Monte Carlo Tree Search Algorithm for SSPs Under the GUBS Criterion

Gabriel N. Crispino[1], Valdinei Freire[2], Karina V. Delgado[2]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)

[2]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (USP)

{crispino}@ime.usp.br,{valdinei.freire, kvd}@usp.br

Abstract. *In probabilistic planning, an usual problem is the Stochastic Shortest Path (SSP) in a Markov Decision Process (MDP), where the objective is to find policies that reach a goal with the minimum expected accumulated cost. In the presence of states from where it is not possible to reach a goal, this criterion is not well-defined in these states. Because of that, there are extensions and other formulations for this particular case, like the GUBS criterion. Although algorithms to solve SSPs under this criterion exist, they need to make sweeps through the entire state space. With that in mind, this work proposes UCT-GUBS: an online planning algorithm based on UCT to solve SSPs under the GUBS criterion. The results of the experiments that were performed show empirically that the algorithm makes the desired trade-off between cost and probability to goal when its parameters are varied, with the ability of making a decision in less or more time depending on the time limit provided to the algorithm, an important characteristic of UCT.*

Resumo. *Em planejamento probabilístico, um problema comum é o de caminho estocástico mais curto (SSP), no qual se deseja encontrar políticas que chegam a uma meta com o menor custo acumulado esperado. A presença de estados a partir dos quais não é possível alcançar uma meta, os dead-ends, torna esse critério indefinido nesses estados. Por isso, existem extensões e novas formulações para esse caso particular, como o critério GUBS. Embora existam algoritmos para solucionar esse critério, todos eles necessitam iterar pelo espaço completo de estados. Tendo isso em vista, no presente trabalho é proposto o UCT-GUBS, um algoritmo online de planejamento baseado no UCT para resolver SSPs sob o critério GUBS. Os resultados dos experimentos realizados mostram empiricamente que o algoritmo demonstra um compromisso entre custo e probabilidade para a meta conforme a variação dos seus parâmetros, podendo tomar uma decisão em um tempo menor ou maior dependendo do tempo limite passado para o algoritmo, que é uma característica importante do UCT.*

1. Introdução

Na área de tomada de decisão sequencial, é preciso modelar a interação do agente com o ambiente. No contexto de planejamento probabilístico, o formalismo mais utilizado é o de Processos de Decisão Markovianos (*Markov Decision Processes* - MDPs) [Puterman 1994], nos quais um agente está em um determinado estado do problema

e, a cada passo, deve escolher entre um conjunto de ações uma ação que será executada, levando-o a um novo estado no próximo passo. Um problema comum nessa área é o de encontrar o caminho estocástico mais curto (*Stochastic Shortest Path* - SSP) [Bertsekas 1995], no qual se procura encontrar um caminho ou política que alcance uma meta, minimizando o custo acumulado esperado. No entanto, esse critério é indefinido em estados a partir dos quais não se pode alcançar a meta, os *dead-ends*.

Isso motiva adaptações do modelo existente de SSP-MDPs e a criação de novos critérios para encontrar soluções para esse tipo de problema. O uso de extensões como fator de desconto ou penalidade para "desistir" [Puterman 1994, Kolobov et al. 2012], são exemplos de soluções para esse problema sob o critério de custo acumulado esperado. Além disso, existem modelos que otimizam critérios diferentes, como o MAXPROB [Kolobov et al. 2011], que encontra políticas que maximizam a probabilidade de chegar à meta. No entanto, nenhum desses modelos permite estabelecer um compromisso com significado entre probabilidade de chegar à meta e custo acumulado esperado. Como uma consequência disso, não é claro como escolher valores da penalidade ou do fator de desconto para que a política ótima encontrada pelo modelo gradualmente priorize mais ou menos caminhos com um risco maior de não se chegar à meta, com um custo respectivamente menor, por exemplo.

O critério GUBS (*Goals With Utility-Based Semantics*) [Freire and Delgado 2017] foi proposto como uma maneira de se encontrar soluções para SSPs com *dead-ends* com uma parametrização com significado para obter um melhor compromisso entre probabilidade de chegar à meta e custo acumulado esperado. Para isso, existem dois algoritmos que resolvem SSPs sob o critério GUBS [Freire and Delgado 2017, Freire et al. 2019], mas ambos necessitam iterar sobre o espaço completo de estados, sendo impraticáveis em problemas de tamanho grande. Tendo isso em vista, é necessário que algoritmos mais eficientes, que não precisam iterar por todos os estados do problema, sejam propostos.

O presente trabalho está organizado da seguinte maneira: a seção 2 apresenta os fundamentos básicos de MDPs, SSPs e alguns dos modelos alternativos para resolvê-los na presença de *dead-ends*. A seção 3 descreve o critério GUBS, demonstrando algumas de suas características e resultados teóricos importantes. A seção 4 apresenta o algoritmo UCT-GUBS, e a seção 5, os experimentos realizados com ele e os resultados obtidos. Por fim, a seção 6 expõe as conclusões do trabalho.

2. Processos de Decisão Markovianos (MDPs)

Na área de planejamento probabilístico, é comum que a interação de um agente com o ambiente seja modelada por meio de um MDP. Nesse processo, a cada passo discreto de tempo t o agente está em um estado s. Ele toma uma ação a que o leva a um novo estado s', seguindo uma função de transição. Cada vez que uma ação é executada em um estado, o agente paga um custo c.

Definição 1. *Um MDP é uma tupla $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, C \rangle$ [Puterman 1994], em que:*

- *$\mathcal{S}$ é o conjunto de estados em que o agente pode estar em cada passo de tempo;*
- *$\mathcal{A}$ é o conjunto de ações que podem ser executadas em cada estado;*
- *$T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ é a função de transição, tal que $T(s, a, s')$ indica a probabilidade do agente estar em um estado s' no próximo passo, dado que no passo atual a ação a é executada no estado s; e*

- $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *é a função de custo, que define o custo associado à execução da ação* a *estando no estado* s*.*

Uma solução para um MDP é uma política: uma função $\pi : \mathcal{S} \to \mathcal{A}$ que mapeia estados em ações. Uma política ótima π^* é uma política que otimiza uma função valor objetivo. Uma abordagem comum é avaliar uma política pelo seu custo acumulado esperado, ao segui-la a partir de um determinado estado:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} c_t | s_0 = s\right]. \tag{1}$$

A partir da qual a seguinte equação pode ser derivada [Puterman 1994]:

$$V^{\pi}(s) = C(s, \pi(s)) + \sum_{s' \in \mathcal{S}} [T(s, \pi(s), s')V^{\pi}(s')]. \tag{2}$$

De maneira análoga, o operador de Bellman [Bellman et al. 1957] $\mathcal{T}$ é definido da seguinte maneira:

$$(\mathcal{T}V)(s) = \min_{a \in \mathcal{A}} \left\{ C(s, a) + \sum_{s' \in \mathcal{S}} [T(s, a, s')V(s')] \right\}. \tag{3}$$

Esse operador tem como ponto fixo a função valor ótima V^*, isto é, a função valor associada à política ótima π^*. Ou seja, no limite, V^* é obtida como o resultado de infinitas aplicações do operador $\mathcal{T}$ em qualquer função inicial V.

Um objetivo usual no contexto de MDPs é encontrar o **caminho estocástico mais curto** (*Stochastic Shortest Path* - SSP) [Bertsekas 1995]. Dado um horizonte indeterminado (quando se sabe que eventualmente o processo chegará a um fim) e um conjunto de estados meta, o objetivo nesse problema é determinar a política ótima que minimiza o custo acumulado esperado a partir de cada estado s.

Definição 2. *Um SSP-MDP (ou SSP, para simplificar) é uma tupla* $\mathcal{M}_s = \langle \mathcal{S}, \mathcal{A}, T, C, \mathcal{G} \rangle$*, em que:*

- $\mathcal{S}, \mathcal{A}, T$ *e* C *são definidos como na Definição 1;*
- $\mathcal{G}$ *é o conjunto de estados absorvedores chamados de meta, tal que* $T(g, a, g) = 1$ *e* $C(g, a) = 0, \forall a \in \mathcal{A}$ *e* $\forall g \in \mathcal{G}$*.*

E as seguintes condições são atendidas:

1. *Existe pelo menos uma política própria. Uma política é própria se, ao segui-la, um agente alcança um estado meta a partir de qualquer estado* s *com probabilidade* 1*; e*
2. *O valor de qualquer política* π *que não é própria é* $V^{\pi}(s) = \infty$*.*

Embora SSPs forneçam uma definição consistente e que existam algoritmos eficientes para resolvê-los, como o LAO* [Hansen and Zilberstein 2001] e LRTDP [Bonet and Geffner 2003], as suas pré-condições podem ser muito restritivas para diversos tipos de situações. Mais especificamente, pode ser necessário resolver um problema que possui estados chamados de ***dead-ends***, a partir dos quais não é possível alcançar a meta. A violação dessa condição implica que a aplicação do operador de Bellman definido na Equação 3 não converge para um ponto fixo. Por isso, é necessário que o modelo tradicional de SSPs seja alterado de maneira que ainda seja possível resolver esse tipo de problema, mesmo na presença desses estados.

2.1. Modelos Alternativos para SSPs com *Dead-ends*

Uma das extensões mais comuns para MDPs em situações que uma política tem valor infinito para algum estado $s \in \mathcal{S}$ é o uso de um fator de desconto γ sob os custos recebidos, tal que $0 < \gamma < 1$ [Puterman 1994]. Dessa maneira, um histórico originalmente definido por $h = \{(s_0, a_0, c_0), (s_1, a_1, c_1), ..., (s_T, a_T, c_T)\}$ tem os seus custos transformados em $\{\gamma^0 c_0, \gamma^1 c_1, ..., \gamma^{T-1} c_T\}$. Com essa modificação, uma política é avaliada por $V^\pi = C(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} [T(s, \pi(s), s') V(s')]$, que tem um valor finito para cada estado s, mesmo que $T = \infty$. Isso possibilita o uso de algoritmos já conhecidos para solucionar SSPs, ao se utilizar essa extensão.

Pode-se também associar uma penalidade finita D ao custo do agente "desistir" do processo, ao alcançar um *dead-end*. Dessa maneira, o conjunto de ações é estendido com uma ação especial, que se escolhida leva o agente a um estado meta, sob a penalidade de pagar o custo D. SSPs com essa extensão são chamados de fSSPUDEs [Kolobov et al. 2012]. A adição de D e da ação especial evita que a função valor divirja nesses estados.

Outros modelos utilizam critérios diferentes para encontrar a solução de SSPs. O critério MAXPROB [Kolobov et al. 2011] avalia políticas com base na probabilidade de encontrar estados meta ao segui-las, de maneira que a política ótima é a que maximiza a probabilidade de chegar à meta, sem considerar os custos do problema. Uma abordagem alternativa é considerar um critério dual. Esse critério encontra políticas que minimizam o custo, tal que essas sejam políticas que maximizam a probabilidade de chegar à meta. Nos iSSPUDEs [Kolobov et al. 2012], a penalidade de se visitar um *dead-end* é considerada como infinita, de maneira que esses estados sejam evitados ao máximo. Por isso, o critério dual é utilizado ao encontrar as políticas MAXPROB, e depois as políticas que minimizam o custo esperado com uma equação de otimalidade específica. S^3Ps [Teichteil-Königsbuch 2012] possuem outra formulação para encontrar políticas que minimizam o custo esperado acumulado dentre as que maximizam a probabilidade de chegar à meta, inclusive para o caso de horizonte infinito, para MDPs com custos estritamente positivos.

3. *Goals with Utility-Based Semantics* (GUBS)

As diferentes formulações para resolver SSPs com *dead-ends* fornecem variadas maneiras para se encontrar soluções exatas e bem definidas para esses problemas. No entanto, um problema que elas compartilham é o fato de não terem uma maneira intuitiva de modificar os seus parâmetros, para encontrar políticas que fazem um compromisso suave entre probabilidade de chegar à meta e custo acumulado esperado. No caso do MAXPROB, o único critério a ser otimizado é a probabilidade de chegar à meta. Nos iSSPUDEs, esse mesmo critério é otimizado como primeiro passo, para depois encontrar a política que minimiza o custo, dentre as políticas que maximizam a probabilidade de chegar à meta. Nos modelos com fator de desconto e penalidade, por exemplo, não é claro como definir valores de γ e D, de maneira que caminhos que priorizam menos a meta em favor da obtenção de um menor custo sejam escolhidos.

Tendo essas questões em vista, o critério GUBS foi proposto como uma maneira de oferecer uma alternativa para amenizar esse tipo de problema.

Definição 3. *O critério GUBS [Freire and Delgado 2017] avalia um histórico com a seguinte função utilidade U:*

$$U(\zeta_T, \beta_T) = u(\zeta_T) + K_g \mathbb{1}_{\beta_T = g}, \tag{4}$$

em que $\zeta_T = \sum_{t=0}^{T-1} c(s_t, a_t)$, $\beta_T = g$ se até o tempo T algum estado meta foi alcançado, e $\beta_T = \neg g$ caso contrário, u é uma função de utilidade sobre o custo acumulado e K_g é a utilidade constante de se alcançar a meta. Um agente segue o critério GUBS se avalia políticas com a seguinte função valor:

$$V^{\pi}(s) = \mathbb{E}[U(\zeta_T, \beta_T) | \pi, s_0 = s]. \tag{5}$$

Os seguintes teoremas mostram que um agente que segue o critério GUBS prioriza metas se certas condições forem atendidas, e como os seus parâmetros podem ser escolhidos para priorizar políticas com mais ou menos probabilidade de se chegar à meta.

Teorema 1. *[Freire and Delgado 2017] Um agente seguindo o critério GUBS prioriza metas, se:*

1. $u : \mathbb{R} \to [U_{min}, U_{max}]$;
2. $u(\zeta)$ *é estritamente decrescente em* ζ;
3. $K_g > U_{max} - U_{min}$.

Teorema 2. *[Freire and Delgado 2017] Seja P_g^{π} a probabilidade de uma política π alcançar a meta. Considerando duas políticas π e π', tal que $P_g^{\pi} > P_g^{\pi'}$, um agente que segue o critério GUBS prefere π sobre π', se:*

$$\frac{P_g^{\pi}}{P_g^{\pi'}} > \frac{K_g}{U_{max} - U_{min} + K_g}. \tag{6}$$

Pode-se ver por meio desses teoremas que o critério GUBS fornece uma maneira mais clara de se priorizar metas tendo a preocupação com o compromisso entre custo e probabilidade para meta. Um dos seus pontos negativos é que a política ótima seguindo esse critério não é markoviana, ou seja, depende do histórico completo para escolher a próxima ação. Isso dificulta que uma solução exata seja encontrada por programação dinâmica.

Em [Freire and Delgado 2017] é proposto um algoritmo para solucionar SSPs sob o critério GUBS. Nesse algoritmo é realizada uma iteração de valor, indo de trás para frente de um custo máximo dado como parâmetro até 0, calculando por programação dinâmica os valores dos estados. Esse algoritmo não garante uma solução ótima, pois não é possível definir *a priori* o custo máximo alcançado ao seguir uma política ótima de um problema em específico. Quando existem *dead-ends* no domínio, o custo da política nesses estados é infinito, o que impossibilitaria a enumeração deles para se encontrar a política ótima.

Em [Freire et al. 2019] é proposto um novo algoritmo que resolve SSPs sob o critério GUBS de maneira exata, sob a condição de que a função de utilidade sobre o custo seja $u(\zeta_T) = e^{-\lambda \zeta_T}$, em que λ é o fator de risco e é positivo. Esse caso particular do GUBS é nomeado de eGUBS, e o algoritmo que soluciona SSPs sob esse critério é chamado de eGUBS-VI.

4. UCT-GUBS

Apesar do eGUBS-VI possuir a propriedade importante de encontrar uma solução exata sob o critério eGUBS, ele ainda é um algoritmo de iteração de valor. Isso significa que é preciso realizar várias passagens em todos os estados em $\mathcal{S}$, o que o torna impraticável em casos nos quais esse conjunto é muito grande. Para resolver esse problema, seria necessário um algoritmo que tenha o mesmo objetivo, mas que consiga lidar com um espaço grande de estados, sem precisar visitá-lo por completo.

Por essa razão, no presente trabalho é proposto o algoritmo **UCT-GUBS** (*Upper Confidence Bounds Applied to Trees Under GUBS*). **UCT-GUBS** é uma modificação do algoritmo de *Monte Carlo Tree Search* UCT (*Upper Confidence Bounds Applied to Trees*) [Kocsis and Szepesvári 2006] utilizada para encontrar políticas sob o critério GUBS. Em linhas gerais, o algoritmo funciona de maneira *online*, isto é, o planejamento é realizado enquanto o agente atua no ambiente. Dessa maneira, dado que o agente está em um estado s, o algoritmo realiza repetidas simulações (também chamadas de *rollouts*) de trajetórias partindo desse estado s, construindo uma árvore, em que cada nó representa um estado encontrado na profundidade de busca d, e cada um dos seus ramos representa a tomada de uma ação específica, levando a um novo estado na profundidade $d + 1$.

Seja $Q_d(s, a, d)$ a qualidade de se executar a ação a, estando no estado s e profundidade d, o algoritmo então atualiza uma estimativa atual do valor de $Q_d(s, a, d)$ para diferentes ações $a \in \mathcal{A}$, ao calcular a média das utilidades dos históricos resultantes dos *rollouts* enquanto o tempo limite não for atingido. Ao fim dessas simulações, a ação escolhida para ser tomada no estado s e profundidade d é a que maximiza $Q_d(s, a, d)$, ou seja, a que fornece maior utilidade. Ao contrário do eGUBS-VI, o UCT-GUBS (assim como o UCT) não encontra uma solução exata. Ao invés disso, ele tenta encontrar soluções boas com pouco tempo de planejamento, de maneira que as estimativas fiquem melhores conforme esse tempo aumenta. O pseudocódigo do UCT-GUBS pode ser visualizado no Algoritmo 1.

Algoritmo 1: UCT-GUBS

Entrada: SSP-MDP $\mathcal{M}_s = \langle \mathcal{S}, \mathcal{A}, T, C, \mathcal{G} \rangle$, função heurística h, função de utilidade sobre o custo u, utilidade constante da meta K_g, estado inicial s e profundidade máxima D

Saída: a

início
  Seja $Q_d : \mathcal{S} \times \mathcal{A} \times \mathbb{N} \to \mathbb{R}$
  repita
    BUSCA$(\mathcal{M}_s, Q_d, h, u, K_g, s', D, 0)$
  até *tempo limite*;
  $a \leftarrow \arg\max_{a' \in \mathcal{M}_s.\mathcal{A}} Q_d(s, a', 0)$
fim
retorna a

Esse algoritmo realiza o processo descrito, chamando a sub-rotina BUSCA a cada iteração, enquanto o tempo limite não for atingido. Quando isso acontece, a ação que

maximiza $Q_d(s, a, 0)$ é a ação escolhida pelo algoritmo. O pseudocódigo do algoritmo de busca é mostrado no Algoritmo 2.

Algoritmo 2: BUSCA

Entrada: SSP-MDP $\mathcal{M}_s = \langle \mathcal{S}, \mathcal{A}, T, C, \mathcal{G} \rangle$, função de qualidade Q_d, função heurística h, função de utilidade sobre o custo u, utilidade constante da meta K_g, estado s, profundidade máxima D e profundidade d

Saída: Custo q obtido no *rollout* a partir do nó atual e indicador g que é verdadeiro se algum estado meta foi alcançado, e falso caso contrário

1 **início**
2 **se** $s \in \mathcal{G}$ **então**
3 $g \leftarrow verdadeiro$
4 **senão**
5 $g \leftarrow falso$
6 **fim**
7 **se** $d = D$ **então**
8 **retorna** $0, g$
9 **fim**
10 **se** s *é folha* **então**
11 $n_{s,d} \leftarrow 0$
12 **para cada** $a \in \mathcal{A}$ **faça**
13 $Q_d(s, a, d) \leftarrow h(s, a)$
14 $n_{s,a,d} \leftarrow 0$
15 **fim**
16 **fim**
17 $a \leftarrow \arg\max_{a' \in \mathcal{A}} UCB1(s, a', d)$
18 $c \leftarrow C(s, a)$
19 $s' \leftarrow$ amostre o próximo estado de acordo com $T(s, a, s'),\ \forall s' \in \mathcal{S}$
20 $q', g' \leftarrow$ BUSCA$(\mathcal{M}_s, Q_d, h, u, K_g, s, D, d + 1)$
21 $q \leftarrow c + q'$
22 $g \leftarrow g \vee g'$
23 **se** g **então**
24 $k \leftarrow K_g$
25 **senão**
26 $k \leftarrow 0$
27 **fim**
28 $Q_d(s, a, d) \leftarrow \frac{n_{s,a,d} Q_d(s,a,d) + u(q) + k}{n_{s,a,d} + 1}$
29 $n_{s,d} \leftarrow n_{s,d} + 1$
30 $n_{s,a,d} \leftarrow n_{s,a,d} + 1$
31 **fim**
32 **retorna** q, g

O processo de busca consiste em executar ações e ir para novos estados, até que um horizonte limite seja atingido (linha 7). No caso que o nó correspondente ao estado s e profundidade d na árvore é folha, ou seja, é um nó que não tem filhos, o seu valor é

inicializado com uma função heurística h (linhas 10-16). Durante a simulação, as ações são escolhidas segundo o critério UCB1 [Auer et al. 2002], na linha 17, definido pela seguinte expressão:

$$UCB1(s, a, d) = Q_d(s, a, d) + C\sqrt{\frac{\ln n_{s,d}}{n_{s,a,d}}}, \tag{7}$$

em que $n_{s,d}$ é o número de vezes que o estado s foi visitado na profundidade d, e $n_{s,a,d}$ é o número de vezes que a ação a foi tomada no estado s e profundidade d.

A ação que maximiza essa equação é a escolhida para ser executada. Essa expressão é uma maneira de lidar com o dilema de *exploration-exploitation*. Quanto mais estimativas forem feitas para uma ação a em um estado, mais o segundo termo da soma diminui para a e aumenta para outras ações no mesmo estado. Por outro lado, quanto maior a utilidade da estimativa atual, maior é o valor desse estado para essa expressão. A constante C é um fator de exploração, que vai dar mais ou menos peso para a exploração de ações que não foram muito escolhidas até então.

O próximo estado s' é então amostrado com base em T, dado o estado atual s e a ação escolhida a (linha 19). O custo de executar essa ação nesse estado é obtido, e a função é chamada recursivamente para o próximo estado (linha 20). Por fim, o novo valor da estimativa atual de $Q_d(s, a, d)$ é calculado ao atualizar a sua média, com a utilidade do histórico obtido no *rollout*. Essa utilidade é a soma de $u(q)$, em que q é custo total que foi obtido, e K_g, se um estado meta foi alcançado (linha 28). O algoritmo devolve dois valores: q e g, em que g é verdadeiro se um estado meta foi atingido, e falso caso contrário.

5. Experimentos e Resultados

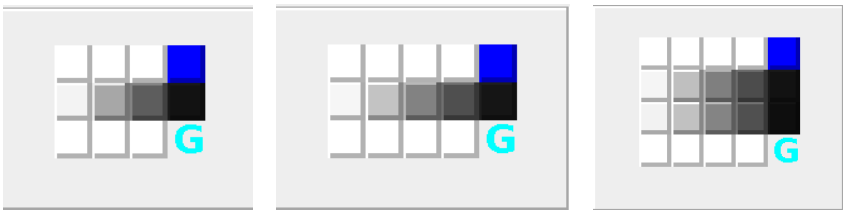
O algoritmo UCT-GUBS[1] foi implementado dentro da infraestrutura do PROST [2][Keller and Eyerich 2012], um planejador do estado da arte que foi utilizado como base pelos vencedores da trilha de planejamento probabilístico totalmente observável das competições internacionais de planejamento (IPC) dos anos de 2011, 2014 e 2018 [Sanner and Yoon 2011, Grzes et al. 2014, Keller 2018]. Essa infraestrutura implementa o algoritmo UCT, juntamente com algumas otimizações que melhoram o seu desempenho.

Os experimentos foram realizados em instâncias do domínio NAVIGATION, da IPC 2011. A função heurística, o valor da constante C e o horizonte máximo dos *rollouts* utilizados foram os padrões da configuração IPC2011 do PROST. Para calcular a heurística é usada uma função de busca de aprofundamento iterativo (*Iterative Deepening Search*); à constante C, no cálculo da expressão $UCB1$ (Equação 7) para a dupla (s, d), é atribuído o valor atual de $\max_{a \in \mathcal{A}} Q_d(s, a, d)$; o valor utilizado para o horizonte máximo dos *rollouts* é 15. A função de utilidade sobre o custo acumulado utilizada foi a mesma do eGUBS-VI [Freire et al. 2019], ou seja, $u(\zeta_T) = e^{-\lambda\zeta_T}$, com $\lambda = 0.1$, na qual ζ_T é o custo acumulado obtido no *rollout*.

Domínio NAVIGATION: esse é um problema do estilo *grid world*, tendo M linhas e N colunas, e cada estado representado por uma coordenada (x, y). Em qualquer estado,

[1]Seu código-fonte está disponível em https://github.com/GCrispino/prost/tree/eniac-2020

[2]Seu código-fonte está disponível em https://github.com/prost-planner/prost



(a) Instância 1 do NAVIGATION (b) Instância 2 do NAVIGATION (c) Instância 3 do NAVIGATION

Figura 1. Visualização das 3 instâncias do domínio NAVIGATION utilizadas.

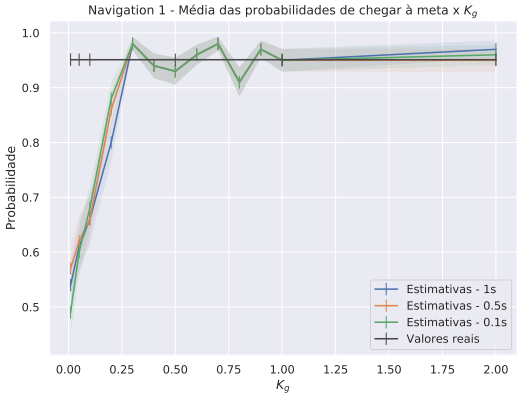
o agente tem como possíveis ações os movimentos nas 4 direções cardeais, e o objetivo é alcançar o estado meta. Na primeira e última linha (estados $(1, y)$ e (M, y), para $1 \leq y \leq N$), as ações são determinísticas. No entanto, nas linhas intermediárias, o agente tem uma probabilidade $P(x, y)$ (quanto mais longe da meta, menor é essa probabilidade) de desaparecer, indo para um estado a partir do qual não pode mais sair (um *dead-end*).

Foram utilizadas 3 instâncias desse domínio, com dimensões de 4×3, 5×3 e 5×4, e cujas respectivas visualizações estão expostas na Figura 1. Em cada uma das delas, quadrados brancos indicam que as transições neles são determinísticas. Nos outros, quanto mais escuro, mais a probabilidade do agente desaparecer ao executar uma ação se aproxima de 1. O estado marcado de azul é o inicial, e o marcado com um G é a meta.

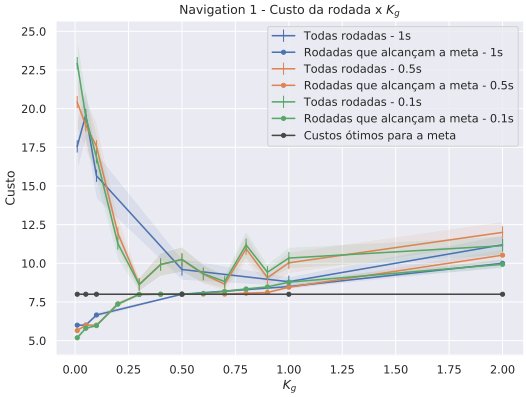
Foram variados os parâmetros de tempo de planejamento para cada estado, e valores de K_g. Para os valores de tempo de planejamento foi utilizado 1 segundo, o padrão do sistema do PROST, além dos valores de 0.5 e 0.1 segundo, para avaliar os resultados do algoritmo mesmo com tempos menores de planejamento. Para K_g, são utilizados valores entre 0.01 e 2. Esses valores foram escolhidos pelo fato de que, nas instâncias testadas, entre esses valores existe uma variação entre as políticas ótimas sob o critério GUBS, com exceção apenas da primeira instância. Dessa maneira, é possível verificar o quanto os resultados do UCT-GUBS se aproximam dos ótimos nesses experimentos.

Para cada uma dessas configurações, foram executados 100 rodadas do algoritmo. Em cada rodada, o planejamento começa do estado inicial, o algoritmo faz as simulações durante o tempo de planejamento, e submete uma ação, indo para um novo estado. Esse processo se repete até que o horizonte máximo tenha sido atingido. Para todas as instâncias desse domínio, o horizonte máximo da rodada, especificado no seu arquivo de entrada, é 40. São analisadas as médias das probabilidades de chegar à meta (divisão do número de rodadas que se chegou à meta sobre o número total de rodadas) e dos custos totais obtidos nas rodadas.

Na instância 1 do NAVIGATION, as probabilidades observadas se aproximam das probabilidades da política exata sob o critério GUBS (Figura 2a), mas a partir de $K_g = 0.3$. Antes disso, o UCT-GUBS escolhe políticas com uma probabilidade menor de chegar à meta do que a política ótima. Ademais, pode-se observar que o comportamento das rodadas com um tempo menor de planejamento não mostram muitas diferenças quanto ao seu desempenho, apresentando resultados semelhantes. No caso dos custos (Figura 2b), é possível visualizar que para valores pequenos de K_g os custos de rodadas que chegam à meta são menores, mas com a penalidade da média dos custos de todas as rodadas ser mais alta que para valores menores, pelo fato do agente priorizar caminhos com um risco

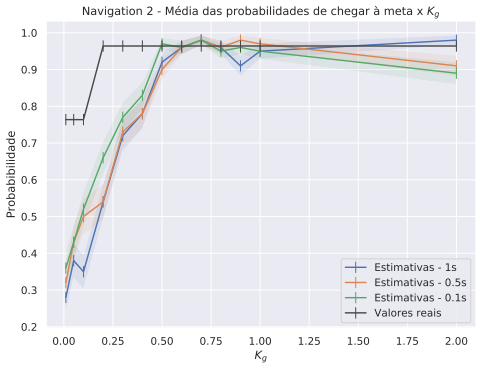


(a) Média das probabilidades de chegar à meta em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 1 do domínio NAVIGATION.

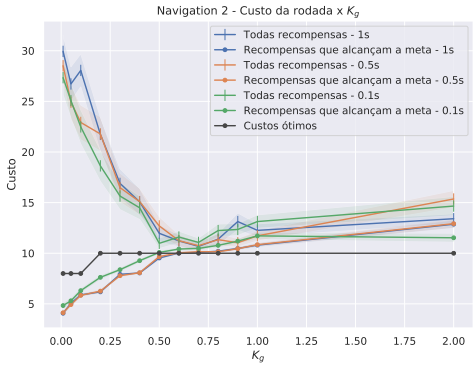


(b) Média dos custos obtidos nas rodadas em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 1 do domínio NAVIGATION.

Figura 2. Resultados da instância 1 do domínio NAVIGATION



(a) Média das probabilidades de chegar à meta em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 2 do domínio NAVIGATION.



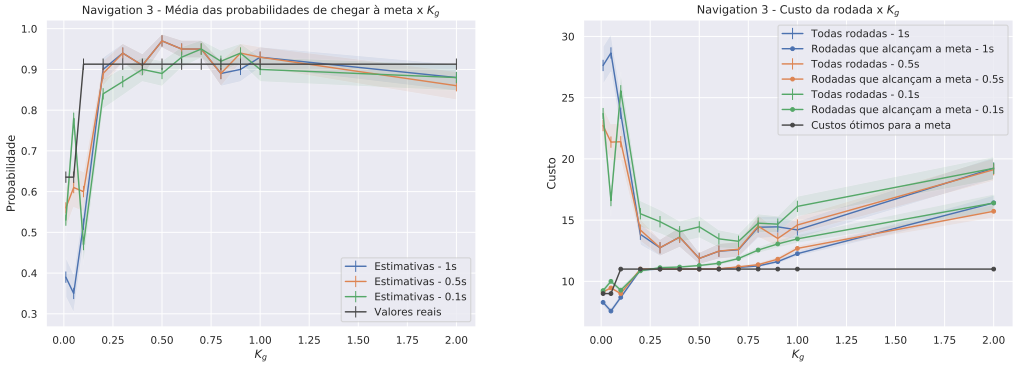
(b) Média dos custos obtidos nas rodadas em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 2 do domínio NAVIGATION.

Figura 3. Resultados da instância 2 do domínio NAVIGATION

maior de não se chegar à meta.

Na segunda instância, percebe-se um comportamento similar quanto às probabilidades e custos obtidos. No primeiro caso, a partir de $K_g = 0.5$, esses valores se aproximam mais aos exatos. No maior valor testado ($K_g = 2$) esses valores sofrem uma maior oscilação, principalmente nos dois testes com menor tempo de planejamento, sugerindo que para maiores valores de K_g pode ser necessário um número maior de amostras para se aproximar do valor exato. O gráfico de custos se apresenta similar aos resultados da primeira instância. Aqui, é interessante notar que os custos obtidos dos testes com tempo de planejamento de 0.5 e de 1 segundo tem uma diferença pequena, praticamente imperceptível nas linhas do gráfico.

Os resultados da terceira e última instância são similares aos anteriores, com uma oscilação menor nas probabilidades — o que pode se explicar pela pequena variação das políticas ótimas exatas sob o critério GUBS para os valores de K_g utilizados — e uma



(a) Média das probabilidades de chegar à meta em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 3 do domínio NAVIGATION.

(b) Média dos custos obtidos nas rodadas em função de valores de K_g, calculadas em 100 rodadas do UCT-GUBS para cada valor na instância 3 do domínio NAVIGATION.

Figura 4. Resultados da instância 3 do domínio NAVIGATION

oscilação maior do custo. Isso pode-se explicar porque em algumas situações, o agente fica "indeciso", por exemplo, se movendo para leste e oeste, atrasando o momento de atravessar pelas linhas com probabilidade de desaparecer e pagando um custo maior que o ótimo.

Em geral, observa-se que, para as instâncias do domínio testado, quando se utilizado um tempo menor de planejamento, não houve grande perda de desempenho no algoritmo. Isso indica que apenas valores ainda menores que 0.1 causariam perda considerável de desempenho, o que pode variar em outros domínios. Além disso, o compromisso entre custo e probabilidade é realizado conforme os valores de K_g são modificados, com a possibilidade de se utilizar um tempo de planejamento adequado ao usuário.

6. Conclusão

No presente trabalho foi proposto o algoritmo UCT-GUBS, que realiza planejamento *online* em SSPs para encontrar políticas sob o critério GUBS. Foram feitos experimentos em três instâncias do domínio NAVIGATION, variando o tempo de planejamento e a utilidade da meta K_g. Foi observado que o algoritmo encontra políticas fazendo o compromisso entre probabilidade de chegar à meta e custo, na variação de valores de K_g. Além disso, não foram observadas mudanças consideráveis quando diminuído o tempo de planejamento para tomar uma decisão. No entanto, é possível que seja observada uma perda de desempenho maior para o caso em que valores menores que 0.1 segundo sejam utilizados, ou em problemas de tamanho maior.

Trabalhos futuros podem estender os experimentos ao analisar diferentes domínios com *dead-ends*. Um exemplo é o CROSSING TRAFFIC, outro domínio da IPC 2011. Variações no algoritmo como o uso de uma heurística que leva em conta as probabilidades da função de transição e a maneira como as estimativas são atualizadas podem melhorar o seu desempenho. Além disso, novos algoritmos para o critério GUBS podem ser propostos, aproveitando ideias de algoritmos que seguem outras abordagens, como no caso do LAO* e LRTDP, que encontram soluções exatas com busca heurística.

Referências

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Bellman, R., Corporation, R., and Collection, K. M. R. (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Bertsekas, D. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Mass.

Bonet, B. and Geffner, H. (2003). Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS*, volume 3, pages 12–21.

Freire, V. and Delgado, K. V. (2017). GUBS: a utility-based semantic for goal-directed markov decision processes. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, pages 741–749.

Freire, V., Delgado, K. V., and Reis, W. A. S. (2019). An exact algorithm to make a trade-off between cost and probability in SSPs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 146–154.

Grzes, M., Hoey, J., and Sanner, S. (2014). IPPC discrete track results. In *International Conference on Automated Planning and Scheduling*. Acessado em julho de 2020.

Hansen, E. A. and Zilberstein, S. (2001). LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62.

Keller, T. (2018). International planning competition 2018 - probabilistic tracks. In *International Conference on Automated Planning and Scheduling*. Acessado em julho de 2020.

Keller, T. and Eyerich, P. (2012). PROST: Probabilistic planning based on UCT. In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, pages 119–127.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Kolobov, A., Weld, D., et al. (2012). A theory of goal-oriented MDPs with dead ends. In *Uncertainty in artificial intelligence : proceedings of the Twenty-eighth Conference [on uncertainty in artificial intelligence] (2012)*, pages 438–447.

Kolobov, A., Weld, D. S., and Geffner, H. (2011). Heuristic search for generalized stochastic shortest path mdps. In *Proceedings of the Twenty-First International Conference on International Conference on Automated Planning and Scheduling*, pages 130–137.

Puterman, M. (1994). *Markov decision processes : discrete stochastic dynamic programming*. Wiley, New York.

Sanner, S. and Yoon, S. (2011). IPPC results presentation. In *International Conference on Automated Planning and Scheduling*. Acessado em julho de 2020.

Teichteil-Königsbuch, F. (2012). Stochastic safest and shortest path problems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1825–1831.