P-BDD

Ramon Ribeiro

Instituto de Computação - UNICAMP

06/11/2018

Tópicos

- Paralelização com OpenMP
- Parallel BDD
- Sequential BDD
- Reed-Muller Soft Decision Decoder
- Referências

Chamada inicial

Em uma função "main"obtemos o numero de processadores p utilizáveis e ajustamos esse numero para um valor 4^n.
Chamamos, então, a função parallel BDD para $p = 4^n$ processadores.

Condição para chamar Sequential BDD

Quando não pudermos diminuir a cifra ou o numero de processadores.

$$p < 4 \text{ ou } tamanho = 1$$

Divisão de cifra

Dividimos a cifra s em duas metades alocadas.
Através delas, geramos mais duas.
Posteriormente, para cada metade alocada, chamamos em paralelo uma função parallel BDD.

Chamada por paralelização em threads de for

Com omp.h podemos dividir o numero de threads igualmente para chamadas de for.

Retornos

Das funções em paralelo, recebemos 4 retornos gaussianos de meio tamanho.
Dos 4 retornos geramos 4 pontos candidatos a serem o ponto que s representa.

Candidatos

Dos candidatos gerados retornamos o candidato mais próximo de s.
Em que Argumento(s-candidato) for o menor entre eles.
A função Parallel BDD retorna esse candidato.

Definindo raio

Definimos como r o raio que limita as aproximações possíveis da aproximação de s ainda possíveis.
O raio inicial é igual ao numero de processadores (1).

Raio máximo

Quando o tamanho de s for menor que 2^r, alcançamos o raio máximo de aproximações.
Para o raio máximo retornamos o arredondamento da cifra atual.

Criando alvos binários

Para cada ponto de s, criamos seu alvo binário e a confiabilidade do alvo.

Chamada de Reed-Muller

Para o alvo binário e a confiabilidade de alvo, chamamos o soft distance decoder de Reed-Muller para determinar a próxima cifra que iremos variar para se aproximar do candidato mais perto da perfeição possível.

Sequential BDD recursivo

Após aproximação de Reed- Muller, chamamos a função recursiva Sequential BDD para um raio incrementado em 1.

Retorno Sequential BDD

Retornamos o resultado da função recursiva reajustada com a aproximação de Reed.

Sem necessidade de calculo

Para um raio r, quando tamanho do alvo $= 2^r$ não ha necessidade de calculo.
Retorna o alvo binário.

Para raio igual a zero

Retornamos binários {0,0,...} ou {1,1,...}, dependendo da confiabilidade dos pontos do alvo.

Chamada recursiva Reed-Muller

Para um raio em que ha a necessidade de calculo de aproximação.
A partir da confiabilidade de cada ponto do alvo binário.
Criamos dois novos alvos binários u e v de meio tamanho, cujos (raios, dimensões) são (r, tamanho/2) e (r-1, tamanho/2), respectivamente.
São criados através de chamadas recursivas do decodificador de Reed usando manipulações dos pontos do alvo.

Retorno de Reed-Muller Recursivo

Retorna as somas de cada ponto de $[u, u + v]$.

Retorno de Reed-Muller

Sendo $\psi(\alpha)$ o retorno da funcão de Reed, temos que:

$$\begin{cases} \psi(\mathbf{0}) = \mathbf{0} \\ \psi(\mathbf{1}) = \mathbf{1} \\ \psi([\mathbf{u}, \mathbf{u} \oplus \mathbf{v}]) = [\psi(\mathbf{u}), \psi(\mathbf{u}) + \psi(\mathbf{v})] \end{cases}$$

Tempo

$$T_1(p,N) = \begin{cases} T_2(0,N) & \text{if } p < 4 \text{ or } N = 1 \\ O(\log N + N/p) + T_1(p/4, N/2) & \text{o/w} \end{cases}$$

$$T_2(r,N) = O(\log N - r)(N \log N).$$

Referências

- Efficient Bounded Distance Decoders for Barnes-Wall Lattices - Daniele Micciancio and Antonio Nicolosi April 30, 2008