Evento	Salão UFRGS 2019: SIC - XXXI SALÃO DE INICIAÇÃO CIENTÍFICA DA UFRGS
Ano	2019
Local	Campus do Vale - UFRGS
Título	Aprendendo a Detectar Estados Dead em Sokoban
Autor	MATEUS DAVI SIMON
Orientador	ANDRÉ GRAHL PEREIRA

Aprendendo a Detectar Estados *Dead* em Sokoban

Bolsista: Mateus Davi Simon **Orientador:** André Grahl Pereira

O objetivo da área de planejamento automatizado é alcançar a capacidade humana em solução de problemas. Uma *tarefa de planejamento* é uma tupla $\Pi = \langle \mathcal{V}, s_0, \mathcal{A}, S^* \rangle$, onde $\mathcal{V}$ são as variáveis de estado, s_0 é um estado inicial, $\mathcal{A}$ é um conjunto de ações e S^* é um conjunto de estado objetivos. Uma tarefa Π induz um *espaço de estados*: um grafo direcionado de todos os estados gerados por atribuições de valores as variáveis. Existe um arco nesse grafo de s para s' sse existe uma ação $a \in \mathcal{A}$ que quando aplicada ao estado s gera o estado s'. Uma *solução* para uma tarefa Π (um estado s) é uma sequência de ações que quando aplicada ao estado inicial s_0 (ao estado s) gera um estado objetivo $s^* \in S^*$. As abordagens mais efetivas para solução de tarefas de planejamento usam uma busca no espaço de estados guiada por uma função heurística h.

Os estados de uma tarefa Π são classificados como *alive* se possuem solução e *dead* caso contrário. O problema de detecção de estados *dead* é essencial para a solução de tarefas de planejamento e os métodos mais efetivos usam as funções heurísticas de abstração *pattern databases* h^{PDBk}. Heurísticas h^{PDBk} representam perfeitamente a informação de subproblemas limitados por um subconjunto de $k = |V|$ variáveis $V \subseteq \mathcal{V}$. Assim, heurísticas h^{PDBk} detectam estados *dead* causados pela interação de no máximo k variáveis. Propomos um modelo de classificação baseado em aprendizado para detecção estado *dead* causados pela interação de mais do que k variáveis. Para o treinamento do modelo, é necessário um conjunto de treinamento com rótulos *alive* ou *dead*. O ponto crucial desse trabalho é determinar como gerar o conjunto de treinamento que maximiza o desempenho de modelo de classificação durante a busca.

Para geração dos estados *alive* propomos quatro métodos baseados em busca *backward* a partir do conjunto de estados objetivo S^* – estados gerados dessa forma possuem solução. O primeiro método é um *breadth-first search* (BFS) o qual gera estados estados próximos ao conjunto de estados objetivo S^*. O segundo método é um A* que usa como estado objetivo o estado inicial s_0 e pretende gerar estados que seriam gerados durante a busca para resolver a tarefa Π. O terceiro método é um *greedy best-first search* (GBFS) que maximiza valor-h em relação ao conjunto S^* e esperadamente gera estados distantes do conjunto de estados objetivo S^*. O ultimo método que chamamos de *reverse* GBFS (rGBFS) minimiza o valor-h em relação ao estado inicial s_0. rGBFS deve gerar estados distantes de S^* e próximos ao estado inicial s_0.

Propomos duas abordagens para geração de estados *dead*. A primeira gera estados aleatórios e verifica usando h^{PDBk} se o estado gerado é *dead*. A segunda abordagem tem por objetivo gerar estados *dead* mais similares aos estados gerados durante a busca para resolver a tarefa Π. Assim, nesse método para cada estado *alive* gerado realizamos uma busca local até um estado *dead* ser gerado, novamente fazemos essa verificação com h^{PDBk}.

Utilizamos o domínio *Sokoban* como caso teste para nossa abordagem. Sokoban é um domínio PSPACE-completo e uma das tarefas de planejamento mais desafiadoras para planejadores automatizados. O modelo de classificação utilizado foi uma rede neural usando a biblioteca *TensorFlow*. A *entrada* do modelo é um vetor de booleano que representa se um valor está atribuído a uma variável no estado. Os métodos usam a heurística h^{MM} que é baseada em um emparelhamento perfeito de custo mínimo em um grafo bipartido. Primeiramente, testamos os quatro algoritmos para geração de estados *alive*. rGBFS apresentou os melhores resultados expandindo o menor número de estados na busca para resolver Π. No segundo teste avaliamos o método de geração de estados *dead* e a utilização da busca local usando estados *alive* se mostrou superior a geração aleatória de estados. Combinados os melhores métodos para geração de estados *alive* e *dead* mostram que o uso do modelo obtém resultados superiores a um A* guiado por h^{MM}, e é comparável a um A* guiado por h^{MM} que usa h^{PDBk} para detecção de estados *dead*. Nesse trabalho mostramos que o modelo tem desempenho superior quando treinado com estados mais similares aos encontrados na busca para resolver a tarefa de planejamento.