Line Monitor: A Line Balancing and Optimization Program

Hongjoon Kim

Naveen Jindal School of Management, University of Texas at Dallas

ITSS 4381.501: Object Oriented Programming with Python

Professor Vatsal Maru

December 11, 2023

Line Monitor: A Line Balancing and Optimization Program

Background & Assumptions

Suppose you are production line manager at a packaging contractor for a world-renowned electronics company. You and your company are tasked with receiving the packaging supplies and the devices themselves to package them and transfer to a third-party outbound logistics vendor for those electronics to be sent to the clients. You would know that you have tangible and urgent goal of packaging thousands of these electronics in one shift. This may seem simple enough, but you will encounter several hiccups, delays, and bottlenecks in the production process. The latter could be a serious problem, as several packaged units may be queued while the pelleting personnels struggle to catch up. How does one overcome this without wasting precious time calculating the optimal worker counts? This is where Line Monitor comes in!

Some background knowledge is needed to understand how it works. The customer-centric metric that all production vendors ought to consider is takt time. *Takt time* is the time it takes for a single unit to pass through the production phase to adequately meet the customer demand.

$$Takt\ Time = \frac{Available\ Production\ Time}{Customer\ Demand}$$

For a manufacturer like you, your customer demand would be the size of the order in question. The time for each unit to pass through the process is called the *cycle time*. The key to line optimization is the inequality below.

$$Takt\ Time\ \geq Cycle\ Time = \sum_i \frac{t_i}{n_i}$$

In the equation above, t_i stands for task time it takes for a unit to pass through the phase i, while n_i represents the number of workers at stage i. The relationship between takt time and cycle time is that the former is the ideal and the latter is the actual state. Not only do we want to satisfy the requirement above in terms of cycle time, but we also want to prevent bottle neck from one stage to the next. This is what is called line balancing.

In addition, the production line is assumed to be broken into three stages. First, once the packager receives the necessary parts, a device enters the *packaging phase*. This is where a device gets put into a unit box, along with accessories like chargers and manuals. The packager, once done assembling the unit box package, places it on the assembly, concluding the packaging phase. Right away, the device takes time to move from the end of the packaging line to the palleting phase. This is called the *assembly line transportation phase*. There is no personnel here, so this stage can be largely ignored. Finally, the *palleting phase* involves scanning and placing the unit boxes on to a pallet, so that it could be taken by the outbound logistics vendor.

There are important assumptions that Line Monitor operates on. First, there is no delay in between the phases. Second, the scope of takt time is from the moment the packager receives all of the necessary parts to the moment the unit is scanned and placed on the pallet. Third, there are parts always available and accessible for the workers.

With these in mind, the role of Line Monitor hopefully becomes clearer: it is a tool to help production managers structure the line in optimal and balanced ways.

How to Use it

So, how do you begin to utilize it? First and foremost, the program, by nature, is a class. Therefore, you need to create an object, or an instance of this class. One needs to input the line name (in terms of str), order quantity (in terms of int), available production hours (in terms of float), and expected task time in seconds per unit in packaging/palleting phase (in terms of float). The parameters are arranged in the order of specified.

Once an object is created, one will be presented with a line name, department, and a range of methods to help optimize the line. The *optimizeTaktTimeWorker* method is the heart of Line Monitor. Given the input at the creation of the object, it will calculate the optimal takt time, its breakdown by each of the three phases, and the optimal worker count per phase. However, suppose that you want to update the available production hours, order size, expected task time, or stage proportion (a three-element list of three positive floats less than 1 representing the proportion of the cycle time each stage occupies). You can use productionHrEdit, orderSizeEdit, taskSecEdit, changeStageProportion methods, respectively. For more details, consult the python documentation for the methods.

Closing Remarks

In conclusion, the program should help the production line manager if he or she keeps the assumptions in mind, you will benefit from this program immensely. It should be obvious that this is a far cry from on-the-floor production usage, being limited by such constraints. Therefore, in the future, incorporating a way to feed operational data such as individual worker performance into the program on a regular basis, so that it can compare real production performance vs the customer-oriented production performance would be immensely beneficial. In addition, some sort of machine learning application to the program may help account for variations in part availability and various kinds of delays for each kind of device. Finally, the ability to customize according to different processes on the production floor would clearly increase flexibility and profitability.