

Production Line Task Scheduling

Stefano Nicolis
A.A. 2021-2022

16 luglio 2022



Embedded and IoT Systems Design

Indice

1	Introduction	2
2	Project structure	3
3	Example of execution	4
4	Notes	5

1 Introduction

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

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

2 Project structure

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 occur, such as a machine breaking down or a new recipe order coming in. As an example, from the file above we can see that the first line says that a new order for three DEMO_ICE recipes has arrived at time 0, the second line says that a new shipment of 48 nails arrives to the warehouse, the fifth line says that at time 32 the WH machine will go offline.

The Recipes.json file instead, stores the specification for each possible recipe, which means a name and a list of operations that compose that recipe. Every operation has a name and a list of possible equipment that can be executed with, one of which must be chosen 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 `Recipe` class is used to represent a recipe in execution, each instance contains:

- **name**: the name of the recipe, taken from `Recipes.json` as a plain string
- **opsData**: short for operationsData, initially it's a copy of the operations field taken from Recipe.json, and during the simulation it is accessed as a dictionary and updated to keep track of the recipe status
- **opIndex**: short for operationsIndex, the index of the operation that the recipe is currently at

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

3 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 recipes 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 26 before executing what line 26 says.

When terminating an operation for a particular recipe and starting a new one, for those that have more than one possible equipment to be executed with, the algorithm 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-chosen 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 recipes might still be in execution, it keeps incrementing `timeTick` and executing the recipes until each one has finished its execution.

Some recipes may not terminate if resources are lacking or the required machine broke down and never came back online. These recipes will be aborted and a warning is given.

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

4 Notes

The assignment left many aspects of how the program should look like up to the developer, some design choices are listed below:

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