Production Line Task Scheduling

Stefano Nicolis

A.A. 2021-2022

20 marzo 2022

Embedded and IoT Systems Design

Università degli Studi di Verona   Computer Science Department

Indice

1 Introduction

The assignment requires the creation of an algorithm for the scheduling of recipies on a production line.

Given a list of events that happen on the production line and the specification of the possible recipies to be executed, the algorithm must output a json file detailing the execution order of the recipies.

2 Assumptions

-The processingTime for an operation is assumed to be greater or equal to 1

-Although the input file has add-materials operations, there is no information in the Recipies.json file about how much material a certain operation uses, so even thought the algorithm checks for the material availability, its a futile check because it will always return positive result

-The algorithm can decide which tools to use for each operation when more than one are available, this is not decided at each start of new operation, so theoretically one could change the requirement while the recipie is rurrning, and optimize the firt part of it for speed and the other for

-Many design decision regarding the timing of certain actions, at first glance may seem incorrect, but please note that this is a descrete system, and reasoning in continuoos time is not comparable

3 Structure of the project

The algorithm is implemented in Python in a declarative manner.

Two files are given, the Recipes.json file and one or more input files in .csv format. An example of input file is:

type	subtype	key	value	time
recipes	new-order	DEMO_ICE	3	0
machine-event	add-materials	nails	48	2
machine-event	add-materials	wood	24	4
machine-event	add-materials	lego	28	25
machine-event	breakdown	WH	True	32

The input file specifies events on the production line and the time at which they occured, such as a machine breaking down or a new recipie order coming in, for example, from the file above we can see that the first line tells me that a new order for three DEMO_ICE recipies has arrived at time 0, the second line tells me that a new shipment of 48 nails is added to the warehouse, the fifth line tells me that at time = 32 the WH machine will go offline.

The Recipies.json file instead, stores the specification for each possible recipie, which means a name and a list of operations that compose that recipie. Every operation has a name and a list of possible equipment that can be executed with, one must be choosen at run time.

The scheduling algorithm is implemented in a declarative manner using Python, inside `main.py`, and it generates a schedule file in the `output` directory for each input file found in the `input` folder.

The `Recipie` class is used to represent a recipie in execution, each instance contains:

- name: the name of the recipie, taken from `Recipes.json` as a plain string
- opsData: short for operationsData, initially its a copy of the operations field taken from Recipie.json, during the execution it's updated to keep track of the recipie status
- opIndex: short for operationsIndex, the index of the operation that the recipie is currently at

The algorithm keeps track of the recipies execution by updating the opsData and opIndex fields of the various `Recipie` instances that have been created during the execution.

4 Example of execution

The algorithm simulates the passing of time with an integer variable, called timeTick, that initially starts at -1, which is incremented accordingly while the input file is read and the recipies are executed.

When the value of timeTick corresponds to the timestamp in the current line from the input file, it means it's time to execute the action specified in the line and then keep executing.

If there is a time gap between two consecutive lines in the input file, for example line 1 has a timestamp of 1 and line two has a timestamp of 26, after having read and executed line 1, the algorithm simulates all the intermediate steps from 2 to to 26 before executing what the second line says.

When terminating an operation for a particular recipie and starting a new one, for those that have more than one possibile equipment to be executed with, the algorithms needs to decide which one to use to execute the operation, so by calling the setEquipment method, a certain policy is applied and the choice is remembered by setting all the non-choosen equipment to empty string.

The current policy is to just choose the first equipment of the list, this could be easily modified to choose the equipment based on power consumption or execution speed.

Once all the input file has been read, given that some recipies might still be in execution, it keeps incrementing `timeTick` and executing the recipies.

Once all the recipies are over, the calculated schedule is written to file.

5 Final remarks

Let's make a brief example to understand how the algorithm works. Let's suppose that the input file only has the following line: The algorithm has to schedule a single `DEMO_ICE` recipe