Abordagem Paralela de Evolução Diferencial Aplicado ao Problema de Predição de Estrutura de Proteína

Renan S. Silva[1], **Rafael S. Parpinelli**[1]

[1]Departamento de Ciências da Computação –
Universidade do Estado de Santa Catarina (UDESC) – Santa Catarina – SC – Brasil

renan.samuel.da.silva@gmail.com, rafael.parpinelli@udesc.com

Resumo. *Este trabalho explora a utilização do algoritmo de Evolução Diferencial paralelizado com MPI. O speedup obtido permite a utilização de operadores de busca de maior custo computacional. Os resultados preliminares indicam que há um speedup significativo da aplicação.*

Proteínas são macromoléculas essenciais para as formas de vida conhecidas, desempenhando papéis regulatórios, metabólicos e estruturais [Walsh 2002]. Sua estrutura tridimensional, que está diretamente associada a sua funcionalidade, é determinada experimentalmente em laboratório através de um processo lento e caro. Atualmente, o problema de predição de estrutura de proteína (PSPP) é considerado um problema em aberto tanto na Ciência da Computação quanto na Bio-informática estrutural e é considerado $\mathcal{NP}$-difícil [Guyeux et al. 2014]. Portanto, a utilização de métodos exatos se torna inviável até mesmo para instâncias de pequeno porte. Sendo assim, abre-se oportunidade para utilização de heurísticas e meta-heurísticas para solução do problema. Considerando a complexidade do problema envolvido, algoritmos meta-heurísticos básicos demonstram limitações e são incapazes de obter soluções significantes em instâncias mais complexas. Logo, o emprego de métodos avançados e a agregação de informações sobre o problema torna-se necessário para uma melhor solução do problema. Além da dificuldade do problema, tem-se ainda a necessidade de avaliar um grande número de funções para determinar a qualidade das soluções durante o processo de otimização, ocupando a maior parte do tempo de execução. Deste modo, a aplicação de uma arquitetura computacional paralela eficiente torna-se necessária para poder escalar a complexidade do PSPP. Com isto em mente, este trabalho tem como objetivo explorar paralelismo no algoritmo de Evolução Diferencial (*Differential Evolution - DE*). O algoritmo DE é um otimizador para problemas de domínio contínuo [Storn and Price 1997] e foi utilizado com sucesso no PSPP em [Narloch and Parpinelli 2017] e [Oliveira et al. 2017]. O modelo de ângulos e torsões é o empregado neste trabalho.

A proposta de paralelização consiste em utilizar uma estratégia *master-slave* para efetuar a avaliação da energia potencial residual das conformações. O processo *master* é responsável por executar todas as operações relacionadas ao DE enquanto que os processos *slave* são responsáveis pela avaliação das conformações. Para medir o desempenho do método mediu-se o tempo médio total gasto em cada execução do DE. Variou-se o número de processos *slave* entre 1 e 4 e comparando com a implementação serial. A máquina utilizada para testes possui um processador Intel Core i5-3570k com 4 núcleos e 16 Gb de RAM. O DE foi executado 10 vezes com os seguintes parâmetros: $Cr = 1.0$ e $F = 0.5$, 5000 gerações e 100 indivíduos. Os testes foram conduzidos na proteína 1ZDD, que possui 34 aminoácidos e 179 ângulos a serem otimizados. Como esperado, as

	Tempo de execução	Speedup
Serial	72m	1.00
1 *slave*	74m	0.97
2 *slaves*	43m	1.82
3 *slaves*	34m	2.07
4 *slaves*	56m	1.28

Tabela 1. Tempo médio por execução e *speedup*

aplicações com e sem paralelização não apresentam diferenças significativas nos resultados da otimização.

Na Tabela 1 pode-se observar que o aumento do número de processos *slave* reduz o tempo médio gasto por execução do algoritmo DE. Observa-se que quando o número total de processos é menor ou igual ao número de núcleos da máquina obtêm-se uma redução no tempo de processamento. Já quando este número é extrapolado o processo *master* passa a competir por recursos e isto leva a atrasos na comunicação, tendo em vista que o processo *master* é o mais sensível a gargalos. Comparando a implementação serial com a paralela utilizando apenas um *slave* nota-se que não há um aumento significativo no tempo de processamento, indicando que existe apenas um pequeno *overhead* devido a utilização do MPI. Analisando-se o *speedup* pode-se observar que o ganho máximo ocorre quando há 3 processos *slave*. O fato do *speedup* estar a quase uma unidade de distância do valor ideal 3 é devido a existência de trechos de códigos seriais no algoritmo. Ao sobrecarregar os núcleos o *speedup* cai consideravelmente.

A implementação foi desenvolvida com python3.5 e mpi4py. O uso destas ferramentas permitiu uma rápida prototipagem do método e foi capaz de apresentar uma redução significativa de velocidade no processo de predição, possibilitando novos direcionamentos na pesquisa. Trabalhos futuros incluem a paralelização de mais porções do código com o objetivo de melhorar o *speedup*. Pode-se ainda utilizar controle automático de parâmetros e aplicar um modelo distribuído em ilhas como modelo de paralelismo.

Referências

Guyeux, C., Côté, N. M.-L., Bahi, J. M., and Bienia, W. (2014). Is protein folding problem really a np-complete one? first investigations. *Journal of bioinformatics and computational biology*, 12(01):1350017.

Narloch, P. H. and Parpinelli, R. S. (2017). The protein structure prediction problem approached by a cascade differential evolution algorithm using rosetta. In *6th Brazilian Conference on Intelligent Systems*, pages 294–299.

Oliveira, M., Borguesan, B., and Dorn, M. (2017). Sade-spl: A self-adapting differential evolution algorithm with a loop structure pattern library for the psp problem. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 1095–1102. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Walsh, G. (2002). *Proteins: biochemistry and biotechnology*. John Wiley & Sons.