorias, em inglês, *Planning Modulo Theories* (PMT). Ao limite dessa pesquisa, pouco foi encontrado sobre esse formalismo.

Assim, o objetivo desta pesquisa é analisar esse o formalismo PMT. Com foco em:

- expor e explicar o planejador, ainda em fase experimental, que resolve diretamente problemas PMT, o PMTPlan, desenvolvido por Gregory e colegas [Gregory et al. 2012];
- propor um novo domínio PMT em que o uso de tempo e recursos são utilizados simultaneamente, usando como base o conhecido problema dos jarros [Rich 1985]; e
- analisar o comportamento do planejador PMTPlan com esse novo problema.

Dessa forma, o restante deste artigo está organizado da seguinte forma: A Seção 2 apresenta conceitos e definições sobre o PMT, assim como a descrição de problemas em PMT e a Seção 3 apresenta o planejador PMTPlan. Na Seção 4 está proposto um novo domínio para o problema dos jarros. Os experimentos e a discussão dos resultados são apresentados na Seção 5, baseados no tratamento de problemas que envolvem tempo e recurso. Por último, as conclusões e trabalhos futuros são discutidos na Seção 6.

2. Planejamento Módulos Teoria

Esta seção define o PMT, além de apresentar seus conceitos e a descrição de problemas em PMT.

2.1. O Problema PMT

O PMT é uma estrutura modular inspirada no SMT (*Satisfiability Modulo Theories*) [Barrett et al. 2009] que permite que o planejamento seja ampliado em um caminho modular de forma semelhante ao SMT. Isso significa que módulos com funções que operam sobre as novas teorias adicionadas ao planejador serão criados e usados em ações de planejamento. De maneira geral, um problema PMT contém as seguintes definições [Gregory et al. 2012]:

- Estado: um estado é uma avaliação sobre um conjunto finito de variáveis, V, no qual cada variável, $v \in V$, tem um domínio correspondente de valores possíveis, D_v. A expressão $s[v]$ denota o valor que o estado s atribui à variável v e $s[v = x]$ é o estado que é idêntico a s, exceto que ele assinala o valor x para a variável v;
- Espaço de estado: para um conjunto de variáveis V é o conjunto de todas as avaliações sobre V;
- Sentença de primeira ordem sobre um espaço de estado S módulo T: é uma sentença de primeira ordem sobre as variáveis do espaço de estado, símbolos constantes, símbolos de função e símbolos predicados, em que T é uma teoria que define os domínios das variáveis e as interpretações para as constantes, funções e predicados;
- Termo sobre S módulo T: é uma expressão construída com os símbolos definidos por S e T;
- Ação: uma ação, A, para um espaço de estado S módulo T, é uma função de transição de estado, compreendendo:
  - Uma sentença de primeira ordem sobre S módulo T, Pre_A (pré-condição de A);
  - Um conjunto, Eff_A (efeitos da ação A), de atribuições a um subconjunto das variáveis do estado S, no qual cada configuração de uma variável para um valor é definido por um termo sobre S módulo T.
- Aplicação de uma ação: uma ação, A, para o espaço de estado S módulo T, é aplicável (ou executável) em um estado $s \in S$ se T, $s \models Pre_A$ (ou seja, a teoria, juntamente com a avaliação s, satisfaz a pré-condição de A). Após a aplicação de A, o estado s é atualizado pelas atribuições em Eff_A para as variáveis que eles afetam. Todas as outras variáveis permanecem inalteradas.
- Um problema PMT, para uma teoria T, é uma tupla <S, A, I, G>, onde:
  - S é um espaço de estado no qual todas as variáveis dos domínios são definidos em T;
  - A é uma conjunto de ações para S módulo T;
  - I é uma avaliação inicial (o estado inicial);
  - G é uma sentença de primeira ordem sobre S módulo T (o objetivo).
- Plano: é uma seqüência de ações $\langle a_1, ..., a_n \rangle$, de modo que a_i é aplicável no estado s_{i-1} e s_i é o resultado da aplicação a_i para $s_i - 1$, onde $s_0 = I$, $s_n| = G$.

2.2. Descrevendo Problemas PMT

Para suportar a descrição dos domínios PMT, Gregory e colegas propuseram uma variante modular do PDDL (*Planning Domain Definition Language*) [McDermott et al. 1998], que visa a adição de teorias. A ideia dessa nova linguagem modular se assemelha ao SMT-lib [Barrett et al. 2016], pois permite adicionar novas teorias, em que novos tipos, com funções e variáveis da teoria lógica de interesse, são proporcionados.

Os arquivos do módulo contêm declarações dos tipos de funções que manipulam os tipos novos e existentes e estão escritas em MDDL (Module Definition Description Language). Cada módulo pode definir um único tipo de teoria com constantes e funções. As funções podem ser sobre o novo tipo e também sobre os tipos definidos em outros módulos.

Por exemplo, pode-se descrever o módulo *set* (teoria dos conjuntos) como mostrado no código abaixo. Neste código estão definidas as funções apropriadas para o tratamento de conjuntos.

```
(define (module set)
    (:type set of a')
    (:functions
        (construct-set  ?x+ - a')                       - set of a'
        (empty-set)                                     - set of a'
        (cardinality    ?s - set of a')                 - integer
        (member         ?s - set of a' ?x - a')         - boolean
        (subset         ?x - set of a' ?y - set of a')  - boolean
        (union          ?x - set of a' ?y - set of a')  - set of a'
        (intersect      ?x - set of a' ?y - set of a')  - set of a'
        (difference     ?x - set of a' ?y - set of a')  - set of a'
        (add-element    ?s - set of a' ?x - a')         - set of a'
        (rem-element    ?s - set of a' ?x - a')         - set of a'
    )
)
```

As definições das variáveis e ações que modelam um domínio de planejamento são fornecidas em arquivos principais, que estão escritos em *Core Domain Description Language* (CDDL).

Um arquivo CDDL que descreve o domínio do problema contém um cabeçalho, que define nomes e valores, os módulos, que são necessários para aquele domínio, e as funções, que definem variáveis de estado para o domínio. Nesse arquivo também são descritas as ações do planejador. As ações, assim como em PDDL, são definidas em duas partes: uma lista de pré-condições e uma lista de efeitos. Um arquivo CDDL que descreve uma instância do problema contém um cabeçalho, que define os objetos requeridos, e uma descrição do estado inicial e objetivo.

Consideremos como exemplo uma instância do problema dos jarros [Rich 1985], no qual existem 4 jarros, inicialmente vazios. Os seguintes trechos de código apresentam a instância do problema dos jarros em CDDL. O objetivo é deixar algum dos jarros com quantidade total de 1 litro.

O trecho abaixo exibe o cabeçalho dessa instância. Quatro objetos do tipo jarro (*jug*) são requeridos.

```
(define (problem jugs)
(:domain jugpour)
(:objects
 jug1 jug2 jug3 jug4 - jug
)
```

O próximo trecho representa o estado inicial da instância. Os 4 jarros, enumerados do 1 ao 4, com capacidade de 3, 6, 9 e 23 litros, respectivamente, estão inicialmente vazios. O espaço livre é igual a capacidade.

```
(:init
(assign (quantity jug1) 0)
(assign (quantity jug2) 0)
(assign (quantity jug3) 0)
(assign (quantity jug4) 0)
(assign (space jug1) 3)
(assign (space jug2) 6)
(assign (space jug3) 9)
(assign (space jug4) 23)
(assign (capacity jug1) 3)
(assign (capacity jug2) 6)
(assign (capacity jug3) 9)
(assign (capacity jug4) 23)
)
```

O fragmento de código abaixo apresenta o estado objetivo da instância: um dos jarros deve estar com a quantidade de 1 litro.

```
(:goal
(member (construct-set (quantity jug1) (quantity jug2)
(quantity jug3) (quantity jug4) ) 1)
)
)
```

Os próximos trechos apresentam o domínio para o problema dos jarros em CDDL. O trecho abaixo apresenta o cabeçalho do domínio, definindo o nome, os tipos de objetos e os módulos usados para a resolução. Nesse exemplo temos o tipo jarro (*jug*) e os módulos de inteiros (*integer*) e conjuntos (*set*).

```
(define (domain jugpour)
  (:types
   jug
  )
  (:modules integer set)
```

O próximo passo é definir as funções que serão usadas. Neste caso, as funções de *quantity*, para especificar a quantidade existente no jarro, de *space*, para informar o espaço restante no jarro e *capacity*, para indicar a capacidade do jarro.

```
  (:functions
     (quantity ?t - jug) - integer
     (space    ?t - jug) - integer
     (capacity ?t - jug) - integer
  )
```

Como exemplo de descrição de uma ação, o trecho abaixo descreve a ação de encher um jarro.

```
(:action fill-jug
  :parameters
   (?t - jug)
  :precondition
   (true)
  :effect
   (
    (assign (quantity ?t) (capacity ?t))
    (assign (space ?t) 0)
   )
)
```

3. O Planejador PMTPlan

O PMTPlan é um planejador em fase experimental proposto por Gregory e colegas [Gregory et al. 2012], em 2012 e, desde então, não houve evolução publicada. Esse planejador trata o problema PMT diretamente no formato PMT, em linguagem CDDL e MDDL.

O PMTPlan está dividido em duas partes: *Modules* e *Core*. A parte denominada *Modules* é encarregada pelos módulos de teorias, estes que contém os tipos e funções e a parte *Core* é responsável pela busca do plano e acessa cada parte do *Modules*. A Figura 1 apresenta uma esquematização do PMTPlan.

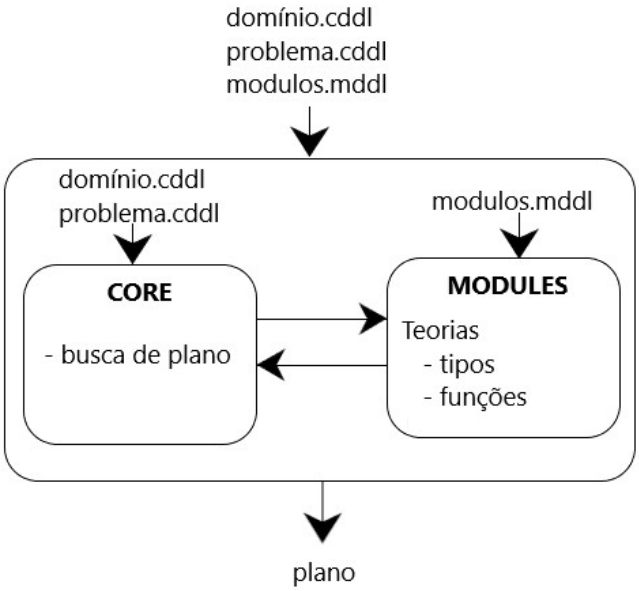

Figura 1. Esquematização do PMTPlan.

Um detalhe interessante do PMTPlan é que, a fim de tornar fácil a criação de novos módulos pelo usuário, a entrada do planejador precisa de um arquivo com a especificação do módulo (teoria). Os módulos não são implementados diretamente no planejador. O usuário precisa descrever em um arquivo MDDL o tipo e funções e usar esse arquivo como parâmetro, na execução do planejador. Dessa forma, os módulos são criados durante a execução do planejador e só serão criados os módulos com as teorias necessárias para aquele problema.

4. Novo Domínio PMT

Aqui apresenta-se um novo domínio para mostrar a eficiência do PMT em lidar com problemas que vão além dos domínios proposicionais, mais especificamente domínios com tempo e recursos. Como não foi encontrado domínios na literatura que tratem dessa composição, propõe-se um novo domínio com base em um já existente.

A proposta é acrescentar uma nova dimensão ao problema dos jarros, que trata apenas a questão de recursos. Assim, esse problema terá duas dimensões: recursos e tempo.

O objetivo desse problema é encher os jarros da melhor forma possível com o menor tempo. Para isso, admite-se que cada litro transferido, colocado ou retirado de cada jarro leva 1 segundo para acontecer.

Assume-se o exemplo da Seção 2 para demonstrar as mudanças na modelagem do problema.

Seja o arquivo de instância do problema. Ao novo cabeçalho adiciona-se um novo objeto, act1, para garantir que cada ação execute e termine. Isso impede que ações sejam executadas ao mesmo tempo.

```
(define (problem jugs)
(:domain jugpour)
(:objects
jug1 jug2 jug3 jug4 - jug
 act1 - act
)
```

Ao estado inicial define-se o objeto act1 com o comando (assign (free-act act1) true). O restante continua igual, assim como o estado objetivo.

Na modelagem do domínio do problema acrescenta-se o módulo temporal e a função (free-act ?a - act) - boolean. Essa função indica se existe ação sendo executada ou não.

Para exemplo de descrição de uma ação, o trecho abaixo descreve a ação de encher um jarro.

```
(:action fill-jug
    :parameters(
       ?t - jug
       ?a - act
    )
    :precondition(
        (duration-of (this) (capacity ?t))
        (< (quantity ?t) (capacity ?t))
        (at-start (free-act ?a))
    )
    :effect(
        (at-start (assign (free-act ?a) false))
        (assign (quantity ?t) (capacity ?t))
        (assign (space ?t) 0)
        (at-end (assign (free-act ?a) true))
    )
)
```

Note a criação do parâmetro a. Esse paramêtro é necessário para controlar a duração de cada ação.

Para calcular o tempo é necessário que o objeto a se inicie livre. Assim, o estabelece como não-livre. Deste modo, a duração para encher o jarro é a capacidade do mesmo. Após o término da ação, marca-se o objeto a como livre, para que outra ação possa ser iniciada.

5. Experimentos e Discussões

Esta seção mostra alguns experimentos e resultados com o uso do PMTPlan. Para a execução dos experimentos foi utilizado uma máquina com 8GiB de memória e processador Intel Core i3.

5.1. Experimentos com Recursos

Aqui apresenta-se os experimentos para o conhecido problema dos jarros, citado na Seção 2. Neste experimento é feito uma comparação dos resultados do PMTPlan com o planejador MetricFF. Este planejador foi escolhido por ser considerado um bom planejador para problemas com domínios numéricos.

A Tabela 1 apresenta os resultados para esse problema e está disposta em dois blocos principais identificando o planejador usado. A primeira coluna de cada bloco apresenta a quantidade de nós criados durante a busca pelo plano, a segunda o tamanho do plano encontrado e a terceira tempo total, em segundos, de execução. Cada linha dessa tabela representa os dados obtidos para instâncias com diferentes quantidades de jarros. A escolha do número de jarros para cada instância do problema se baseou nos domínios já existente.

Tabela 1. Análise de desempenho para o problema clássico dos jarros.

	PMTPlan			MetricFF		
Nº de Jarros	**Nós**	**Tamanho do Plano**	**Tempo (seg)**	**Nós**	**Tamanho do Plano**	**Tempo (seg)**
2	17	16	1.37	18	17	0.00
4	16	6	3.70	136	17	0.00
6	24	6	7.88	582	17	0.01
8	1044	8	298.25	2516	17	0.10
10	27	6	26.05	10564	17	0.64
12	25	6	33.15	28740	17	2.73
14	26	6	60.53	73558	17	9.76
16	31	6	95.09	186206	17	35.21
18	47	6	286.23	–	–	–
20	50	6	418.63	–	–	–
22	36	6	408.08	–	–	–
24	60	6	824.07	–	–	–
26	42	6	768.76	–	–	–
28	63	6	1527.27	–	–	–
30	47	6	1645.70	–	–	–

Observa-se que para as instâncias testadas o PMTPlan apresentou resultados inferiores quando comparado ao MetricFF nas colunas 'Nós' e 'Tamanho do Plano'. Nota-se que as linhas abaixo da instância com 16 jarros não apresentam resultados para o MetricFF, isso se deve ao fato de que esse planejador ultrapassou a quantidade de memória da máquina utilizada ao executar a busca pelo melhor plano.

Também, observa-se que na questão de tempo total da execução o MetricFF teve uma melhor performance, apresentando tempos menores em comparação ao PMTPlan.

Apesar do MetricFF ser mais rápido, ele não consegue resolver os problemas que foram resolvidos pelo PMTPlan. Isto mostra que o PMTPlan usa menos memória e consegue executar testes maiores, além de encontrar planos consideravelmente menores em relação ao MetricFF.

5.2. Experimentos com Recursos e Tempo

Aqui exibe-se os experimentos feitos utilizando o domínio proposto na Seção 4. Desta forma, se quer analisar o desempenho e viabilidade do PMTPlan encontrar algum plano para esse problema. Visto que esse domínio baseou-se em um existente, no qual acrescentou-se noções temporais, pode-se comparar esses dois domínios.

A Tabela 2 traz os resultados obtidos com esses experimentos e está preparada da seguinte forma: a primeira coluna apresenta a quantidade de jarros usada em cada instância, seguindo o padrão do experimento anterior a quantidade de jarros varia de 2 até 18, buscando quantidades pares. A segunda coluna apresenta a quantidade de nós criados durante a busca e escalonamento do plano, a terceira o tamanho do plano encontrado e a quarta o tempo total, em segundos, de execução.

Tabela 2. Análise de desempenho para o problema dos jarros com noções temporais.

	PMTPlan		
Nº de Jarros	**Nós**	**Tamanho do Planho**	**Tempo (seg)**
2	78	16	4.03
4	23	5	6.58
6	51	5	19.05
8	29	5	22.31
10	48	5	63.64
12	46	5	96.78
14	83	5	381.64
16	57	5	416.03
18	80	5	882.03

Com esses resultados pode-se perceber que o PMTPlan consegue encontrar bons planos para a resolução das instâncias do problema dos jarros com noções temporais.

Observa-se que comparado com a Tabela 1 houve um aumento significativo no número de nós criados durante a busca pelo plano solução. Isso acontece devido ao uso do fator tempo, então, o planejador precisa que as ações demarquem tempo de início e tempo de finalização. Assim cada ação é transformada em duas: uma demarcando seu início e outra demarcando o seu fim. Ao final da execução da busca as ações voltam ao normal para o escalonamento de tempo. Desta maneira, com o aumento da criação de nós, o tempo total de execução também aumenta.

Assim, com esses experimentos assegura-se que o PMTPlan é capaz de lidar com esse tipo de desafio, onde recursos e tempo são usados no mesmo problema, além de encontrar bons planos (parecidos com os encontrados na Tabela 1) para a resolução das instâncias do problema dos jarros com noções temporais.

6. Conclusão

Neste trabalho foi aplicado um novo formalismo de planejamento, o PMT, que permite que o planejamento possa ser ampliado a um caminho modular, para resolver problemas que utilizam tempo e recursos simultaneamente. Também foi utilizado um planejador que resolve diretamente problemas PMT, o PMTPlan, desenvolvido por Gregory e colegas para resolver esse tipo de problema. Como não foi identificado na literatura domínios para esse problema, foi proposto um novo domínio, utilizando como base o problema dos jarros, em que o uso de noções temporais foi adicionado ao problema.

A partir disso, foi demonstrado o comportamento e a eficiência do PMTPlan em lidar com problemas em que o uso de tempo e recursos são usados simultaneamente.

Também foi demonstrado a eficiência do PMTPlan, mesmo sendo um planejador em fase experimental em comparação com outro planejador, o MetricFF, ao resolver um problema clássico de planejamento com recursos.

Em trabalhos futuros planeja-se aprimorar o planejador PMTPlan, acrescentando a ele métricas em relação a quantidade e, principalmente, tempo. Além disso, pretende-se criar novos domínios para o novo formalismo, no qual um deles foca no estudo de caso para problema de transporte ferroviário.

Referências

Barrett, C., Fontaine, P., and Tinelli, C. (2016). The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, acessado em 13/01/2018.

Barrett, C., Sebastiani, R., Seshia, S., and Tinelli, C. (2009). Satisfiability modulo theories. In Biere, A., Heule, M. J. H., van Maaren, H., and Walsh, T., editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

Gregory, P., Long, D., Fox, M., and Beck, J. C. (2012). Planning Modulo Theories : Extending the Planning Paradigm. *International Conference on Automated Planning and Scheduling*, pages 65–73.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language.

Rich, E. (1985). Artificial intelligence and the humanities. *Computers and the Humanities*, 19(2):117–122.

Russell, S. J. and Norvig, P. (2003). *Artificial intelligence: a modern approach*. Elsevier.

Weld, D. S. (1999). Recent Advances in AI Planning. *AI Magazine*, 20(2):93.