Processos Markovianos de Decisão com heurísticas, junção de abordagens backward e forward para transferência de conhecimento baseados em políticas

R. G. Pontes and V. F. Silva

***Abstract*— This article discusses studies conducted in the area of artificial intelligence for problems developed in Abstract Markov Decision Processes, involving algorithms for feature selection, based on previous studies for greedy search strategies "backward" and "forward", adopting a mixed strategy which, despite being computationally expensive, avoids local optimum and ensures best sets of attributes. Also are discussed strategies that adopt heuristics in order to reduce this computational cost, achieving success in comparison to previous algorithms, developed without blended approach and without adopting heuristics.**

***Keywords*— Markov Decision Process, selection of features, greedy search.**

I. INTRODUÇÃO

PROCESSOS Markovianos de Decisão (em inglês *Markov Decision Process, MDPs)* são formulações muito utilizadas em ambientes de decisão sequenciais e estocásticos (Puterman, 1994; Mausam & Kolobov, 2012; Kaelbling et al., 1998). As principais características que compõem os MDPs são as transições probabilísticas entre estados, a possibilidade de se observar o estado corrente do agente de acordo com a ação executada pelo mesmo e a possibilidade de se alterar este processo, em períodos nominados como épocas de decisão. A solução para um MDP consiste na definição de uma política que mapeia para cada estado uma ação; enquanto a solução ótima maximiza as recompensas recebidas dependendo de uma função valor que especifica como recompensas devem ser acumuladas (Coelho, 2009).

Na confecção de algoritmos para a resolução de MDPs, são propostos duas abordagens para a análise dos problemas: na primeira, iteração de valor, utiliza-se a avaliação no espaço de valores, onde para cada estado mapeado é atribuído um valor, atualizado de acordo com as ações executadas e recompensas acumuladas durante as épocas de execução. A execução deste algoritmo é interrompida quando não há mais alterações significativas nos valores atribuídos, ou seja, quando a variação não passa de um determinado valor atribuído pelo desenvolvedor do algoritmo (um *threshold).* Já a outra abordagem envolve avaliação no espaço de políticas, onde também há o cálculo dos valores, mas o critério de parada do algoritmo é diferente, pois envolve a análise das melhores ações a serem tomadas em cada estado para a obtenção da melhor recompensa, de acordo com as ações realizadas e valores de recompensa coletados até então, envolvendo as interações com os estados adjacentes ao mesmo. A atualização de valores é interrompida quando, em uma determinada época de execução, não é detectada troca de ação ótima para o conjunto de estados, comparando a época de execução imediatamente anterior, evitando assim uma dependência do algoritmo de um *threshold* determinado.

A transferência de conhecimento é importante no âmbito dos problemas sequenciais resolvidos com MDPs, visto que a carga computacional utilizada para a execução de modelos de aprendizagem (tanto em abordagens no espaço de valores quanto no de políticas) é elevada. Bogdan e Silva (2012) propuseram um modelo onde, utilizando abordagens *backward* e *forward* separadamente, facilitariam o cálculo de modelos de aprendizado mais genéricos, onde o conhecimento acumulado em um determinado problema fosse repassado como um modelo para outro problema, com algumas características semelhantes e que não necessitaria passar pelo treinamento completo.

Uma abordagem para possibilitar a transferência de conhecimento é a utilização de estados abstratos. A abordagem com estados abstratos se desenvolve tendo como base a existência de atributos para um determinado espaço a ser analisado. Os atributos são características que, em conjunto, podem determinar a unicidade de um estado. Por exemplo, em um problema envolvendo um conjunto de carros podemos utilizar seu identificador único (número da placa, por exemplo) ou um conjunto de atributos que o identifique (cor, modelo, ano, marca, etc). Utilizando-se deste recurso, podemos aplicar um modelo desenvolvido a partir de características genéricas para problemas semelhantes. Por exemplo, utilizando o conjunto dos veículos, podemos estender o aprendizado desenvolvido em um conjunto para outro que apresente elementos com características semelhantes, como por exemplo esquemas de rodízio para veículos com o final da placa de número semelhante.

Dependendo do problema em questão pode ser interessante escolher um subconjunto de atividades. Primeiro, esse subconjunto pode facilitar o aprendizado em uma tarefa a partir do zero, já que diminui a quantidade de observações possíveis. Segundo, a escolha de um subconjunto pode evitar

R. G. Pontes, Universidade de São Paulo (USP), São Paulo, SP, Brasil, rodrgarcia@gmail.com

V. F. Silva, Universidade de São Paulo (USP), São Paulo, SP, Brasil, valdinei.freire@usp.br

"*overfitting*", isto é, facilitar a generalização na transferência de conhecimento entre tarefas. O conjunto não pode ter um número grande de atributos ao ponto de identificar unitariamente cada elemento do conjunto original (para não haver baixa abstração dos estados) nem ser vazio o suficiente para que não agregue os conhecimentos necessários para o desenvolvimento dos problemas a serem enfrentados. Para realizar esta análise, duas abordagens foram utilizadas por Bogdan e Silva (2012), abordagens *backward* e *forward*.

Na abordagem *backward*, são considerados inicialmente todos os atributos e, recursivamente, o algoritmo identifica o atributo mais supérfluo e o descarta, até chegar a um ponto com zero atributos. Durante a execução, ele vai analisando os valores dados aos estados, executando o algoritmo MDP com o conjunto atual daquela época de execução, e vai armazenando em um vetor, que mostra ao final a eficácia de cada conjunto de atributos selecionados em relação ao conjunto original.

Já a utilização de abordagem *forward* é inversa à abordagem *backward*, pois na mesma o conjunto é iniciado sem atributos e é computado em cada época de execução o melhor atributo a ser incluído no conjunto, ordenando o vetor de atributos desde o elemento mais relevante até o menos relevante, considerando todas as execuções realizadas.

Buscamos então, através deste artigo, analisar as características e melhorias apontadas ao se desenvolver um algoritmo que une as duas abordagens *backward* e *forward* dinamicamente executadas.

II. Arcabouço teórico

A seguir, são apresentadas definições acerca dos aspectos inerentes ao desenvolvimento deste artigo, tais como definição de um MDP (Pellegrini e Wainer, 2007), abstrações e seleção de atributos (Bogdan e Silva, 2012).

A. *Formalização de um MDP*

Os MDPs são representados formalmente por uma tupla (S, A, T, R) com os seguintes componentes: S é o conjunto total de estados que o MDP pode estar em algum determinado momento de sua execução; A é um conjunto de ações que o agente pode tomar em algum determinado momento. Vale ressaltar que uma ação pode levar a uma mudança de estado ou permanência no mesmo; T(s,a,s') é uma distribuição de probabilidade sobre a chance do sistema transitar para um estado s' estando anteriormente em um estado s através da realização de uma ação a. Se for aplicado sobre uma política não-determinística, é um valor graduado entre 0 e 1 para cada possível s', se for determinística, é mapeado diretamente a um estado s'; e R(s) é a função recompensa e retorna a recompensa quando o processo está no estado "s" (pertencente a S). Essa recompensa pode ser positiva ou negativa (penalidade).

A solução de um MDP é uma política que mapeia para cada estado uma ação; enquanto a solução ótima maximiza as recompensas recebidas dependendo de uma função valor que especifica como as recompensas devem ser acumuladas. Usualmente $V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t V_t | \pi, S_0 = S]$ e π^* é ótimo se $V^{T^{\pi}}(s) > V^{T}(s)$ para todo $s \in S$ e toda política π.

B. *Transferência de conhecimento em MDPs*

Os MDPs podem ser descritos não apenas como um conjunto de estados e ações, seus estados também podem ser desmembrados em estruturas menores conhecidas como atributos. Os atributos possibilitam uma abstração do problema a ser modelado como um MDP de tal forma que possibilita a transferência de conhecimento entre problemas semelhantes (Otterlo, 2004).

Os estados podem ser abstraídos ao considerar um conjunto de atributos binários $F = \{f_1, f_2, ..., f_n\}$, que são atributos relevantes ao conjunto S de estados, tal que a tupla $f_i : s \rightarrow [0,1]$ indica a presença do atributo f_i no estado s pertencente ao conjunto S. Já σ é um estado pertencente ao conjunto S_σ, que é o conjunto de estados abstratos, ou seja, identificados pela presença de seus atributos. Cada σ é único e mapeado não através de um valor numérico ou índice, mas sim pelo conjunto $F(\sigma)$ de atributos que o compõe; e cada estado concreto s pode ter um estado σ atribuído a ele.

Utilizando estes atributos, pode-se realizar a transferência de conhecimento entre dois problemas desenvolvidos sob a forma de um MDP, pois durante a execução dos algoritmos de iteração de valores ou de políticas, ao invés de se determinar um valor para o estado atual s, é atribuído o valor ao estado abstrato σ e a todos os atributos que o compõe ($f_1, f_2 .... f_n$), formulando uma política ótima para os mesmos. Assim, em caso onde seja necessário comparar dois problemas com atributos semelhantes, caso seja aplicável, o aprendizado realizado sobre um conjunto de dados é relevante na solução do problema a ser solucionado no outro conjunto de dados.

Análogo ao caso dos estados abstratos também temos, no conjunto dos MDPs com características abstratas, ações abstratas. Estas ações $\alpha \in A_{ab}$ induzem a um conjunto de ações A_α. Como para cada estado $s \in A$ são permitidas a execução apenas de ações do conjunto A_s, também há uma restrição a quais ações abstratas pertencentes a A_{ab} possam ser executáveis pelos estados abstratos. A regra para a validade de execução de uma ação abstrata é dada pela seguinte fórmula:

$$A_\sigma = \{\alpha | \alpha \in A_{ab} \wedge \forall s \in S_\sigma \exists \alpha \in A_\alpha a \in A_s\}$$

Assim, cada ação abstrata α poderá ser executada por um estado abstrato σ apenas se for possível correlacioná-lo a pelo menos uma ação não abstrata a representada pela ação abstrata α (ou seja, uma ação $a \in A_\alpha$), em relação a cada estado s correlacionado ao estado abstrato σ (ou seja, um estado $s \in S\sigma$).

Um exemplo prático disso é a utilização do aprendizado para a resolução de problemas envolvendo navegação robótica, onde atributos comuns à tarefa são levados em conta durante o treinamento e confecção da política (proximidade a objetos, como porta e corredores; distância entre o robô e a porta, noções de localização em geral, se o mesmo está distante do objeto, se está próximo da meta, etc.). A política obtida então é aplicada sobre o outro ambiente não avaliado anteriormente, porém como o treinamento foi realizado sobre a abstração de estados através de atributos comuns, o que foi

aprendido durante o treinamento pode ser aproveitado no outro ambiente.

Para realizar a transferência de conhecimento, deve-se considerar políticas abstratas π_{ab} que mapeia estados abstratos $\sigma \in S_{ab}$em ações abstratas $\alpha \in A_{ab}$.

Como através desta abordagem existem casos onde não existam ações abstratas possíveis de serem realizadas por certos estados abstratos, é recomendado que, durante a implementação do problema, seja implementada para todos os estados abstratos uma ação abstrata válida α_R, onde em caso de sua execução seja realizada uma ação válida qualquer dente as presentes em A_s, para que assim não haja problemas durante a execução.

A partir da formulação de suas componentes, podemos declarar uma política estocástica abstrata representada por π_{ab} como sendo $\pi_{ab} : S_{ab}$ x $A_{ab} \rightarrow [0,1]$, onde após recebeu um estado abstrato σ é executada uma ação abstrata α com probabilidade $\pi_{ab}(\sigma,\alpha)$ variando entre 0 (não será executada) e 1 (será executada com 100% de certeza).

C. Formalização do algoritmo AbsProb-PI

O algoritmo AbsProb-PI é um algoritmo que implementa a iteração de políticas baseado no gradiente (Silva et al., 2012). É um algoritmo importante, pois, através do mesmo, é possível localizar políticas abstratas localmente ótimas, em relação a recompensa acumulada.

Define-se uma matriz de transição $T^{\pi_{ab}}(s,s')$ e uma função-valor $V^{\pi_{ab}}(s)$ como segue:

$$T^{\pi_{ab}}(s,s') = \sum_{\alpha \in A_{ab}} \pi_{ab}(\alpha|\sigma(s)) * \sum_{a \in A_\alpha \cap A_s} \frac{1}{|A_\alpha \cap A_s|} T(s,a,s1),$$

$$V^{\pi_{ab}} = (\mathrm{I} - \gamma T^{\pi_{ab}})^{-1} r,$$

aonde r, $T^{\pi_{ab}}$ e $V^{\pi_{ab}}$ são representações vetoriais da função recompensa $r(s)$, da matriz de transição $T^{\pi_{ab}}(s,s')$ e da função valor de estados $V^{\pi_{ab}}(s)$, respectivamente.

Análoga à matriz $T^{\pi_{ab}}$, é definida uma matriz de transição $T^{\alpha,\varepsilon}$, responsável por realizar a escolha de execução entre uma ação abstrata α com probabilidade de ocorrência 1 – ε e o conjunto das outras ações abstratas possíveis de execução com probabilidade ε, sendo ε>0. Também definimos uma função de desconto gradual $\delta(i) = (1 + \frac{i}{k})^{-1}$, onde i é a iteração de execução do algoritmo e k é uma constante. Considerando b^0 como sendo a representação vetorial da distribuição inicial de $b^0(s)$, os passos do algoritmo AbsProb-PI são descritos desta maneira, de acordo com o que foi definido por Silva et al. (2012):

1. Inicialize a política abstrata probabilística π_{ab} aleatoriamente, tal que:

$$\pi_{ab}(\sigma,\alpha) \geq \frac{\varepsilon}{|A_{ab|}} \forall \sigma \in S_{ab}, \alpha \in A_{ab} \sum_{a \in A_{ab}} \pi_{ab}(\sigma,\alpha) = 1 \forall \sigma \in S_{ab}$$

2. A cada iteração i realizar as seguintes etapas:

(a) Calcular o valor da função $V^{\pi_{ab}}$

(b) Calcular o produto de $C = \gamma b^{o\mathrm{T}}(\mathrm{I} - \gamma \mathrm{T}^{\pi_{ab}})^{-1}$

(c) Para cada $\alpha \in A_{ab}$ calcular $\Delta^{\alpha,\pi_{ab}} = (\mathrm{T}^{\alpha,\varepsilon} - \mathrm{T}^{\pi_{ab}})\mathrm{V}^{\pi_{ab}}$

(d) Para cada $\sigma \in S_{ab}$ e cada $\alpha \in A_{ab}$ calcular $W(\sigma,\alpha) = \sum_{s \epsilon S_\sigma} C(s)\Delta^{\alpha,\pi_{ab}}(s)$

(e) Para cada $\sigma \in S_{ab}$ procurar a melhor direção $\alpha_\sigma^* = \arg\max_{\alpha \in A_{ab}} W(\sigma,\alpha)$

(f) Selecionar um valor de desconto $\delta(i)$ e atualizar a política π_{ab} da seguinte maneira:

$$\pi_{ab}(\sigma,\alpha) \leftarrow \begin{cases} (1-\delta(i))\pi_{ab}(\sigma,\alpha) + \delta(i)\left(\frac{\varepsilon}{|A_{ab}|} + (1-\varepsilon)\right), \text{if } \alpha = \alpha_\sigma^* \\ (1-\delta(i))\pi_{ab}(\sigma,\alpha) + \delta(i)\frac{\varepsilon}{|A_{ab}|}, \text{if } \alpha \neq \alpha_\sigma^* \end{cases}$$

D. Abordagens para seleção de atributos

Bogdan e Silva (2012) definem quatro tipos de algoritmos diferentes para seleção de atributos no âmbito dos MDPs abstratos, baseadas em duas abordagens. Uma é conhecida como *backward*, e percorre o conjunto de atributos de um MDP desde cheio (com todos os atributos) até o conjunto vazio (sem nenhum atributo). Já o outro é conhecido como *forward* e realiza o inverso do *backward*, ou seja, percorre desde o conjunto vazio de atributos até um conjunto com todos os atributos. Também existem duas variantes destas abordagens, *backward* heurístico e *forward* heurístico, que se utilizam de heurística para reduzir substancialmente o tempo de execução dos algoritmos. Todos serão discutidos minuciosamente a seguir.

Os quatro algoritmos dependem de uma função de avaliação $V(F)$ que avalia o conjunto de atributos *F*.

Para a obtenção do valor *V* do conjunto F_{f-} (*F* menos o atributo *f*), Bogdan e Silva (2012) propõem dois métodos distintos, um denominado *full backward*, executa uma avaliação plena do valor dos atributos e outro, denominado *backward* por estimativa de peso heurística (ou apenas "*backward* com heurísticas", para simplificar), que pode até ter $[\max\{\frac{N_{passos}}{2}|F_{max} - 2\}]$ vezes menor custo do que a execução plena do método *full backward*.

O método *full backward* envolve a avaliação de um conjunto de atributos pelo valor da política ótima abstrata no conjunto de estados abstratos $S_{ab}^{F_{f-}}$. O valor determinado do conjunto de atributos seguirá a seguinte definição, envolvendo a execução do algoritmo AbsProb-PI:

$\hat{\pi}_{ab}^* \leftarrow \mathrm{AbsProb{-}PI}\left(S_{ab}^{F_{f-}}, N_{passos}\right), V(F_{f-}) = \mathrm{b}^{0\mathrm{T}}(\mathrm{I} - \gamma \mathrm{T}^{\hat{\pi}_{ab}^*})^{-1}\mathrm{r}$.

É desta abordagem que deriva o *backward* com heurísticas. A diferença consta na menor quantidade de cálculos computacionais que o mesmo executa, pois utiliza o vetor C que o algoritmo AbsProb-PI cria, vetor este que representa o desconto acumulado esperado para o valor de recompensa obtido em cada $s \in S$ do MDP seguindo a política abstrata π_{ab}. Esta heurística consiste em pegar estes valores do vetor C para criar uma estimativa do peso de cada estado baseado nas políticas anteriores. A política $\hat{\pi}_{ab}^{F_{f-}}$ é determinada pela

equação $\hat{\pi}_{ab}^{F_{f-}}(\sigma,\alpha) = \frac{\sum_{s\in S_\sigma} C(s)\hat{\pi}_{ab}^{F}(\sigma^F(s),\alpha)}{\sum_{s\in S_\sigma} C(s)}$, onde a política $\hat{\pi}_{ab}^{F_{f-}}$ é obtida através da execução do algoritmo AbsProb-PI sobre o conjunto original de atributos F e a função $\sigma^F: S \rightarrow S_{ab}^F$ mapeia cada estado original no conjunto de estados abstratos.

Após a obtenção da política $\hat{\pi}_{ab}^{F_{f-}}$, podemos definir os valores de $V(F_{f-})$ como sendo $V(F_{f-}) = \mathrm{b}^{0\mathrm{T}}(\mathrm{I} - \gamma \mathrm{T}^{\hat{\pi}_{ab}^{F_{f-}}})^{-1}\mathrm{r}$.

Como fizeram para a obtenção dos valores $V(F_{f-})$ dos atributos no algoritmo *backward*, Bogdan e Silva (2012) também buscaram a obtenção dos valores do conjunto F_{f+} (F mais o atributo f) desenvolvendo dois métodos distintos, um denominado *full forward*, que executa uma avaliação plena do valor dos atributos e outro, denominado *forward* por estimativa heurística baseada em gradientes (ou apenas "*forward* com heurísticas", para simplificar) que, assim como o algoritmo *backward* com heurísticas faz em relação ao *full backward*, pode até ter $[\max\left\{\frac{N_{passos}}{2}|F_{max} - 2\right\}]$ vezes menor custo do que a execução plena do método *full forward*.

O cálculo heurístico do método *forward* com heurísticas envolve o gradiente $W(\sigma,\alpha)$ calculado pela execução do algoritmo AbsProb-PI, sendo que este gradiente calculado é a forma representada de como (e, quantitativamente, quanto) a política corrente pode ser melhorada com a adição de um atributo. Dada a política corrente $\hat{\pi}_{ab}^{F}$ obtida na execução do algoritmo AbsProb-PI, a estimativa heurística baseada em gradiente define o valor dado a um conjunto de atributos F_{f+} da seguinte forma:

$$V(F_{f+}) = \sum_{\sigma\in S_{ab}^{F_{f+}}} \max_{\alpha\in A_{ab}} W(\sigma,\alpha)\,,$$

E. Execução gulosa dos algoritmos backward e forward

Os algoritmos *backward* e forward têm em comum a característica de serem executados de forma gulosa, considerando a iteração atual como sendo a última. Porém esta escolha gulosa acaba por minar a qualidade do conjunto em um todo, gerando uma solução subótima. Por exemplo na execução de um algoritmo *backward* onde um atributo que, de início, não fosse muito relevante ao problema, fosse retirado, poderia ganhar considerável importância a partir de outras iterações, considerando outras combinações de atributos, mas por ser uma execução gulosa do algoritmo este atributo nunca irá adentrar no conjunto de atributos posteriormente.

Para minimizar tal problema propomos uma abordagem mista na resolução de MDPs, alternando a execução de "funções *backward*" e "funções *forward*" no mesmo algoritmo, dando a chance de que certos atributos que foram descartados (ou adicionados precocemente) possam ser readmitidos (ou eliminados, respectivamente), caso haja relevância do atributo no conjunto naquela iteração corrente.

III. ABORDAGENS MISTAS PARA SELEÇÃO DE ATRIBUTOS

Nossas propostas de abordagens mistas baseiam-se nos algoritmos *backward* e *forward* já descritos anteriormente, e suas variantes com heurísticas.

Os experimentos desenvolvidos por Bogdan e Silva (2012) predispõem-se a realizar as iterações sobre os conjuntos de atributos de maneira isolada, ou seja, uma execução completa para o algoritmo *backward*, outra para o algoritmo *forward* e uma para cada variante dos dois algoritmos supracitados. Assim, pode-se realizar uma análise em separado da eficácia de cada abordagem por si só. Sob o ponto de vista da eficácia da seleção dos atributos, é uma alternativa válida, mas podem ocorrer problemas devida a natureza gulosa dos algoritmos, como já foi citado anteriormente.

A nossa proposta envolve, num mesmo algoritmo, a utilização de sequências alternadas de execução das duas modalidades de abordagem para seleção de atributos (*backward* e *forward*, com ou sem heurísticas). O mesmo pode iniciar com conjunto de atributos vazio ($F \leftarrow \emptyset$) e iterar inicialmente utilizando a abordagem *forward*, cuja abordagem nomeamos como algoritmo *Forward-Backward*; ou iterar sobre um conjunto de atributos inicialmente cheio ($F \leftarrow F_{max}$) e iterar inicialmente utilizando a abordagem *backward*, cuja abordagem nomeamos como algoritmo *Backward-Forward*. Estas mesmas proposições valem para os algoritmos baseados em heurísticas, nomeados respectivamente de algoritmo *Forward-Backward* com heurísticas e *Backward-Forward* com heurísticas, respectivamente.

A seguir, os passos de execução para a abordagem *Forward-Backward*:

1. Inicialize o conjunto de atributos $F \leftarrow \emptyset$, ou seja, com nenhum atributo contido no conjunto F.

2. Enquanto o valor do complemento de F for maior do que 1 ($|F^C > 1|$), ou seja, ainda existirem atributos a serem acrescentados ao conjunto F, fazer o seguinte:

(a) Para todos os atributos f pertencentes à F^C, simular a adição do mesmo no conjunto F, construindo o conjunto $F_{FB} = F \cup \{f\}$ e avaliar este novo conjunto, utilizando-se do algoritmo AbsProb-PI para calcular o valor $V(F_{FB})$.

(b) Encontrar o atributo f^a mais relevante do conjunto de estados abstratos, através da função $f^a = \arg\max_{f\in(F^C)} V(F_{FB})$, e adicioná-lo ao conjunto F, fazendo $F = F \cup \{f^a\}$ e eliminando f^a do conjunto F^C.

(c) Armazenar f^a em um vetor F_{add}

(d) Para todos os atributos f pertencentes à F^C, simular a remoção do mesmo no conjunto F, construindo o conjunto $F_{FB} = F_{FB} - \{f\}$ e avaliar este novo conjunto, utilizando-se do algoritmo AbsProb-PI para calcular o valor $V(F_{FB})$

(e) Encontrar o atributo f^b menos relevante do conjunto de estados abstratos, através da função $f^b = \arg\max_{f \in F} V(F_{FB})$.

(f) Caso $f^a \neq f^b$, eliminar o atributo f^b do conjunto F e do vetor F_{add} e repetir os passos (d), (e) e (f) até $f^a = f^b$

3. Como resultado, retornar a lista F_{add} de atributos adicionados, ordenados por relevância.

Segue os passos de execução para a abordagem *Backward-Forward*:

1. Inicialize o conjunto de atributos $F \leftarrow F_{max}$, ou seja, com todos os atributos contidos no conjunto F.

2. Enquanto o valor de $|F| > 1$, ou seja, ainda existirem atributos no conjunto F, fazer o seguinte:

(a) Para todos os atributos f pertencentes à F, simular a remoção do mesmo, construindo o conjunto $F_{BF} = F - \{f\}$ e avaliar este novo conjunto, utilizando-se do algoritmo AbsProb-PI para calcular o valor $V(F_{BF})$.

(b) Encontrar o atributo f^a menos relevante do conjunto de estados abstratos, através da função $f^a = \arg\max_{f \in F} V(F_{FB})$, e eliminá-lo do conjunto F, fazendo $F = F - \{f^a\}$.

(c) Armazenar f^a em um vetor F_{el}

(d) Para todos os atributos f pertencentes à F, simular a inserção do mesmo no conjunto F, construindo o conjunto $F_{BF} = F_{BF} \cap \{f\}$ e avaliar este novo conjunto, utilizando-se do algoritmo AbsProb-PI para calcular o valor $V(F_{BF})$

(e) Encontrar o atributo f^b mais relevante do conjunto de estados abstratos, através da função $f^b = \arg\max_{f \in (F^C)} V(F_{BF})$.

(f) Caso $f^a \neq f^b$, adicionar o atributo f^b ao conjunto F e ao vetor F_{el} e repetir os passos (d), (e) e (f) até $f^a = f^b$

3. Como resultado, retornar a lista F_{el} de atributos, ordenados por sua relevância.

Os mesmos passos são válidos para os algoritmos *Forward-Backward* com heurísticas e *Backward-Forward* com heurísticas, sendo que, assim como os algoritmos originais desenvolvidos por Bogdan e Silva (2012), a porção referente à ação *forward* destes algoritmos utiliza-se do vetor C (desconto acumulado esperado para o valor de recompensa obtido em cada $s \in S$ do MDP seguindo a política abstrata π_{ab}) que o algoritmo AbsProb-PI desenvolve para compor a política abstrata $\hat{\pi}_{ab}^{F_{f-}}$ e, em seguida, desenvolver os valores de $V(F_{f-})$, conforme seguem fórmulas:

$$\hat{\pi}_{ab}^{F_{f-}}(\sigma,\alpha) = \frac{\sum_{s \in S_\sigma} C(s)\hat{\pi}_{ab}^{F}(\sigma^F(s),\alpha)}{\sum_{s \in S_\sigma} C(s)}, V(F_{f-}) = \mathrm{b}^{0\mathrm{T}}(\mathrm{I} - \gamma \mathrm{T}^{\hat{\pi}_{ab}^{F_{f-}}})^{-1}\mathrm{r}$$

O mesmo é válido para a porção referente à ação *backward*, que utiliza-se do vetor de gradientes $W(\sigma,\alpha)$ e da política atual $\hat{\pi}_{ab}^{F}$ desenvolvidos pela execução do algoritmo AbsProb-PI, conforme segue:

$$V(F_{f+}) = \sum_{\sigma \in S_{ab}^{F_{f+}}} \max_{\alpha \in A_{ab}} W(\sigma,\alpha)$$

Como percebe-se pela composição dos algoritmos, a cada execução dos algoritmos *Backward-Forward* e *Forward-Backward* são executados os núcleos dos seus algoritmos de inspiração, tanto para as versões sem heurística quanto para as versões heurísticas, em relação a seus pares de origem. Portanto, a execução sempre será de, no mínimo, duas vezes a execução de uma das abordagens de origem; por exemplo, uma execução do algoritmo *Backward-Forward* será, no mínimo, duas vezes mais demorada do que a execução do algoritmo *Backward* somente. O que compensa esta abordagem mais demorada é que os resultados de otimização do conjunto de atributos permite melhores resultados do que a execução de uma abordagem não mista.

IV. Experimentos e Resultados

Para a realização de experimentos avaliativos, desenvolvemos no ambiente de desenvolvimento Matlab os algoritmos propostos por Bogdan e Silva (2012) *backward*, *forward*, *backward* com heurísticas e *forward* com heurísticas, além de nossas propostas realizadas com as abordagens mistas dos mesmos.

A. *Descrição dos ambientes utilizados*

Foi escolhido um problema simplificado de navegação robótica, onde o agente tem como ações disponíveis para a exploração "caminhar ao norte", "caminhar ao sul", caminhar a leste" e "caminhar a oeste", com 90% de chances de executar a ação e 10% de ir às outras direções, garantindo assim que, mesmo que custosamente, sempre o algoritmo tenha chances de chegar a sua meta proposta.

Foram desenvolvidos dois ambientes, "Original" e "Teste".

O ambiente original é composto por um conjunto ordenado de 273 possíveis estados, sendo apenas 155 alcançáveis por exploração. São subdivididos em estados "parede", "porta", "corredor" e "sala", conforme a imagem 1.

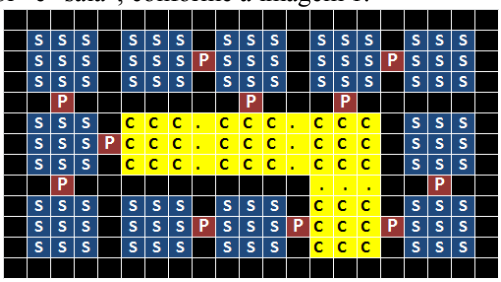


Imagem 1. Ambiente original de exploração, onde os estados marcados com "S" são pertencentes a uma sala, "C" ou "." a um corredor e "P" são portas.

Já o ambiente de testes envolve a mesma formulação do ambiente original, porém é composto de mais estados

possíveis (total de 1025, sendo 650 exploráveis), conforme a imagem 2.

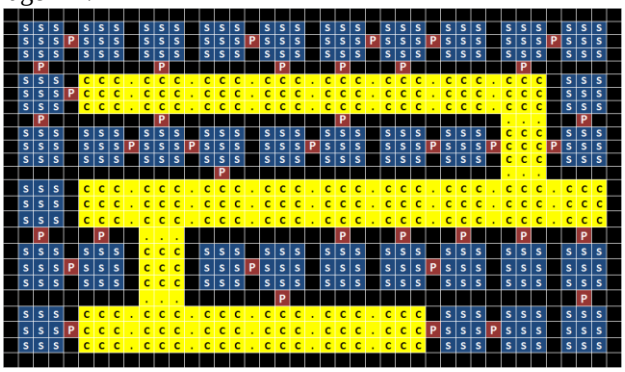


Imagem 2. Ambiente de exploração "teste", onde os estados marcados com "S" são pertencentes a uma sala, "C" ou "." a um corredor e "P" são portas.

No ambiente original, são designados 11 estados-meta, ou seja, são pontos no mapa (geralmente centros de salas) designados como uma tarefa a ser realizada, e a meta do explorador é justamente chegar até este ponto, utilizando suas ações padrão (caminhar ao norte, ao sul, a leste e a oeste), sendo que enquanto ele for explorando vai perdendo 1 ponto por iteração, forçando o algoritmo a procurar os estados-meta no menor tempo, visando a menor perda de pontos possíveis. Já no ambiente de teste o número de estados-meta aumenta para 35, e os valores de pontos perdidos por iteração e de alcance da meta mantém-se equivalentes em relação ao ambiente original.

O MDP descrito tem como atributos o mesmo conjunto descrito por Bogdan e Silva (2012) para a resolução dos algoritmos *backward* e *forward*, como segue:

TABELA 1. VARIÁVEIS REPRESENTANDO OS ATRIBUTOS (FEATURES)

Feature	Notation
see_empty_space	Es
see_door_far	Df
see_room	Sr
see_corridor	Sc
see_ambience	Sa
see_anything	An
in_room	In
near_to_goal	Ng
almost_near_to_goal	Ang

As propriedades de cada atributo são as seguintes:

- see_empty_space (Es): é um atributo que, quando ativo, representa que no estado corrente o explorador vê um espaço alcançável e vazio;

- see_door_far (Df): é um atributo que, quando ativo, representa que no estado corrente o explorador vê uma porta a mais de 3 passos de distância;

- see_room (Sr), see_corridor (Sc) e see_ambience (As) são atributos que, ao verificar proximidade dos itens citados (porta, corredor e ambiente/sala) retorna positivo;

- see_anything (An) é um atributo que retorna positivo quando see_room, see_corridor ou see_ambience retorna positivo;

- in_room (In): quando ativo, representa que o estado corrente está localizado dentro de uma sala;

- near_to_goal (Ng): quando ativo, representa que, no estado corrente, o explorador está a 5 passos (ou menos) da meta;

- almost_near_to_goal (Ang): quando ativo, representa que, no estado corrente, o explorador está entre 8 e 5 passos da meta.

Além dos atributos representados acima, para o conjunto de atributos {Es, Df, Sr, Sc, Sa, An} são considerados duas variantes, conforme nomenclatura adotada: (atributo)*_opposite_to_goal* e (atributo)*_ahead_to_go*. Assim, totalizamos 15 atributos a serem avaliados nos dois ambientes.

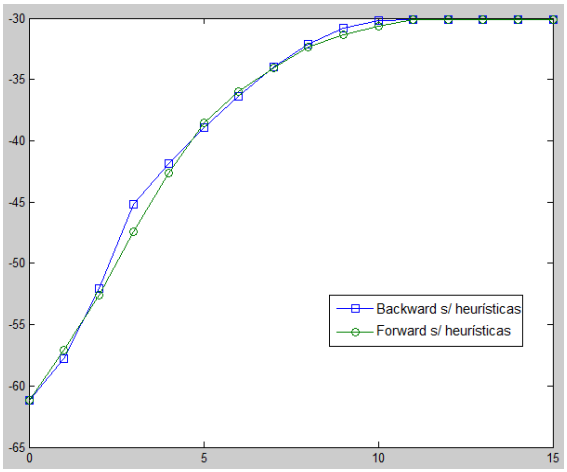
A descrição de dois ambientes, Original e Teste, foi necessária para a avaliação dos algoritmos em relação a eficácia dos conjuntos de atributos e políticas determinadas encontrados durante a etapa de aprendizado no ambiente Original para as tarefas designadas a ocorrerem no ambiente Teste; e pela proposição dos ambientes reflete a característica de utilização dos métodos de resolução de problemas com MDPs na realidade, onde em um ambiente simulado e pequeno são realizados os testes (para diminuição de custos computacionais), para posteriormente ter o aprendizado aplicado num ambiente de tamanho maior e com grau de complexidade igualmente grande.

B. Execução dos testes

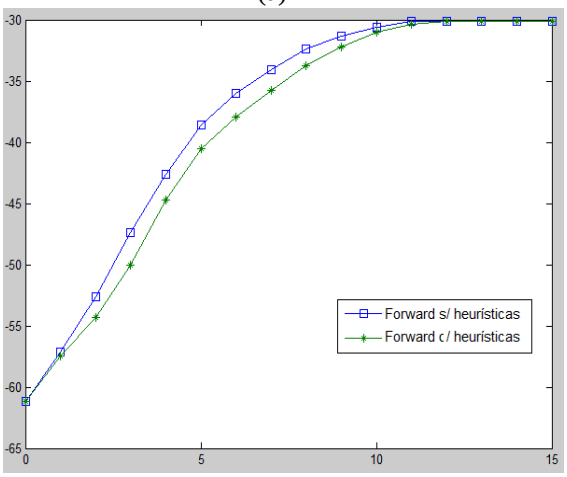
Os primeiros testes avaliaram os algoritmos baseados nos propostos por Bogdan e Silva (2012) em nossos dois ambientes propostos (Original e Teste). Para a execução do algoritmo AbsProb-PI determinamos o n_{passos} = 1000; ou seja, 1000 iterações a serem executadas e, em cada iteração, todos os estados-meta são considerados nos ambientes.

Para a realização das avaliações, foram realizados 20 testes para cada conjunto (Original e Teste), onde são selecionados metade dos estados-alvo para treinamento do conjunto e a outra metade é utilizada para avaliação das políticas e atributos.

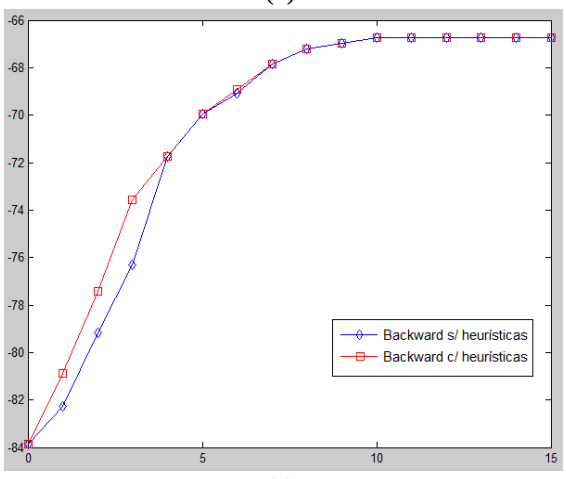
Na imagem 3, são apresentados os resultados das execuções dos algoritmos, sendo que, na escala horizontal, quanto mais próximo da extremidade direita, maior o número de atributos no conjunto de atributos considerado (F) e o valor vertical representa o valor do conjunto atual de atributos considerado (quanto maior o valor, melhor é o conjunto).



(a)



(b)



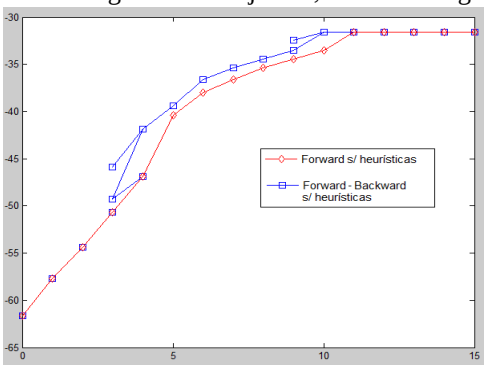
(c)

Imagem 3. Média das avaliações dos algoritmos forward com e sem heurísticas no ambiente Original (a) e comparativo de eficácia da política prescrita pelos algoritmos com e sem heurística na abordagem forward (b) e backward (c) no ambiente original, após 20 execuções.

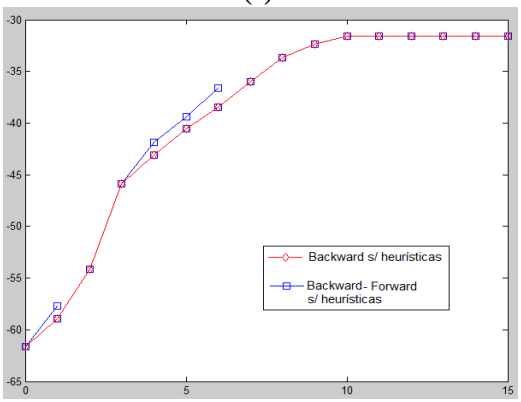
Os resultados obtidos nos gráficos da imagem 3 são semelhantes aos obtidos por Bogdan e Silva (2012) em seus experimentos, salvo as devidas adaptações referentes aos ambientes de testes e parâmetros de execução utilizados, que são ligeiramente diferentes entre si; por exemplo o valor adotado de n_{passos}, que nos testes da proposta original fora utilizado com o valor 280 e, em nossa proposta, valor 1000.

Conforme os resultados das execuções anteriores foram se adequando ao já estudado, partimos para a execução dos algoritmos próprios, tanto as versões baseadas em heurísticas quanto as que não eram baseadas neste modelo.

Através da análise dos gráficos fica visível a melhora de desempenho dos algoritmos conjuntos, conforme segue:



(a)



(b)

Imagem 4. Média das avaliações dos conjuntos de atributos com os algoritmos Forward comparado ao algoritmo Forward-Backward (a) e Backward comparado ao algoritmo Backward-Forward (b), no ambiente Original.

Conforme verificado tanto no gráfico (a) quanto no gráfico (b) da imagem 4, os algoritmos de abordagem mista realizam alternativamente a inserção e remoção de atributos, dependendo do valor do conjunto de atributos que as fórmulas de *backward* e *forward* fornecem adicionando e retirando elementos diversos do conjunto. Isto faz com que haja uma menor influência do fator limitante de eficiência aplicado pela abordagem gulosa dos dois algoritmos propostos por Bogdan e Silva (2012), que levava a diversos ótimos locais que nem sempre representavam a melhor opção para determinado conjunto de atributos selecionados, algo facilmente visível comparando, no gráfico (b) da imagem 4, o conjunto retornado pelo algoritmo *backward* sem heurísticas comparado ao algoritmo misto *Backward-Forward* no conjunto de seis atributos, onde verifica-se uma diferença de eficácia entre os dois conjuntos propostos de cerca de 1,5 ponto a mais para a abordagem mista.

TABELA 2. TEMPO DE EXECUÇÃO DOS ALGORITMOS ABORDADOS NO AMBIENTE ORIGINAL

Algoritmo	Tempo de execução (seg)
Backward sem heurísticas	1968,3 segundos
Forward sem heurísticas	1941,7 segundos
Backward com heurísticas	655,43 segundos
Forward com heurísticas	632,77 segundos
Backward-Forward sem heurísticas	5734,4 segundos
Forward-Backward sem heurísticas	5123,6 segundos
Backward-Forward com heurísticas	1509,2 segundos
Forward-Backward com heurísticas	1370,3 segundos

A utilização de abordagem mista demanda em média 2,77 vezes mais tempo de execução, comparada aos algoritmos *backward* e *forward* separadamente. Porém a utilização de heurísticas combinada a abordagens mistas consegue ser reproduzida, em média, em 73,6% do tempo que seria demandado para a execução dos algoritmos *backward* e *forward* separadamente, e em comparação a seus pares mistos sem heurística, este índice chega a promissores 26,53% de tempo de execução (ver tabela 2).

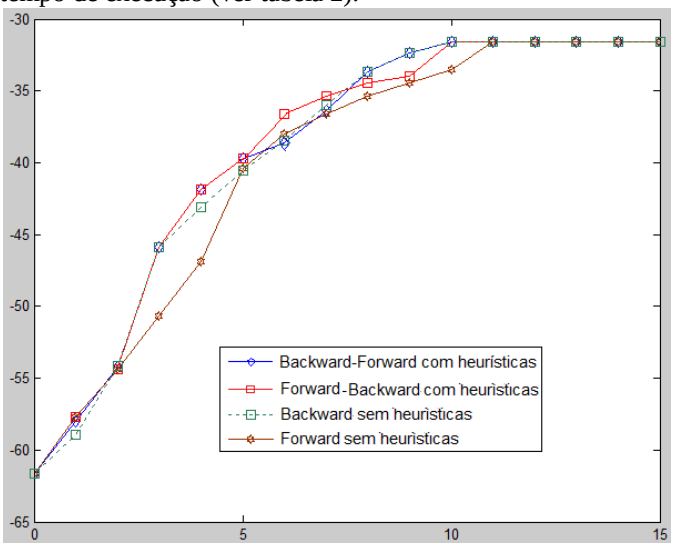


Imagem 5. Média das avaliações dos conjuntos de atributos com quatro modelos de abordagens do algoritmo de seleção de atributos.

Na imagem 5 pode-se ver a comparação entre a eficácia dos algoritmos mistos com heurísticas aos algoritmos *backward* e *forward* sem heurística, separadamente. Como podemos ver no gráfico, especialmente na faixa de 4 atributos selecionados, os algoritmos combinados, mesmo que se valendo de heurísticas para agilizar o processo de cômputo dos valores das políticas e atributos selecionados, conseguem ser melhores do que as abordagens *full* dos algoritmos *Backward* e *Forward* para certos conjuntos, e com poucos ou muitos atributos selecionados as abordagens mistas conseguem ser equivalentes às abordagens *full* supracitadas. Considerando que o tempo demandado para a execução de uma iteração dos algoritmos mistos com heurísticas seja menor do que o de *full Backward* e de *full Forward* e apresentam resultados satisfatórios, é considerada válida a utilização dos mesmos para solucionar problemas referentes a seleção de atributos em MDPs abstratos.

A imagem 6 compara a diferença entre a pior abordagem em um determinado conjunto de atributos e as outras abordagens realizadas. O gráfico têm as mesmas propriedades do gráfico da imagem 5, ou seja, na escala horizontal são representadas as quantidades de atributos naquela iteração e, na vertical, a diferença em pontos entre a pior abordagem e as outras.

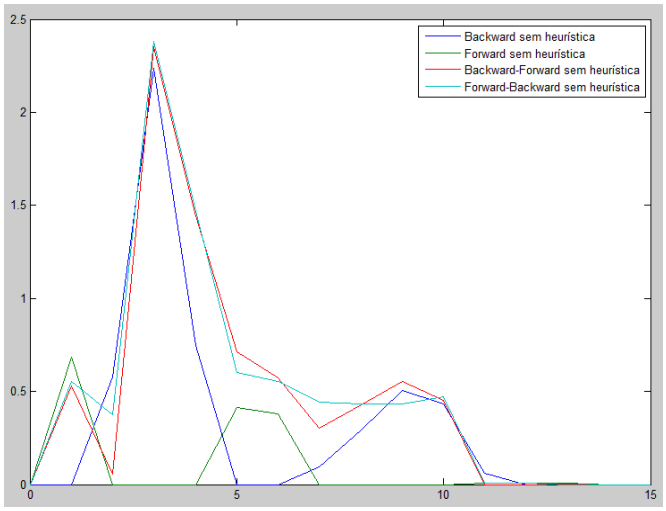
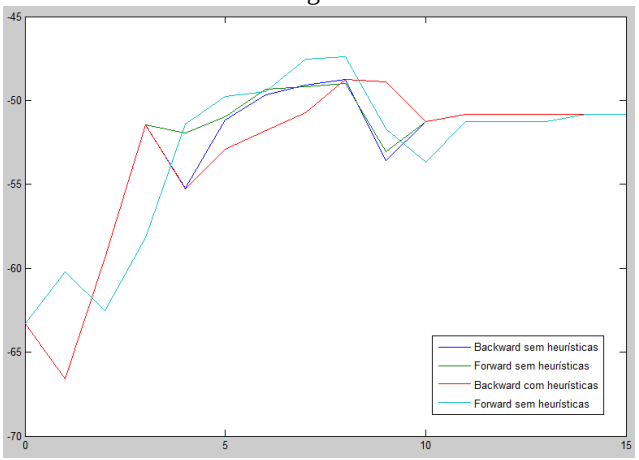


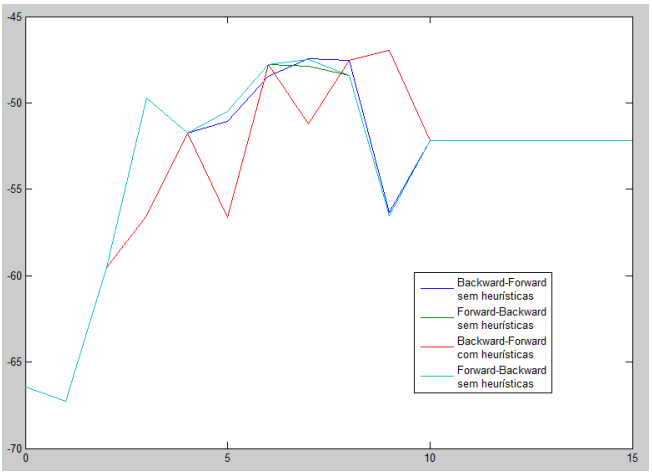
Imagem 6. Diferença computada entre a pior abordagem em um determinado conjunto de atributos e as outras abordagens, através da média de valores de políticas e atributos de 20 execuções para cada quantidade de atributos.

Com os dados apresentados, fica mais fácil notar que, na maior parte dos casos analisados, os algoritmos mistos são melhores do que os algoritmos *full*, sem heurísticas; por exemplo no conjunto entre 3 e 8 atributos ininterruptamente as abordagens mistas com heurísticas obtiveram melhor avaliação e menor custo computacional de tempo do que suas contrapartes.

No âmbito da transferência de conhecimento, foram realizados testes de migração da política computada no ambiente Original e avaliada no ambiente Teste. Para isso, foram realizadas duas execuções do algoritmo para cada valor de conjunto de atributos (de 0 a 15) e computada a média. Os resultados são exibidos na imagem 7.



(a)



(b)

Imagem 7. Valor do conjunto de atributos calculado no conjunto Original e aplicada no conjunto Teste para as oito abordagens estudadas, quatro com heurísticas e quatro sem heurísticas, sendo que em (a) são retratados os algoritmos não mistos e, em (b), as abordagens mistas.

É importante salientar que, apesar da imprecisão dos dados colhidos, devido a execução de apenas duas iterações dos algoritmos, os mesmos já apresentam comportamento semelhante ao observado por Bogdan e Silva (2012) em relação aos algoritmos propostos por eles. No caso, a partir de, aproximadamente, 2/3 da quantidade total de atributos, os algoritmos não conseguem executar mais uma política satisfatória de transferência de conhecimento entre os problemas, pois há *overfitting*. Já em relação à faixa entre 1/3 e 2/3 do total de atributos, temos uma variância entre as abordagens dos algoritmos e, com exceção de dois conjuntos de atributos (com 5 e 9 atributos, no exemplo do gráfico (b) da imagem 7), sempre obtêm uma política melhor do que a partir de 10 atributos (2/3 do total).

V. Conclusão

Apresentamos dois algoritmos que melhoram a abordagem anterior, no âmbito de seleção de atributos, ao agregar uma abordagem mista entre a técnica *backward* e *forward* propostos separadamente por Bogdan e Silva (2012) em estudos anteriores. Também realizamos experimentos para avaliar a qualidade das políticas apontadas pelos algoritmos criados e realizamos comparações de eficácia em relação aos algoritmos não mistos, obtendo uma melhor avaliação em relação aos mesmos, mesmo considerando a situação de comparação entre algoritmos mistos heurísticos e algoritmos não mistos e não heurísticos, onde obtivemos melhor qualidade da política gerada em um menor tempo de execução.

Como já foi analisada a eficácia dos conjuntos de abordagens mistas em relação aos algoritmos não mistos, é interessante realizar em futuras pesquisas a avaliação dos conjuntos de atributos obtidos no conjunto Original aplicados para treinamento e avaliação no conjunto de Testes e um maior refinamento na análise da migração da transferência de políticas entre os dois conjuntos bidirecionalmente (treinamento em Original e avaliação em Teste e vice versa).

Referências

Coelho, L. G. (2009) “Processo de Decisão Markoviano e Aprendizado por Reforço”. PCS 5019 – Probabilistic Methods in Robotics ans Vision, USP, Brasil.

Pellegrini, J., Wainer, J. (2007) “Processo de Decisão de Markov: um tutorial”. Instituto de Computação – Unicamp, Brasil.

Silva, V. F., Pereira, F., Costa, A. H. (2012) “Finding memoryless probabilistic relational policies for inter-task reuse”, Universidade de São Paulo, Brasil.

Bogdan, K., Silva, V. (2012) “Backward and Forward Feature Selection for Knowledge Transfer between MDPs”, Universidade de São Paulo, Brasil.

Otterlo, M. V. (2004) “Reinforcement Learning for Relational MDPs”. Em: Machine Learning Conference of Belgium and the Netherlands, p. 138-145 [S.1]

Mausam, A., Kolobov, A. (2012) “Planning with Markov Decision Processes: An AI Perspective”. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers.

Puterman, M. L. (1994) “Markov Decision Process – Discrete Stochastic Dynamic Programming”. John Wiley & Sons, Inc., New York/NY, EUA.

Kaelbling, L, P., Littman, M. L., Cassandra, A. R. (1998) “Planning and acting in partially observable stochastic domains”. Em: Artificial Inteligente, vol. 101, p. 99 – 134.