ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Applied Logic Programming

TENSOR-PROLOG: A LOGIC PROGRAMMING FRAMEWORK FOR TRAINING NEURAL NETWORKS

CANDIDATE
John Smith

SUPERVISOR
Prof. Mario Rossi

Academic year 2020-2021

Session 1st

dedicated(X) :- friend(X).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Rationale

The adoption of Artificial Intelligence (AI) is growing rapidly, leading to the development of increasingly complex models that demand significant computational resources. This growth results in high carbon emissions, raising concerns about sustainability in AI research and deployment. Consequently, there is a need to monitor and reduce the environmental impact of AI computations.

In this thesis, we investigate the problem of optimizing hardware architecture selection and configuration for AI algorithms while adhering to carbon emission constraints. An existing tool, HADA (HArdware Dimensioning of AI Algorithms), has previously tackled hardware dimensioning considering budget, runtime, and solution quality constraints. Our work extends HADA by integrating carbon emission constraints, introducing the concept of Sustainable Hardware Dimensioning. [1]

Chapter 2

Related Works

2.1 Carbon Footprint in AI

Several studies have addressed the issue of AI's carbon footprint. Tools like **Green Algorithms** and **CodeCarbon** have been developed to estimate and monitor emissions.

CodeCarbon is an open-source tool designed to track the energy consumption of computational resources and estimate the corresponding carbon emissions. The formula used is:

$$CO2eq = C \times E \tag{2.1}$$

where:

- **C** represents the carbon intensity of electricity (kg CO2e per kWh), varying by country and energy mix.
- **E** is the total electricity consumed during computation (kWh).

By monitoring CPU, GPU, and RAM consumption, CodeCarbon estimates the total emissions associated with a computation. It retrieves the carbon intensity based on the geographical location and logs results at user-defined intervals (default: 15 seconds).

Installation:

```
pip install codecarbon
```

Chapter 3

Metodology

3.1 Empirical Model Learning in HADA

HADA employs the **Empirical Model Learning (EML)** paradigm, which integrates **Machine Learning (ML)** models into an optimization framework. EML involves:

1. **Data Collection**: Running target algorithms under various hyperparameter configurations and hardware settings to collect performance data.

2. **Surrogate Model Creation**: Training ML models (e.g., Decision Trees) to approximate the relationship between input configurations and performance metrics (e.g., runtime, memory, carbon emissions).

3. **Optimization**: Using the learned models within a combinatorial optimization framework to find the best hardware configuration.

HADA was originally applied to the **ANTICIPATE** and **CONTINGENCY** stochastic algorithms used in energy management. These algorithms compute energy production schedules while minimizing cost and considering uncertainties.

3.2 Integration of CodeCarbon in HADA

To extend HADA for sustainable AI, we integrate CodeCarbon to track emissions in:

- ANTICIPATE and CONTINGENCY algorithms.
- MaxFlow Algorithms:
  - Boykov-Kolmogorov (BK)
  - Excess Incremental Breadth First Search (EIBFS)
  - Hochbaum's Pseudo Flow (HPF)

Chapter 4

Experimental Analysis

4.1 Benchmarking on Different Hardware Platforms

Experiments were conducted on:

- MacBook Pro (2019)
- Leonardo Supercomputer (CINECA HPC)

Each algorithm was executed on 30 instances with hyperparameter values ranging from 1 to 100, generating datasets with 6,000 records per algorithm per hardware platform.

Chapter 5

HADA-as-a-Service

5.1 HADA Web Application

Benchmark data was integrated into the HADA web application, requiring:

- Creation of JSON configuration files for each algorithm-hardware combination.
- Specification of hyperparameters and performance targets.

Example JSON structure:

```
{
"name": "anticipate",
"HW_ID": "macbook",
"hyperparams": [
{"ID": "num_scenarios", "type": "int", "LB": 1, "UB": 100}
],
"targets": [
{"ID": "time", "LB": null, "UB": null},
{"ID": "memory", "LB": null, "UB": null},
{"ID": "emissions", "LB": null, "UB": null}
]
}
```

Chapter 6

Conclusions

This work extends HADA by integrating carbon emission constraints, enhancing its applicability for sustainable AI hardware selection. Through experimental benchmarks on laptops and HPC systems, we validated the framework's ability to balance performance and environmental impact. The web-based prototype enables users to make informed decisions when configuring AI workloads under sustainability constraints.

Bibliography

[1] J. Wielemaker et al. “Swi-prolog”. In: *Theory and Practice of Logic Programming* 12.1-2 (2012), pp. 67–96.

Acknowledgements

I'm very grateful to the inventor of the Prolog language, without whom this thesis couldn't exist. I'd also like to acknowledge my advisor Prof. Mario Rossi by tail-recursively acknowledging my advisor.