Finding critical constraints in schedules for re-entrant Manufacturing Systems

Author: Amir van Delft
Supervisors: Eghonghon Eigbe, dr. Neil Yorke-Smith

Introduction

We consider the flexibility of flexible manufacturing systems (FMSs) schedules subject to processing time, setup times and due dates. Our goal is to determine which constraints can break a given schedule and which do not participate at all. In this article we propose a method to distinguish these two different types of constraints.

Motivation

Understanding how a schedule can be made more flexible and how we can change constraints without worrying about the schedule breaking. Finding points of infinite flexibility in a schedule, and forcing such points on a benchmark.

Solution approach

We represent the problem as a graph and break it down to its strongly connected components(SCCs) using algorithms such as kosaraju's algorithm[1] to extract critical edges. Finding a defining pattern for SCCs in FMSs lead us to come up with a domain specific solution by iterating the operation sequence for the re-entrant machine

Contact

A.vanDelft@student.tudelft.nl

TU Delft

Research question

- How can we use and alter known algorithms to isolate what constraints could possibly cause a schedule to become infeasible and which do not participate at all
- How can we find relationships between constraints that might have an accumulating contribution to making a schedule infeasible
- How can we force certain constraints to be non-participating

Finding critical constraints

Our approach separates critical edges with potential to break the schedule from those that do not participate at all quickly enough to be run online. We also find such constraints follow a certain pattern that can be forced in specific places in the graph.

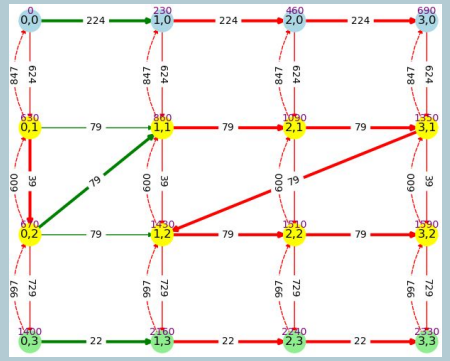


Figure 1: Graph representation of a schedule with red edges representing critical constraints and green ones, non-participating constraints

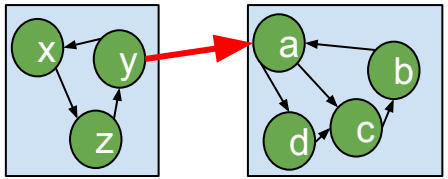


Figure 2: Graph with strongly connected components, and an acyclic edge connecting those components

Finding tightly bound constraints

We come up with a definition for tightly dependent constraints which can have an accumulating contribution for breaking a schedule.

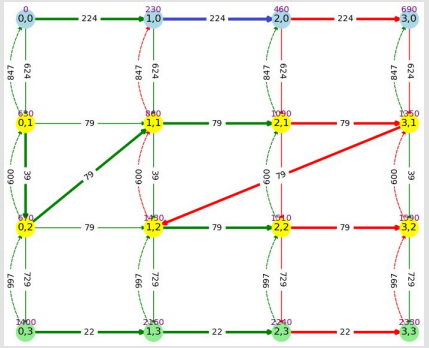
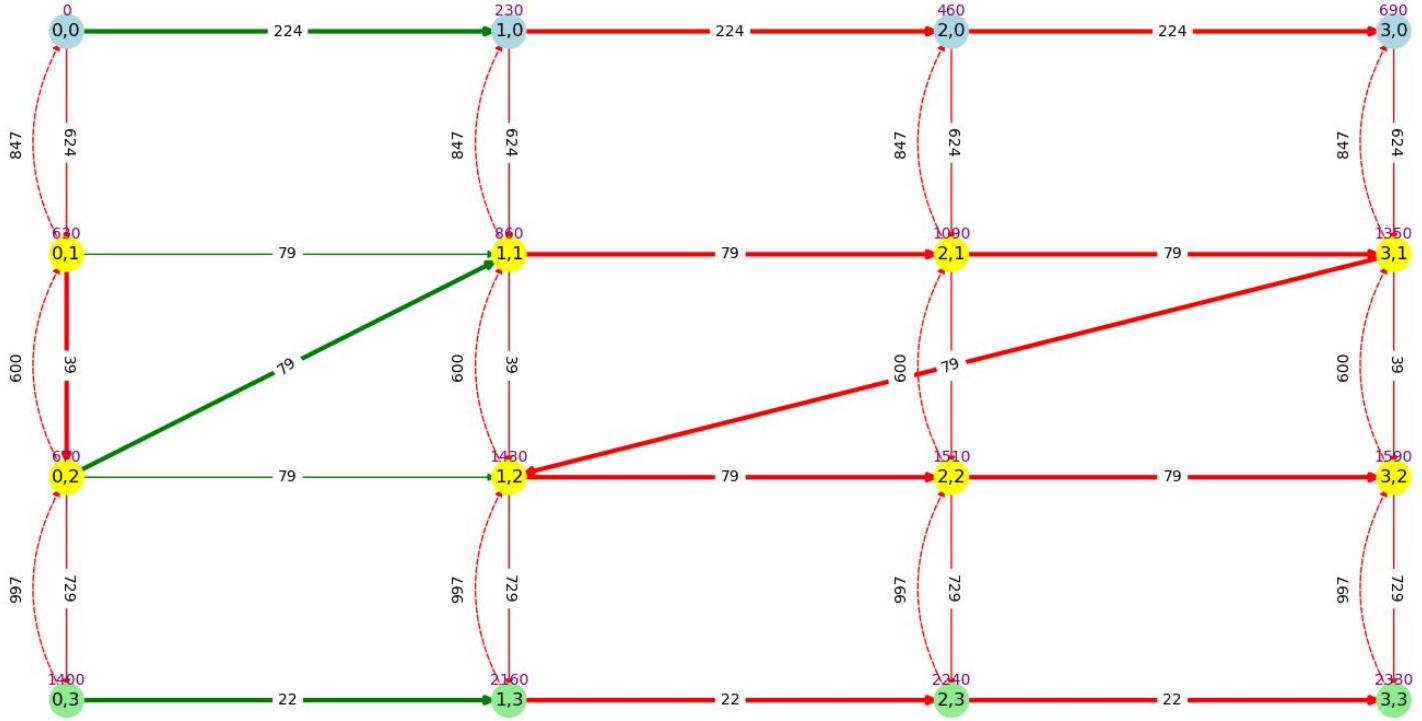


Figure 3: A selected constraint(in blue) with its tightly bound constraints in red

References

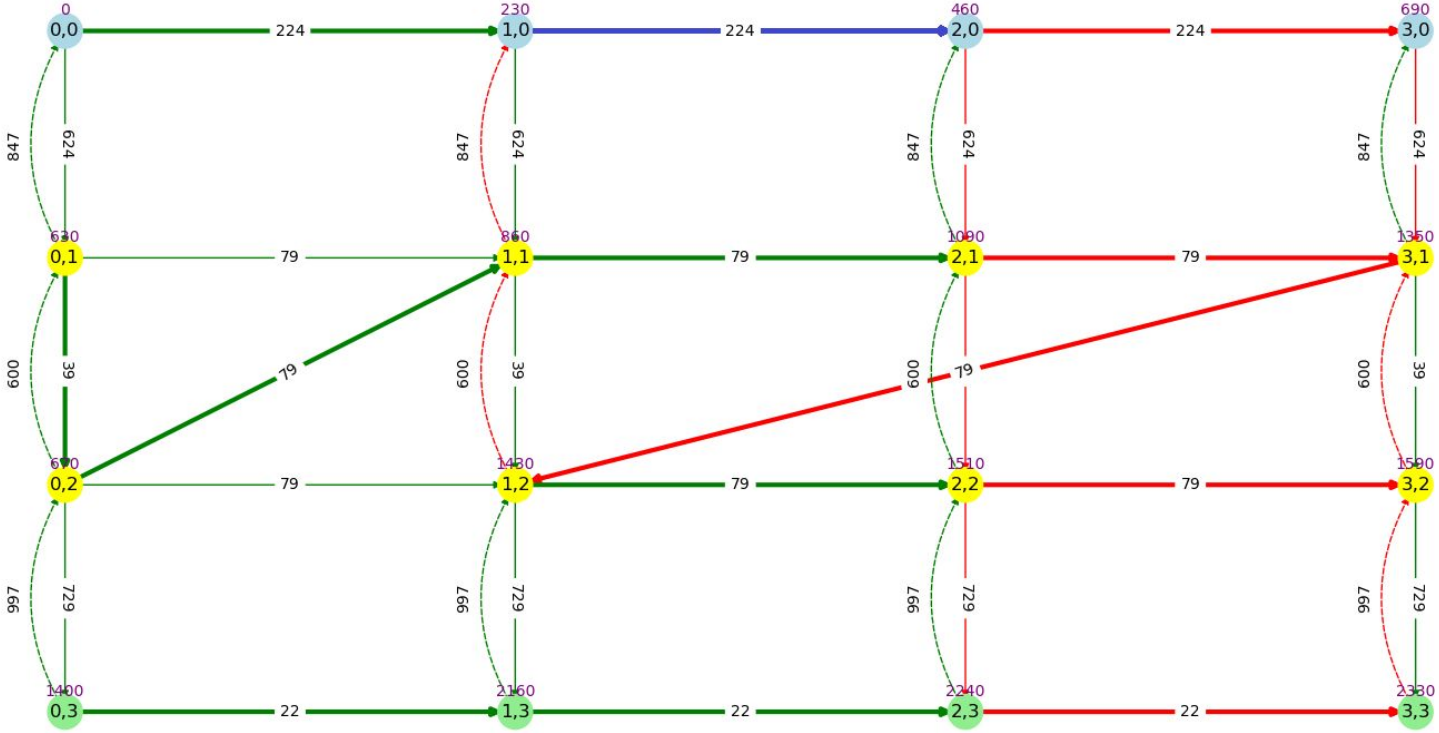
[1] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers & Mathematics with Applications*, vol. 7, no. 1, pp. 67–72, 1981.

Figure 1:
Graph representation of a schedule with red edges representing critical constraints and green ones, non-participating constraints



TU Delft

Figure 3:
A selected constraint (in blue) with its tightly bound constraints in red, and those that are loosely bound with it in green



TU Delft