Ice Cream Production with Random Machine Breakdown

Gabriel Yiu, Zi Yang Chen, Guangzhi Sun

January 13, 2021 (Updated Version)

Abstract:

Optimizing a production policy that minimizes the cost of production and inventory costs is a common problem many manufacturing facilities face. The complexity of creating an optimal policy increases if random manufacturing difficulties arise during the course of production. In this paper, we construct a production model that incorporates random machine breakdowns during production. We then solve for an optimal policy that schedules ice cream production using a Q-learning method. Our major findings are that the solutions to our model are robust, stable, and insensitive to changes in the cost of lateness and breakdown probability.

Introduction

Many researchers primarily consider optimizing a production line in ways that could practically be implemented into a factory production line deterministically. These research results are useful and applicable to real-world situations. In his book *Dynamic Optimization*, Bryson analyzes an ice cream production problem where he solves for an optimal schedule using deterministic methods (1975). However, we consider optimizing an ice cream production that incorporates random machine failure that aims to meet a deadline and maximize profit.

Georgiadis et al. (2020) focus on optimizing weekly production schedules for a Spanish canned fish production factory using a mixed-integer linear programming (MILP) based strategy. They use a mixed discrete and continuous-time model in their solution. They propose two useful and realistic assumptions for our model. First, they assume that meeting demand is necessary for a factory to maintain its credibility, so the deadline is hard (i.e. missing the deadline is not allowed). Second, the order cannot be split into multiple ones. This paper inspired us to use a high cost of lateness to simulate a hard deadline.

Gholami et al. (2009) solve the hybrid flow shop scheduling problem with stochastic machine breakdowns. They consider that both the interval between two adjacent machine failures and their repair times follow the exponential distribution, which is memoryless. This research leads to us to choose a geometric distribution for the number of days of operation and repair.

Model Framework and Assumptions

Chakraborty et al. (2008) suggest a framework where there are two states the production can be in: “in-control” and “out-of-control.” During the in-control state, production quantities are fixed and known according to the decisions of the producer. On the day, the machine transitions from an in-control state to an out-of-control state, production produces a random number of defective units that is dictated at the time of breakdown during the day. While the machine is out-of-control, production halts and there are no decisions to be made. Gholami et al. (2009) suggested that the interval between two adjacent machine breakdowns and their repair times follow an exponential distribution.

We consider an analogous discrete framework where days between adjacent machine breakdowns and their repair times follow a geometric distribution with parameters p_B and p_R respectively. The production machines have the same probability of breakdown p_B each day because the geometric distribution is memoryless. Embedded in this framework is the assumption that machines are as good as new at the start of each day because they have the same probability of breaking down each day. One essential assumption in the ice cream production operation, therefore, is that preventative maintenance is conducted at the end of each day to maintain the quality of the machine. The cost of maintenance is embedded in the cost function to produce. Let us introduce the following variables and related assumptions for the formulation of our model.

Variable	Definition	Dimensions
Δx_t	The amount of ice cream produced on day t. Δx_t will be determined by a policy π.	L^{-3}
$x(t)$	The amount of inventory on day t. Note that $x(t) = \sum_{i=1}^{t-1} \Delta x_i$.	L^{-3}
t_D	The deadline to fill the order. The order is considered late if production occurs on t_D.	T
B	The order of ice cream units to be filled by time t_D.	L^{-3}
M	The maximum units of ice cream that can be produced in one day.	L^{-3}
C_I	The cost of inventory per unit of ice cream per day.	$L^{-3}T^{-1}$
$C_P(\cdot)$	The cost of producing Δx_t units in one day. The function is increasing and convex. Assume that $C_p(0) = 0$.	None
C_L	The cost of being late per day. The quantity is constant and does not increase with time.	T^{-1}
p_B	Probability of breakdown per day if machine is working.	None
p_R	Probability of repair per day if machine is in repair.	None
$I_S(\cdot)$	$I_S(t) = \begin{cases} 1, & \text{in} - \text{control} \\ 0, & \text{otherwise} \end{cases}$	None
$I_L(\cdot)$	$I_L(t) = \begin{cases} 1, & t \geq t_D \\ 0, & t < t_D \end{cases}$	None
T_C	A random variable that indicates the time of completing the order. We note that $x(T_C) = B$.	T

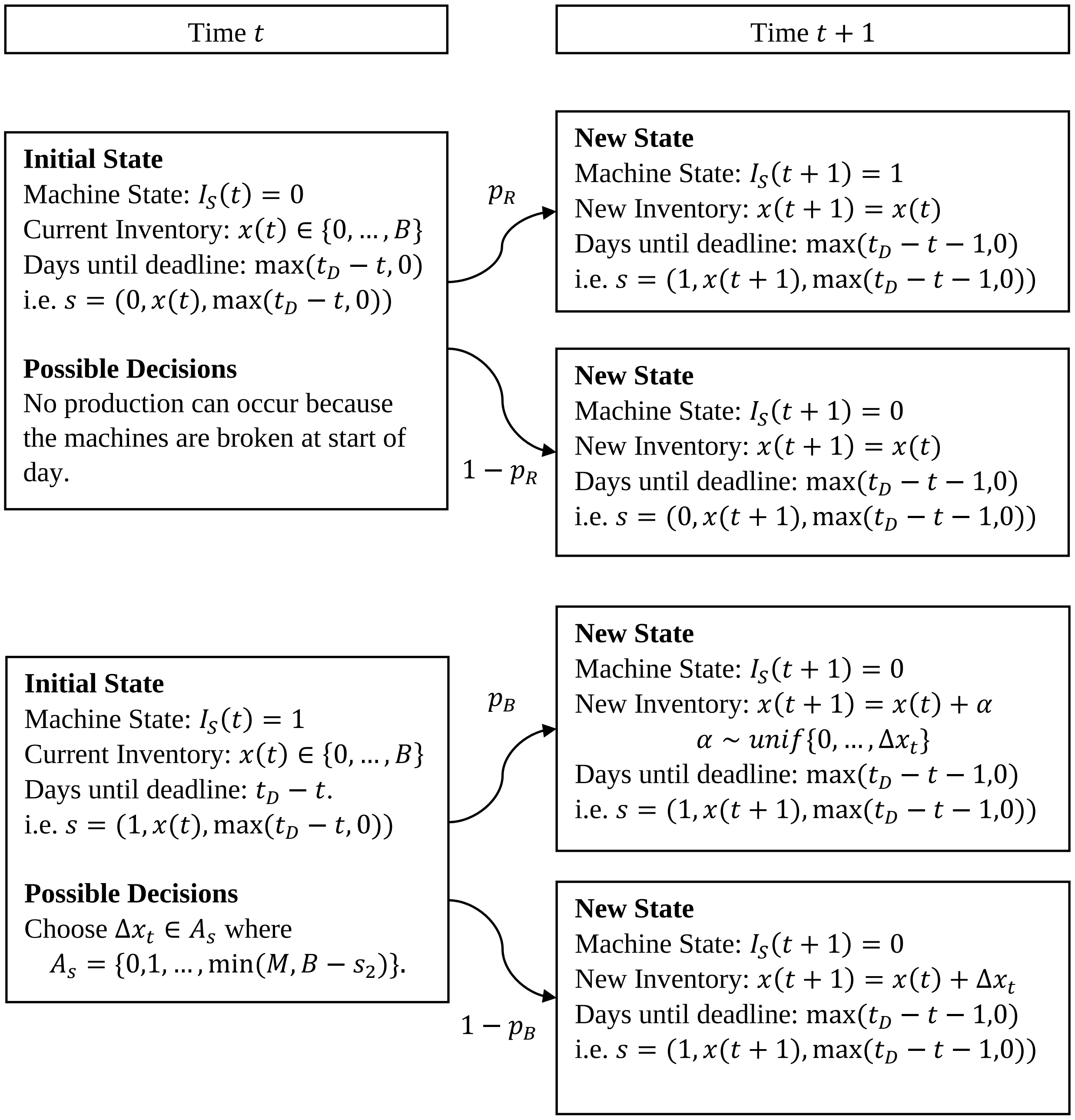
Figure 1: Table of variables

In this framework, we cannot deterministically solve for an optimal schedule ahead of time because the probability of machine failure on each day may disrupt our planned schedule. We, instead, need to find a policy that determines how much ice cream Δx_t to make at the beginning of each day t. The policy should be a function π of the machine state, current inventory, and the days until the deadline. If the current day $t \geq t_D$, we set the days until the deadline to zero because the future expected costs will be the same for any $t > t_D$ if $x(t) = x(t_D)$ and $I_S(t) = I_S(t_D)$. Once Δx_t is determined by our policy, our total cost incurred at the end of day t will be:

$$C_t = C_I x(t) + C_p(\Delta x_t) + C_L I_L(t).$$

We can formulate our problem into a Markov Decision Process with the following definitions.

1. Set of possible states is $S = \{0,1\} \times \{0, \dots, B\} \times \{0, \dots, t_D\}$. If $s = (s_1, s_2, s_3) \in S$, the first, second, and third entries of s are the machine state, current inventory, and days until the deadline, respectively.
2. Set of possible actions at state s is $A_s = \{0, 1, \dots, \min(M, B - s_2)\}$. Note that policy functions $\pi(s) \in A_s$ if $s_1 \neq 1$.
3. The transition possibilities and their probabilities are given in *Figure 2*.
4. Our immediate reward is $-C_t$.



Time t
Time $t+1$
Initial State
Machine State: $I_S(t) = 0$
Current Inventory: $x(t) \in \{0, \dots, B\}$
Days until deadline: $\max(t_D - t, 0)$
i.e. $s = (0, x(t), \max(t_D - t, 0))$
Possible Decisions
No production can occur because the machines are broken at start of day.
p_R
$1 - p_R$
New State
Machine State: $I_S(t+1) = 1$
New Inventory: $x(t+1) = x(t)$
Days until deadline: $\max(t_D - t - 1, 0)$
i.e. $s = (1, x(t+1), \max(t_D - t - 1, 0))$
New State
Machine State: $I_S(t+1) = 0$
New Inventory: $x(t+1) = x(t)$
Days until deadline: $\max(t_D - t - 1, 0)$
i.e. $s = (0, x(t+1), \max(t_D - t - 1, 0))$
Initial State
Machine State: $I_S(t) = 1$
Current Inventory: $x(t) \in \{0, \dots, B\}$
Days until deadline: $t_D - t$.
i.e. $s = (1, x(t), \max(t_D - t, 0))$
Possible Decisions
Choose $\Delta x_t \in A_s$ where
$A_s = \{0, 1, \dots, \min(M, B - s_2)\}$.
p_B
$1 - p_B$
New State
Machine State: $I_S(t+1) = 0$
New Inventory: $x(t+1) = x(t) + \alpha$
$\alpha \sim unif\{0, \dots, \Delta x_t\}$
Days until deadline: $\max(t_D - t - 1, 0)$
i.e. $s = (1, x(t+1), \max(t_D - t - 1, 0))$
New State
Machine State: $I_S(t+1) = 0$
New Inventory: $x(t+1) = x(t) + \Delta x_t$
Days until deadline: $\max(t_D - t - 1, 0)$
i.e. $s = (1, x(t+1), \max(t_D - t - 1, 0))$

Figure 2: Markov Chain

Our objective, therefore, is to find a policy π^* that minimizes the following expected cost:

$$\mathbb{E}\left\{\sum_{t=1}^{T_c} C_I x(t) + C_p(\pi^*(I_S(t), x(t), \max(t_D - t, 0))) + C_L I_L(t)\right\}.$$

The following quantity is extremely difficult to optimize analytically; hence, we use a reinforcement learning algorithm to solve for π^*. The framework of the problem is set up such that we can apply a Q-Learning algorithm.

Q-Learning

The model has been constructed in the format of a Markov Decision Process. Reinforcement learning algorithms, more specifically Q-learning, will be used to solve for π^*. The Q-learning algorithms train the policy based on trial and error, encouraging good decisions and punishing bad ones. Inspired by Sutton and Barto (2018), the basic idea and the pseudocode are as follows:

```
Set parameters
Initialize $Q(x, a)$ (the objective function) with arbitrary value;
Repeat (for number of episode):
    Initialize $x$;
    Repeat (for number of steps in an episode):
        Choose $a_t$ from $s_t$ based on Ɛ-greedy strategy;
        Take action $a_t$, evaluate $r_t$ and $x_{t+1}$
        $Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t \left[r_t + \gamma \max_a Q_t(x_{t+1}, a) - Q_t(x_t, a_t)\right]$
        $x = x_{t+1}$
```

Where x_t and a_t are current state and action, x_{t+1} is the state after a_t is taken, α_t is the learning rate, r_t is the reward, γ is the discount factor, $\max_a Q_t(x_{t+1}, a)$ is the estimated optimal value. The learning rate determines weights new information based on the new reward and old information that it has previously learned. The discount factor tells us how much we should weight expected future rewards with current rewards. The Ɛ-greedy strategy allows the model to pick a random action instead of the already learned optimal with probability Ɛ so that the model gets rid of the local extreme and keeps updating.

The learning constants we chose for α_t, γ and Ɛ are 0.8, 1, and 0.2, respectively. The model, therefore, prefer exploring 20% of the time instead of exploiting the old information. In addition, we set the discount factor at one so the cost of lateness will be fully recognized.

Comparison to Deterministic Model

To test the robustness of our model, it is beneficial to compare the deterministic solution where there is no probability of breakdown (i.e. $p_B = 0$) with the policies produced when there are high costs of lateness and low probabilities of breakdown. These two properties mimic the deterministic case as closely as possible where there is no chance of being late and no probability of breakdown. We trained the model with and without the consideration of machine failure and obtained two policies. The following example compares the deterministic solution π_D to the corresponding learned policy π_R under the following table of parameters (Figure 3).

t_D	B	M	C_I	C_L	p_B	p_R	$C_p(\Delta x_t)$
10	100	50	4	50	0.01	0.7	Δx_t^2

Figure 3: Parameters of the model

The policy π_R produces the following schedule in these scenarios. In the subsequent schedules, red indicates machine breakdown.

Day	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Costs**
Deterministic Solution π_D	1	3	5	7	9	11	13	15	17	19	-	-	**2470**
No Breakdown π_R	1	4	3	7	1	13	20	19	15	17	-	-	**2616**
Breakdown on Day 7 and repaired on Day 8 π_R	1	4	3	7	1	13	20	3	0	3	16	29	**3624**

Figure 4: Solution table for three cases

If we look at the pattern of growth of the inventory *(Figure 5)* under the policy π_R and the pattern of growth under π_D, there is a clear similarity in the pattern.

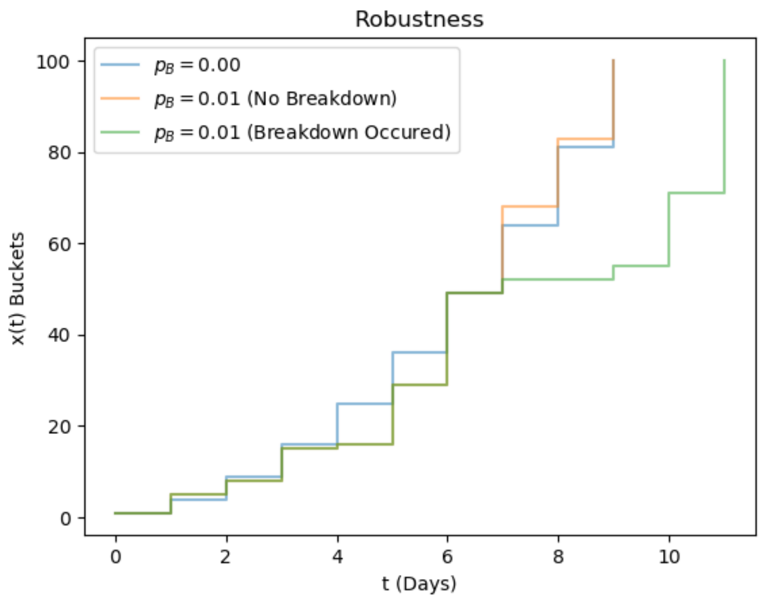


Figure 5: Robustness

The major difference between the production schedules of π_D and π_R is where the bulk of production occurs. The deterministic solution π_D has a steady monotonic increase in production per day as time approaches the deadline whereas π_R ramps up production on days 5-8 and has slightly less production on the last days up until the deadline. In the no breakdown scenario, π_R attempts to avoid late fees by ramping up production a little earlier compared to π_D but also risks producing late to minimize inventory costs. However, the differences between the two production schedules is relatively small when the breakdown does not occur. This result suggests that our model is robust.

It is interesting to note that, in the scenario where failure occurred on day seven, the policy thought it best to spread out production over three days in days 9-11 instead of increasing the rate of production and meeting the deadline. The convex nature of C_p likely encouraged the policy to make this decision as extremely high rates of production would be more costly than the late fees.

Stability of Solution

Another issue to address regarding the robustness of our model is to test the stability of our algorithm in producing optimal policies. Given the same set of parameters, do different runs of the Q-Learning algorithm produce the same or similar policies? One way of testing this question is to train two algorithms under the same set of parameters, then we can simulate production runs and compare policies' distribution of costs. Let us train two policies π_1 and π_2 under the following parameters.

t_D	B	M	C_I	C_L	p_B	p_R	$C_p(\Delta x_t)$
10	100	50	8	50	0.05	0.7	Δx_t^2

Figure 6: parameters used to test stability.

After training π_1 and π_2 under these parameters, we simulated 100 production runs and obtained two samples of costs.

Policy	Mode	Mean	St. Dev	Min	Max
π_1	4846	5313.95	1348.7	4013	13074
π_2	4480	5514.91	2003.81	4330	15183

Figure 7: statistical summary

We observe that the summary statistics are similar except for the difference in standard deviation. If we plot overlaying histograms *(Figure 8)* of the data, we can see that the distributions are extremely similar. This result suggests that our solution is stable.

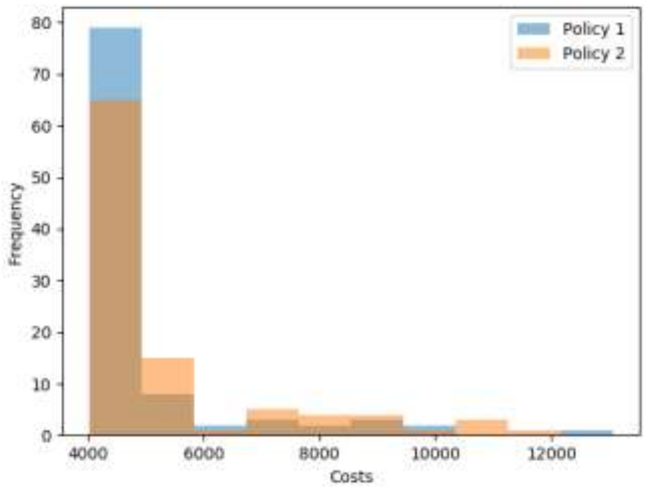


Figure 8: Histogram of Costs

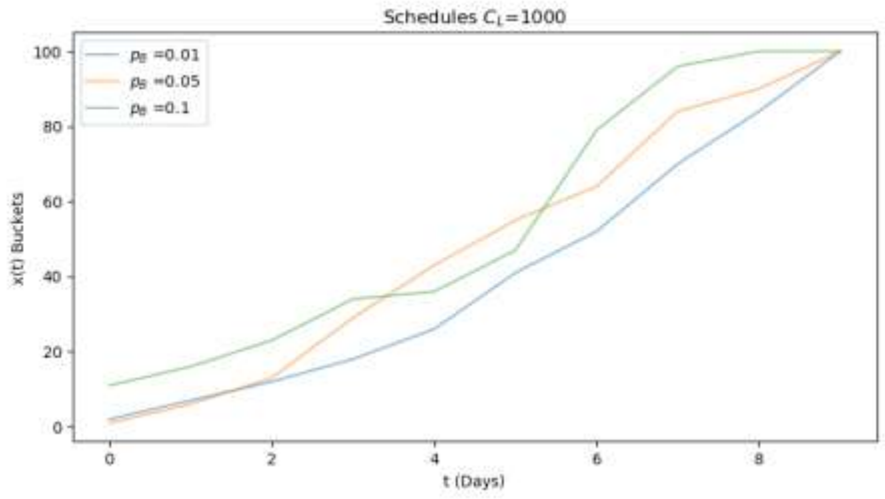
Sensitivity to Lateness Fee and Probability of Breakdown

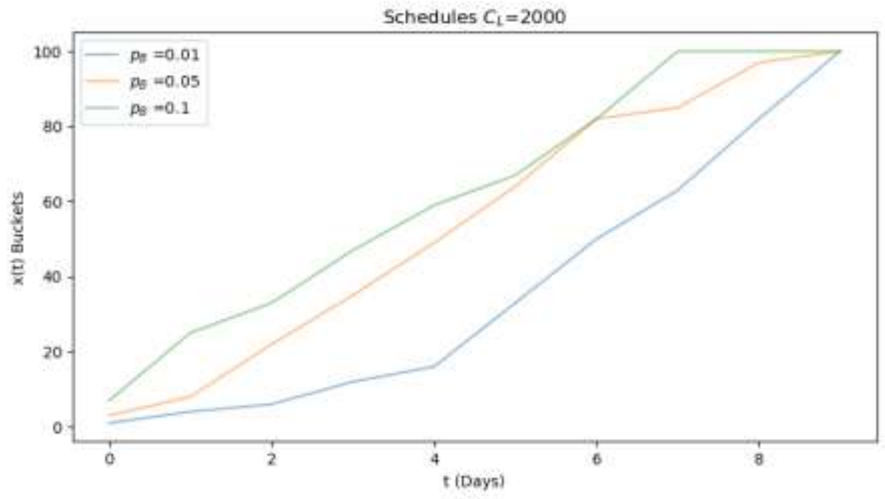
We are particularly interested in general shifts in the production schedule when lateness fees and the probability of breakdown changes. How dramatically does the schedule change and how do these changes affect costs in response to changes in lateness fees and the probability of breakdown. To look at how the algorithm changes with respect to these two variables, train nine algorithms π_{C_L,p_B} where $C_L = 1000, 2000, 3000$ and $p_B = 0.01, 0.05, 0.1$ holding the remaining parameters fixed at these values $t_D = 10, B = 100, M = 50, C_I = 4, p_R = 0.7, C_p(\Delta x_t) = \Delta x_t^2$. The resulting schedules are produced below (see Figure 9 for visualizations of schedule).

Production Schedule Under Situation with No Failures											
Policy	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	**Costs**
$\pi_{1000,0.01}$	2	5	5	6	8	15	11	18	14	16	**2524**
$\pi_{1000,0.05}$	1	5	7	16	14	12	9	20	6	10	**2828**
$\pi_{1000,0.1}$	11	5	7	11	2	11	32	17	4	0	**3538**
$\pi_{2000,0.01}$	1	3	2	6	4	17	17	13	19	18	**2566**
$\pi_{2000,0.05}$	3	5	14	13	14	15	18	3	12	3	**3086**
$\pi_{2000,0.1}$	7	18	8	14	12	8	15	18	0	0	**3470**
$\pi_{3000,0.01}$	3	1	2	6	9	11	8	27	12	21	**2686**
$\pi_{3000,0.05}$	4	5	10	12	11	17	16	14	7	4	**2908**
$\pi_{3000,0.1}$	4	10	22	11	17	24	12	0	0	0	**3942**

As expected, we observe a general shift towards earlier production for higher costs of lateness and a higher probability of breakdown. We analyze the relationship between optimal production schedule and cost of lateness by fixing a probability of breakdown and compare results with different lateness costs. The optimal schedule produced by the algorithm seems to respond to the movement of the probability of breakdown and cost of lateness in a multiplicative fashion. For production schedules produced by policies trained on $p_B = 0.01$, we observe there is not much difference between the schedules on different costs of lateness. Different costs of lateness lead to similar production schedules. The increase in the cost of lateness does not significantly affect the optimal production schedule. However, there are noticeable production schedule differences by policies trained on $p_B = 0.1$. As the cost of lateness increase from 1000 to 3000, the optimal production schedule shifts from finishing one day before the deadline to three days before the deadline. A possible explanation for this result is that when the probability of breakdown is low, the model requires are an exceptionally large increase in lateness costs for a significant change in the production schedule to occur. A low probability of breakdown occurring directly relates to the probability that the factory incurs the lateness cost. When the probability of breakdown is high, an increase in lateness costs similar to the above comparison would be enough to significantly change the production schedule.

Production Schedule Inventories





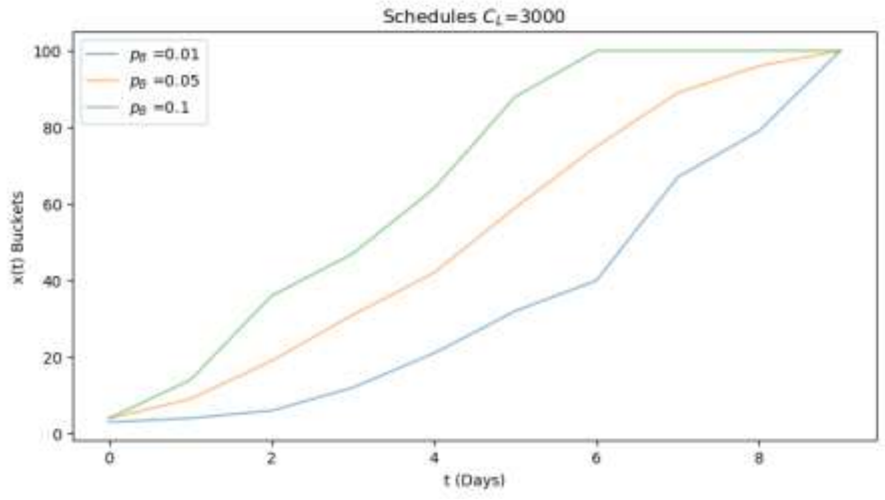


Figure 9

To better reflect real-world circumstances, we train the model and run simulations with random breakdowns. As before, we vary p_B and keep C_L constant at 1500. Then we change C_L with some fixed p_B and observe how these changes affect the total cost. Since randomness is involved, we simulate 30 trails for each case to observe the general pattern.

p_B *and* C_L	Mean(Total cost)	St. Dev(Total cost)
$C_L = 1000, p_B = 0.01$	2874.24	481.29
$C_L = 2000, p_B = 0.01$	3241.02	1577.44
$C_L = 3000, p_B = 0.01$	2934.74	490.92
$C_L = 1000, p_B = 0.05$	4903.74	2504.03
$C_L = 2000, p_B = 0.05$	5009.41	2750.32
$C_L = 3000, p_B = 0.05$	5076.70	2190.35
$C_L = 1000, p_B = 0.1$	5022.86	2574.61
$C_L = 2000, p_B = 0.1$	5646.41	3645.53
$C_L = 3000, p_B = 0.1$	6192.21	3008.12

Figure 10

The basic result remains the same as the no breakdown circumstance, p_B and C_L is positively correlation with the total cost. As stated before and shown in Figure 10, the total cost becomes more sensitive with respect to C_L as p_B gets larger. This is because when more breakdowns occur during the training section, the algorithm focuses more on the impact of breakdown and will be punished more often for lateness. This causes the algorithm to respond to a higher C_L. On the other hand, the standard deviation of total cost gets larger as p_B and C_L increase. In reality, even if the optimized schedule is given by the model, breakdowns can still significantly increase the total cost. An effective way to lower the total cost might be to update the machine and lower the probability of breakdown.

Conclusions and Findings

Our model of incorporating random machine breakdowns in ice cream production produces solutions that are relatively robust, stable, and insensitive to the cost of lateness or breakdown probability when the other parameter is low. We find that fluctuations exist in cases where breakdown probability is present. Our analysis shows that if one of C_L (cost of lateness) and p_B (probability of breakdown), the optimal policy does not deviate from an optimal policy as much. Therefore, when the value of one of these parameters is fixed at a low value, a very large change in the other is required to significantly affect the optimal production schedule. The total cost gets more sensitive to one of the parameters when the other parameter increases.

There are many future problems that can be studied within this model framework. An important question that should be studied is how much change in the cost of lateness or probability of breakdown necessary to cause the algorithm to change policies significantly. One can try to find the mathematical relationship between lateness cost, probability of breakdown and the production policies produced. There may be a threshold for the lateness cost that separates the situations in which the factory chooses to risk being late or chooses to finish early to avoid late costs.

References

Bellman, R. (2010). *Dynamic Programming.* Princeton, NJ: Princeton University Press.

Bryson, A. E. (1975). *Applied Optimal Control - Optimization, Estimation and Control.* New York, NY: Taylor & Francis Group.

Georgiadis, G. P., Pampín, B. M., Cabo, D. A., & Georgiadis, M. C. (2020). Optimal Production Scheduling of Food Process Industries. *Computers & Chemical Engineering, 134*, 106682. Retrieved from https://doi.org/10.1016/j.compchemeng.2019.106682

Gholami, M., Zandieh, M., & Alem-Tabriz, A. (2008). Scheduling Hybrid Flow Shop with Sequence-Dependent Setup Times and Machines with Random Breakdowns. *The International Journal of Advanced Manufacturing Technology, 42*(1-2), 189-201. Retrieved from https://doi.org/10.1007/s00170-008-1577-3.

Kopanos, G. M., Puigjaner, L., Georgiadis, M. C., & Bongers, P. M. (2011). An Efficient Mathematical Framework for Detailed Production Scheduling in Food Industries: The Icecream Production Line. *Computer Aided Chemical Engineering, 29*, 960-964. Retrieved from https://doi.org/10.1016/B978-0-444-53711-9.50192-9.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction. Second Edition.* Cambridge, MA: MIT Press.

Tulika Chakraborty, B. G. (2008). Production lot sizing with process deterioration and machine breakdown. *European Journal of Operational Research, 185*(2), 606-618. Retrieved from https://doi.org/10.1016/j.ejor.2007.01.011.