Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

A SIMULATION OPTIMIZATION APPROACH FOR MANAGING PRODUCT TRANSITIONS IN MULTISTAGE PRODUCTION LINES

Atchyuta Bharadwaj Manda
Karthick Gopalswamy
Sara Shashaani
Reha Uzsoy

Edward P. Fitts Department of Industrial and Systems Engineering
North Carolina State University
111 Lampe Dr
Raleigh, NC 27695, USA

ABSTRACT

We explore the problem of managing releases into a multistage production system transitioning from producing a mature product in high volume to a new one whose production process is initially unreliable but improves as experience is accumulated. We use simulation optimization to develop solutions and examine the impact of learning at a single machine on the rest of the system. This work lays the foundation for studying product transitions using realistic fab scale simulation models.

1 INTRODUCTION

Effective management of product transitions, in which a product that a firm is currently producing and selling is replaced by another throughout its supply chain (Bilginer and Erhun 2010; Billington et al. 1998), is critical to survival in today's rapidly changing global economy. A closely related problem, that of ramp-up, involves bringing a new product into high volume production in a new plant. In the semiconductor industry, both business and retail customers have come to expect frequent introduction of improved products at lower prices (Rash and Kempf 2012). Delays in bringing a new product into high volume manufacturing, especially when a competitor is successful in doing so, can have serious, long-term financial impacts on a firm. Salomon and Martin (2008) estimated that a month's delay in bringing a new semiconductor wafer fab into volume production costs a firm $22,000,000 in lost revenue. Most of the extensive literature on product transitions (Lim and Tang 2006) has treated product transitions as a strategic problem, using simplified models and generally ignoring the impact of the new product on products not involved in the transition. However, a large firm active in multiple market segments may have multiple new products launched into its factories in a year. Difficulties in producing the new product may significantly increase the variability of the effective processing time distribution in the factory, adversely affecting cycle time and throughput for all products sharing the same production resources. As experience with the new product accumulates, and problems are identified and remediated, these negative impacts gradually diminish.

In previous studies of product transitions in a single-stage production system (Manda and Uzsoy 2020; Manda and Uzsoy 2018), we found that careful management of work releases of both old and new products can significantly improve factory performance, considering the effects of learning by experience and by explicit experimentation that consumes production capacity in order to improve processing capabilities. In this paper we extend our previous approaches to a multiple stage production line, which poses significant challenges to simulation optimization due to the very large search space and the highly variable production

environment. The work is exploratory in nature, seeking to lay the foundation for future work studying the impact of product transitions using realistic simulation models of semiconductor wafer fabs.

The following section briefly reviews related work on learning in production systems, production planning with learning, and simulation optimization. We then describe our learning model, our simulation model and our simulation optimization approach in Section 3, and report our computational results in Section 4. We conclude with a discussion of our principal findings and some directions for future work.

2 Literature Review

Learning in production systems was first discussed in the context of the improvements observed in the cost or duration of a task as experience accumulates by repetition, leading to the classical literature on learning curves (Yelle 1979; Anzanello and Fogliatto 2011). Models of this type have been used to capture the improvement in semiconductor device yield as experience producing a device is accumulated (Tirkel 2013). A widely used proxy for experience is the cumulative production of the product up to a particular point in time. Production planning models incorporating these learning effects assume that the mean time to produce a unit of output will decrease with experience (Ebert 1976; Liao 1979; Hiller and Shapiro 1986).

Subsequent efforts expanded the scope of the problem to include learning by deliberate experimentation that consumes capacity but does not yield a saleable product. Chand et al. (1996), Fine (1986, 1989) and Terwiesch and Bohn (2001, 2004) address the problem of how much capacity to dedicate to revenue-generating production as opposed to experiments aimed at improving process quality and yield. Kim and Uzsoy (2008, 2013) develop models of a single production resource whose behavior is represented using a clearing function (Missbauer and Uzsoy 2020). Learning is a function not of cumulative production, but of cumulative experimentation, modelled as engineering lots that cannot be sold but improve the mean processing time of the new product after a time lag required to implement the insights obtained from the experiments, but its effect on variability of processing is not explicitly modelled. They present deterministic models to optimize the releases of both production and engineering lots, and use the Kuhn-Tucker optimality conditions to examine the marginal value of additional experimentation. Manda and Uzsoy (2020) extend this work by modelling the impact of learning by experience on both the mean and variance of the effective processing time of a single production resource. This approach is extended to consider both learning by experience and by experimentation using simulation optimization with a genetic algorithm as the search engine (Manda and Uzsoy 2018). This paper extends these efforts to a multistage production line where learning can occur at different stations, as a first step towards studying wafer fab models of realistic size.

Several authors have studied product transitions using simulation. Crist and Uzsoy (2011) examine the relative prioritisation of engineering and production lots using a simulation model of a simplified wafer fab. Nemoto et al. (2000) use simulation to quantify the benefits of cycle time reduction on yield during fab ramp-up, while Leachman and Ding (2011) develop an analytical solution. Haller et al. (2003) use a simulation model to examine the problem of managing cycle time during ramp-up. Simulation models have the ability to represent the behavior of production resources and different learning mechanisms in great detail, subject to the availability of data to validate them, making simulation optimization an attractive option for managing product transitions in the event scalable procedures can be developed.

Simulation optimization refers to a family of stochastic optimization techniques that are used when the performance of a solution cannot be calculated analytically, but instead must be estimated using a simulation model. Given a stochastic simulation model in which each replication of a single decision input returns simulated outputs that contain noise, finding a globally optimal solution is particularly hard with a high dimensional continuous search space. The computational burden of running the simulation model to evaluate each feasible solution is also significant. These concerns motivate a need for an efficient and reliable optimizer for this environment. The three main approaches to simulation optimization over a continuous domain are (i) direct search and heuristics, (ii) line-search and Newton-like methods, (iii) and model-based and trust-region methods. While in i) new solutions are directly generated with an evolutionary mechanism or pattern search, (ii) chooses the direction and length of the step size using gradient information, and (iii)

does so implicitly through a constructed local model whose step lengths are bounded by the size of the neighborhood in which the model is built and validated. An important factor in the choice of simulation optimization methods is the availability of direct gradient observations from the stochastic simulation, whose absence renders the problem *derivative-free*. Suitable solution methods for the problem at hand are hence direct search methods such as genetic algorithms or the Nelder-Mead simplex algorithm (1965), stochastic approximation with finite-differences (Robbins and Monro 1951) or more recently developed stochastic derivative-free trust-region methods (Shashaani et al. 2018). We have experimented with both a genetic algorithm and stochastic approximation, which we describe in more detail in Section 4.

3 Modelling and Analysis: A Simulation Optimization Procedure

The simulation model is a modification of that by Li et al. (2011) with simple linear routings without reentry, essentially a multi-stage version of the simulation model developed by Manda and Uzsoy (2019) using non-preemptive engineering activity based disruptions and mechanisms of learning from production lots at each station. Batch processing machines and machine failures are eliminated to ensure that the principal source of variability in the system is the presence of the new product.

3.1 Notation and Formulation

The simulation takes place over a time horizon of T discrete periods each of length Δ. The production system consists of 11 stations in series, each with a single machine, except Station 4 which has two machines. The processing times at each station are log-normally distributed to avoid negative values. Production lots of both products pass through all the stations and can be used to meet demand. For this exploratory work, we assume that the new product induces engineering disruptions at only one station in the line, station n, whose location relative to the bottleneck is varied to examine the effect on throughput and cycle time. At the start of the planning horizon, the line produces Product 1 to meet a known stable demand with a fully debugged process. Over the planning horizon, demand for Product 1 is gradually replaced by that for a newer product, Product 2. While there are many ways in which a new product may disrupt processing in an already operating factory, we assume that all such disruptions are manifested as a process event, leading to a stoppage of production while the problem is identified and remedied (an engineering hold). The mature product, Product 1, has a natural processing time with mean $\mu^1(n)$ and standard deviation $\sigma^1(n)$. A process event requiring engineering activity with mean duration $P^1(n)$ and standard deviation of $\sigma_p^1(n)$ occurs at station n on average once in the processing of every $Q^1(n)$ units, which represents the expected number of units successfully processed between engineering holds in a completely debugged process. Product 2 has the same natural processing time and engineering activity duration distributions as Product 1, but induces disruptions that require engineering activities more frequently. However, as experience producing Product 2 accumulates, the mean number of lots processed between disruptions increases, eventually reaching the steady state value $Q^2(n)$. Denoting the average number of units of Product 2 successfully processed at station n between process events by $Q_t^2(n)$, the effect of production experience on the frequency of process events at station n is described by the exponential learning model

$$Q_t^2(n) = Q_0^2(n) + (Q^2(n) - Q_0^2(n))(1 - e^{-\alpha X_t^2(n)}), \tag{1}$$

where $Q_t^2(n)$ denotes the average number of lots between disruptions at station n in period t, $X_t^2(n)$ the cumulative production of Product 2 at the start of period t and the parameter α controls the rate of learning.

The presence of this time-dependent learning mechanism results in significant transient behavior over the planning horizon, as discussed in our previous work on single-stage systems (Manda and Uzsoy 2020). The engineering holds induced by Product 2 increase both the mean and the variability of the effective service time distribution at the learning station n (due to the effect of the engineering holds) and at downstream stations (due to the variability of the output from station n). Maintaining the total releases of both products at a constant level ignores these effects, resulting in higher resource utilization and variability, and hence

longer, more variable cycle times and WIP levels, reducing throughput until the adverse effects of the engineering holds are eliminated through learning. Neglecting these reductions in processing efficiency can result in utilization temporarily exceeding 1, causing large accumulations of WIP and very long cycle times. However, as learning takes place and processing efficiency and variability improve, releases can again be increased to take advantage of the improved situation.

The simulation model takes as inputs the values of the processing parameters, the unit price π_t^i and deterministic demand D_t^i, the unit backorder cost s_t^i, the unit work in process (WIP) and finished inventory holding costs w_t^i and h_t^i, the raw materials costs r_t^i and a release vector R_t^i specifying the number of production lots of each product $i \in \{1, 2\}$ released at uniform intervals of Δ/R_t^i time units over the period t. Our performance measure is the total contribution, given by sales revenue minus the inventory holding, WIP holding and backordering costs, over the planning horizon. Let S_t^i denote the number of units of product i backlogged at the end of period t, W_t^i the work in progress (WIP) inventory of product i at the end of period t, I_t^i the amount of product i in finished goods inventory at the end of period t. The stochastic optimization problem we seek to solve can now be stated as follows:

$$\begin{aligned} \max_{\boldsymbol{R}=[R_t^i]} \quad & E_\xi \left[\sum_{t=1}^{T} \sum_{i=1}^{2} \left[\pi_t^i (D_t^i + S_{t-1}^i(\xi) - S_t^i(\xi)) - (r_t^i R_t^i + w_t^i W_t^i(\xi) + h_t^i I_t^i(\xi) + s_t^i S_t^i(\xi)) \right] \right] \\ \text{s.t.} \quad & R_t^i \geq 0 \end{aligned} \tag{2}$$

where the release quantities $\boldsymbol{R} = [R_t^i]$ are the vector of decision variables, D_t^i denotes the deterministic demand and $S_t^i(\xi), W_t^i(\xi)$, and $I_t^i(\xi)$ represent the random variables whose values are evaluated by the simulation model with the random seed ξ. Using Monte Carlo sampling (Shapiro 2003), we estimate the expected total contribution of a release vector with the sample average of the fixed number of replications, under the sample average approximation (SAA) framework (2015). Thus problem (2) can be rewritten as

$$\max_{\boldsymbol{R}=[R_t^i]} \; f(\boldsymbol{R}, m) := \frac{1}{m} \sum_{j=1}^{m} F(\boldsymbol{R}, \xi_j), \tag{3}$$

where $F(\boldsymbol{R}, \xi)$ is the stochastic simulation output, m the number of calls to the simulation (replications) at $\boldsymbol{R}$ and $f(\cdot, m)$ the sample average that is the new objective function.

Our initial experiments revealed that efficiently obtaining a globally optimal solution for this problem is challenging. A high degree of noise in the simulated outputs requires a large number of replications to obtain even moderately accurate estimates of the objective function value. This results in lengthy runs and numerically unstable gradient estimates, greatly compromising their usefulness within the optimization. We also encountered premature termination of sequential search procedures due to the presence of a large number of local optima for the highly non-convex objective function. An effective solution procedure for this problem must thus combine sufficient diversification to search a wide range of the solution space efficiently with sufficient intensification to ensure that the solutions are at least locally optimal. Our first choice for this study, building on our previous work for single stage systems (Manda and Uzsoy 2020), was a genetic algorithm (GA). However, in addition to the high computation time the GA requires to converge to a global optimum, the sample size used for sample averages may become insufficient to provide sufficiently precise estimates of the objective function values as the search approaches the critical region, further slowing convergence. We therefore also experiment with stochastic approximation (SA) with finite-differences where the step size is in the direction of the steepest descent with a pre-defined length a_k. The recursion at iteration k is then

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k - a_k \hat{\nabla} f(\boldsymbol{R}_k), \tag{4}$$

with a_k decreasing at a rate no faster than $1/k$, and $\hat{\nabla} f(\boldsymbol{R}_k)$ denoting the approximated gradient computed using finite differences at the incumbent solution $\boldsymbol{R}_k$. One can also approximate the gradient with

simultaneous perturbations (Spall et al. 1992), which can save some evaluations but results in slow convergence due to the added bias. Finally, using information from past iterations in addition to the most recent one can reduce the effect of the noise in the search direction (Byrd et al. 2016) through incorporating approximated Hessian values B_k that are updated separately, i.e.,

$$\boldsymbol{R}_{k+1} = \boldsymbol{R}_k - a_k \boldsymbol{B}_k^{-1} \hat{\nabla} f(\boldsymbol{R}_k), \tag{5}$$

where $\boldsymbol{B}_k$ denotes the approximated Hessian that can be updated with computationally cheap methods such as BFGS or its limited memory counterparts (Liu and Nocedal 1989). The GA and SA can be viewed as complementary approaches. Operating with a population of solutions, the GA can search a wide area of the solution space in each iteration, providing excellent diversification. However, once a promising region of the solution space has been identified, it may fail to exploit the region thoroughly. In contrast, SA searches the neighborhood of an incumbent solution intensively, but can fluctuate heavily due to variability in the gradient approximation and terminate in a local optimum far from the global optimum.

3.2 Solution Procedure

In each experiment, we first execute the simulation optimization algorithm of interest, henceforth solver, until termination with an initial solution $\boldsymbol{R}_0$ and m fixed random number streams (seeds) $\boldsymbol{\xi} = \{\xi_j\}_{j=1}^{m}$, a process known as common random numbers (Fu 2015), to reduce the variability in the optimization. In addition to the random number seeds for the simulation, we also specify those for the solver, due to its stochasticity, denoted by $\boldsymbol{\zeta} = \{\zeta_\ell\}_{\ell=1}^{L}$. Each random number seed for the solver constitutes a *macro-replication* that produces a single solution $\boldsymbol{R}_\ell^*(\boldsymbol{\xi})$. We thus report L solutions for each solver and compute the mean and 95% confidence interval of their evaluated objective function, assuming normality with sufficiently large L. The evaluated objective function uses a fresh set of seeds $\boldsymbol{\xi}' = \{\xi_\iota\}_{\iota=1}^{m'}$ to call the simulation at each reported solution and estimate $f(\boldsymbol{R}_\ell^*(\boldsymbol{\xi}), m')$ following the computation in (3), where m' is the number of post replication seeds. The purpose of this *post-processing* step is to avoid optimization bias in the original estimates at the solution points (Mak et al. 1999). The termination criterion in this paper is the maximum number of iterations, although one can also specify a maximum simulation budget or maximum clock time.

4 Numerical Experiments

We begin this section by providing some details of our implementation of the GA to the simulation optimization problem described in Section 3, for which we used the GA Toolbox in MATLAB. Following extensive preliminary experiments, we set the population size to 240, the elite count to 25 and the crossover probability to 0.7; mutation, scale and shrink and migration options are left at their default values. The release vectors for the initial population are generated by randomly perturbing the demand data (Manda and Uzsoy 2018), and the algorithm is terminated after 40 generations. Using the Parallel Computing Toolbox of Matlab to parallelize the function evaluations required for each chromosome on an Intel Xenon processor equipped system with 64 cores, one generation of the GA requires 18 minutes of CPU time with a population size of 200. Due to the high level of noise encountered in the simulation output, we use $m_1 = 30$ simulation replications to estimate the objective function value for each candidate solution corresponding to a release vector.

We performed 10 independent macro-replications of the GA with independent random number streams for each of the experiments. At the end of each GA run, the release vector obtained was subjected to the post-processing procedure to eliminate optimization bias. We report the average TC value for each experiment along with a confidence interval constructed using the results of the ten runs.

We consider a product transition in which demand for a mature product is replaced by that for a new product over a 15 period planning horizon. As each period represents four weeks, the total planning horizon is about two and a half years. The values of the learning parameters in (1) for the learning station are the same in both the experiments, with $Q^i(n) = 60, Q_0^2(n) = 5$, and $\alpha = 0.004$. Demand for Product 1 is

Table 1: Estimated Total Contribution in LBB and LAB Experiments for 10 macro-replications

	GA Result		RED Result	
Experiment	Mean	95% CI	Mean	95% CI
LAB	83510	[82900, 84120]	70223	[68681, 71765]
LBB	84127	[83884, 84370]	74060	[73009, 75111]

600 units in periods 1 through 5, 300 units in period 6, and 0 thereafter. Demand for Product 2 is 0 in periods 1 through 5, 300 in period 6, and 600 thereafter. Cost parameters for both products are identical and time-stationary with $\pi_t^i = 10$, $s_t^i = 5$, $w_t^i = 2$ and $h_t^i = 3$ for all $t = 1, \cdots, T$ and $i = 1, 2$.

To examine how the location of the learning station impacts the performance of the line, we conduct two experiments where the learning station n is located upstream of the bottleneck (Experiment 1) and downstream of it (Experiment 2). As a benchmark we use the Releases Equal to Demand (RED) policy, where the releases in each period are set equal to the demand for that period.

4.1 Experiment 1: Learning Station Before Bottleneck (LBB)

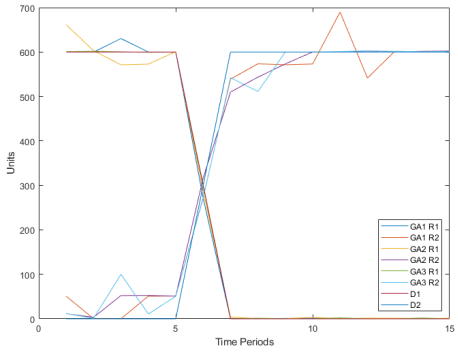
In this experiment, which we shall refer to as LBB, Product 2 causes disruptions at Station 2. The processing times for both products at all stations, except Stations 2 and 5, are lognormally distributed with mean $\mu^i(n) = 60$ minutes and standard deviation $\sigma^i(n) = 5$ minutes for $i = 1, 2$. Station 2 has a mean processing time of $\mu^1(2) = 50$ for Product 1 and $\mu^2(2) = 42$ minutes for Product 2 with standard deviation of 5 in both cases. On average $Q_t^2(2)$ units of Product 2 are processed between engineering holds at Station 2, and the average hold duration is $P^2(n) = 750$ minutes. Station 5 has a higher mean processing time of $\mu^i(5) = 60$ minutes, and an average utilization of 89.2% when the line is producing only Product 1 (in the early periods). When Product 2 is introduced, Station 2 becomes the bottleneck due to the engineering interventions. As production experience is gained, the value of $Q_t^2(2)$ decreases, eventually shifting the bottleneck back to Station 5 when $\mu^2(2)$ drops below 60 minutes. The average total contribution values and their 95% confidence intervals for the ten GA macro-replications and RED are given in Table 1. The results of three release vectors obtained by the GA are shown in Figure 1, and the average WIP at each station over the 30 replications for both m (in optimization) and m' (in post-processing) in Figure 2.

Under RED, the high processing time at Station 2 when Product 2 is introduced causes WIP accumulation as seen in Figure 1, causing high cycle time and reducing total contribution by increasing backorders. As the effective processing time improves with learning, throughput increases and WIP decreases. The GA solutions, in contrast, release small amounts of Product 2 early in the horizon to permit learning at Station 2, improving its effective processing time and greatly reducing WIP buildup as seen in Figure 1.

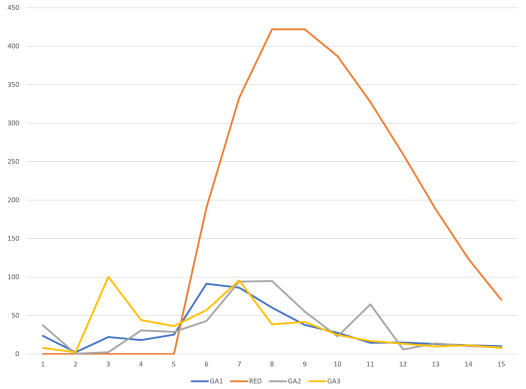
4.2 Experiment 2: Learning Station After Bottleneck (LAB)

In this experiment, referred to as LAB, the new product induces engineering holds at Station 10, downstream of the bottleneck, applying the parameter values for Station 2 in Experiment 1 to Station 10. The processing time distributions for the other stations remain as in LBB. The average total contribution values for ten GA macro-replications and RED are given in Table 1 and their releases and average WIP in Figure 1.

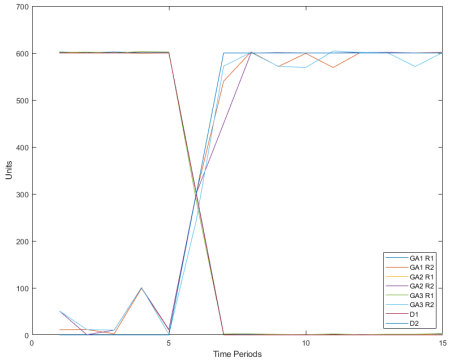
In this experiment RED has a higher TC and a smaller confidence interval than in LBB. When Product 2 is introduced, material moves through the line with very low variability until it reaches Station 10, while the disruptions at Station 10 affect only Station 11, which operates at low utilization. Hence, the average WIP is lower for RED in this experiment than in LBB. The TC values of the GA solutions are similar to those in LBB. In Figure 1(c), we see an increase in the releases for the new product in the GA solution to improve the effective processing time distribution. The similarity of the GA solutions for both experiments is somewhat surprising considering the location of the learning station is near the end of the line and yielded a better RED solution. The reasons for this behavior are examined in the next section.



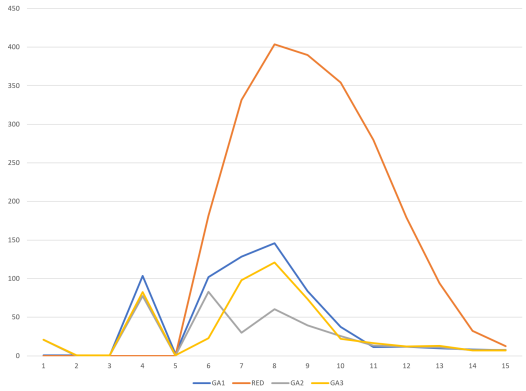
(a) Product Releases for LBB Experiment



(b) Average WIP at End of Period for LBB Experiment



(c) Product Releases for LAB Experiment



(d) Average WIP at End of Period for LAB Experiment

Figure 1: Release and WIP profiles for LBB and LAB experiments

4.3 Comparing the LBB and LAB Solutions

To examine the differences in behavior between LBB and LAB, the solutions for best GA solution fromm each experiment are simulated for both locations of the learning station. We plot the average throughput at key stations for both cases in Figure 2. In Experiment 1, throughput at Station 2, where the learning takes place, lags behind the releases when Product 2 is introduced in large quantities after period 5. As learning occurs, the throughput recovers and eventually exceeds releases, as the WIP buildup reduces idle time due to flow variability, before settling at a level equal to the releases. In Experiment 2, where the learning is at Station 10, throughput is equal to releases at Station 2, while Station 5 has the highest mean processing time once learning is complete. Thus in LBB, the high cycle time at Station 2, combined with the long processing time at Station 5, leads to loss of throughput at Station 5. In LAB, where material flow to Station 5 is unobstructed, throughput at Station 5 closely tracks the releases, except in period 1 where it falls slightly behind because of the high release quantity.

In LBB, Station 10's throughput remains consistent with that of Station 5. However, in LAB, where learning takes place at Station 10, we see a drop in throughput slightly greater than that observed at Station 2 in LBB. Due to the time it takes for the releases in a given period to reach Station 10, the time available at the station to produce throughput in a period decreases. Due to the exponential learning function this lost output results in slower improvement, leading, in turn, to more lost throughput. Thus the drop in throughput at Station 10 in LAB is higher than that at Station 2 in LBB.

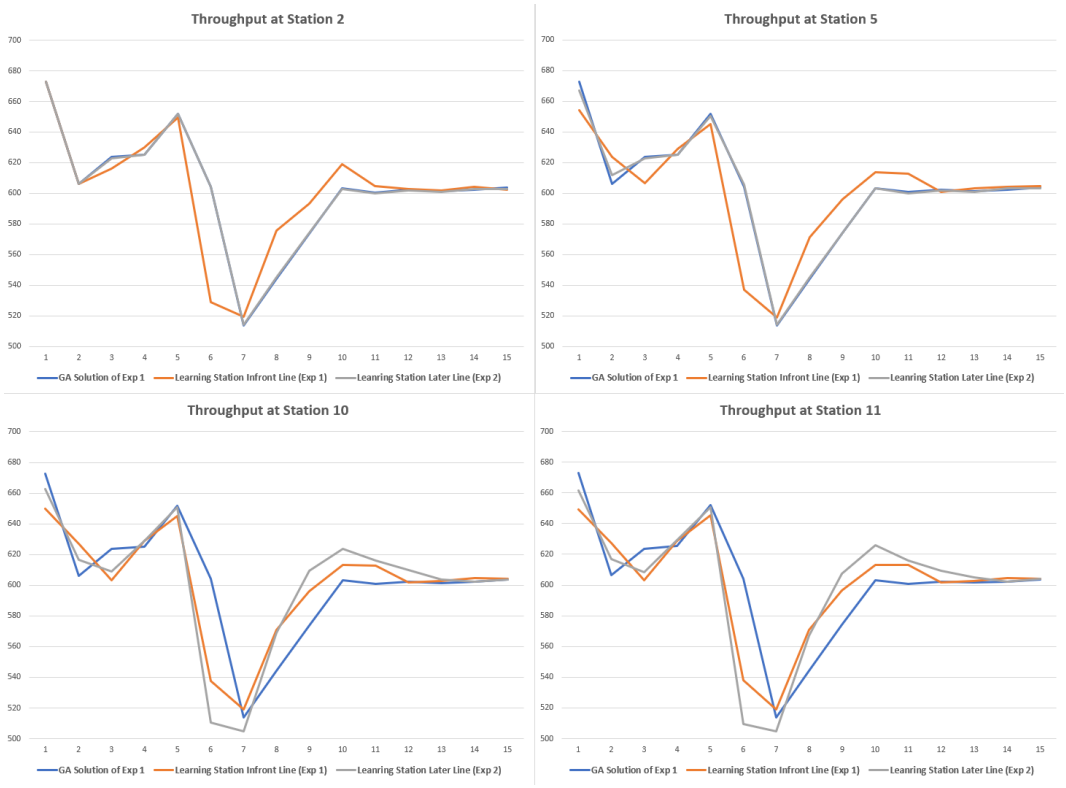


Figure 2: Average Throughput at Key Stations for LBB Solution

The GA solution from LBB expedites learning at Station 2 while avoiding overloading the system to an extent that downstream stations, especially Station 5, see an increase in cycle time. This is achieved by gradual releases of Product 2 in the early periods without exceeding the capacity of Station 5. This ensures that the throughput at Stations 2 and 5 (and thus the rest of the line) remains quite similar. Thus the impact of the variability at Station 2 on the downstream machines is reduced by managing the releases. However, fewer units reaching Station 10, so when the learning machine is at Station 10, instead of Station 2, this release vector results in lower total throughput and, ultimately, lower TC.

The reason the GA solution of LAB performs better in LAB than in LBB can be seen in Figure 3. The LAB solution seeks to push material through the system to speed up learning at Station 10. This is achieved by higher releases of Product 2 in period 4 compared to the more gradual releases in the LAB solution. In Experiment 2 with learning at Station 10, this does not cause any issues at Station 2 whose throughput is equal to releases. In Experiment 1, however, this sudden increase in releases causes throughput at Station 2 to lag behind releases. At Station 5, the drop in throughput is further exacerbated by the increased flow variability due to the engineering holds at Station 2, causing increased cycle time and WIP. In Experiment 2 the decrease in throughput at Station 5 is far lower as there is very little variability in its arrival process, allowing it to operate at high utilization without a drastic increase in cycle time. Thus, more units reach Station 10, accelerating learning. While throughput at Station 10 still lags releases, it performs better than LBB for this release vector. The LAB solution thus seeks to increase the throughput of Station 10 by maximizing the output of the bottleneck (Station 5) without excessive WIP cost. In contrast, when Station 2 is the learning station, increasing releases merely increases the WIP in the line without benefit to learning. Hence the LBB solution implements gradual releases of Product 2 to ensure that learning takes place with minimal impact on the downstream machines.

We can draw some interesting conclusions from these two simple experiments. Firstly, as might be expected, the location of the learning station in the line relative to the bottleneck has a strong impact

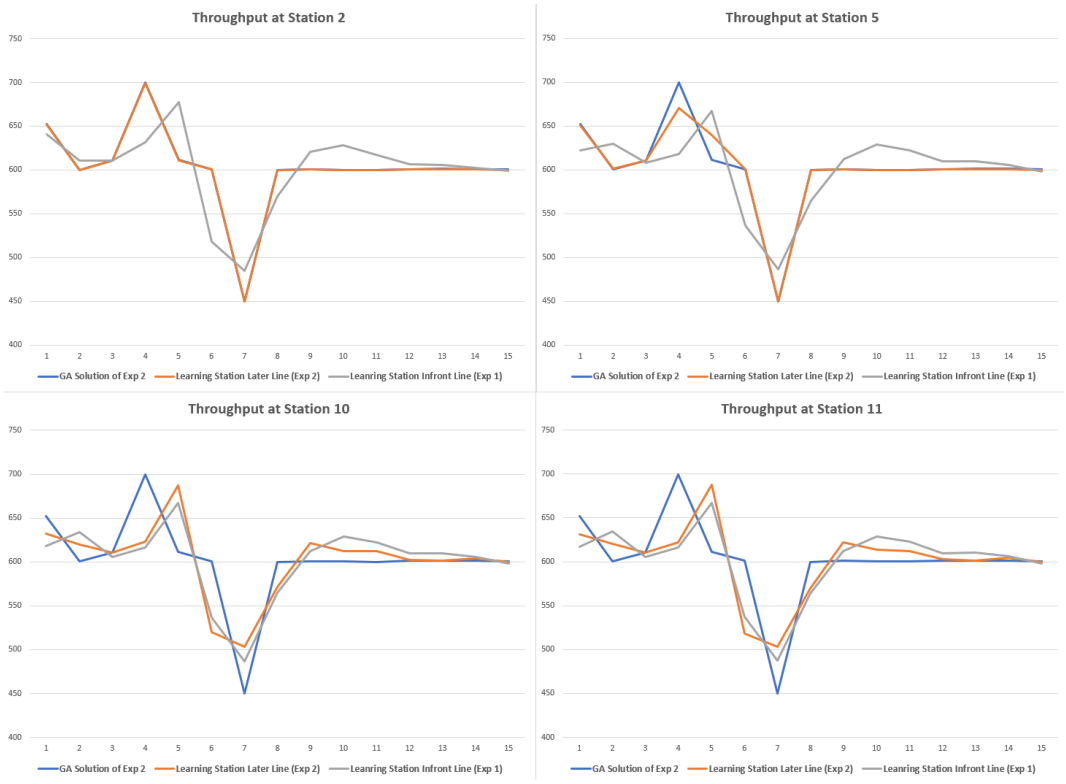


Figure 3: Average Throughput at Key Stations for LAB solution

on performance. Having the learning station early in the production line, upstream of the bottleneck, allows releases to easily reach the station, accelerating learning and improving the mean and variance of its throughput. However, a learning station early in the line induces flow variability that leads to higher WIP and longer cycle times at downstream stations, especially those operating at high utilization. A learning machine downstream in the line takes longer to learn, as it takes material longer to reach it after release into the line, especially if it must pass through a bottleneck.

Secondly, the results of the two experiments show that how releases should be managed depends on the location of the learning station in the line. Unlike in a single stage system, releases need to be made such that the cycle time of the line as a whole is kept in check. The complexity of accomplishing this in a wafer fab with reentrant product flows is immediately apparent.

5 Solutions with Stochastic Approximation

While the GA described above yields results consistent with those from our previous work, its computational requirements are quite substantial, to the point that it cannot be expected to scale up to larger fab simulations with re-entrant flows. Some of this computational burden arises from the fact that a GA, while able to quickly reach the neighborhood of a local optimum, often requires many iterations to converge to the local optimum itself, especially with significant noise in the simulation outputs used as the fitness function.

To address this issue we sought to use SA to intensify the search locally with multi-starts to explore the solution space. SA in its original form is a solver for continuous optimization problems and the decision space in this study is integer based. However, since the magnitude of the product release vectors is in the hundreds, we use a heuristic to enable the use of SA for this problem by scaling down the decision space by 1000, which then resembles a continuous space. Then by experimenting with a number of perturbation

Table 2: Multi-start SA mean and 95% confidence intervals across 10 macro-replications.

Initial point		1	2	3	4	5
LAB	Mean	82260	22135	47119	34145	36373
	C.I.	[81625, 82895]	[20938, 23332]	[44385, 49852]	[30528, 37762]	[35938, 36807]
LBB	Mean	82504	22011	31578	40842	32575
	C.I.	[81689, 83320]	[21148, 22875]	[29345, 33811]	[38215, 43469]	[32069, 33081]

sizes in the scaled-down decision space and rescaling back to the original, we choose the best perturbation size for this problem to be 3; that is,

$$[\hat{\nabla} f(\boldsymbol{R}_k, m)]_i = \frac{f(\boldsymbol{R}_k + \delta e_i, m) - f(\boldsymbol{R}_k - \delta e_i, m)}{2\delta},$$

where $[\hat{\nabla} f(\boldsymbol{R}_k, m)]_i$ is the i-th element of the approximated gradient and e_i is the i-th column of the identity matrix. The step size is a decreasing function with the iteration number of the form $a_k = 0.01k^{-0.7}$.

For a comparison of SA and GA, we execute the SA with 5 start points. Each initial point experiment starts with a randomly selected $\boldsymbol{R}_0$ and executes 10 macro-replications using the same seeds used in the GA. At each $\boldsymbol{R}_k$ during the optimization, the average of $m = 30$ replications estimates the objective function value, and the release vector at the final iteration of the SA for each macro-replication is evaluated with $m' = 30$ independent replications for post-processing. Both of these procedures use the exact same seeds used in the GA. Table (2) provides the mean and 95% C.I. for the LBB and LAB experiments; Learning Before Bottleneck (LBB) and Learning After Bottleneck (LAB). It is clear from the results that the SA can achieve objective value close to GA solution given a good starting point (Initial point 1). While we observe consistent decrease in the objective estimate, SA is only able to converge locally and within a single trajectory it does not explore the broader solution space. The behavior of the algorithm is susceptible to the small step size to avoid divergence. SA run time is competitive with GA, and good starting points, tuned step size and perturbation size as well as more curvature information in the form of an approximated Hessian are likely to speed the convergence and improve the solution quality. However, due to the derivative-free nature of finite-differences, the computational effort of SA increases with the dimension. Moving towards unbiased gradient estimators that are directly observed from the simulation, and adaptive coordinate updates in finite-difference setting can reduce the computational cost.

6 Conclusions and Future Directions

This work lays the foundation for planning releases for multi-stage production systems during product transitions. Our experiments show that the location of the learning station can have significant impact on the performance of the line, and that this impact can be mitigated by carefully managing releases.

This work revealed a number of problem characteristics that render the use of simulation optimization challenging. The problem is by its nature a finite-horizon, transient phenomenon, and high variability induced by the learning results requires many simulation replications to obtain accurate estimates of system performance for a given solution. The problem is highly non-convex, with many local optima, requiring a high level of diversification across the solution space for a near-optimal solution to be found. This imposes a high computational burden on any search-based optimization procedure.

This work represents an early stage of research in this area and opens up several directions of future work. A more detailed experimental design with different learning rates, costs, bottleneck utilization, and especially re-entrant product flows, is needed. The current paper focuses entirely on learning by experience; another direction for future work is to incorporate learning by experimentation through engineering lots, targeted at specific stations, and how different stations should be targeted optimally for improvement using engineering lots. An important direction for future work is the exploration of improved simulation optimization methods. While initial experiments with an SA method shows success, further experimentation is needed with different

parameter combinations to understand the applicability of the method for this problem. Another possible area of exploration is trust-region methods (Nocedal and Wright 2006). These methods define a region around the current iterate within which the model is considered to be an adequate representation of the the objective function and then choose the direction and length of the step simultaneously. A potential candidate in this class of methods is a derivative-based stochastic trust-region algorithm called Adaptive Sampling Trust-Region Optimization (ASTRO) (Vasquez et al.). This algorithm generates iterates that are guaranteed to converge almost surely, to a first-order or a second-order critical point of the objective function by incorporating adaptively sampled function and gradient estimates within a trust region framework.

REFERENCES

Anzanello, M. J., and F. S. Fogliatto. 2011. "Learning curve models and applications: Literature review and research directions". *International Journal of Industrial Ergonomics* 41(5):573–583.

Bilginer, Ö., and F. Erhun. 2010. "Managing product introductions and transitions". In *Wiley Encyclopedia of Operations Research and Management Science*, 1–12. Wiley Online Library.

Billington, C., H. L. Lee, and C. S. Tang. 1998. "Successful strategies for product rollovers". *Sloan Management Review* (Spring):23 – 30.

Byrd, R. H., S. L. Hansen, J. Nocedal, and Y. Singer. 2016. "A stochastic quasi-Newton method for large-scale optimization". *SIAM Journal on Optimization* 26(2):1008–1031.

Chand, S., H. Moskowitz, A. Novak, I. Rekhi, and G. Sorger. 1996. "Capacity allocation for dynamic process improvement with quality and demand considerations". *Operations Research* 44(6):964–975.

Crist, K., and R. Uzsoy. 2011. "Prioritising production and engineering lots in wafer fabrication facilities: a simulation study". *International Journal of Production Research* 49(11):3105–3125.

Ebert, R. J. 1976. "Aggregate planning with learning curve productivity". *Management Science* 23(2):171 – 182.

Fine, C. H. 1986. "Quality improvement and learning in productive systems". *Management Science* 32(10):1301–1315.

Fine, C. H., and E. L. Porteus. 1989. "Dynamic process improvement". *Operations Research* 37(4):580–591.

Fu, M. C. 2015. *Handbook of Simulation Optimization*. New York: Springer.

Haller, M., A. Peikert, and J. Thoma. 2003. "Cycle time management during production ramp-up". *Robotics and Computer-Integrated Manufacturing* 19(1-2):183–188.

Hiller, R. S., and J. F. Shapiro. 1986. "Optimal capacity expansion planning when there are learning effects". *Management Science* 32(9):1153 – 1163.

Kim, S., R. Pasupathy, and S. G. Henderson. 2015. "A guide to sample average approximation". In *Handbook of Simulation Optimization*, 207–243. Springer.

Kim, S., and R. Uzsoy. 2013. "Modeling and analysis of integrated planning of production and engineering process improvement". *IEEE Transactions on Semiconductor Manufacturing* 26(3):414–422.

Kim, S., and R. M. Uzsoy. 2008. "Integrated planning of production and engineering process improvement". *IEEE Transactions on Semiconductor Manufacturing* 21(3):390–398.

Leachman, R. C., and S. Ding. 2011. "Excursion yield loss and cycle time reduction in semiconductor manufacturing". *IEEE Transactions on Automation Science and Engineering* 8(1):112 – 116.

Liao, W. M. 1979. "Effects of learning on resource allocation decisions". *Decision Sciences* 10(1):116–125.

Lim, W. S., and C. S. Tang. 2006. "Optimal product rollover strategies". *European Journal of Operational Research* 174:905 – 922.

Liu, D. C., and J. Nocedal. 1989. "On the limited memory BFGS method for large scale optimization". *Mathematical Programming* 45(1-3):503–528.

Liu, J., C. Li, F. Yang, H. Wan, and R. Uzsoy. 2011. "Production planning for semiconductor manufacturing via simulation optimization". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 3612–3622. Phoenix, Arizona: Institute of Electrical and Electronics Engineers, Inc.

Mak, W.-K., D. P. Morton, and R. K. Wood. 1999. "Monte Carlo bounding techniques for determining solution quality in stochastic programs". *Operations Research Letters* 24(1-2):47–56.

Manda, A. B., and R. Uzsoy. 2018. "Optimizing releases during new product introductions". Technical report, Edward P. Fitts Department of Industrial and Systems Engineering, NCSU, Raleigh, NC 27519.

Manda, A. B., and R. Uzsoy. 2019. "Optimizing engineering and production lots during product transitions in semiconductor manufacturing". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K. H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y. J. Son, 2292–2303. National Harbor, Maryland: Institute of Electrical and Electronics Engineers, Inc.

Manda, A. B., and R. Uzsoy. 2020. "Optimizing releases during new product introductions". *IEEE Transactions on Semiconductor Manufacturing* 33(2):240 – 251.

Missbauer, H., and R. Uzsoy. 2020. *Production Planning with Capacitated Resources and Congestion*. New York: Springer.

Nelder, J. A., and R. Mead. 1965. "A simplex method for function minimization". *The Computer Journal* 7(4):308–313.

Nemoto, K., E. Akcali, and R. M. Uzsoy. 2000. "Quantifying the benefits of cycle time reduction in semiconductor wafer fabrication". *IEEE Transactions on Electronics Packaging Manufacturing* 23(1):39–47.

Nocedal, J., and S. Wright. 2006. *Numerical Optimization*. New York: Springer Science & Business Media.

Rash, E., and K. Kempf. 2012. "Product line design and scheduling at Intel". *Interfaces* 42(5):425–436.

Robbins, H., and S. Monro. 1951. "A stochastic approximation method". *The Annals of Mathematical Statistics*:400–407.

Salomon, R., and X. Martin. 2008. "Learning, knowledge transfer and technology implementation performance: A study of iime-to-build in the global semiconductor industry". *Management Science* 54(7):209–213.

Shapiro, A. 2003. "Monte Carlo sampling approach to stochastic programming". In *ESAIM: Proceedings*, Volume 13, 65–73. EDP Sciences.

Shashaani, S., F. S. Hashemi, and R. Pasupathy. 2018. "ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization". *SIAM Journal on Optimization* 28(4):3145–3176.

Spall, J. C. et al. 1992. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". *IEEE Transactions on Automatic Control* 37(3):332–341.

Terwiesch, C., and R. E. Bohn. 2001. "Learning and process improvement during production ramp-up". *International Journal of Production Economics* 70(1):1–19.

Terwiesch, C., and Y. Xu. 2004. "The copy-exactly ramp-up strategy: Trading-off learning with process change". *IEEE Transactions on Engineering Management* 51(1):70–84.

Tirkel, I. 2013. "Yield learning curve models in semiconductor manufacturing". *IEEE Transactions On Semiconductor Manufacturing* 26(4):564–571.

Vasquez, D., R. Pasupathy, and S. Shashaani. "Astro for derivative-based stochastic optimization: Algorithm description & numerical experiments". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K. H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y. J. Son, 3563–3574. National Harbor, Maryland: Institute of Electrical and Electronics Engineers, Inc.

Yelle, L. E. 1979. "The learning curve: Historical review and comprehensive survey". *Decision Sciences* 10(2):302–328.

AUTHOR BIOGRAPHIES

ATCHYUTA BHARADWAJ MANDA is a Doctoral student in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. He holds a Masters in Industrial and Systems Engineering from North Carolina State University, and a Bachelor of Technology degree in Mechanical Engineering from GITAM University. His research interests are in production systems, supply chain management and new product introduction, focusing on data oriented decision analysis using stochastic simulation. His e-mail address is amanda@ncsu.edu.

KARTHICK GOPALSWAMY is a research scientist at Walmart Labs. He holds Masters and Ph.D degress in Industrial and Systems Engineering from North Carolina State University. His research focuses on data oriented decision analysis, revenue optimization in retail and experimental design. His e-mail address is kgopals@ncsu.edu.

SARA SHASHAANI is an assistant professor in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. Her research interests are probabilistic data-driven modeling and simulation optimization. Her email address is sshasha2@ncsu.edu.

REHA UZSOY is Clifton A. Anderson Distinguished Professor in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. He holds BS degrees in Industrial Engineering and Mathematics and an MS in Industrial Engineering from Bogazici University, Istanbul, Turkey. He received his Ph.D in Industrial and Systems Engineering in 1990 from the University of Florida. His teaching and research interests are in production planning and supply chain management. His email address is ruzsoy@ncsu.edu.