1
Introdução

1.1
Descrição do problema

A programação genética (PG) é uma meta-heurística utilizada para gerar programas de computadores, de modo que o computador possa resolver problemas de forma automática [1, 2, 3]. Em PG, uma população de programas de computadores é evoluída baseada no princípio da seleção natural de Darwin. Os programas de computadores, ou indivíduos, são criados, modificados, avaliados e selecionados com base na sua aptidão para dar origem a indivíduos com melhores características. Desde sua criação, a PG tem sido aplicada com sucesso a diversos problemas reais tais como *design* automático, reconhecimento de padrões, controle de robôs, mineração de dados e análise de imagens [2, 4, 5, 6, 7, 8].

A programação genética, entretanto, apresenta um elevado custo computacional. O espaço de busca de alguns problemas pode ser gigantesco e a função de avaliação muito custosa. Quando é preciso avaliar milhões de indivíduos durante o processo evolutivo, o tempo de processamento pode se tornar inviável. Este custo computacional se torna o principal limite para se permitir que problemas realistas sejam avaliados. Diante deste cenário, pesquisadores têm buscado reduzir o tempo de processamento da PG para evoluir problemas mais complexos através do emprego de técnicas de processamento paralelo [9, 10]. A PG permite explorar processamento paralelo em diferentes níveis: múltiplos indivíduos podem ser avaliados ao mesmo tempo ou múltiplas amostras de dados de um mesmo indivíduo podem ser avaliadas em paralelo. Tradicionalmente, estratégias de processamento paralelo têm sido empregadas em PG utilizando máquinas com multiprocessadores ou *clusters* de computadores [11, 12, 13].

Recentemente, o surgimento da programação de propósito geral em Unidades de Processamento Gráfico (ou GPUs) fornece uma oportunidade para se reduzir drasticamente o tempo computacional de problemas custosos tais como a PG. As GPUs têm se tornado populares devido ao alto poder computacional, ao baixo custo, à impressionante capacidade de realizar operações de ponto flutuante e à alta taxa de transferência de memória. Estas características fazem das GPUs uma plataforma bastante atrativa para acelerar o processo evolutivo da PG. A natureza paralela da PG é muito adequada ao

paralelismo de granularidade fina fornecido pela arquitetura da GPU. Além disso, GPUs estão presentes na grande maioria dos computadores modernos, inclusive em aparelhos celulares *smartphones* e outros dispositivos móveis, a um custo relativamente baixo.

O poder computacional das GPUs tem sido empregado para acelerar a PG em diversos trabalhos anteriores. Podemos dividir estes esforços em duas metodologias principais: (i) *compilação* [14, 15, 16, 17]; e (ii) *interpretação* [18, 19, 20, 21]. Na metodologia de *compilação*, cada programa ou indivíduo da PG é dinamicamente compilado para código de máquina de GPU e em seguida avaliado em paralelo na GPU. Por outro lado, na metodologia de *interpretação*, é utilizado um único interpretador residente na GPU, que executa os programas imediatamente para avaliar cada indivíduo do processo evolutivo da PG. Neste caso, os indivíduos também são avaliados em paralelo na GPU.

Estas metodologias têm sido usadas com níveis variados de sucesso e apresentam diferentes vantagens e desvantagens. Na metodologia de *compilação*, o paralelismo com granularidade fina das GPUs pode ser explorado através da avaliação de múltiplos indivíduos e de múltiplas amostras de dados simultaneamente. Entretanto, o tempo gasto compilando cada indivíduo da PG influencia consideravelmente o tempo de execução, tornando o processo muito lento. Quando o processo evolutivo da PG precisa avaliar milhões de indivíduos, gastar alguns segundos para compilar um único programa de GPU se transforma num grande obstáculo para produzir resultados dentro de um período razoável de tempo. Já na metodologia de *interpretação*, o interpretador utilizado é compilado uma única vez e reutilizado milhões de vezes. Esta abordagem elimina o tempo gasto com a compilação, mas adiciona o custo de interpretar cada instrução para cada programa evoluído. A metodologia da *interpretação* geralmente funciona bem para programas com poucas instruções e com poucas amostras de dados para treinamento.

1.2 Objetivos

O objetivo deste trabalho é acelerar a computação da PG explorando o paralelismo maciço oferecido pela GPU sob duas novas metodologias: *compilação em linguagem intermediária* e *criação de indivíduos em código de máquina*. Estas duas metodologias propostas permitem acelerar o processo evolutivo da PG executando a avaliação dos indivíduos com alto nível de paralelismo: múltiplos indivíduos e múltiplas amostras de dados podem ser avaliadas simultaneamente em paralelo. A metodologia baseada em linguagem intermediária utiliza a linguagem Pseudo-assembly ou PTX [22] definida pela

nVidia e requer compilação, contudo, o tempo gasto é centenas de vezes menor do que o tempo necessário para compilar o código original de GPUs, permitindo que problemas maiores possam ser tratados. A metodologia de criação de indivíduos em código de máquina, permite eliminar totalmente o tempo gasto com a compilação de indivíduos da PG, sem a necessidade de adicionar um custo de interpretação das instruções, uma vez que trabalha modificando diretamente o código de máquina da GPU para cada indivíduo.

As metodologias propostas neste trabalho são baseadas em programação genética linear e utilizam um algoritmo evolucionário inspirado em computação quântica. Na programação genética linear, cada programa é constituído de uma sequência linear de instruções [23, 24, 25]. Este método, portanto, é mais adequado para trabalhar com programas em código de máquina, uma vez que as arquiteturas de computadores requerem que os programas sejam fornecidos como sequencias lineares. Os computadores não executam naturalmente programas representados por estruturas em árvore, sendo necessário, neste caso, empregar compiladores ou interpretadores [3].

Algoritmos com inspiração quântica representam um dos avanços mais recentes da computação evolucionária [26]. São baseados nos princípios da mecânica quântica, particularmente nos conceitos de bit quântico e de superposição de estados, e podem representar diversos indivíduos de uma maneira probabilística. Desta forma, oferecem um mecanismo evolucionário diferente que em alguns casos é mais eficiente do que os algoritmos evolucionários tradicionais. A representação probabilística da inspiração quântica reduz o número de cromossomos necessários para garantir a adequada diversidade de busca. A interferência quântica contribui para tornar mais rápida a convergência para a melhor solução, através da inclusão da história passada dos indivíduos. Esta abordagem é usada para guiar a população de indivíduos explorando as vizinhanças das soluções atuais na busca da melhor solução.

Além de explorar o alto grau de paralelismo da GPU para acelerar a PG, este trabalho explora também o fato de que aceleradores como GPUs são sempre conectados a uma CPU. Isto permite que tanto a GPU como a CPU possam ser utilizadas na PG. O segundo objetivo deste trabalho é avaliar duas formas de divisão de trabalho entre a CPU e a GPU. O processo de PG consiste basicamente de três tarefas principais: (i) inicializar os indivíduos com sua representação probabilística herdada da inspiração quântica; (ii) observar os indivíduos quânticos para gerar indivíduos clássicos; (iii) avaliar os indivíduos clássicos; (iv) selecionar o melhor indivíduo; e (v) atualizar a evolução ou geração. Para avaliar a divisão de tarefas em um ambiente

heterogêneo composto de GPU e CPU, duas abordagens foram propostas: solução híbrida (CPU-GPU) e solução GPU. A solução híbrida (CPU-GPU) implementa apenas a tarefa (iii) na GPU. A avaliação dos indivíduos é o principal gargalo da PG. Ela é implementada na GPU explorando-se o paralelismo no nível dos indivíduos e no nível das amostras de dados. O restante das tarefas é executado na CPU. Na solução GPU, todas as cinco tarefas são executadas na GPU e a CPU permanece ociosa. A quantidade de dados a serem tratados nas tarefas (i), (ii) e (iv) pode levar a diferenças de desempenho entre as soluções híbridas e GPU. Quando a quantidade de dados é suficiente para explorar o paralelismo maciço da GPU, é mais vantajoso deixar a CPU ociosa e executar todas as tarefas da PG na GPU. Caso contrário, o uso da CPU passa a ser proveitoso.

1.3 Contribuições

Este trabalho provê as seguintes contribuições:

- Proposta e implementação de duas novas metodologias para a implementação da PG em GPUs que exploram a linguagem intermediária e a linguagem de máquina da GPU para a geração dos indivíduos, reduzindo (ou eliminando) o tempo de compilação dos indivíduos sem incluir *overheads* em tempo de execução para sua interpretação.
- Utilização de modelo de computação quântica para a PG em GPU de modo a obter convergência rápida, capacidade de busca global e inclusão da história passada dos indivíduos.
- Proposta e implementação de uma metodologia para a captura do código de máquina de uma GPU.
- Avaliação de duas formas de divisão de trabalho entre a CPU e a GPU.
- Proposta e implementação de otimização da seleção do melhor indivíduo na metodologia que explora a linguagem de máquina da GPU.
- Comparação do desempenho das metodologias propostas com as metodologias propostas na literatura: compilação e interpretação.
- Caracterização do desempenho de *benchmarks* que representam problemas reais complexos como previsão de séries temporais, filtros de processamento de imagens e classificação de intrusos na rede.

1.3.1 Publicações

As contribuições deste trabalho geraram por enquanto as seguintes publicações:

– Cupertino, L., Silva, C. P., Dias, D. M., Pacheco, M. A. C., Bentes, C. Evolving CUDA PTX Programs by Quantum Inspired Linear Genetic Programming. *In: GECCO '11: Genetic and Evolutionary Computation Conference, Workshop on Computational Intelligence on Consumer Games and Graphics Hardware,* 2011.

– Silva, C. P., Dias, D. M., Cupertino, L., Bentes, C., Pacheco, M. A. C. Evolving GPU Machine Code. *Journal of Machine Learning Research.* Artigo aceito em fase de revisão final.

Os resultados obtidos por esta tese despertaram grande interesse da comunidade de PG. Eles foram requeridos para constar em capítulo de livro de destaque na área:

– Langdon, William B. (2014). Large Scale Bioinformatics Data Mining with Parallel Genetic Programming on Graphics Processing Units. In: Shigeyoshi Tsutsui and Pierre Collet ed. *Massively Parallel Evolutionary Computation on GPGPUs.* Springer, pp 311–347.

1.4 Organização do trabalho

Este trabalho encontra-se organizado da seguinte forma: O Capítulo 2 descreve os trabalhos relacionados. Os conceitos básicos de Programação Genética são apresentadas no Capítulo 3. O Capítulo 4 descreve o modelo de computação quântica para PG. A arquitetura das GPUs é apresentada no Capítulo 5. O Capítulo 6 descreve a proposta da utilização da compilação em linguagem intermediária para acelerar PG. A proposta da criação de indivíduos em código de máquina é descrita no Capítulo 7, juntamente com a proposta de diferentes divisões de trabalho entre a CPU e a GPU. O Capítulo 8 apresenta os resultados experimentais e o Capítulo 9, as conclusões.