EXTRAÇÃO DE CARACTERÍSTICAS DE PROGRAMAS E VALIDAÇÃO DA ACURÁCIA DE UM SISTEMA INTELIGENTE DE GERAÇÃO DE CÓDIGO

Eduardo Felipe Ribeiro (PIBIC/CNPq/FA/Uem), Anderson Faustino Da Silva (Orientador), e-mail: ra99571@uem.br .

Universidade Estadual de Maringá /Departamento de Informática /Maringá, PR.

Ciências Exatas e da Terra / Ciência da Computação

Palavras-chave: Sistemas inteligentes, aprendizado de máquina, inteligência artificial.

Resumo:

O processo de geração de código realizado pelo compilador busca trazer otimizações para o código final sem alterar sua semântica, aplicando conjuntos de transformações que alteram a estrutura do código de entrada. Uma resposta efetiva para as tomadas de decisões realizadas por um compilador durante esse processo é a utilização de aprendizado de máquina, principalmente devido a sua capacidade de extrair conhecimento de contexto e aplicá-lo de forma satisfatória. Este artigo tem como objetivo avaliar a acurácia do sistema inteligente de geração de código YaCoS diante o uso de diferentes atributos de programas e assim descobrir qual grupo de atributos obteve maior êxito ao ser utilizado. Dos 4 grupos de atributos avaliados, LLVM-stats se mostrou ser o melhor.

Introdução

Realizado por compiladores, o processo de geração de código busca trazer otimizações ao código final aplicando transformações à sua estrutura, porém sem alterar sua semântica. É esperado que esse processo aumente a qualidade do código final. Porém essa premissa nem sempre é verdadeira, pois uma determinada transformação que melhora determinado programa pode não melhorar outro. Encontrar as melhores transformações a serem realizadas em um programa é considerado um problema complexo, devido a dificuldade de escolher quais transformações de código devem ser utilizadas para cada programa (Muchnick, 1997).

Com sua capacidade de extrair, classificar e acumular conhecimento, sistemas inteligentes reduzem o custo da resolução de problemas complexos. Sistemas inteligentes de geração de código utilizam técnicas do aprendizado de máquina para criar um modelo de predição, capaz de encontrar um bom conjunto de transformações para programas ainda não vistos pelo sistema (Lima e Outros, 2013).

O modelo de predição é baseado nos atributos dos programas podendo representar diferentes aspectos dos programas, tais como: quantidade de instruções, acessos à memória, dados de compilação, entre outros. Nesse contexto, uma questão ainda em aberto é a validação da acurácia de sistemas inteligentes de geração de código, mediante o uso de diferentes características. Este é o objetivo deste artigo.

Materiais e Métodos

YaCoS (Filho e Outros, 2018), um sistema inteligente de geração de código, utiliza raciocínio baseado em casos (Riesbeck e Schank, 1989) para encontrar um bom conjunto de transformações para um programa não visto. Para este fim, YaCoS armazena, em uma fase de treino, relações entre programas e bons conjuntos de transformações. Em outras palavras, YaCoS armazena um vetor numérico, o qual representa o programa, e bons conjuntos de transformações. Na fase de teste, YaCoS extrai os atributos do programa teste e os compara com os atributos de treino, para encontrar um programa treino similar. Tal tarefa é realizada calculando a distância Euclidiana entre os vetores treinos e o vetor teste. Por fim, YaCos aplica o conjunto de transformações do programa treino mais similar ao programa teste.

Sendo um sistema parametrizável, YaCoS é um excelente sistema para avaliar diferentes atributos de programas. Este trabalho avalia 4 diferentes grupos de atributos, descritos a seguir.

1. LLVM-stats: atributos extraídos do processo de compilação, como por exemplo quantidade de operações aritméticas removidas.
2. LLVM-inst: atributos que representam a quantidade de cada instrução LLVM.
3. LLVM-bc: atributos provenientes da análise do código gerado, como por exemplo a quantidade de acessos à memória.
4. MSF: atributos que representam as relações entre as entidades que compõem o código fonte do programa, como por exemplo a quantidade de blocos básicos.

Todos esses atributos são valores numéricos, portanto independente da categorização de programas que YaCoS utilize, um programa é representado por um vetor numérico.

Resultados e Discussão

Para avaliar diferentes atributos de programas, YaCoS foi treinado com 4000 programas e os testes foram realizados com 1000 programas. O objetivo foi encontrar um bom conjunto de transformações para cada programa teste que seja superior ao conjunto de transformações padrões Oz. Este é o melhor conjunto de transformações da infraestrutura LLVM para redução de código.

A Tabela 1 sintetiza os resultados obtidos, por cada grupo de atributos.

Tabela 1: Ganhos obtidos por cada grupo de atributos

Todos os programas teste				
	LLVM-bc	LLVM-stats	LLVM-inst	MSF
Maior ganho	71,13	68,56	77,19	77,19
Menor ganho	-1622,22	-1633,33	-1622,22	-1622,22
Ganho médio	-2.334	-1,30	-1,95	-1,45
Apenas programas com ganho de desempenho				
	LLVM-bc	LLVM-stats	LLVM-inst	MSF
Maior ganho	71,13	68,56	77,19	77,19
Menor ganho	0,28	0,17	0,12	0,18
Ganho médio	8,78	8,82	8,61	8,47
Programas	516	580	535	551

Como pode ser visto na Tabela 1, dentre os 1000 programas teste YaCoS conseguiu encontrar conjuntos de transformações que ocasionaram um ganho de desempenho de até 77,19% superior ao conjunto padrão Oz. Porém, alguns programas de teste obtiveram um valor de ganho negativo, o que significa que para estes casos YaCoS não foi capaz de encontrar uma boa solução, havendo assim perda de desempenho. Na base de treino existem conjuntos de transformações que ocasionam uma queda de desempenho de 16 vezes. Isto é uma situação não desejável e indica que existem conjuntos de transformações que são bons apenas para um determinado programa. Analisando todos os programas o ganho médio é negativo, independente do grupo de atributos. Por outro lado, observando apenas os programas que obtiveram ganho de desempenho é possível perceber que independente do grupo de atributos, YaCoS encontrou bons conjuntos de transformações para não mais que 50% dos programas teste. Embora o ganho médio seja significativo, a quantidade de programas que obtiveram ganho de desempenho pode ser considerada baixa. O ideal seria ter uma quantidade maior de programas com ganho de desempenho, mesmo com um ganho médio menor que 8%.

Além do ganho de desempenho foi analisado a importância dos atributos. Estes resultados são apresentados na Tabela 2.

Tabela 2: Importância dos atributos

	Valor da importância			Distribuição	
	Maior	Média	Menor	Acima	Abaixo
LLVM-bc	9794,60	774,86	0	12	59
LLVM-stats	5729,59	553,44	0	15	38
LLVM-inst	5311,11	373,37	0	11	52
MSF	34777,08	2844,82	0	9	46

Como pode ser observado, independente do grupo de atributos poucos tem uma importância superior do que a média. De fato, atributos importantes não ultrapassam 28% (LLVM-stats). Outro ponto a ser observado é o fato dos atributos MSF terem uma importância superior aos atributos dos outros grupos. Por fim, em todos os grupos existem atributos sem importância; portanto poderiam ser removidos do grupo.

Conclusões

Sistemas inteligentes de geração de código, como YaCoS, empregam estratégias de aprendizagem de máquina com o objetivo de gerarem código de boa qualidade.

Esse trabalho avaliou a acurácia do sistema YaCoS, na tarefa de encontrar melhores conjuntos de transformações do que o conjunto padrão Oz, parametrizando tal sistema para utilizar diferentes grupos de atributos para caracterizar programas.

Os resultados obtidos indicam que YaCoS possui o melhor desempenho quando utiliza, como atributos de programas, atributos extraídos do processo de compilação.

Agradecimentos

Agradeço ao CNPq pelo apoio financeiro a esta pesquisa.

Referências

Muchnick S S. **Advanced Compiler Design and Implementation**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

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.

Filho, J.F., Rodriguez, L.G.A. & da Silva, A.F. **Yet Another Intelligent Code-Generating System: A Flexible and Low-Cost Solution**. J. Comput. Sci. Technol. 33, 940–965 (2018).

Riesbeck, C. K., & Schank, R. C. (1989). **Inside case-based reasoning**. Lawrence Erlbaum Associates, Inc.