# Paralelização do algoritmo de competição e cooperação de partículas

**Raul Moreira de Souza, Fabricio Aparecido Breve, UNESP Rio Claro, IGCE, Ciências da Computação, raul_ms02@yahoo.com.br, FAPESP.**

## Introdução

O algoritmo proposto por Breve et al.(2012)[1] é um modelo de aprendizado de máquina semi-supervisionado que pertence ao subcampo de ciência da computação chamado computação inspirada pela natureza, que consiste no desenvolvimento de algoritmos baseados em comportamentos naturais. É um método para classificação de dados baseado em grafos que trata os dados pré-classificados como seres vivos genéricos(chamados de partículas) movimentando-os pelo grafo simulando a cooperação e competição de espécies no meio ambiente e classificando os dados com base na dominação de cada grupo de partículas sobre o grafo.

## Objetivos

O objetivo da pesquisa foi entender o algoritmo proposto e paralelizá-lo para execução tanto em CPU quanto em GPU aumentando sua velocidade sem prejudicar sua precisão, utilizando técnicas de programação concorrente.

## Material e Métodos

Para a paralelização do algoritmo foram utilizadas duas plataformas diferentes: OpenMP para programação concorrente em CPU e a arquitetura CUDA em GPU.

Em CPU, como o número de threads normalmente é inferior ao número de partículas, cada thread processa a movimentação de um grupo de partículas durante todas as iterações do algoritmo. Inicialmente, a proposta era implementar técnicas de mutexes para evitar o conflito de acesso à dados, porém, a forma como as threads e variáveis dentro destas foram organizadas foi suficiente para evitar conflitos, então tais técnicas foram retiradas pelo seu custo computacional.

Em GPU a maior dificuldade está relacionada ao fator de ocupação da placa de vídeo. As GPU's com arquitetura CUDA foram desenvolvidas para executar milhares de threads. Se o processamento não puder ser dividido em um grande número de threads, a placa de vídeo não atinge sua eficiência máxima. Este algoritmo na maioria dos casos não pode ser dividido em número tão grande, mesmo deixando uma partícula para cada thread. A utilização de operações atômicas para evitar conflito de acesso à dados foi mantida em GPU pela maior quantidade de threads, fato que torna o processo mais lento. As dificuldades da execução em GPU a tornam, na maioria dos casos, mais lenta que em CPU com processamento paralelo, porém, na maioria dos casos continua mais rápido que em processamento linear em CPU.

## Resultados e Discussão

**Tabela 1.** Comparação de tempo de execução do algoritmo em milisegundos.

| nº de partículas | CPU linear | GPU | CPU paralelo |
|---|---|---|---|
| 512 | 1997 | 379 | 177 |
| 1024 | 4156 | 391 | 219 |
| 1536 | 6533 | 448 | 283 |
| 2048 | 9082 | 587 | 358 |

Os testes da tabela 1 foram realizados sobre a base de dados shuttle[2](utilizando 20000 das 50000 instâncias). Executando 100 vezes com 10000 iterações para cada teste em um computador com processador Intel Core i7 2600K, 32.0GB de memória RAM e duas GPU's NVidia GEForce GTX 560Ti.

Como pode ser visto na tabela, o melhor desempenho foi obtido com processamento paralelo em CPU, chegando a ser mais de 25 vezes mais rápido que o obtido com processamento linear. O algoritmo em GPU, apesar de não ter sido tão eficiente quanto em CPU paralelo, foi mais rápido que em CPU linear em todos os casos testados.

## Conclusões

A paralelização foi realizada com sucesso nas duas plataformas propostas com aumento de velocidade considerável.

## Agradecimentos

Agradeço a FAPESP por ter financiado minha pesquisa.

[1]Breve F. A.; Zhao L.; Quiles M. G.; Pedrycz W.; Liu J. "Particle competition and cooperation in networks for semisupervised learning". *IEEE Transactions on Knowledge and Data Engineering (PrePrints)*, 2012. DOI 10.1109/TKDE.2011.119, URL:http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.119

[2] Frank, A.; asuncion, A. "UCI Machine Learning Repository", 2010. Disponível em: <http://archive.ics.uci.edu/ml>.