Preditor de Desempenho de GPUs aplicado à Exploração do Espaço de Projetos ciente de Dark Silicon

Liana Duenha, Rhayssa Sonohata, Danillo C. A. Arigoni, Ricardo Santos

[1]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)– Campo Grande – MS – Brasil

{liana.duenha,rhayssa.sonohata,danillo.arigoni, ricardo.santos}@ufms.br

Abstract. *Performance simulators of GP-GPU heterogeneous systems aims to provide performance accuracy at the cost of execution time. This work handles the problem of time-cost in simulations of design space exploration systems based on GPUs. We have developed and evaluate performance preditors from machine learning techniques focusing on high accuracy and low runtime overhead. We measure the prediction accuracy of the preditors by using the coefficient correlation* (R^2)*, training score, and cross-validation. We have adopted Random Forest and SVR-based preditors since they have met the best compromise in performance and accuracy.*

Resumo. *Simuladores de sistemas heterogêneos GP-GPU procuram oferecer acurácia de desempenho ao custo de elevado tempo de execução. Com o objetivo de evitar o custoso processo de simulação durante as etapas de exploração arquitetural de sistemas baseados em GPUs, desenvolvemos e avaliamos diversos preditores de desempenho de GPUs baseados em algoritmos de aprendizado de máquina com acurácia e baixo custo computacional. A qualidade dos preditores desenvolvidos neste trabalho foi avaliada por meio de métricas como coeficiente de determinação, score de treinamento e validação cruzada. Preditores baseados nas técnicas de Random Forest e SVR apresentaram os melhores resultados tanto em acurácia quanto performance.*

1. Introdução

A Lei de Moore e a escala de Dennard [Dennard et al. 1974] direcionaram, por muitos anos, a evolução do processo tecnológico na indústria de semicondutores. Entretanto, os projetos de circuitos com transistores fabricados em processos tecnológicos abaixo de 90nm não seguem a escala prevista. Segundo Dennard [Dennard et al. 1974], isto é causado pelo aumento exponencial da corrente de fuga, o que causa um aumento na densidade de potência e na potência dissipada total do chip. Como consequência, a área de um chip que pode funcionar com sua frequência máxima vem caindo exponencialmente a cada geração, forçando projetos atuais a inutilizar ou diminuir a frequência em uma parte considerável do chip. A porcentagem do chip que deve ser desligada é denominada *dark silicon* [Santos et al. 2017, Taylor 2012].

Diante destas limitações, fabricantes buscaram alternativas para contornar o problema de *dark silicon* e garantir a continuidade do aumento do desempenho dos sistemas computacionais. Os aceleradores como as Unidades de Processamento Gráfico (GPUs),

em conjunto com múltiplos núcleos de processamento, passaram a compor sistemas heterogêneos para processamentos de demandas computacionais de propósito geral, comumente chamados de sistemas GP-GPU.

A heterogeneidade introduziu uma série de novos desafios, por exemplo: o desenvolvimento de ferramentas e técnicas para programação neste novo cenário; a garantia de escalabilidade; a adaptação de aplicações genéricas para o contexto de GPUs, entre outros. E ainda, no contexto do projeto de hardware: a definição de quais características arquiteturais alcançam melhor desempenho; a busca por desempenho sem comprometer a dissipação de potência e consumo energético; limitações relativas à área, entre outros.

Embora demandas como estas possam ser exploradas por meio de simulação, as ferramentas de modelagem e simulação de sistemas compostos por processadores e unidades de processamento gráfico (do inglês: *General Purpose-Graphics Processing Units* - GP-GPUs), além de escassas, são pouco amigáveis. Adicionalmente, avaliações baseadas em simulação requerem alto custo computacional para garantir a precisão dos resultados. Posto isso, torna-se útil o desenvolvimento de um preditor de desempenho para GPUs que venha a substituir o custoso processo de simulação durante o ciclo de vida de projeto de sistemas GP-GPUs, sem perda de acurácia, com taxas de erros controladas e baixo custo computacional.

Este trabalho apresenta o desenvolvimento e avaliação de modelos de predição de desempenho de GPUs utilizando técnicas de aprendizado de máquina, resultando nas seguintes contribuições diretas:

- Definição de preditores de desempenho de GPUs a partir da avaliação de vários modelos como: Regressão Linear, Regressão Polinomial, kNN, Random Forest e SVMs com kernel Linear e RBF;
- Metodologia de avaliação experimental para determinação do modelo mais adequado para a aplicação neste trabalho;
- Aplicação de um preditor baseado em Random Forest em um conjunto de dados previamente definido por um sistema de exploração de espaço de projetos e comparação com um preditor de desempenho otimista.

O artigo é organizado da seguinte maneira: a Seção 2 descreve trabalhos anteriores que fundamentaram teoricamente nossa pesquisa; a Seção 3 apresenta a metodologia para implementação dos preditores, incluindo as métricas de avaliação e o conjunto de treinamento; a Seção 4 mostra os dados resultantes das avaliações dos modelos; a Seção 5 apresenta a aplicação de um dos preditores à ferramenta MultiExplorer para exploração do espaço de projetos de sistemas GP-GPU; Seção 6 conclui este trabalho[1].

2. Trabalhos Relacionados

Uma das demandas iniciais para desenvolvimento desta pesquisa é encontrar um simulador adequado para sistemas GP-GPUs, pois a partir do processo massivo de simulação gera-se um conjunto de dados de treinamento que permitam treinar e avaliar os modelos de predição. Nos próximos parágrafos citaremos ferramentas e metodologias para modelagem e simulação de sistemas com GPUs.

[1]Agradecemos à CAPES e à Fundect pelo apoio financeiro.

Stargazer é um framework automatizado baseado em uma modelagem por regressão [Jia et al. 2012]. Por meio de parâmetros coletados pela simulação de modelos de GPU, a ferramenta provê estimativa de desempenho do sistema e traça os parâmetros mais importantes da arquitetura com menos de 1,1% de erro.

Bakhoda et al. [Bakhoda et al. 2009] propõem um modelo de predição de desempenho para plataforma CUDA GP-GPU. O modelo proposto analisa o pseudo-código de um kernel CUDA para obter uma estimativa de desempenho, de forma similar às análises assintóticas.

Gupta et al. [Gupta et al. 2016] apresentam um modelo de desempenho de tempo de execução adaptativo para GPUs integradas que prevê o tempo de processamento do quadro. O modelo estima a sensibilidade do tempo de quadro à frequência da GPU. Testes com benchmarks comuns de GPU mostram que o erro médio percentual e o erro absoluto médio na previsão do tempo do quadro e na estimativa da sensibilidade do tempo do quadro é de 3,1% e 3,9%, respectivamente.

Outros trabalhos são baseados em alternativas de sistemas heterogêneos como, por exemplo, chips FPGAs (*Field Programmable Gate Array*), para melhoria de desempenho com foco em eficiência energética. O trabalho de Morris e Aubury [Morris and Aubury 2007] explora a utilização de FPGAs em comparação com GPUs em exploração de espaço de projeto por meio da simulação de Monte Carlo [Mooney 1997].

O GPGPU-Sim [Fung et al. 2007] é uma ferramenta de modelagem e simulação de arquiteturas para alguns modelos de GPUs com precisão de ciclo, que fornece estatísticas de desempenho mediante a execução de aplicações baseadas em CUDA e OpenCL. Em 2013, Leng et al. [Leng et al. 2013] acoplaram ao GPGPU-Sim a ferramenta GPUWattch, um estimador de parâmetros físicos de projetos de hardware baseados no McPAT [Li et al. 2013], melhorando a robustez da ferramenta original, que passou a fornecer, além de parâmetros e estatísticas de desempenho, também parâmetros físicos de área e potência dos componentes arquiteturais das GPUs. GPGPU-Sim foi a ferramenta escolhida para gerar os dados do conjunto de treinamento para o desenvolvimento dos preditores neste trabalho.

3. Modelagem de Preditores de Desempenho para GPUs

Esta seção apresenta a metodologia adotada para desenvolvimento e avaliação dos preditores de desempenho de GPUs, incluindo o levantamento dos dados de treinamento, a definição da métrica de desempenho para rotular os dados e as técnicas para avaliação dos modelos.

3.1. Conjunto de treinamento

Os dados do conjunto de treinamento foram obtidos por meio da simulação de distintas configurações de GPUs descritas na Tabela 1 no GPGPU-Sim. Os dados relativos à tecnologia de fabricação e frequência são fornecidos pelos fabricantes e os parâmetros de área da unidade de computação (UC) e potência foram estimados pela ferramenta GPGPU-Sim, com base na configuração da plataforma e no processo tecnológico. O GPGPU-Sim utiliza, como motor de estimativas físicas, a ferramenta McPAT [Li et al. 2013], que possui imprecisões de estimativas de área (erros máximos de 20%).

Tabela 1. Arquiteturas de GPUs usadas para gerar o conjunto de treinamento.

Modelos de GPUs	**Tecnologia** (nm)	**Frequência** (MHz)	**Potência** (W)	**Área da UC** (mm^2)
Quadro FX5600	65	650	16,00	32,41
GeForce GTX8800	65	570	19,69	13,98
GeForce GTX9800	65	670	18,65	16,31
GeForce GTX285	65	640	41,06	18,81
GeForce GTX480	65	700	28,52	77,29
GeForce GTX580	65	770	30,05	77,29

Foram executadas cinco aplicações paralelas do CUDALibraries (Tabela 2) nas plataformas de GPUs da Tabela 1, variando a quantidade de UCs de 4 em 4, iniciando em 1 UC e terminando em 40 UCs.

Uma vez realizadas as simulações, foi necessário estabelecer uma métrica para o desempenho da plataforma sobre todo conjunto de aplicações, que levasse em consideração as diferenças de carga de trabalho. Para cada aplicação i, definimos K_i como o fator de ponderação que refletirá no peso dessa aplicação no cálculo de desempenho, como mostrado na equação 1:

$$K_i = \frac{N_i}{\sum_{i=1}^{5} N_i} \tag{1}$$

O fator de ponderação considera a quantidade de ciclos para executar a aplicação (N_i) com relação ao total de ciclos para todo o *benchmark*. Uma vez conhecidas as quantidades de instruções e de ciclos necessários para executar cada aplicação, calcula-se o IPC (instruções por ciclo) de cada aplicação e aplica-se a Equação 2 para obter a métrica de desempenho que será utilizada para rotular o conjunto de treinamento. Note que a utilização da frequência de operação na equação 2 garante que a métrica resultante expresse valores em *milhões de instruções por segundo (MIPS)*.

$$D = F \times \sum_{i=1}^{5} (IPC_i \times K_i) \tag{2}$$

Tabela 2. Quantidade de Instruções executadas por aplicação da CUDA-Libraries.

Aplicações	**Instruções**
MonteCarlo	1.220.425.764
simpleMultiGPU	335.691.776
asyncAPI	218.103.808
scalarProd	22.083.328
clock	2.114.304

Definimos, então, o conjunto de treinamento $T \in \mathbb{R}^m \times \mathbb{R}$ como o conjunto de q pares (X_j, D_j), $(1 \leq j \leq q)$, onde X_j corresponde ao vetor de entrada com m parâmetros que definem a plataforma e $D_j \in \mathbb{R}$ corresponde ao desempenho obtido a partir da aplicação da equação 2. Os parâmetros usados para definir a plataforma de GPU do conjunto de treinamento são os seguintes:

- Quantidade de UCs da plataforma;
- Frequência de operação das UCs
- Frequência de operação da memória;
- Capacidade de armazenamento da memória;
- Largura de banda da memória;

3.2. Modelos de Aprendizado de Máquina

Esta seção apresenta características gerais de modelos de aprendizado de máquina aplicados à regressão, os quais apresentam-se como os mais adequados para utilização como preditores de desempenho.

3.2.1. Regressão Linear e Polinomial

A regressão linear é um dos algoritmos mais simples de aprendizado de máquina e uma das primeiras formas de análise regressiva utilizada em aplicações práticas. Utilizamos uma regressão linear para predizer um valor numérico resultante de uma função linear aplicada sobre um conjunto de dados de entrada. A regressão não linear ou polinomial de grau *n* advém de combinações dos parâmetros de entrada presentes no conjunto de treinamento, resultando em um polinômio com o grau desejado [Harrington 2012, Santos et al. 2019a].

3.2.2. Random Forest

Random Forest é um sistema de aprendizado de máquina baseado em uma combinação de árvores de decisão, que são modelos estatísticos utilizados para classificação e problemas de regressão. Uma árvore de decisão é representada por nós, contendo, cada um, um teste sobre algum atributo. Cada ramo descendente corresponde a um possível valor desse atributo e as folhas correspondem a uma saída possível (denominada classe). Cada percurso da árvore, da raiz à folha, corresponde a uma regra de classificação [Breiman 2001].

3.2.3. K-Nearest Neighbours - kNN

kNN é um método utilizado para classificação e reconhecimento de padrões. A ideia fundamental do algoritmo é classificar um ponto com base na análise de k vizinhos próximos, sendo k parametrizável. Esse método é utilizado para classificação de objetos com n-dimensões ou, em outras palavras, um objeto que possui n atributos, baseado na similaridade que possuem com outros objetos com m-dimensões [Parsian 2015, Santos et al. 2019a].

3.2.4. Máquinas de Vetores de Suporte

SVMs (do inglês, *Support Vector Machines*) são sistemas de aprendizado que analisam os dados e reconhecem padrões, usados para classificação ou análise de regressão. Esta estratégia de aprendizado teve, em poucos anos de sua introdução, resultados melhores do que a maioria dos sistemas em uma ampla variedade de aplicações. As SVMs adaptadas para casos de regressão mantêm todas as principais características do algoritmo original e são chamadas de *Support Vector Regression Machines (SVRs)*. SVMs são amplamente utilizadas em aplicações de reconhecimento de imagem, classificação independente de

aspecto, classificação baseada em núcleos de processamento, reconhecimento de dígitos escritos à mão, detecção de homologia de proteína, entre outras [Santos et al. 2019a].

3.3. Metodologia de Avaliação

O *score* é o coeficiente de determinação, também conhecido como R^2, utilizado como a medida de qualidade do ajuste do modelo. O *score* é definido na equação 3, na qual u e v representam a soma residual de quadrados e a soma total de quadrados, respectivamente, considerando que y corresponde às respostas já conhecidas e validadas, y_{pred} corresponde à predição realizada pelo modelo, q corresponde ao número de exemplos utilizados e $mean$ é a média simples de todas as respostas utilizadas [Santos et al. 2019a].

$$score = (1 - \frac{u}{v}) = 1 - \frac{\sum_{i=1}^{q}(y - y_{pred})^2}{\sum_{i=1}^{q}(y - mean)^2} \tag{3}$$

Utiliza-se o coeficiente de determinação de duas formas: o *score de treinamento* reflete a comparação da resposta do preditor com as respostas dos exemplos usados para treinar o modelo. Em seguida é obtido o *score de validação cruzada*, a partir da divisão do conjunto de treinamento em duas partes, uma para treinamento e outra para validação do modelo. Neste trabalho, utilizamos 80% do conjunto total de dados para treinamento, enquanto 20% foi separado para avaliação do modelo. Após o treinamento, o *score* do grupo de 20% é calculado, permitindo a avaliação do comportamento do modelo em casos de entradas desconhecidas.

Uma abordagem necessária para reduzir *overfitting* é a aplicação de técnicas de regularização que penalizam o aumento da especialização do modelo por meio da diminuição de coeficientes, reduzindo, portanto, a relevância dos atributos a eles relacionados. Desta forma, busca-se inibir que o modelo se especialize excessivamente em seu conjunto de treinamento, tornando-o mais genérico e com maior taxa de acerto [Santos et al. 2019a]. O parâmetro C é o fator de regularização usado nos modelos SVMs, o qual tem o objetivo de ponderar a parcela relativa ao erro de dentro da função objetivo [de Bastos 2007]. Uma forma de otimizar o preditor baseado em SVMs é realizar a calibração do parâmetro C, sendo um passo vital nas melhores práticas no uso de SVMs.

4. Avaliação dos Preditores

Esta seção apresenta resultados experimentais do treinamento e avaliação dos modelos de aprendizado de máquina descritos na Seção 3. Utilizamos os regressores disponíveis na biblioteca Scikit-learn [Pedregosa et al. 2011], que além de fornecer modelos já validados, também oferece suporte a diversas técnicas de avaliação de modelos, tais como validação cruzada, *Grid Search* e calibração dos parâmetros.

A Tabela 3 apresenta os resultados de avaliação de diversos modelos de preditores utilizando *score* de treinamento e *score* de teste (ou R^2) como métricas. O modelo linear atingiu *score* de teste 0,56 e *score* de treinamento 0,89. Os modelos polinomiais de segundo a quinto graus atingiram coeficiente de determinação médio de 0,90 e os *scores* de treinamento aumentam (de 0,90 até 0,99) a medida que o grau do polinômio aumenta. Independentemente do grau da função de regressão, nenhuma calibração de parâmetro interfere significativamente nos resultados. Potencialmente, o modelo se especializaria mais, podendo gerar *overfitting*.

Tabela 3. *Score* de treinamento e R^2 dos modelos de predição de desempenho

Modelo	***Score* de treinamento**	**R^2 ou *Score* de teste**
Linear	0,89	0,56
Polinomial (graus 2 a 5)	0,90 0,97 0,98 0,99	0,88 0,92 0,91 0,93
Random Forest	0,99	0,98
kNN	0.73	0,43
SVR (kernel linear)	0.78	0,40
SVR (kernel RBF)	0.94	0,77

O modelo *Random Forest* atingiu um *score* de treinamento de 0,99 e um coeficiente de determinação de 0,98. O modelo kNN configurado com $k = 5$ foi o que obteve o melhor resultado com os experimentos ele alcançou *score* de treinamento de 0,73 e coeficiente de determinação de 0,43. O modelo SVR com *kernel* linear apresentou coeficiente de determinação de 0,40 e o modelo SVR com *kenel* RBF apresentou coeficiente de determinação 0,77.

A aplicação de validação cruzada resultou nos seguintes valores: *scores* de teste próximos a zero ou negativos para os modelos linear e polinomiais de graus 2 a 5, demonstrando a baixa confiabilidade destes modelos; o modelo *Random Forest* atingiu 0,83 no *score* de teste; o modelo kNN atingiu apenas 0,38 no *score* de teste, mostrando pior confiabilidade quando comparado ao modelo *Random Forest*; o modelo SVR com validação cruzada apresentou *score* de teste de 0,35 para o kernel linear e 0,81 para o kernel RBF.

A calibração de parâmetros do modelo SVR utilizado foi o *GridSearch* , o parâmetro C foi atrelado a uma lista começando em 100 e terminando em 5000000, variando de 26 em 26. O *GridSearch* fez uma vasta busca, visando encontrar a melhor combinação de parâmetros. Após a calibração, o parâmetro C foi fixado em 2269618 e foi gerada a curva de aprendizado mostrada na Figura 1. A partir da curva de aprendizado é possível verificar que entre 30 e 40 amostras de teste não há aumento significativo nos *scores* de treinamento e de teste com validação cruzada.

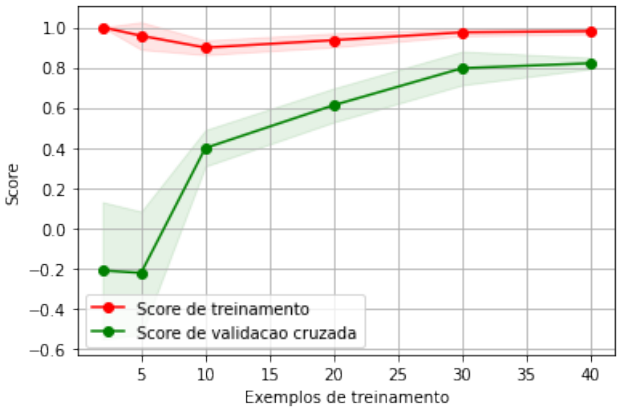

Figura 1. Gráfico de aprendizado.

Uma vez que o objetivo é utilizar modelos preditores em aplicações de exploração de espaço de projetos para sistemas heterogêneos, faz-se necessário analisar, além da precisão, quais modelos têm menor tempo de resposta. A Tabela 4 mostra a quantidade

de predições por segundo alcançada em cada um dos preditores.

Tabela 4. Vazão de predição dos modelos.

Modelos	Milhões de predições por segundo
Linear	4,8
Polinomial (graus 2 a 5)	0,30 0,15 0,22 0,01
kNN	0,33
Random Forest	0,14
SVR RBF	0,37
SVR Linear	0,70

Embora seja o modelo mais preciso com *score* de teste em 0,98, o *Random Forest* apresenta custo computacional maior quanto comparado aos demais modelos, excetuando-se os modelos polinomiais de graus 2 e 5. O modelo SVR-RBF é $2,6\times$ mais rápido que o *Random Forest*, porém mesmo com as devidas calibrações, não foi possível melhorar o *score* do modelo, que alcançou valor máximo de 0,81. Desta forma, ambos os modelos podem ser adotados para os experimentos de exploração do espaço de projeto. Se a quantidade de configurações a serem avaliadas forem excessivamente grandes, o mais adequado é utilizar o modelo de predição menos preciso e mais rápido (SVR-RBF); entretanto, para experimentos com um espaço de projetos menor, sugere-se a aplicação do *Random Forest* pela sua acurácia.

5. Aplicação dos preditores na exploração do espaço de projetos de sistemas GP-GPUs ciente de *dark-silicon*

Em 2019 apresentamos em [Santos et al. 2019b] a aplicação de uma metodologia para exploração do espaço de projetos ciente de *dark silicon* em sistemas heterogêneos GP-GPU, utilizando a ferramenta MultiExplorer [Santos et al. 2017]. O desempenho das plataformas era estimado de maneira otimista. Inicialmente, estimava-se o desempenho de uma unidade de computação de GPU e dos núcleos (*cores*), presentes no banco de dados, por meio de simuladores; em seguida, o desempenho das plataformas heterogêneas era calculado como a soma dos desempenhos dos seus núcleos.

É conhecido que o desempenho dos sistemas computacionais não escala linearmente à medida que aumentamos a quantidade de núcleos processamento. Desta forma, com o objetivo de refinar a metodologia de exploração de espaço de projeto no tocante à acurácia do desempenho, foram aplicados à metodologia de DS-DSE, preditores de desempenho para sistemas *multicore* validados em [Santos et al. 2019a]. Entretanto, não havia ainda um trabalho que integrasse ao processo a predição de sistemas baseados em GPU, o que nos motivou a desenvolver o presente trabalho.

Isto posto, o objetivo desta seção é apresentar resultados preliminares da integração do preditor de GPU *Random Forest* à metodologia aplicada no MultiExplorer para realização de DS-DSE em sistemas heterogêneos, utilizando como base os experimentos realizados em [Santos et al. 2019b].

5.1. Experimento-base

Os experimentos de DS-DSE de [Santos et al. 2019b] utilizando o MultiExplorer foram realizados a partir dos núcleos e unidades de computação apresentados na Tabela 5. A

última coluna utiliza a métrica MIPS (milhões de instruções por segundo) para expressar o desempenho das plataformas a partir da execução dos softwares dos benchmarks SPLASH-2, Parsec e CUDA-Libraries. A simulação de cada modelo gera o valor de MIPS relativo a cada aplicação. O MIPS relativo a todo o *benchmark* é calculado por meio da média ponderada dos MIPS obtidos para cada aplicação separadamente, considerando como fator de ponderação a carga de trabalho de cada aplicação. Assim, aplicações mais robustas serão mais relevantes no cômputo do MIPS final do que as aplicações mais leves. As estimativas dos parâmetros físicos foram obtidas por meio das ferramentas McPAT (para núcleos de CPU) e GPGPU-Sim (para unidades de computação da GPU).

Tabela 5. Núcleos e unidades de computação presentes no banco de *dados*

Núcleos	**Tecnologia**	**Frequência** (GHz)	**Potência** (W)	**Área do núcleo** (mm^2)	**Desempenho de núcleos/UCs**
1 - Smithfield	90nm	2,8	8,9	111,12	4411
2 - Quark x1000	32nm	0,4	1,06	11,32	381,4
3 - ARM A53	22nm	1,6	5,5	6,82	3374,48
4 - ARM A57	22nm	2,0	12,13	6,79	4411,06
5 - Atom Silvermont	22nm	0,5	2,51	6,15	596,67
6 - Quadro FX5600	65nm	0,65	16,00	32,41	20542,16
7 - GeForce GTX8800	65nm	0,57	19,69	13,98	15382,05
8 - GeForce GTX9800	65nm	0,67	18,65	16,31	15391,29
9 - GeForce GTX285	65nm	0,64	41,06	18,81	52144,24
10 - GeForce GTX480	65nm	0,70	28,52	77,29	7932,07
11 - GeForce GTX580	65nm	0,77	30,05	77,29	7935,44
12 - GeForce GTX680	65nm	1,00	34,96	178,11	3943,93
13 - GeForce GTX780ti	65nm	0,87	66,78	182,97	3324,29

A estimativa de *dark silicon* em GPUs apresentada em [Santos et al. 2019a] baseia-se na metodologia original para núcleos de CPU apresentada em [Santos et al. 2017]. Utiliza-se uma GPU modelada com tecnologia de $90nm$ e considera-se, com base na escala de Dennard [Dennard et al. 1974], que o projeto está livre de *dark silicon*. Então, evolui-se esse projeto utilizando processos tecnológicos de $65nm$, $45nm$, $32nm$ e $22nm$ adicionando mais unidades de computação para atingir uma área próxima à área do projeto original. A presença de *dark silicon* no projeto será estimado a partir da densidade de potência do novo projeto, quando comparado com o projeto original livre de *dark silicon*.

Na solução proposta, cada plataforma computacional possui componentes com diferentes densidades de potência, os quais são os candidatos para uma estratégia de exploração de espaço de projeto, uma vez que mudanças na configuração desses componentes alterará a densidade de potência total da plataforma e, como consequência, a área em *dark silicon* no *chip*. Com base nos dados de densidade de potência e área do circuito de referência e do total de potência excedida do projeto, estima-se a área de *dark silicon* do projeto. Em plataformas compostas por GPUs, as unidades de computação (*compute units*) são os componentes com maior densidade de potência. A metodologia de estimativa de *dark silicon* foi aplicada a todos os processadores e GPUs da Tabela 5, escalando a quantidade de unidades de computação e processos tecnológios de $65nm$ a $22nm$ (limite máximo suportado pelo GPGPU-Sim) [Santos et al. 2019a].

Escolheu-se como plataforma-base para os próximos experimentos a GPU GTX8800 com 16 unidades de computação. A partir dessa plataforma-base foi utilizada a ferramenta MultiExplorer com o objetivo de estimar a área do *chip* em *dark silicon*,

usando uma unidade de computação como circuito de referência. Foi encontrado que o chip da GTX8800 apresenta 16,71% de sua área em *dark silicon* em um projeto realizado com tecnologia $65nm$. Ou seja, o experimento aponta que uma área correspondente a essa porcentagem deve ser "desligada" para que a plataforma, na tecnologia especificada, possa ter a mesma potência de sua plataforma-base ($90nm$). Essa informação de *dark silicon* consiste na devida motivação para se realizar um processo de exploração do projeto dessas plataformas, uma vez que, a partir de $32nm$, a maioria das plataformas tem mais da metade da área do *chip* comprometida com *dark silicon* [Santos et al. 2019a].

5.2. Exploração do espaço de projetos ciente de *dark silicon* utilizando o preditor de desempenho de GPUs

Considerando que a plataforma-base tem 16 unidades de computação GTX8800 com 16,71% da área do chip em *dark silicon*, aplica-se o algoritmo de exploração do espaço de projetos do MultiExplorer baseado em força-bruta para realizar a busca e avaliação de todas as soluções alternativas possíveis. Tais alternativas devem ser heterogêneas com até dois tipos de núcleos de computação, sendo que deve haver pelo menos uma unidade de computação original (GTX8800) e outros núcleos alternativos da Tabela 5. As soluções alternativas devem obedecer restrições de área e densidade de potência, e são *rankeadas* pelo desempenho que podem alcançar.

A Tabela 6 apresenta os resultados de soluções encontradas pelo algoritmo de força bruta, todas obedecendo as restrições de área menor ou igual a $493,04$ e densidade de potência menor ou igual a $0,14$ (que correspondem à área de densidade de potência da plataforma original GTX8800 com 16 unidades computacionais). Cada solução é expressa pelos seguintes parâmetros: número de unidades de computação originais (N_O) da plataforma da solução alternativa, a quantidade de novas unidades de computação sugeridas (N_{IP}), o tipo dessas unidades de computação de acordo com a Tabela 5 (T_{IP}), a área ($AT(mm^2)$), densidade de potência ($DP(\frac{W}{mm^2})$) e duas estimativas de desempenho (D_{OT} e D_{RF}) da nova plataforma.

O desempenho otimista (D_{OT}) foi obtido pela soma do desempenho das unidades de computação presentes na plataforma. Claramente, essa é uma estimativa otimista e inalcançável, mas é a metodologia inicialmente empregada no MultiExplorer. Após a integração do preditor de desempenho *Random Forest* à ferramenta MultiExplorer é possível substituir esse cálculo pela predição do desempenho das plataformas (D_{RF}).

Tabela 6. GTX8800: Soluções com o algoritmo de Força Bruta - $65nm$**. Restrições:** $AT \leq 493,04$ **e** $DP \leq 0,14$

	N_O	N_{IP}	T_{IP}	$AT(mm^2)$	$DP(\frac{W}{mm^2})$	D_{OT}	D_{RF}
S1	2	2	11	461,14	0,09	673.158	433.765
S2	2	2	10	461,14	0,09	608.905	423.191
S3	2	1	10	299,13	0,13	352.072	423.191
S4	3	4	1	446,36	0,13	168.570	434.569

As soluções **S1** a **S4** da Tabela 6 foram obtidas sem o uso do preditor durante a execução do algoritmos e estão ranqueadas pela coluna D_{OT}. Como exemplo, descrevemos a primeira solução alternativa (**S1**) composta por 2 unidades de computação

GTX8800, 2 unidades de computação da arquitetura alternativa GTX580 (tipo 11 da Tabela 5), com área de 461,14mm^2 e densidade de potência de 0,09 W/mm^2.

Ao estimar, a posteriori, o desempenho das soluções heterogêneas encontradas, verifica-se que a plataforma $S4$ com 3 núcleos GTX8800 e 4 núcleos Intel Smithfield ($T_{IP} = 1$) apresenta D_{RF} maior do que o predito para $S1$, que corresponde a uma plataforma com 2 GTX8800 e 2 GTX580. Isso nos leva a concluir que, por oferecer uma estimativa mais justa, pode influenciar em melhores escolhas para o algoritmo de exploração do espaço de projetos da ferramenta.

6. Conclusão

Este trabalho apresentou modelos de predição de desempenho para GPUs baseados em técnicas de aprendizado de máquina. Realizamos um abrangente trabalho de análise, modelagem e avaliação de vários modelos de regressão: regressão linear, polinomiais de graus 2 a 5, kNN, Random Forest, SVMs ou máquinas de vetores de suporte adaptadas para regressão (SVR) com diferentes *kernels* (linear e RBF). Para avaliar os modelos, o coeficiente de determinação R^2, *score* de treinamento e validação cruzada, curva de aprendizado e calibração do parâmetro C foram utilizados para modelos baseados em SVMs.

O modelo *Random Forest* apresentou os melhores resultados nas diferentes metodologias de avaliação, atingindo um *score* de teste 0,98 com validação cruzada. Entretanto, esse modelo não apresenta uma taxa de predições por segundo tão alta como o modelo baseado em SVMs.

Durante o processo de exploração do espaço de projetos do MultiExplorer, milhares de configurações alternativas são avaliadas com base no desempenho que podem alcançar. Dois parâmetros de qualidade desse processo são essenciais: a acurácia da estimativa de desempenho e o tempo utilizado nessa estimativa. O método atualmente utilizado para estimar o desempenho de cada solução é muito rápido (apenas algumas operações aritméticas), porém o valor alcançado tem baixa precisão e seria inalcançável no projeto físico. A nossa proposta de integração do preditor de desempenho à metodologia de exploração do espaço de projeto do MultiExplorer garante um valor mais realista na estimativa de desempenho das soluções GP-GPU, sem comprometimento do tempo de execução.

References

Bakhoda, A., Yuan, G. L., Fung, W. W., Wong, H., and Aamodt, T. M. (2009). Analyzing cuda workloads using a detailed gpu simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 163–174. IEEE.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

de Bastos, F. A. C. (2007). *Classificador de modulaçoes baseado em máquinas de vetores de suporte*. PhD thesis, UNIVERSIDADE FEDERAL DO RIO DE JANEIRO.

Dennard, R. H., Gaensslen, F. H., Yu, H.-N., Rideout, V. L., Bassous, E., and LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268.

Fung, W. W., Sham, I., Yuan, G., and Aamodt, T. M. (2007). Dynamic warp formation and scheduling for efficient gpu control flow. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pages 407–420. IEEE.

Gupta, U., Campbell, J., Ogras, U. Y., Ayoub, R., Kishinevsky, M., Paterna, F., and Gumussoy, S. (2016). Adaptive performance prediction for integrated gpus. In *Proceedings of the 35th International Conference on Computer-Aided Design*, pages 1–8.

Harrington, P. (2012). *Machine learning in action*, volume 5. Manning Greenwich, CT.

Jia, W., Shaw, K. A., and Martonosi, M. (2012). Stargazer: Automated regression-based gpu design space exploration. In *2012 IEEE International Symposium on Performance Analysis of Systems & Software*, pages 2–13. IEEE.

Leng, J., Hetherington, T., ElTantawy, A., Gilani, S., Kim, N. S., Aamodt, T. M., and Reddi, V. J. (2013). Gpuwattch: enabling energy optimizations in gpgpus. *ACM SIGARCH Computer Architecture News*, 41(3):487–498.

Li, S., Ahn, J., Strong, R., Brockman, J., Tullsen, D., and Jouppi, N. (2013). The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(1):5.

Mooney, C. Z. (1997). *Monte carlo simulation*, volume 116. Sage publications.

Morris, G. W. and Aubury, M. (2007). Design space exploration of the european option benchmark using hyperstreams. In *2007 International Conference on Field Programmable Logic and Applications*, pages 5–10. IEEE.

Parsian, M. (2015). *Data algorithms: recipes for scaling up with Hadoop and Spark*. " O'Reilly Media, Inc.".

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Santos, M., Sonohata, R., Krebs, C., Segovia, D., Santos, R., and Duenha, L. (2019a). Performance models for heterogeneous systems applied to the dark silicon-aware design space exploration. *Proccedings of the 31st International Symposium on Computer Architecture and High Performance Computing*.

Santos, R., Duenha, L., Silva, A. C., Sousa, M., Tedesco, L. A., Melgarejo, J. C., Santos, T., Azevedo, R., and Moreno, E. (2017). Dark-silicon aware design space exploration. *Journal of Parallel and Distributed Computing*.

Santos, R., Sonohata, R., Krebs, C., Catelan, D., Duenha, L., Segovia, D., and Santos, M. T. (2019b). Exploração do projeto de sistemas baseados em gpu ciente de dark silicon. In *Anais Principais do XX Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 358–369. SBC.

Taylor, M. B. (2012). Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 1131–1136. IEEE.