International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 10, Issue 1, No 2, 2022 ISSN: 2348-6600
Reference ID: IJCS-394 PAGE NO: 2714-2718

REGISTER ALLOCATION USING GRAPH NEURAL NETWORKS

Piyush Jadhav
jpiyush2599@gmail.com
Pune, India.

***Abstract*—The execution lifecycle of a program goes through various optimizations and code generation. Register allocation is critical part of optimization phase of compiler. Graph coloring is one of the NP hard problem and it applies to various applications such as register allocation. Deep Learning methods has various state-of-art techniques in domains like computer vision, natural language processing.**

In this paper we propose a solution for register allocation problem of compiler using Deep Learning networks based on graphs called as graph neural networks (GNN). We start with basic idea that two connected nodes with temporaries will have different registers allocated.

Keywords*—*Register allocation, graph coloring, graph neural networks

I. INTRODUCTION

The compilation of a program goes various phases of optimizations and code generation. Most of the problems in this phases are NP-hard problems. One of the important and critical part in code optimization is physical register allocation from intermediate code. Typical intermediate code uses too many temporary variables for simplification, but it makes complicated when this code has to be finally translated to assembly using physical register, as these physical register are limited for any given processor.

Execution life cycle of a program is divided into various phases of compilation. These phases are part in compiler front-end where it includes syntax and semantic analyzer. It also includes intermediate code representation (one of the most used example is LLVM IR). This code representation uses an unlimited amount of virtual registers/variables for optimization. Compiler back-end includes code generation and optimizations. Hence for this phase of generating code, this must be allocated to either a physical register or the space of memory.

With advancement of GPUs and CPUs, the register instructions are executed faster than through memory process. Optimized use of the registers is very crucial at the time of code generation. It is important to tell which variables are used in which registers and to find the registers to which they are used at each phase of basic block. The allocation in basic block must consider the conflicts in variables used in its execution. The issue in getting the allocation of temporary to the registers by considering these conflicts is called the register allocation issue [1]. Register allocation is considered as old problem during initial days of compiler when used in original FORTRAN compiler in the 1950s. Heuristic approaches were used to solve the problem, but a breakthrough came when register allocation was considered as graph coloring problem by Chaitin[2].

Graph coloring is assignment of color to each node in an undirected graph where no two adjacent (neighbor) color should have exact color. It can be considered as an optimization problem to find least number of colors needed to color all the nodes. Similar analogy can be used for register allocation problem where nodes in the graph are the temporaries used in the basic block and edges between two nodes indicate that these two temporaries live at the concurrent time. This is known as the register interference graph (RIG). Register interference graph (RIG) is architecture independent. Here the constraint is that two temporary variables i.e. nodes can be assigned to same register if and only if there is no edge between them.

Deep Learning has various state-of-art techniques for domains like computer vision, natural language processing. One of the models with relational structure into a neural model is graph neural networks. Graph neural networks (GNN) recently has shown great success due to its representation of graph structural data. One of the blueprint for GNN is message passing scheme where aggregation of neighbor feature node is done.

II. Related Work

One of the global approach to solve register allocation was graph coloring allocation. Chaitin[2] initial to publish the register allocating with graph coloring. Briggs et al. [4, 5, 6] publish some variation, such that quantity of spilling variables was decreased largely. On the other hand Bernstein [7] et al and Lueh [8], used different approach for spilling nodes.

Approach given by Chaitin[2] was in assembly instructions. The STORE assembly was after intermediate variable and LOAD assembly before its application. Bergner et al. [12] and Lueh [8] used heuristic approach for placement of assembly, thus to decrease spill and more use of parallelism. Callahan and Koblenz [11] gave solution to graph coloring problem by executing the program with segment coloring. Chow and Hennessy [10] gave solution to graph coloring problem by assigning the registers in function in the basic block where count of variable referred and execution count of basic block of the program.

Graph neural network had spread in several fields of research. Gori, Monfardini and Scarselli (2005)[15] proposed their GNN model on three versions: connection problems – neighboring group size problem, labeling problem – classifys the parity of a true/false vector which is assigned to each node and a main problem of subgraphs identification. [13] gave a GNN approach to the NP complete true/false satisfiability problem (SAT) achieved accuracy 85% on SAT instances with 40 variables. Also, model could decode assignment operations even it was trained to output true/false answer.

III. Methodology

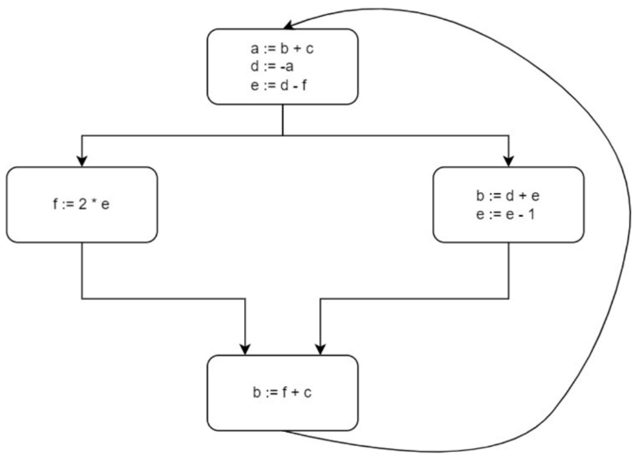
The main principle of register allocation using graphs is,

Temporary variables t_1 and t_2 can share the same register if and only if at any point in the basic block at most one of t_1 or t_2 is live.

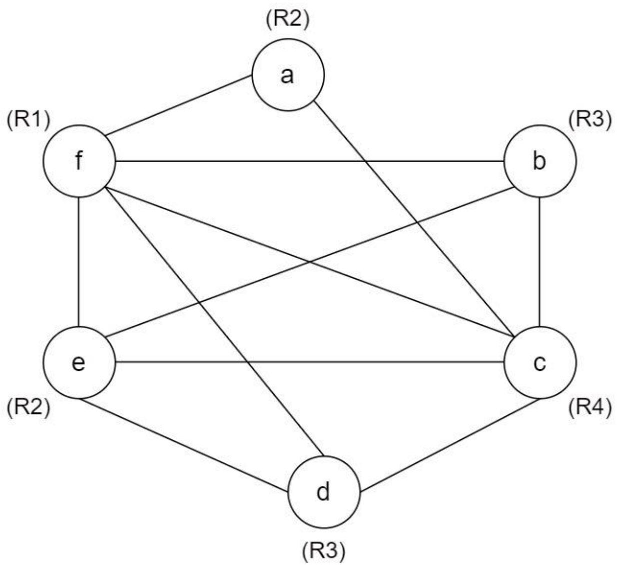
A. *Constructing Register Interference Graph*

We need to construct an undirected graph where each node can be considered as temporary variable and an edge is represented between t_1 and t_2 if they are live at the same time at some point of the basic block. Thus register interference graph (RIG) is created where two temporaries can be allocated to the same register if there is no edge connecting them. With analogy to graph coloring problem, if RIG is k-colorable then there is a register allocation that uses no more than k registers, where k is number of machine registers.

Consider following case,



Above flowchart represents operations that is performed in a basic block using variables a, b, c, d, e, f. Corresponding register interference graph considering its constraints would be,



This is a 4-coloring graph thus, at least 4 registers would be required for above operations in a basic block to be performed.

B. *Correlating RIG with GNN*

GNNs have been applied to solve a set of graph-related problems, and a few attempts have been made for the graph coloring problem. However, GNNs are designed generally as a node embedding scheme, which has a totally different objective from the register allocation context. GNNs often map connected nodes into similar node embeddings while the heuristics for register allocation is to assign the connected nodes to registers.

While iterations of message passing, adjacent information of node embeddings are refined. These filtered messages goes through aggregation function to update for the nodes. Recurrent Neural Network (RNN) is used to compute embedding update for the nodes.

IV. Model

A. *Implementation*

Given a graph $G = (V, E)$ and register $R \in N \mid R > 2$ each register is assigned with any initial value $\in \mathbb{R}^r$ over a uniform distribution, $R^{t=0}[i] \sim U(0, 1) \mid \forall\, i \in R$ and each node is set to the same embedding value $\in \mathbb{R}^v$ sampled from a normal distribution.

Following the procedure of Prates[2] and [14], this random initial embedding will be a trained parameter learned by the model. Node-to-node adjacency matrix $M_{VV} \in \{0,1\}^{|V|\times|V|}$ for communication among neighboring vertices and node-to-register adjacency matrix $M_{VR} \in \{1\}^{|V|\times|R|}$ for communication among vertices and registers, connecting every register to all nodes hence, no existing information is obtained for model and any node can be designated to any register available. Further, adjacent nodes and registers synchronize and change their embedding value throughout p_{max} message passing epoch. Hence resulting node embedding value are resolved by a Multilayer Perceptron (MLP) which outputs a probability with respect to the model's prediction for the answer to the decision version of problem: "does the graph G accept a R-registers?". Below is the pseudocode for the model,

```
GNN_REG_ALLOC (G = (V, E), R):
    // Node-to-node adjacency matrix
    $\mathbf{M}_{VV}[i]\,[j] = 1$ iff $(\exists e \in E \mid e=(v_i, v_j)) \mid \forall v_i \in V, v_j \in V$
    // Node-to-register adjacency matrix
    $\mathbf{M}_{VR}[i]\,[j] = 1\ \forall v_i \in V, r_j \in R$
    //Initialize V[$i$] and R[$i$] according to available numbers
    For p=1 to $p_{max}$: //loop for message passing iteration
        //Update node embeddings with hidden states
        $V^{p+1}, V_h^{p+1} = v_i(V_h^p, \mathbf{M}_{VV} \times V^p, \mathbf{M}_{VR} \times R_{msg} \times R^p)$
        $R^{p+1}, R_h^{p+1} = r_i(R_h^p, \mathbf{M}_{VR}{}^T \times V_{msg} \times V^p)$
    // node embeddings into logit probabilities,
        $\sigma = \overline{V_{vote} \times V^{p_{max}}}$
    //Sigmoid function
```

According to GNN algorithm, node and register embedding is updated with its hidden states. Message functions implemented with MLP are passed to the model R_{msg}: $\mathbb{R}$ translates register embeddings to message to update node function and vice versa for V_{msg}.

B. Training Methodology

We trained these GNN model (MLP and RNN) using Stochastic Gradient Descent algorithm implemented with help of TensorFlow Adam optimizer. For training, the input is a set of 10,000 samples of adjacency matrix of RIGs provided in a .csv file. Hence for full adjacency matrix usability, we need to add the adjacency vector for each node one by one. The output consists of booleans with highest value of 100 steps. Output was 1 register number per node. The optimized output received in the training sample is confirmed against the registers allotted by the model during each epoch.

The Multilayer Perceptron (MLP) as a consequence for computing message are three layered (64,64,64) with ReLU nonlinear activation function for every layers except for the linear activation for output layer. The RNNs are the only LSTM cells with normalisation layer and ReLU activation function. Here p_{max} value was 32 times message passing steps.

V. Result And Conclusion

Dataset was divided to batch normalization. We achieved 80% of accuracy over 16 batches of 1,000 epochs. In this approach solution we tried to show how GNN models can be used for register allocation problem in compiler domain. This approach can also be used for new path of resource allocation for GNN which is still in developing phase.

Spilling of registers is also a phenomenon that needs to be considered in register allocation. This can be the point for future work of the model.

REFERENCES

[1] A. V. Aho and J. D. Ullman. Principles of Compiler Design. Addison-Wesley, 19

[2] G. J. Chaitin, 1982. Register allocation and Spilling via
Graph Coloring. Proceedings of the 1982 SIGPLAN Symposium on Compiler construction - SIGPLAN '82.

[3] M. Prates, P. Avelar, H. Lemos, L. Lamb, and M. Vardi, "Learning to solve NP-complete problems - a graph neural network for decision TSP" arXiv preprint arXiv:1809.02721, 2018.

[4] Preston Briggs, Keith D. Cooper, and Linda Torczon. Rematerialization. ACM SIGPLAN Notices, pages 311–321, 1

[5] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to Graph Coloring Register Allocation. ACM Transactions on Programming Languages and Systems, pages 428–455, 19

[6] Preston Briggs, Keith D. Cooper, Ken Kennedy, and Linda Torczon. Coloring Heuristics for Register Allocation. ACM SIGPLAN Notices, pages 275–284

[7] D. Bernstein, D. Q. Goldin, M. C. Golumbici, H. Krawczykand Y. Mansour, I. Nahshon, and R. Y. Pinter. Spill Code Minimization Techniques for Optimizing Compilers. ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 258–263,

[8] Guei-Yuan Lueh. Issues in Register Allocation by Graph Coloring. Technical Report CMU-CS-96-171, School of Computer Science, 19

[9] S. Thevenin, N. Zufferey, and J. Potvin, "Graph multi-coloring for a job scheduling application," Discrete App. Math., vol. 234, pp. 218 – 235, 2018.

[10] F. C. Chow and J. L. Hennessy. The Priority-based Coloring Approach to Register Allocation. ACM Transactions on Programming Languages and Systems, pages 501–536, 19

[11] David Callahan and Brian Koblenz. Register Allocation Via Hierarchical Graph Coloring. Conference on Programming Language Design and Implementation, pages 192–203

[12] Peter Bergner, Peter Dahl, David Engebretsen, and Matthew T. O'Keefe. Spill Code Minimization Via Interference Region Spilling. SIGPLAN Conference on Programming Language Design and Implementation, pages 287–295

[13] D. Selsam, M. Lamm, B. Bunz, P. Liang, L. de Moura, and D. Dill, "Learning a sat solver from single-bit supervision," arXiv preprint arXiv:1802.03685, 2018.

[14] H. Lemos, M. Prates, P. Avelar and L. Lamb, "Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems," 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, pp. 879-885, doi: 10.1109/ICTAI.2019.00125.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2009.