CARACTERIZAÇÃO DE PROGRAMAS: TAREFA ESSENCIAL À APRENDIZAGEM DE MÁQUINA

Maria Fernanda Almeida Oliveira (PIBIC/CNPq/FA/Uem), Anderson Faustino da Silva (Orientador), e-mail: ra118597@uem.br.

Universidade Estadual de Maringá / Centro de Tecnologia / Maringá, PR.

Ciência da Computação / Sistema de Computação

Palavras-chave: programa, representação, aprendizagem de máquina

Resumo

O conjunto de transformações, aplicado pelo compilador durante o processo de geração de código final, irá impactar o desempenho do código final. Devido às otimizações alterarem a estrutura do código original; embora sem alterar a sua semântica. De fato, é esperado que o código final tenha uma melhor qualidade, quando comparado com o código original. Contudo, tal premissa nem sempre é verdadeira; pois uma determinada transformação pode melhorar a qualidade de um determinado programa mas não necessariamente de outro. Isto indica, que o problema de seleção de transformações é um problema dependente de programa. Sistemas deste porte que empregam estratégias de aprendizagem de máquina é uma ótima opção para o problema de seleção de transformações. Isto por apresentar bons resultados para o problema em questão, além de possuir um baixo tempo de resposta. Por sua vez, sistemas de aprendizagem de máquina requerem um conjunto de dados de treinamento. Neste contexto, este trabalho disponibiliza um conjunto de treinamento para sistemas de seleção de otimização baseados em aprendizagem de máquina.

Introdução

Sistemas inteligentes são sistemas que podem interagir e aprender sobre contextos específicos. Estes sistemas reduzem o custo para resolver problemas complexos devido a sua capacidade de extrair, classificar e acumular conhecimento. Nos últimos anos, vários trabalhos têm aplicado sistemas inteligentes para diferentes contextos, por exemplo, diagnóstico de vitaminas e deficiência mineral no ser humano (Ula e Hendriana, 2016), anomalias arquiteturais em reuso de software, reutilização de mercadorias (Nascimento e Outros, 2017), e geração de código (Fabricio Filho e Outros, 2018).

A geração de código é um processo realizado por compiladores e é considerado um problema complexo por causa da dificuldade em escolher quais transformações de código devem ser utilizadas. Neste contexto,

sistemas inteligentes tem sido largamente utilizados para selecionar transformações e, consequentemente, aplicá-las ao programa sendo compilado. Sistemas inteligentes de geração de código empregam técnicas de aprendizado de máquina e são projetados para serem capazes não só de extrair e classificar o conhecimento, mas também acumular e reutilizá-lo. Tais sistemas criam um modelo de previsão na fase de treinamento, baseado no comportamento dos programas treino e na fase de teste, encontra o melhor conjunto de transformações para um programa não visto pelo sistema (Lima e Outros, 2013).

O modelo de previsão cria uma relação entre efetivos conjunto de transformações e características do programa. Estas características podem ser: (1) contadores de desempenho, (2) grafos de fluxo de controle, (3) dados de compilação, (4) recursos que descrevem laços e estruturas vetoriais do programa, (5) características numéricas, (6) ou uma representação simbólica similar a um DNA.

Embora a literatura apresente diversos sistemas inteligentes para seleção de código, baseados em aprendizagem de máquina, não existe uma base de dados de treinamento que seja comum que possa ser utilizada amplamente. Desta forma, este trabalho tem por objetivo formar uma base de treinamento composta por *N* programas.

Materiais e métodos

O processo de seleção de amostras (programas) para compor a base de dados de treinamento segue três princípios: (1) o programa deve ser compilado sem a ocorrência de erros; (2) deve ser possível gerar código executável; e (3) a execução do código final deve ocorrer sem erros. O uso de tais princípios tem por objetivo fornecer uma base de dados que possa ser utilizada para diferentes fins, por exemplo por sistemas cujo objetivo seja redução do tamanho de código e/ou redução do tempo de execução.

Vale ressaltar dois pontos: (1) o presente trabalho não tem por objetivo definir quais métricas serão utilizadas pelo sistema inteligente, (2) nem definir quais atributos devem ser extraídos dos programas. Como pode ser observado no trabalho de Fabricio Filho e Outros (2018), diferentes sistemas inteligentes utilizam uma diferente representação de programa, consequentemente extraindo atributos diferentes. Desta forma, a definição de quais atributos de programas utilizar foge do escopo deste trabalho. Portanto, este trabalho tem como objetivo atender as necessidades de sistemas como o apresentado por Fabricio Filho e Outros (2018) e outros sendo: “*obter um conjunto de programas cuja execução não possua erros.*”

A seleção considerou os seguintes conjuntos de programas: MiBench[1], cBench[2], Polybench[3] e o conjunto de teste LLVM[4]. Tal seleção gerou uma base de dados com 282 programas, onde cada programa segue os três princípios norteadores para seleção.

Resultados e Discussão

Um questão importante na disponibilização de uma base de dados é estabelecer uma metodologia que a descreva. Para este fim os 282 programas foram categorizados nos seguintes grupos: (1) quantidade de entradas; (2) quantidade de funções; e (3) quantidade de instruções.

A base de dados possui 60 (30 programas do grupo cBench e 30 programas do grupo Polybench) programas com diferentes entradas. Embora esta quantidade seja apenas 21% dos programas, ela é razoável para sistemas que avaliam diferentes entrada de dados. Vale ressaltar que tais sistemas avaliam o comportamento do programa em execução e desta forma cada entrada gera um comportamento diferente. Portanto, nossa base de dados fornece (30x10) + (30x5) comportamentos diferentes.

A Tabela 1 apresenta a caracterização da base de dados de acordo com a quantidade de funções dos programas.

Tabela 1 Caracterização baseada na quantidade de funções e instruções.

Percentil	Funções	Instruções	Percentil	Funções	Instruções
10	*3*	*184,3*	**60**	*14*	*1766,8*
20	*5*	*361,8*	**70**	*19*	*3081,9*
30	*7*	*469,3*	**80**	*27*	*3897,0*
40	*11*	*647,6*	**90**	*75,5*	*12631,0*
50	*13*	*996,5*			

A caracterização dos programas indica que 60% dos programas possuem menos que 15 funções. Isto indica que a maioria dos programas da base são programas *pequenos*. Por outra lado, para um sistema que analisa funções individuais, ao invés do programa completo, a base de dados fornece 15011 funções distintas. Tal quantidade é considerável para um sistema de aprendizagem de máquina, baseado em funções. Observando a quantidade de instruções dos programas é possível perceber que uma quantidade pequena de programas, aproximadamente 10%, possuem mais de 10000 instruções.

Tais resultados indicam que pelo menos 30% dos programas possuem características bem distintas dos outros. Isto é um ponto negativo da base

1 https://github.com/pulp-platform/mibench
2 https://ctuning.org/wiki/index.php/CTools:CBench
3 https://web.cse.ohio-state.edu/~pouchet.2/software/polybench/
4 https://llvm.org/

formada, pois ao agrupar os programas existirão programas cujo "comportamento" sobressai. Tal fato tem o potencial de ocasionar perda de desempenho em sistemas de aprendizagem de máquina, a medida que os dados de treinamento, de uma mesma classe, são divergentes.

Organização da base de dados

A base de dados foi organizada por grupos de programas pertencentes ao mesmo *benchmark,* sendo*:* ACO-Metaheuristic,_Purple, ASC_Sequoia, BenchmarkGame, BitBench, Bullet, CAPBenchmark, cBench, Dhrystone, DOE_ProxyApps_C, Fhourstone, FreeBench, Linpack, llubenchmark, mafft, MallocBench, McCat, McGill, mediabench, mediabench2, MiBench, minisat, Misc, nbench, Olden, Polybench, Prolangs-C, Ptrdist, rodinia, Shootout, Stanford, Trimaran, TSVC e VersaBench. Cada grupo está organizado em distintos diretórios que possuem *scripts* para compilação e execução.

Conclusões

Este trabalho apresentou uma base de dados de programas, que podem ser utilizados por sistemas inteligentes de geração de código durante o treinamento.

Agradecimentos

Agradecemos ao Programa Institucional de Bolsa de Iniciação Cientifica

Referências

Lima E D, Souza Xavier T C, Silva A F, Ruiz L B. **Compiling for performance and power efficiency**. In Proc. the 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation., Sept. 2013, pp.142-149.

Nascimento R J O, Fonseca C A G, Medeiros Neto F D. **Using expert systems for investigating the impact of architectural anomalies on software reuse**. IEEE Latin America Transactions, 2017, 15(2): 374-379.

Fabricio Filho J, Rodrigues L. G. A., Silva A. F. **Yet Another Intelligent Code-Generating System: A Flexible and Low-Cost Solution**. Journal of Computer Science and Technology, v. 33, p. 940-965, 2018.

Ula M, Mursyidah, Hendriana Y, Hardi R. **An expert system for early diagnose of vitamins and minerals deficiency on the body**. In Proc. the International Conference on Information Technology Systems and Innovation, Oct. 2016.