Constrained Deep Learning for Scheduling Problems

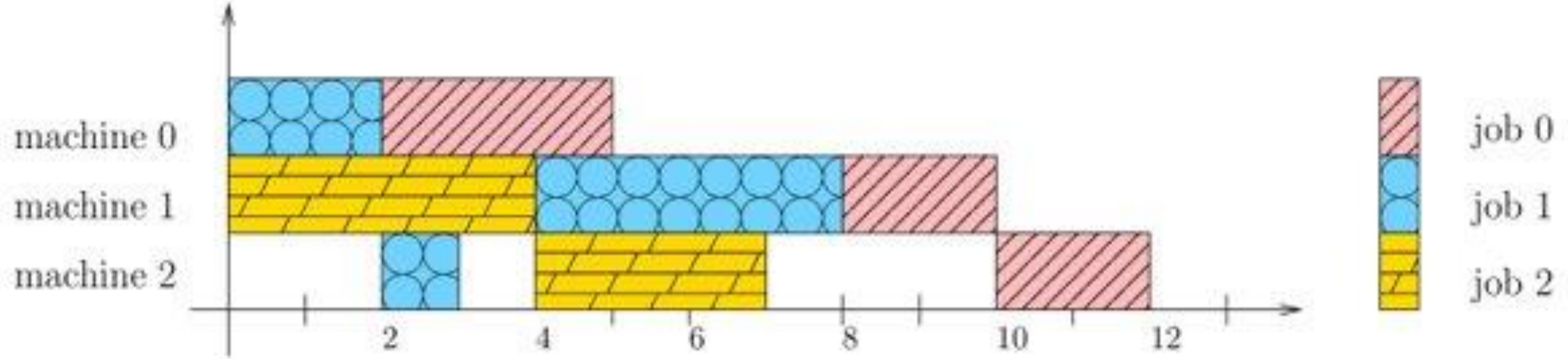
Security and Privacy of Machine Learning

Final Presentation

James Kotary

Job Shop Scheduling Problem

- M machines in a system each process tasks
- J jobs are each composed of T tasks
- Each task is assigned a machine and a processing time (duration)
- Tasks within the same job must be processed in a specified order (Task-Precedence Condition)
- No two tasks may occupy the same machine at the same time (No-Overlap Condition)
- Goal: Schedule the tasks in such a way that the endtime of the final task is minimized



Job shop scheduling

Linear Integer Program

$$\min \max_{1 \le j \le n} s[j, t_j]$$

$$\text{s.t.} \quad s[j,t] + dur[j,t] \le s[j, t+1] \quad \forall_j$$

$$s[j_1, t_1] + dur[j_1, t_1] \le s[j_2, t_2]$$

or

$$s[j_2, t_2] + dur[j_2, t_2] \le s[j_1, t_1] \qquad \forall_{\substack{j_1, j_2 \\ t_1, t_2}}$$

whenever

$$mach[j_1, t_1] = mach[j_2, t_2]$$

Learning Goal:

Train a deep learning model to predict, given a job shop problem instance, a schedule that is (close to) optimal and feasible

Training Data:

Job shop problem instances: Durations and machine assignments for each task. The order of the input data imply the specified order of tasks within jobs

Training Labels:

Optimal solution schedules for the specified training problems. Obtained by solving with constraint programming

Limitation:

A learning system will be specific to one problem shape: M machine, J jobs, T tasks per job

Software tools

- Python 3.7.3 programming language
- Google OR-Tools constraint programming SAT solver, for solving training problem instances
- Pytorch machine learning library

Dataset Development

- Machine assignments chosen uniformly at random between 1 and M

- Durations chosen uniformly at random, centered around a chosen mean

- Mean durations are chosen uniformly between a chosen min and max

- Solutions are generated using the Google OR-Tools SAT solver

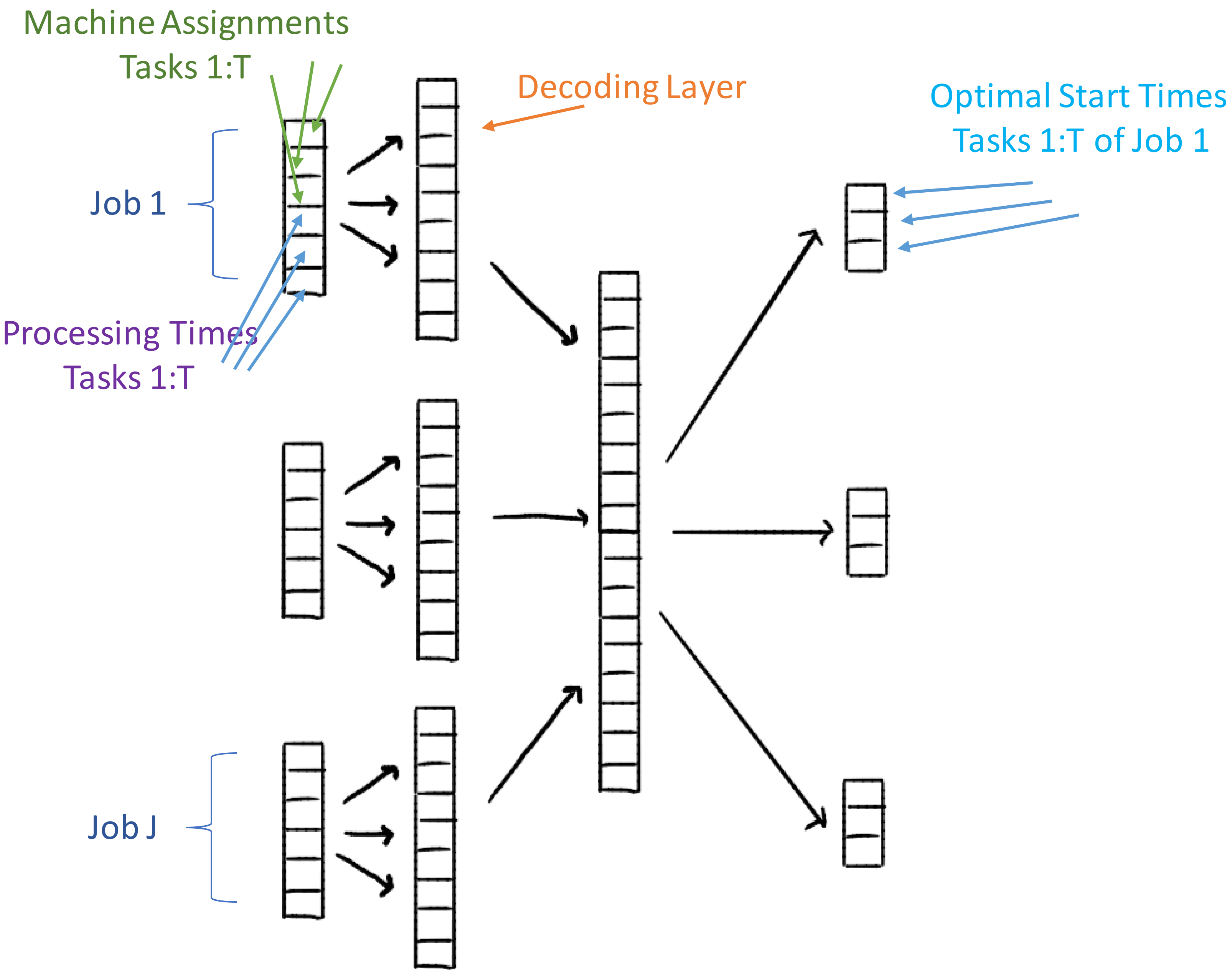
- Problems generated this way tend to be "easy" to solve, while having understandable structure and statistical properties

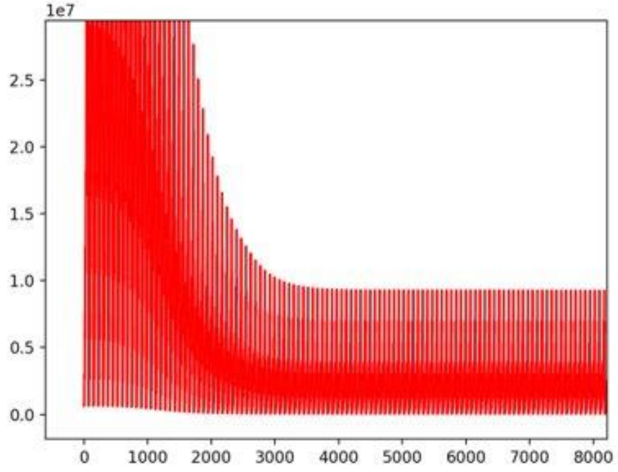
- All training problems have the same dimensions: M=J=T=5
- The problems are of manageable size for solving and visualization

- Dataset size: 1500 problems

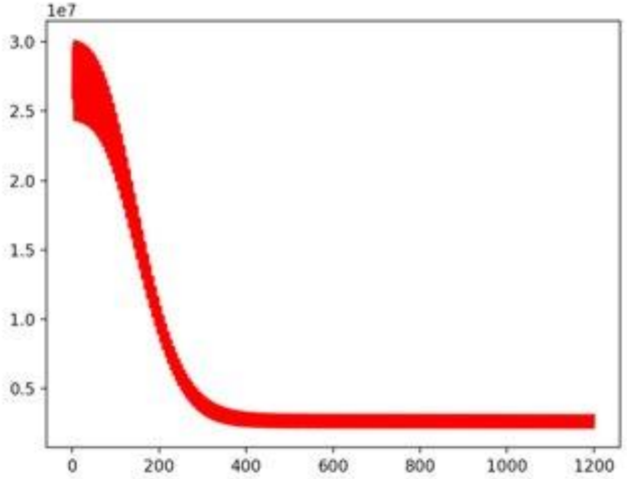
- More carefully designed problems will be added when we are ready to study the effect of problem difficulty on the learning system

Neural Network Design





Training loss curves for model with single vs multiple input and decoding layers



Loss Function - Lagrangian Relaxation

$$\mathcal{O} = \underset{y}{\operatorname{argmin}}\, f(y) \text{ subject to } g(y) \leqslant 0.$$

$$f_\lambda(y) = f(y) + \lambda g(y)$$

$$LR_\lambda = \underset{y}{\operatorname{argmin}}\, f_\lambda(y)$$

$$LD = \underset{\lambda \geqslant 0}{\operatorname{argmax}}\, f(LR_\lambda)$$

$$\mathcal{L}_\lambda(\tilde{y}_l, y_l, d_l) = \mathcal{L}(\tilde{y}_l, y_l) + \lambda\, \nu(g(\tilde{y}_l, d_l) \leqslant 0).$$

$$w^*(\lambda) = \underset{w}{\operatorname{argmin}} \sum_{l=1}^{n} \mathcal{L}_\lambda(\mathcal{M}[w](d_l), y_l, d_l)$$

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}}\, \min_{w} \sum_{l=1}^{n} \mathcal{L}_\lambda(\mathcal{M}[w](d_l), y_l, d_l)$$

The loss functions *g* account for each constraint in the problem, they are of two types:

- Task-Precedence violations take the form of a ReLU acting on a static linear layer
- No-Overlap violations take the form of a minimum over two such ReLU's

This Lagrangian dual-based method is due to [1]

Two Types of Constraint Violation Loss Function

Computing Task Precedence Violation Degrees

No violation:

$$s_i + d_i \leq s_{i+1} \quad \forall i$$

Is violation:

$$s_i + d_i > s_{i+1}$$

$$\text{violation degree} = \max(s_i + d_i - s_j, 0)$$

$$= \mathrm{ReLU}(s_i + d_i - s_j)$$

Computing No-Overlap Violation Degrees

No violation when:

$$s_2 + d_2 \leq s_1$$

or

$$s_1 + d_1 \leq s_2$$

Is violation when:

$$s_2 + d_2 > s_1$$

and

$$s_1 + d_1 > s_2$$

$$\text{magnitude (if nonzero)} = \min(s_2 + d_2 - s_1, \; s_1 + d_1 - s_2)$$

Minimum adjustment to restore no-overlap

In general:

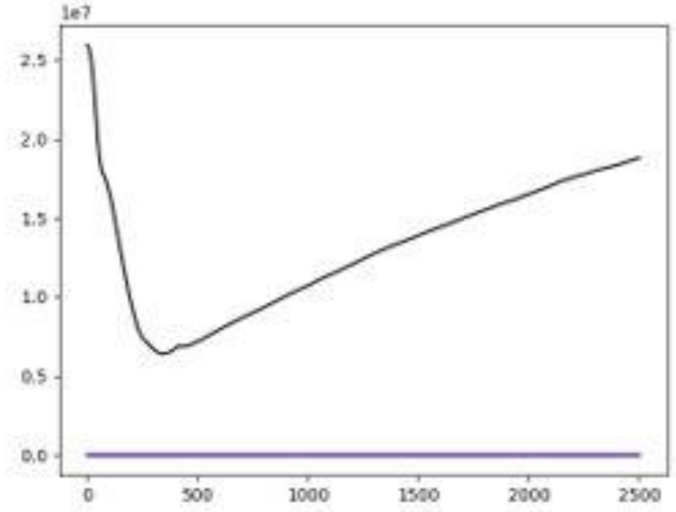
$$= \min\left(\max(s_2 + d_2 - s_1, 0), \max(s_1 + d_1 - s_2, 0)\right)$$

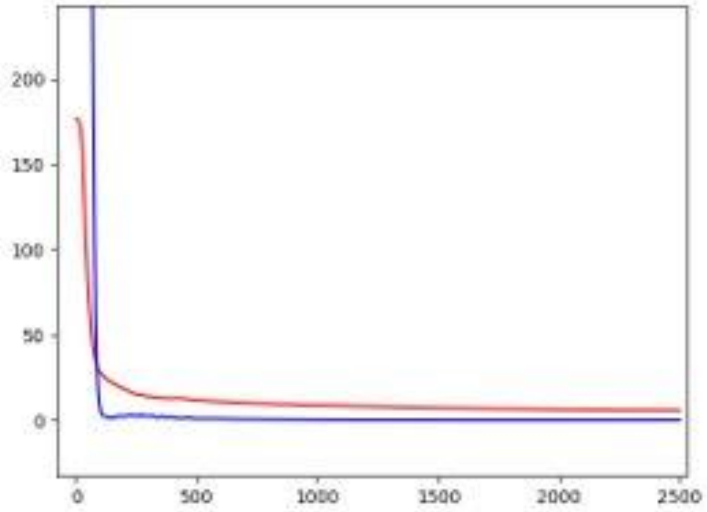
Training with the Lagrangian Relaxation

Observations

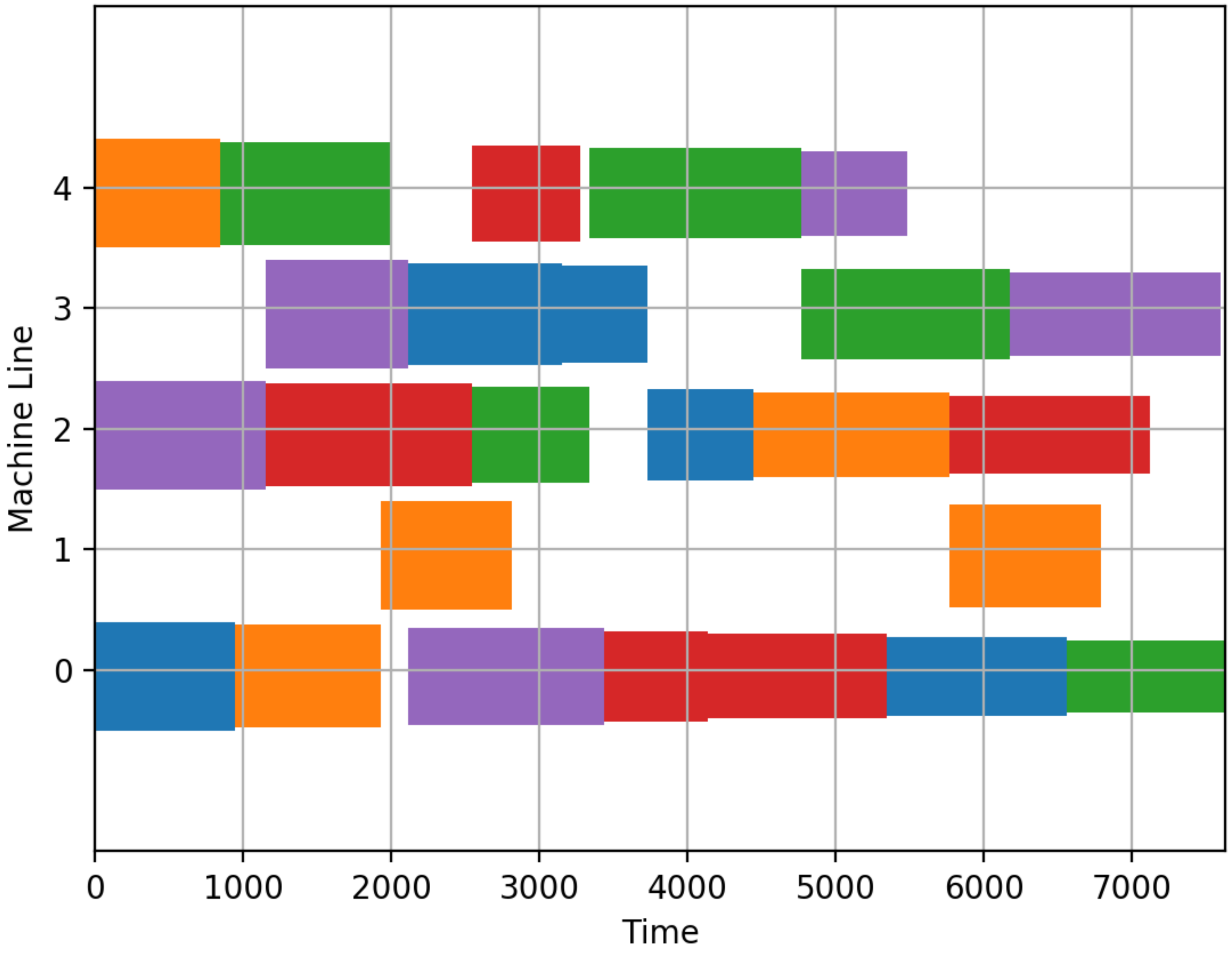
- The No-Overlap violations improve more slowly during training, converge to higher values on average and tend to consist of a few highly-overlapped tasks
- The Task-Precedence violations almost always converge close to zero
- Depending on the relative values of the learning rate and subgradient step-size, the resulting trained model may favor schedule predictions with longer makespans (less optimal) or more task overlap (less feasible)
- The primary objective (MSE Loss over task start times) can show possible signs of overtraining before the constraint violations converge

Concurrent test-set error curves for MSE loss (top) and No-Overlap vs Task-Precedence constraint loss (bottom) for a parameter tuning that yields high feasibility





Comparing Predicted Solutions



Optimal schedule for a particular 5x5x5 problem instance

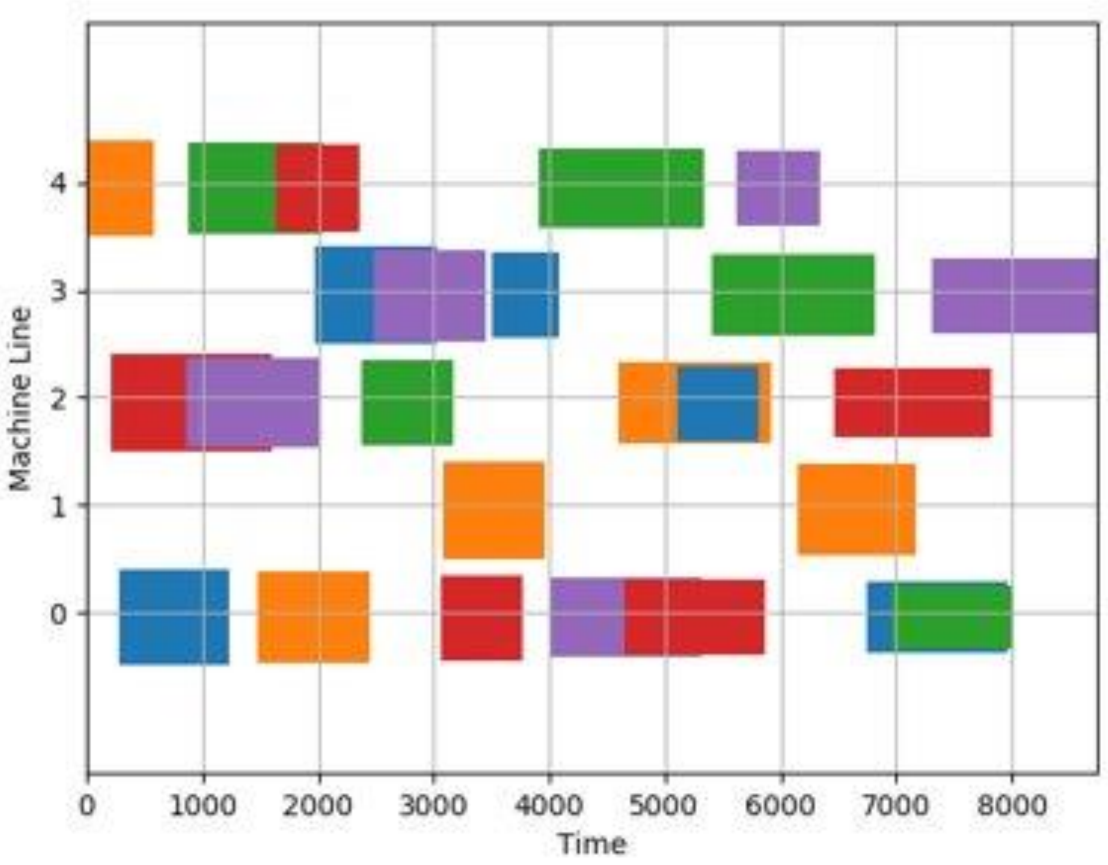
Comparing Predicted Solutions

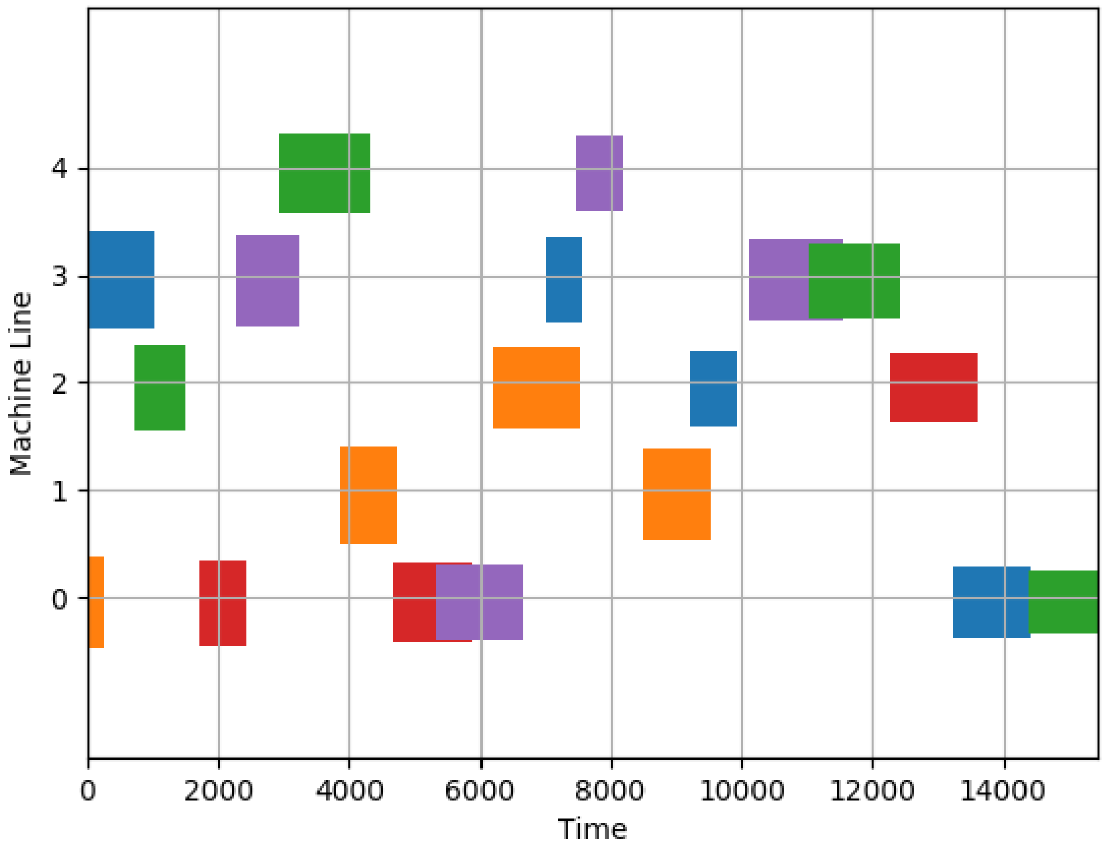
Model A

- Favors Optimality
- Short makespan / close to optimal
- High overlap degree / infeasible

Model B

- Favors Feasibility
- Long makespan / suboptimal
- Low overlap degree / almost feasible



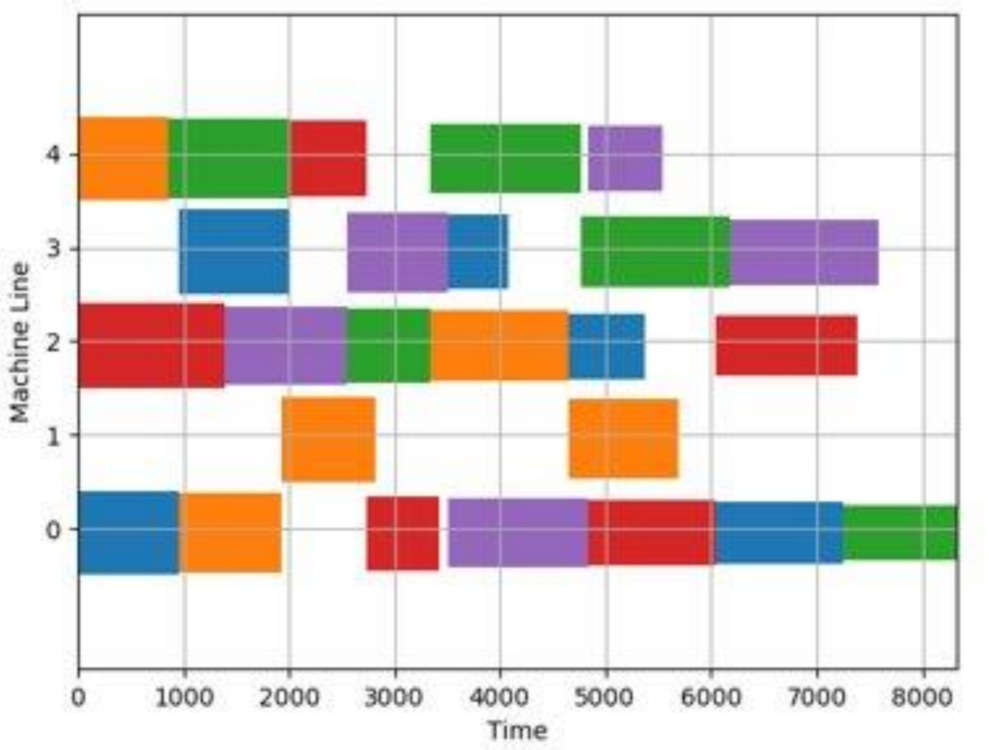


Schedule Rectifying Linear Program

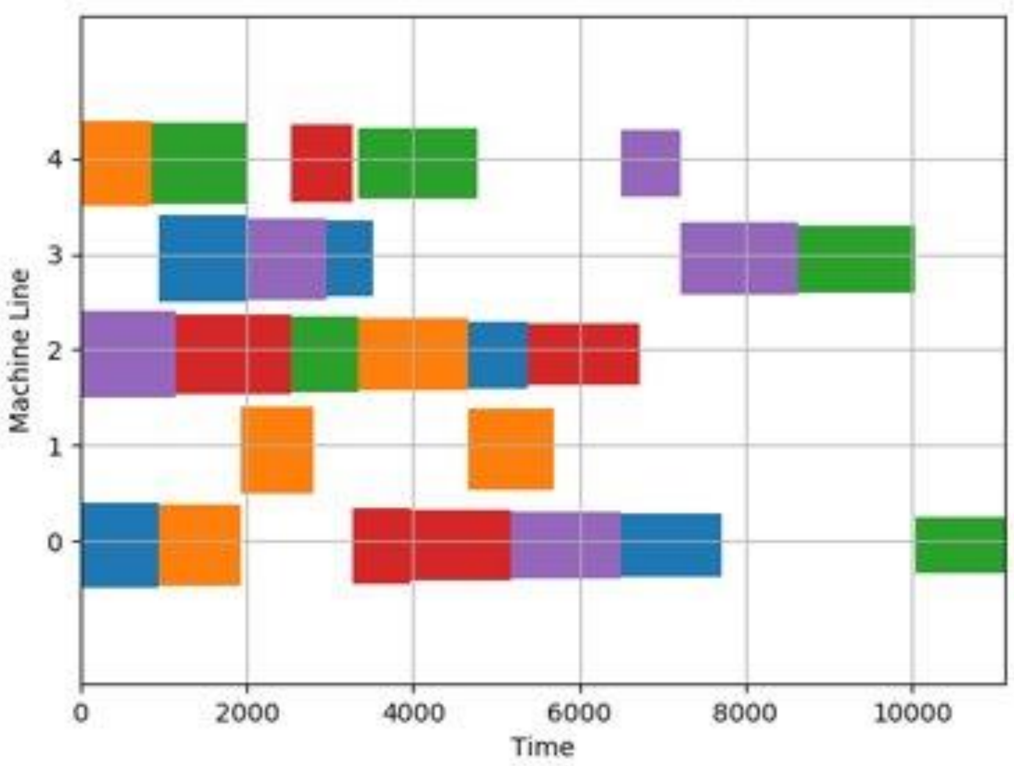
- Input a (suboptimal or infeasible) predicted schedule from the model
- Extract only the task-rankings from the prediction and assemble the optimal solution subject to the rankings over each job and machine
- Solution is guaranteed to be feasible to the original problem, and also optimal subject to the prediction's rankings
- Constraints are pairwise inequalities denoting precedence of task start times due to the defined rankings within each job, and the inferred rankings within each machine line
- Objective is the same as in the typical job shop problem
- Fast solving (linear in the number of matrix inversions)

Rectified Schedules – Typical Result

Model A – High quality



Model B – Low quality



Potential fast and accurate schedule predictor: generate preliminary solutions with a Lagrangian dual predictor that favors optimality, and rectify feasibility using LP

Future Developments

- We intend to extend the concept of schedule assembly from rankings using linear programming in an integrated DNN model.
- Task rankings may be predicted directly from problem instances, as in [2]
- A quadratic programming layer for deep learning is described in [3]. Adaptation to linear programming may potentially be explored either through reduction to the degenerate linear case, or through quadratic regularization as in [2].

References

[1] A Lagrangian Dual Framework for Deep Neural Networks with Constraints *Ferdinando Fioretto, Terrence W.K. Mak, Federico Baldo, Michele Lombardi, Pascal Van Hentenryck (ArXiv 2020)*

[2] Fast Differentiable Sorting and Ranking *Mathieu Blondel, Olivier Teboul, Quentin Berthet, Josip Djolonga (ArXiv 2020)*

[3] OptNet: Differentiable Optimization as a Layer in Neural Networks *Brandon Amos, J. Zico Kolter (arXiv 2019)*