REGAL: Transfer Learning For Fast Optimization of Computation Graphs

Aditya Paliwal, Felix Gimeno, Vinod Nair, Yujia Li, Miles Lubin, Pushmeet Kohli, Oriol Vinyals
Optimising Computation Graphs

- Device placement
- Scheduling
- NP-hard
Motivation & Related Work

• AutoTVM
  • No transfer across models
• Learning to super optimise programs
  • Handcrafted instance & small graphs
• Parallel task scheduling
  • Traditionally not learning-based
• Little attempt to learn to transfer to new graphs on a large scale
REGAL: Transfer Learning For Fast Optimization of Computation Graphs
Pipeline

Input → Graph Neural Net Policy → BRKGA → Output → Evaluation

- Input
- Graph Neural Net Policy
- BRKGA: Mutants, Elites, crossover, copy
- Output: Placement, Schedule
- Evaluation: Performance Model → Peak Memory / Running Time
Objective
Peak memory minimisation

Input
Graph Neural Net Policy
BRKGA
Output
Evaluation

Performance Model
Peak Memory / Running Time
BRKGA
Biased random key genetic algorithm
BRKGA
Evolution

Population K

Elite solutions

Non-elite solutions

Population K+1
BRKGA

Evolution

Population K

Elite solutions

Copy Elites

Population K+1

Elite solutions

Non-elite solutions
BRKGA
Evolution

Population K
Elite solutions
Non-elite solutions

Copy Elites

Population K+1
Elite solutions
Mutant solutions

Generate Mutants
BRKGA Evolution

Population K

Elite solutions

Non-elite solutions

Copy Elites

\[ \rho_i \]

Crossover

\[ 1 - \rho_i \]

Generate Mutants

Population K+1

Elite solutions

Mutant solutions
BRKGA
Encoding & Decoding

- Chromosome
  - n-d vector $[0,1]^n$
  - Ops-device affinity
  - Scheduling priorities
  - Tensor transfer priorities
- Fitness function
  - $f : [0,1]^n \rightarrow \mathbb{R}$
GNN policy
GNN policy

- Aim to generate
  - Parameters of chromosome generation distribution $\mathcal{D}$
  - Elite biases $(\rho_i)$
  - As a vector $y_{\nu}$ for each node $\nu$
GNN policy

• Aim to generate
  • Parameters of chromosome generation distribution $\mathcal{D}$
  • Elite biases $(\rho_i)$
  • As a vector $y_v$ for each node $v$
• GNN
  • Representation vectors $h_v$ for each node $v$
  • Structural information of the graph
GNN policy

How do we go from $h_v$ to $y_v$?

- Conditionally independent predictions
- Autoregressive predictions
- Actions & Rewards (Aka RL)
GNN policy

REINFORCE

• Sample action vector $y$ from $p(y | G)$

• Reward $r = - \frac{o_a(G)}{o_s(G)}$

• Maximise

$$L = \mathbb{E}_G \left[ \sum_y p(y | G) r(y, G) \right]$$
Results
Vs Baselines

- Constraint programming
- Graph partition
- Local search (greedy)
- Graph-As-Sequence

Table 1: Performance for all methods, averaged over the graphs in the test set of the TensorFlow and XLA datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TensorFlow dataset (test)</th>
<th>XLA dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% Improv. over BRKGA5K</td>
<td>% Gap from best</td>
</tr>
<tr>
<td>CP SAT</td>
<td>-1.77%  13.89%</td>
<td></td>
</tr>
<tr>
<td>GP + DFS</td>
<td>-6.51%  16.63%</td>
<td></td>
</tr>
<tr>
<td>Local Search</td>
<td>0.63%   8.65%</td>
<td></td>
</tr>
<tr>
<td>BRKGA 5K</td>
<td>0%      9.65%</td>
<td></td>
</tr>
<tr>
<td>Tuned BRKGA</td>
<td>0.8%    8.54%</td>
<td></td>
</tr>
<tr>
<td>GAS</td>
<td>0.16%   9.33%</td>
<td></td>
</tr>
<tr>
<td>REGAL</td>
<td>3.56%   4.44%</td>
<td></td>
</tr>
</tbody>
</table>
Discussion
Ablation analysis

Table 3: Performance of REGAL with various subsets of actions.

<table>
<thead>
<tr>
<th>Placement</th>
<th>Scheduling</th>
<th>Elite Bias</th>
<th>Valid</th>
<th>Test</th>
<th>XLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>-0.4%</td>
<td>-0.2%</td>
<td>-0.4%</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>4.4%</td>
<td>3.65%</td>
<td>1%</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>4.67%</td>
<td>3.56%</td>
<td>3.74%</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>-1.53%</td>
<td>-1.1%</td>
<td>-2.2%</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>2.47%</td>
<td>1.4%</td>
<td>-0.4%</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>2.58%</td>
<td>1.88%</td>
<td>-0.7%</td>
</tr>
</tbody>
</table>
Comments

• Extensive evaluation and impressive results
• Transfer learning through policy network
• Objectives other than peak memory minimisation
• Too many optimisation layers, very complex system
• Justification of BRKGA
Conclusions

- Optimisation *all the way down*
- Input -> GNN -> REINFORCE -> BRKGA -> Decision
- Transfers well
References


• Mauricio G. C. Resende: Biased random-key genetic algorithms: A tutorial, 2012

• Zak Singh, R244 Paper Presentation on REGAL, 2021