Solving Classical Aproach of RNA Secondary Structure in Graphics Processing Unit

Marcos Negreiros
Mestrado Profissional em Computacão Aplicada, MPCOMP
Universidade Estadual do Ceara, UECE
Fortaleza, Brazil
negreiro@graphvs.com.br

Pedro Jorge de Abreu Figueredo
Laboratório de Computação Científica, LCC
Universidade Estadual do Ceara, UECE
Fortaleza, Brazil
pedro.jorge@aluno.uece.br

***Resumo*—Este artigo tem como intenção mostrar a utilização de um ambiente massivamente paralelo em GPU com a utilização da API CUDA da NVIDIA para abordar problemas que usam paradigma de Programação Dinâmica para serem resolvidos. Para o mesmo é utilizado o Problema Clássico da predição da RNA Secondary Structure mostrando a classificação do problema, a definição do modelo, criação da estrutura de memória a ser mantida em GPU com ênfase no coalesced access, sincronização de fases com sincronização interblocos e utilização de warp reduction. Gerando resultados com speedup de até 100 vezes em relação a implementação serial e mostrando a integração com a plataforma Gephi para melhor visualização da estrutura resultante.**

***Keywords*-Programação Dinamica; GPGPU; RNA; Warp Reduce; Sincronização InterBloco;**

I. Introduction

Algoritmos que usam programação dinâmica(DP) são geralmente algoritmos que resolvem problemas de optimização discretos, aonde deve se decompor um problema em problemas menores e ao resolver os mesmos pode-se utilizar uma função de composição para resolver o problema pai, essa função é a função recursiva do problema. Como nesses problemas o mesmo subproblema pode ser utilizado para resolver multiplos subproblemas em uma camada superior podemos memorizar a solução em uma tabela de "memoization"assim evitando ter que descer a partir folha na árvore de recursão novamente, se aplicarmos esse pensamento para toda a estrutura teremos um algoritmo que utiliza o paradigma DP. Porém, mesmo utilizando técnicas de DP multiplos desses algoritmos possuem complexidades temporais de valor elevado em CPU como o problema do RNA Secondary Structure Problem, Traveling Salesman Problem, Knapsack Problem como formulados em [1]. O que é proposto é seja feita a resolução de subproblemas em paralelo porém pra isso é necessario descobrir o nivel de independência entre os subproblemas e para isso temos a classificação de [2] das formulações DP:

Serial	Se subproblemas dependem apenas de subproblemas de camadas anteriores.
Non-serial	Se subproblemas não dependem apenas de subproblemas de camadas anteriores.
Monodiac	Se a função recursiva possui apenas um fator de resolução.
Polyadic	Se a função recursiva possui mais de um fator de resolução.

O algoritmo clássico de RNA Folding to find the Secondary Structure utiliza como critério criar o maior números de ligações entre bases seguindo o complemento de Watson-Crick, ou seja, dada as bases b_i e $b_j \in \{(A)denine, (C)ytosine, (G)uanine, (U)racil\}$ apenas pareamentos C-G e A-U são permitidos, devem haver no mínimo 4 bases entre i e j e as ligações de dois pares diferentes não podem se cruzar. Dados esse parâmetros e que $OPT(i, j)$ é o número máximo de bases entre i e j podemos chegar a seguinte função de recursão:

TODO:SUBFIGURE FUNÇÃO RECURSIVA a). TODO:SUBFIGURE dependencia triangular b).

Vendo a função a) podemos classifica-lá como Non-serial Polyadic Dynamic Programming(NPDP) esse tipo de DP se caracteriza por possuir um dependência triangular como mostrado em b) diferente das outras classificações que geralmente são retangulares.

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Type style and Fonts

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your system, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

III. Conclusion

The conclusion goes here. this is more of the conclusion

Acknowledgment

The authors would like to thank... more thanks here

REFERÊNCIAS

[1] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[2] V. Kumar, *Introduction to Parallel Computing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.