Reducing the runtime of an NP-Hard algorithm using deep learning on historical data

Christoffer Lindkvist

September 10, 2025

Abstract

SILKSONG SOON SHAW

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

This thesis is an extension to the Volvo Truck Assembly Line problem [**?**]; Today trucks are placed manually by management workers based solely on their own tacit knowledge, thus none of those experience this is not written down anywhere. The algorithm in the works will using data from Volvo help place the trucks so that there are as few overlaps as possible. My idea is that the algorithm can gain a faster runtime by defaulting to "safe" combinations which are already used today.

1.2 Research Problem

The truck assemly line problem at Volvo is considered NP-Hard, and the method today to solve this problem is based entirely on unwritten knowledge. If this tacit knowledge could be emulated based on historical data we could get a better starting point and thus reduce the runtime needed for the heuristic algorithm.

1.3 Objectives

The objectives of this thesis are:

1. To design a deep learning model capable of emulating the tacit knowledge of Volvo's management workers using historical data.
2. To integrate the model as a preprocessing step in the existing heuristic algorithm in order to reduce its runtime.
3. To provide a visualization tool that intuitively illustrates the scheduling flow, highlighting overlaps and bottlenecks across stations and time.

1.4 Visualization of the Problem

1.4.1 UI/UX Problem

The UI will visualize the flow of the assembly line on two axes. One per station, and one in clockcycles. One clockcycle is the time it takes the theoretical items to make it from one station to the next. Hence the items must be displayed in a way that conveys that some stations take longer and shorter time to complete. One clockcycle is defined as a rudimentary unit of time, the size is arbitrary, and it does not reflect real life.

Issues start to appearing when we start to consider that different stations S_n and S_m may take different times to complete. If we then step a clockcycle, then we can never keep our items in sync. The main issue is that if we compare the station S_n and S_m we'll see that each station have a different time to finish, then the clockcycle system will not be perfect or even realistic as stations with differing times will each finish in different times and thus an item n might make it to the station S_{n+2} from S_n in the same time it takes item m to make it to the same station as S_{m+1}.

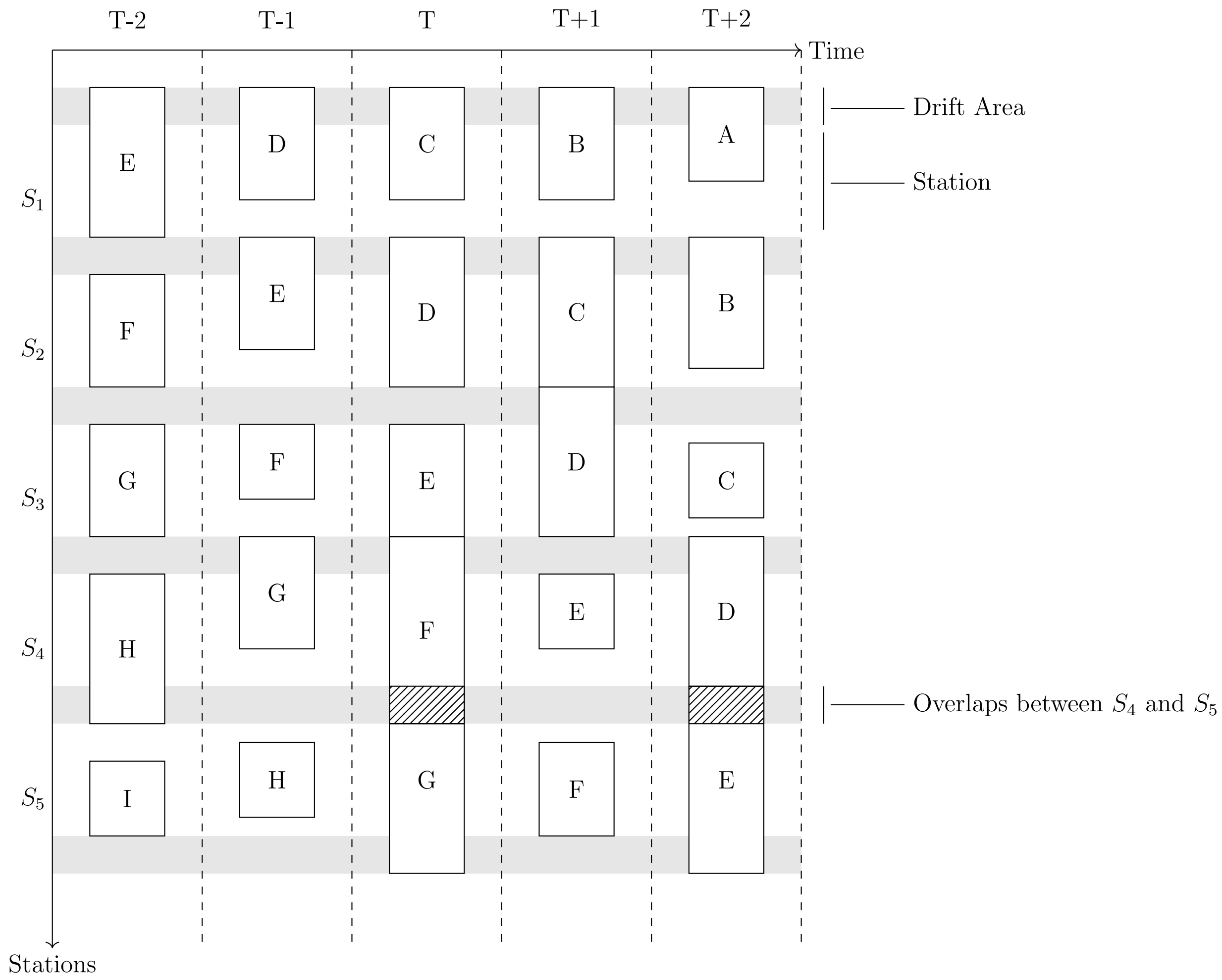

Figure 1.1: Assembly Line Example with Uniform Station and clockcycles

Thus we find the difficulty in displaying it properly in a UI/UX. If we wish to display each station as uniform sizes, then we also have to stretch the items to make up for it visually. But doing this we have no intuitive way of knowing that S_4 could be 20 seconds long in real life, while S_3 could be 90.

As pictured in figure 1.1, D will take a lot of time on S_4 and is forced to utilize some time from S_3 and S_5, which works well in a vacuum. The problems start to arise when E also has to utilize some additional time from its neighbouring stations, causing an overlap between D and E at $T + 2$.

The same problem arises with F and G at T as both items need to borrow time from the stations before and after. Thus we run into another overlap.

Do note that on T, E does not utilize the drift area which results in it sitting flush with F on the timeline, this may look good on paper but can result in overlap in practice due to the human workers at the assembly line occasionally taking a bit longer than presumed. This can be resolved by borrowing some time from S_2 and moving E into the drift area.

The same issue arises at $T + 1$ where C and D just barely get enough time, but it cannot get resolved by simply moving D forward, as D on $T + 2$ will require all of the time it can get on S_4.

1.5 Thesis Outline

Chapter 2

State of the art analysis

2.1 LSTM

In previous works addressing similar scheduling problems, researchers have applied Recurrent Neural Networks (RNNs), often with Long Short-Term Memory (LSTM) units, in a sequence-to-sequence (seq2seq) framework [2]. Seq2seq models are traditionally used in language-based tasks such as machine translation or text summarization, but the same architecture can be adapted to the assembly line problem. Each truck can be represented as a feature vector, and a day's worth of incoming trucks forms an input sequence. The seq2seq model can then be trained to output a corresponding placement sequence, effectively learning to replicate the tacit knowledge of Volvo's management workers in arranging trucks to minimize overlaps.

Although LSTMs are typically trained in a supervised manner, historical assembly line data can serve as training examples, where the input features correspond to product characteristics, and the output labels correspond to the placement decisions made by the servicemen.

2.2 Seq2Seq Models

2.3 Transformer-based Approaches

2.4 Heurestic Scheduling Methods

Chapter 3

Methodology

3.1 The Heuristic Approach

The problem to properly order manufacturing assembly lines with as few overlaps as possible is considered an NP-Hard problem.

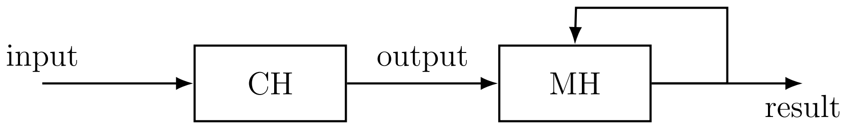

Figure 3.1: Heuristic solution

The Algorithm designed to solve this problem is a heuristic solution that will be made out of a Construction Hieuristic (*CH*) that produces a starting point based on pre-defined constraints, that feeds into a Meta Heuristic (*MH*) that finds a better solution starting from the output of the construction heuristic and self-improving until an acceptable result is returned.

3.2 Complementing the Heuristic Approach using Machine Learning

Due to the fact that servicemen today place the items manually using unwritten knowledge (tacit) that they have accumulated over the years.

The idea is that if they have knowledge of a good enough solution from the get-go with some risk of overlap, then we can train a Deep Learning Model (*ML*) on such previous data to give the algorithm a better starting point, thus reducing the runtime of that algorithm.

However it is worth to consider that such an approach can prove redundant or yield worse results if the problem at hand is an easy problem where many solutions can be

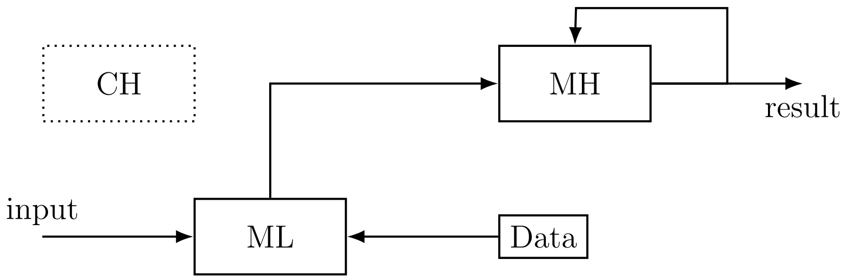

Figure 3.2: ML solution

found quickly, as opposed to a hard problem where a desired solution may not even be found.

3.3 System Architecture

Chapter 4

Experiments and results

4.1 Dataset

4.2 Evaluation Metrics

4.3 Runtime Analysis

Chapter 5

Discussion

5.1 Interpretation of Results

5.2 Limitations

5.3 Future Work

Bibliography

[1] J. Abbasi, *Predictive Maintenance in Industrial Machinery using Machine Learning*, Master's thesis, Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, 2021.

[2] A. Dupuis, C. Dadouchi, and B. Agard, *A decision support system for sequencing production in the manufacturing industry*, *Computers & Industrial Engineering*, vol. 185, p. 109686, 2023.

[3] J. Lindén, *Understand and Utilise Unformatted Text Documents by Natural Language Processing Algorithms*, Master's thesis, Mid Sweden University, Department of Information and Communication Systems (IST), Spring 2017.

[4] Author, Title, Journal, Year.

[5] Author, Title, Journal, Year.

[6] Author, Title, Journal, Year.

[7] Author, Title, Journal, Year.

[8] Author, Title, Journal, Year.

[9] Author, Title, Journal, Year.

[10] Author, Title, Journal, Year.

[11] Author, Title, Journal, Year.

[12] Author, Title, Journal, Year.

[13] Author, Title, Journal, Year.

[14] Author, Title, Journal, Year.

[15] Author, Title, Journal, Year.

[16] Author, Title, Journal, Year.

[17] Author, Title, Journal, Year.

[18] Author, Title, Journal, Year.

[19] Author, Title, Journal, Year.

[20] Author, Title, Journal, Year.