DeepSched: A Deep Representation Of Scheduling Policies For Heterogeneous Distributed Systems

Team #10

Mohamed Shawky Zaky, Sec: 2, BN: 16

Remonda Talaat, Sec: 1, BN: 20

Mahmoud Osman Adas, Sec: 2, BN: 21

Evram Youssef, Sec: 1, BN: 9

Demonstration Audio

DeepSched: A Deep Representation Of Scheduling Policies For Heterogeneous Distributed Systems

Mohamed Shawky	Remonda Talaat	Mahmoud Adas	Evram Youssef
SEC 2, B.N 16	SEC 1, B.N 20	SEC 2, B.N 21	SEC 1, B.N 9

{mohamedshawky911, remondatalaat21, mido3ds, evramyousef}@gmail.com

Introduction

- **Task scheduling** is a very critical factor to the performance of all systems.
- **Heterogeneity** of the systems adds extra layers of complexity to the scheduling problem.
- Heterogeneous task scheduling is divided into **Heuristic** and **Approximate** methods.

Introduction

- We study the approximation of heuristic methods using predictive models, specifically **neural networks**.
- We use **HEFT** as our heuristic baseline, **genetic algorithms** as our approximate baseline.

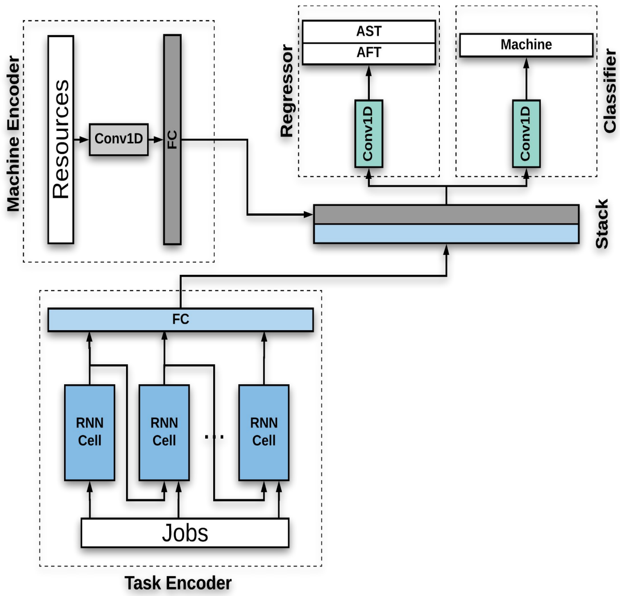
Proposed network

The network consists of 4 main blocks:

1. Task Encoder.
2. Machine Encoder.
3. Classification Module.
4. Regression Module.

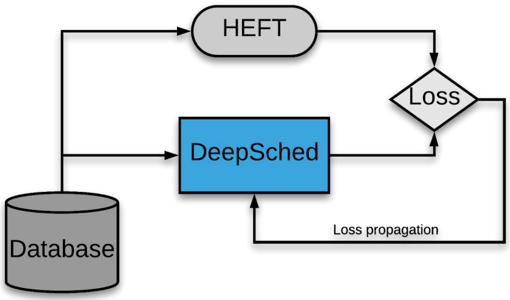
The network dimensions are adjusted on the **maximum** number of tasks and resources.

Full network code is available at:
https://github.com/DarkGeekMS/deepsched



Training Framework

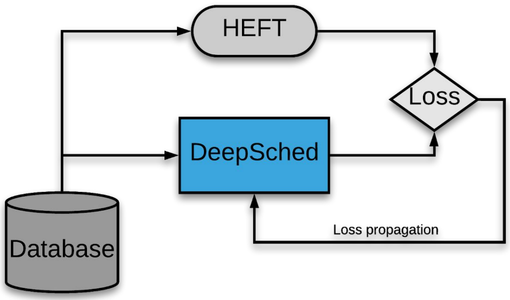
- A database of tasks and resources is provided to both **DeepSched** network and **HEFT**.
- The network is trained on the loss between its predicted schedules and the schedules produced by HEFT.



Training Framework

The **network loss** is mainly composed of two parts:

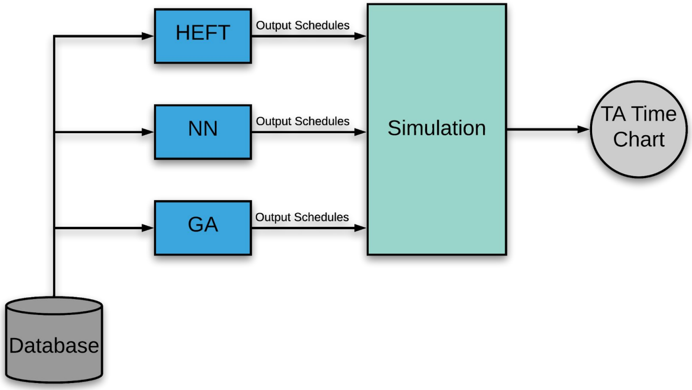
- **Classification loss** of the running machine for each task.
- **Regression loss** for actual start and finish time of each task (to add some constraints on the loss function).



The **summation** of these two losses is used to train the network.

Evaluation Framework

- A database of both **seen** and **unseen** data is provided to the **three** discussed methods.
- The output schedules of the three methods are passed to a **simulation stub** that runs the schedules and calculates the **average turnaround time**.
- This is repeated on **multiple input sizes**.

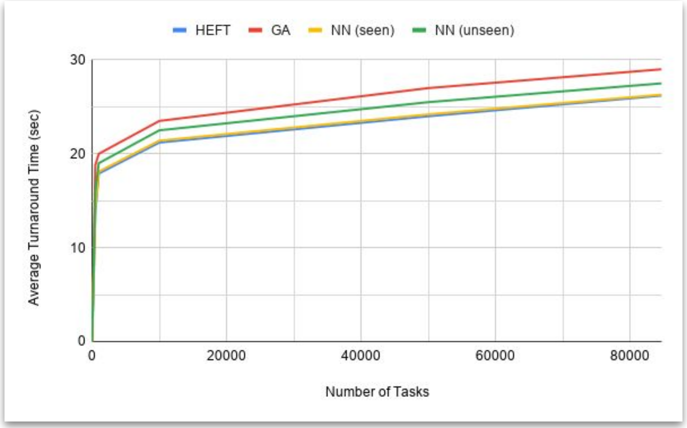


Results Discussion

Our results are divided into two parts:

1) **Performance** analysis:

- The opposite graph shows the **average turnaround time** for different input sizes.
- We can see that the network can offer the **same performance** as HEFT on **seen** data and **slightly lower performance** on **unseen** data. However, it still outperforms GA.



Results Discussion

Our results are divided into two parts:

2) **Time** analysis:

- We can see in the opposite table that the execution time of the **network** is **constant** over different input sizes, while the execution time of **HEFT** and **GA grows rapidly**.

Input Size	HEFT	GA	NN
486	31.5	67.5	0.125
10000	305.5	602.5	0.125
84654	670.0	3967.0	0.125

Future work and limitations

- This work can be improved in **two** possible ways:

1) **Approximation** of other scheduling algorithms, where we can go with some **optimal algorithms**.
2) Dealing with **dynamic** scenarios through **online network optimization** or **reinforcement learning environments**, because **static scenarios** and **maximum predefined input size** are the main **drawbacks** of the proposed method.