Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Wave function collapse hierárquico

Autor: João Pedro Moura Oliveira

Brasília, DF
2023

Sumário

1 Domínio

Em um sistema quântico, é recorrente a manifestação de superposições de estados. Ao ser submetido à observação por um agente, emerge a possibilidade de ocorrer o colapso da função de onda, culminando no sistema "transitando"para um dos estados fundamentais anteriormente sobrepostos. Este princípio da mecânica quântica, fortalecido pelas teorias de entropia, viabiliza a formulação do algoritmo de colapso de onda, conhecido como *Wave Function Collapse*.

Portanto, ao empregar este algoritmo no domínio da computação, torna-se possível a concepção de uma aplicação capaz de gerar conjuntos de mapas a partir de segmentos menores do mesmo, de modo que o resultado apresente similaridades locais. Por outro lado, em contraste com os preceitos da mecânica quântica, nota-se que este algoritmo dispensa o uso de números complexos para representar os estados sobrepostos, distanciando-se levemente de sua base teórica.

O algoritmo base segue o seguinte fluxo (GUMIN, 2016):

- Leitura da entrada dos padrões, sendo opcional a geração de rotações e reflexões;
- Criação de um *array* que representa a "onda". Cada elemento dessa lista corresponde a um estado possível, conforme os padrões instanciados;
- Inicialização da "onda"em um estado totalmente não observado, implicando que todos os estados de cada elemento do *array* são considerados possíveis;
- Processo de observação e propagação
  - Observação: Busca pelo elemento com menor entropia no sistema, excluindo o zero. Se nenhum elemento for identificado, esta etapa é concluída, passando-se para a última fase. Se encontrado, o elemento é colapsado para o padrão possível.
  - Propagação: Propagação do estado e das informações do elemento colapsado.
- Finalização do algoritmo com todos os elementos colapsados ou o sistema se encontra em um estado contraditório. Em caso de contradição, o algoritmo não consegui encontrar uma expansão válida para o sistema.

1.1 Propriedades

O algoritmo exibe uma propriedade forte e outra fraca. A propriedade forte denota que a saída deve incluir exclusivamente os padrões presentes na entrada do programa. A

propriedade fraca implica que a distribuição dos padrões no resultado deve assemelhar-se à distribuição em um conjunto extenso de resultados. Em outras palavras, a probabilidade de identificar um padrão específico na saída deve ser equivalente à probabilidade de encontrar esse padrão na entrada (GUMIN, 2016).

1.2 Motivação

Para alcançar uma solução que satisfaça a primeira propriedade, dado que o algoritmo pode incorrer em contradição levando a falha, portanto, classifica-se o problema como NP-hard. Assim, para obter um resultado para um padrão de entrada, é imperativo abordar um problema NP-completo que seja redutível em tempo polinomial. Dessa maneira, essa característica representa a principal motivação para a implementação de uma estrutura hierárquica no algoritmo.

1.3 *Benchmark*

Para a execução do *benchmark* foram criados 20 problemas para cada *track*. Cada problema se diferencia no tamanho do mapa, 2x2 até 4x4 na *track* AGL, 6x6 até 10x10 na *track* SAT e 6x6 até 20x20 na *track* OPT. Além disso, cada mapa pode iniciar com N padrões já colapsados, seguindo a seguinte probabilidade: até 30% para a *track* AGL, até 60% para a *track* SAT e até 50% para a *track* OPT.

As Tabelas 1, 2 e 3 representam os resultados utilizando os planejadores PandaPI e TLE (compilado localmente) na máquina GPU2. Código e instruções de execução podem ser encontrados no seguinte link: https://github.com/Joao-Moura/wave-function-collapse-hddl

Tabela 1 – Resultados *benchmark* Agile

Mapa	PandaPI	PandaDealer
map_2x2_0_6	0.181s	0.307s
map_2x2_1_4	0.240s	0.255s
map_2x2_1_9	0.199s	0.226s
map_2x2_1_12	0.260s	0.208s
map_2x2_1_14	0.214s	0.174s
map_3x3_0_1	0.312s	TLE
map_3x3_1_15	0.351s	53.734s
map_3x3_1_17	0.335s	74.101s
map_3x3_1_18	0.314s	56.859s
map_3x3_2_2	0.434s	7.423s
map_4x4_1_0	0.706s	TLE
map_4x4_1_3	0.709s	TLE
map_4x4_1_8	0.830s	TLE
map_4x4_3_5	0.688s	TLE
map_4x4_3_13	59.332s	TLE
map_4x4_3_19	0.803s	TLE
map_4x4_4_7	5.815s	TLE
map_4x4_4_10	2.931s	132.103s
map_4x4_4_16	0.699s	TLE
map_4x4_5_11	0.676s	TLE

Tabela 2 – Resultados *benchmark* SAT

map_6x6_6_8	TLE	TLE
map_6x6_7_1	TLE	TLE
map_7x7_9_7	TLE	TLE
map_7x7_18_14	TLE	TLE
map_7x7_19_2	TLE	TLE
map_7x7_20_16	TLE	TLE
map_7x7_22_5	TLE	TLE
map_8x8_1_19	TLE	TLE
map_8x8_3_9	TLE	TLE
map_8x8_11_11	TLE	TLE
map_8x8_13_18	TLE	TLE
map_8x8_36_17	TLE	TLE
map_8x8_38_6	TLE	TLE
map_9x9_5_4	TLE	TLE
map_9x9_11_0	TLE	TLE
map_9x9_11_15	TLE	TLE
map_9x9_26_10	TLE	TLE
map_9x9_29_3	TLE	TLE
map_9x9_29_12	TLE	TLE
map_9x9_47_13	TLE	TLE

Tabela 3 – Resultados *benchmark* OPT

map_8x8_11_18	TLE	TLE
map_8x8_27_7	TLE	TLE
map_9x9_5_4	TLE	TLE
map_10x10_9_3	TLE	TLE
map_10x10_17_0	TLE	TLE
map_10x10_20_8	TLE	TLE
map_10x10_34_13	TLE	TLE
map_10x10_35_17	TLE	TLE
map_12x12_63_6	TLE	TLE
map_12x12_68_11	TLE	TLE
map_13x13_40_1	TLE	TLE
map_14x14_85_15	TLE	TLE
map_15x15_61_5	TLE	TLE
map_15x15_109_19	TLE	TLE
map_17x17_19_2	TLE	TLE
map_18x18_48_14	TLE	TLE
map_18x18_57_12	TLE	TLE
map_18x18_70_16	TLE	TLE
map_18x18_76_9	TLE	TLE
map_18x18_82_10	TLE	TLE

Referências

GUMIN, M. Wavefunctioncollapse. 2016. Disponível em: <https://github.com/mxgmn/WaveFunctionCollapse>. Citado 2 vezes nas páginas 3 e 4.