Dynamic Product Assembly and Inventory Control for Maximum Profit

Michael J. Neely and Longbo Huang

***Abstract*— We consider a manufacturing plant that purchases raw materials for product assembly and then sells the final products to customers. There are M types of raw materials and K types of products, and each product uses a certain subset of raw materials for assembly. The plant operates in slotted time, and every slot it makes decisions about re-stocking materials and pricing the existing products in reaction to (possibly time-varying) material costs and consumer demands. We develop a dynamic purchasing and pricing policy that yields time average profit within ϵ of optimality, for any given $\epsilon > 0$, with a worst case storage buffer requirement that is $O(1/\epsilon)$. The policy can be implemented easily for large M, K, yields fast convergence times, and is robust to non-ergodic system dynamics.**

***Index Terms*— Queueing analysis, pricing, optimization**

I. Introduction

This paper considers the problem of maximizing time average profit at a product assembly plant. The plant manages the purchasing, assembly, and pricing of M types of raw materials and K types of products. Specifically, the plant maintains a storage buffer for each of the M materials, and can assemble each product from some specific combination of materials. The system operates in slotted time with normalized slots $t \in \{0, 1, 2, \ldots\}$. Every slot, the plant makes decisions about purchasing new raw materials and pricing the K products for sale to the consumer. This is done in reaction to material costs and consumer demand functions that are known on each slot but can change randomly from slot to slot according to a stationary process with a possibly unknown probability distribution.

It is well known that the problem of maximizing time average profit in such a system can be treated using dynamic programming and Markov decision theory. A textbook example of this approach for a single product (single queue) problem is given in [1], where inventory storage costs are also considered. However, such approaches may be prohibitively complex for problems with large dimension, as the state space grows exponentially with the number of queues. Further, these techniques require knowledge of the probabilities that govern purchasing costs and consumer demand functions. Case studies of multi-dimensional inventory control are treated in [2] using a lower complexity neuro-dynamic programming framework, which approximates the optimal value function used in traditional dynamic programming. Such algorithms fine-tune the parameters of the approximation by either offline simulations or online feedback (see also [3][4]).

In this paper, we consider a different approach that does not attempt to approximate dynamic programming. Our algorithm reacts to the current system state and does not require knowledge of the probabilities that affect future states. Under mild ergodicity assumptions on the material supply and consumer demand processes, we show that the algorithm can push time average profit to within ϵ of optimality, for any arbitrarily small value $\epsilon > 0$. This can be achieved by finite storage buffers of size cT_ϵ/ϵ, where c is a coefficient that is polynomial in K and M, and T_ϵ is a constant that depends on the "mixing time" of the processes. In the special case when these processes are i.i.d. over slots, we have $T_\epsilon = 1$ for all $\epsilon > 0$, and so the buffers are size $O(1/\epsilon)$. The algorithm can be implemented in real time even for problems with large dimension (i.e., large K and M). Thus, our framework circumvents the "curse of dimensionality" problems associated with dynamic programming. This is because we are not asking the same question that could be asked by dynamic programming approaches: Rather than attempting to maximize profit subject to finite storage buffers, we attempt to reach the more difficult target of pushing profit arbitrarily close to the maximum that can be achieved in systems with *infinite buffer space*. We can approach this optimality with finite buffers of size $O(1/\epsilon)$, although this may not be the optimal buffer size tradeoff. A dynamic program might be able to achieve the same profit with smaller buffers, but would contend with curse of dimensionality issues.

Prior work on inventory control with system models similar to our own is found in [5] [6] [7] and references therein. Work in [5] considers a single-dimensional inventory problem where a fixed number of products are sold over a finite horizon with a constant but unknown customer arrival rate. A set of coupled differential equations are derived for the optimal policy using Markov decision theory. Work in [6] provides structural results for multi-dimensional inventory problems with product assembly, again using Markov decision theory, and obtains numerical results for a two-dimensional system. A multi-dimensional product assembly problem is treated in [7] for stochastic customer arrivals with fixed and known rates. The complexity issue is treated by considering a large volume limit and using results of heavy traffic theory. The work in [7] also considers joint optimal price decisions, but chooses all prices at time zero and holds them constant for all time thereafter.

Our analysis uses the "drift-plus-penalty" framework of stochastic network optimization, developed for queueing networks in [8][9][10]. Our work is most similar to the work

Michael J. Neely and Longbo Huang are with the Electrical Engineering department at the University of Southern California, Los Angeles, CA. (web: http://www-rcf.usc.edu/∼mjneely and http://www-scf.usc.edu/∼longbohu).

This material is supported in part by the NSF Career grant CCF-0747525.

in [11], which uses this framework to address *processing networks* that queue components that must be combined with other components. The work in [11] treats multi-hop networks and maximizes throughput and throughput-utility in these systems using a *deficit max-weight* algorithm that uses "deficit queues" to keep track of the deficit created when a component cannot be processed due to a missing part. Our paper does not consider a multi-hop network, but has similar challenges when we do not have enough inventory to build a desired product. Rather than using deficit queues, we use a different type of Lyapunov function that avoids deficits entirely. Our formulation also considers the purchasing and pricing aspects of the problem, particularly for a manufacturing plant, and considers arbitrary (possibly non-ergodic) material supply and consumer demand processes.

Previous work in [12] uses the drift-plus-penalty framework in a related revenue maximization problem for a wireless service provider. In that context, a two-price result demonstrates that dynamic pricing must be used to maximize time average profit (a single price is often not enough, although two prices are sufficient). The problem in this paper can be viewed as the "inverse" of the service provider problem, and has an extra constraint that requires the plant to maintain enough inventory for a sale to take place.

The outline of this paper is as follows: In the next section we specify the system model. The optimal time average profit is characterized in Section III, where the two-price behavior is also noted. Our dynamic control policy is developed in Section IV for an i.i.d. model of material cost and consumer demand states, and extensions to more general ergodic and non-ergodic systems are considered in Section V.

II. System Model

There are M types of raw materials, and each is stored in a different storage buffer at the plant. Define $Q_m(t)$ as the (integer) number of type m materials in the plant on slot t. We temporarily assume all storage buffers have infinite space, and later we show that our solution can be implemented with finite buffers of size $O(1/\epsilon)$, where the ϵ parameter determines a profit-buffer tradeoff.

Let $\boldsymbol{Q}(t) = (Q_1(t), \ldots, Q_M(t))$ be the vector of queue sizes, also called the *inventory vector*. From these materials, the plant can manufacture K types of products. Define β_{mk} as the (integer) number of type m materials required for creation of a single item of product k (for $m \in \{1, \ldots, M\}$ and $k \in \{1, \ldots, K\}$). We assume that products are assembled quickly, so that a product requested during slot t can be assembled on the same slot, provided that there are enough raw materials.[1] Thus, the plant must have $Q_m(t) \geq \beta_{mk}$ for all $m \in \{1, \ldots, M\}$ in order to sell one product of type k on slot t, and must have twice this amount of materials in order to sell two type k products, etc. The simplest example is when each raw material itself represents a finished product, which corresponds to the case $K = M$, $\beta_{mm} = 1$ for all m, $\beta_{mk} = 0$ for $m \neq k$. However, our model allows for more complex assembly structures, possibly with different products requiring some overlapping materials.

Every slot t, the plant must decide how many new raw materials to purchase and what price it should charge for its products. Let $\boldsymbol{A}(t) = (A_1(t), \ldots, A_M(t))$ represent the vector of the (integer) number of new raw materials purchased on slot t. Let $\tilde{\boldsymbol{D}}(t) = (\tilde{D}_1(t), \ldots, \tilde{D}_K(t))$ be the vector of the (integer) number of products sold on slot t. The queueing dynamics for $m \in \{1, \ldots, M\}$ are thus:

$$Q_m(t+1) = \max\left[Q_m(t) - \sum_{k=1}^{K} \beta_{mk}\tilde{D}_k(t), 0\right] + A_m(t) \quad (1)$$

Below we describe the pricing decision model that affects product sales $\tilde{\boldsymbol{D}}(t)$, and the cost model associated with purchasing decisions $\boldsymbol{A}(t)$.

A. Product Pricing and the Consumer Demand Functions

For each slot t and each commodity k, the plant must decide if it desires to offer commodity k for sale, and, if so, what price it should charge. Let $Z_k(t)$ represent a binary variable that is 1 if commodity k is offered and is 0 else. Let $P_k(t)$ represent the per-unit price for product k on slot t. We assume that prices $P_k(t)$ are chosen within a compact set $\mathcal{P}_k$ of price options. Thus:

$$P_k(t) \in \mathcal{P}_k \ \ \forall k \in \{1, \ldots, K\}, \forall t \quad (2)$$

The sets $\mathcal{P}_k$ include only non-negative prices and have a finite maximum price $P_{k,max}$. For example, the set $\mathcal{P}_k$ might represent the interval $0 \leq p \leq P_{k,max}$, or might represent a discrete set of prices separated by some minimum price unit. Let $\boldsymbol{Z}(t) = (Z_1(t), \ldots, Z_K(t))$ and $\boldsymbol{P}(t) = (P_1(t), \ldots, P_K(t))$ be vectors of these decision variables.

Let $Y(t)$ represent the *consumer demand state* for slot t, which represents any factors that affect the expected purchasing decisions of consumers on slot t. Let $\boldsymbol{D}(t) = (D_1(t), \ldots, D_K(t))$ be the resulting *demand vector*, where $D_k(t)$ represents the (integer) amount of type k products that consumers want to buy in reaction to the current price $P_k(t)$ and under the current demand state $Y(t)$. Specifically, we assume that $D_k(t)$ is a random variable that depends on $P_k(t)$ and $Y(t)$, is conditionally i.i.d. over all slots with the same $P_k(t)$ and $Y(t)$ values, and satisfies:

$$F_k(p, y) = \mathbb{E}\left\{D_k(t) \mid P_k(t) = p, Y(t) = y\right\} \ \ \forall p \in \mathcal{P}_k, y \in \mathcal{Y} \quad (3)$$

The $F_k(p, y)$ function is assumed to be continuous in $p \in \mathcal{P}_k$ for each $y \in \mathcal{Y}$.[2] We assume that the current demand state $Y(t)$ is known to the plant at the beginning of slot t, and that the demand functions $F_k(p, y)$ are also known to the plant. The process $Y(t)$ takes values in a finite or countably infinite set $\mathcal{Y}$, and is assumed to be stationary and ergodic with steady state probabilities $\pi(y)$, so that:

$$\pi(y) = Pr[Y(t) = y] \ \ \forall y \in \mathcal{Y}, \forall t$$

[1] Algorithms that yield similar performance but require products to be assembled one slot before they are delivered can be designed based on simple modifications [13].

[2] Continuity is not crucial. It is assumed only for convenience to ensure the expression (26) has a maximum over $\mathcal{P}_k$, which is always true in the special case when $\mathcal{P}_k$ is a finite set of points (in which case the mathematical definition of continuity also holds trivially).

The probabilities $\pi(y)$ are not necessarily known to the plant.

We assume that the maximum demand for each product $k \in \{1, \ldots, K\}$ is deterministically bounded by a finite integer $D_{k,max}$, so that regardless of price $\boldsymbol{P}(t)$ or the demand state $Y(t)$, we have:

$$0 \leq D_k(t) \leq D_{k,max} \quad \text{for all slots } t \text{ and all products } k$$

This boundedness assumption is useful for analysis. Such a finite bound is natural in cases when the maximum number of customers is limited on any given slot. The bound might also be artificially enforced by the plant due to physical constraints that limit the number of orders that can be fulfilled on one slot. Define $\mu_{m,max}$ as the resulting maximum demand for raw materials of type m on a given slot:

$$\mu_{m,max} \triangleq \sum_{k=1}^{K} \beta_{mk} D_{k,max} \tag{4}$$

If there is a sufficient amount of raw materials to fulfill all demands in the vector $\boldsymbol{D}(t)$, and if $Z_k(t) = 1$ for all k such that $D_k(t) > 0$ (so that product k is offered for sale), then the number of products sold is equal to the demand vector: $\tilde{\boldsymbol{D}}(t) = \boldsymbol{D}(t)$. We are guaranteed to have enough inventory to meet the demands on slot t if $Q_m(t) \geq \mu_{m,max}$ for all $m \in \{1, \ldots, M\}$. However, there may not always be enough inventory to fulfill all demands, in which case we require a *scheduling decision* that decides how many units of each product will be assembled to meet a subset of the demands. The value of $\tilde{\boldsymbol{D}}(t) = (\tilde{D}_1(t), \ldots, \tilde{D}_K(t))$ must be chosen as an integer vector that satisfies the following *scheduling constraints*:

$$0 \leq \tilde{D}_k(t) \leq Z_k(t) D_k(t) \quad \forall k \in \{1, \ldots, K\} \tag{5}$$

$$Q_m(t) \geq \sum_{k=1}^{K} \beta_{mk} \tilde{D}_k(t) \quad \forall m \in \{1, \ldots, M\} \tag{6}$$

B. Raw Material Purchasing Costs

Let $X(t)$ represent the *raw material supply state* on slot t, which contains components that affect the purchase price of new raw materials. Specifically, we assume that $X(t)$ has the form:

$$X(t) = [(x_1(t), \ldots, x_M(t)); (s_1(t), \ldots, s_M(t))]$$

where $x_m(t)$ is the per-unit price of raw material m on slot t, and $s_m(t)$ is the maximum amount of raw material m available for sale on slot t. We assume that $X(t)$ takes values on some finite or countably infinite set $\mathcal{X}$, and that $X(t)$ is stationary and ergodic with probabilities:

$$\pi(x) = Pr[X(t) = x] \ \forall x \in \mathcal{X}, \forall t$$

The $\pi(x)$ probabilities are not necessarily known to the plant.

Let $c(\boldsymbol{A}(t), X(t))$ be the total cost incurred by the plant for purchasing a vector $\boldsymbol{A}(t)$ of new materials under the supply state $X(t)$:

$$c(\boldsymbol{A}(t), X(t)) = \sum_{m=1}^{M} x_m(t) A_m(t) \tag{7}$$

We assume that $\boldsymbol{A}(t)$ is limited by the constraint $\boldsymbol{A}(t) \in \mathcal{A}(X(t))$, where $\mathcal{A}(X(t))$ is the set of all vectors $\boldsymbol{A}(t) = (A_1(t), \ldots, A_M(t))$ such that for all t:

$$0 \leq A_m(t) \leq \min[A_{m,max}, s_m(t)] \quad \forall m \in \{1, \ldots, M\} \tag{8}$$

$$A_m(t) \text{ is an integer} \quad \forall m \in \{1, \ldots, M\} \tag{9}$$

$$c(\boldsymbol{A}(t), X(t)) \leq c_{max} \tag{10}$$

where $A_{m,max}$ and c_{max} are finite bounds on the total amount of each raw material that can be purchased, and the total cost of these purchases on one slot, respectively. These finite bounds might arise from the limited supply of raw materials, or might be artificially imposed by the plant in order to limit the risk associated with investing in new raw materials on any given slot. A simple special case is when there is a finite maximum price $x_{m,max}$ for raw material m at any time, and when $c_{max} = \sum_{m=1}^{M} x_{m,max} A_{m,max}$. In this case, the constraint (10) is redundant.

C. The Maximum Profit Objective

For each $k \in \{1, \ldots, K\}$, define α_k as a fixed (non-negative) cost associated with assembling one product of type k. Define a process $\phi(t)$ as follows:

$$\phi(t) \triangleq -c(\boldsymbol{A}(t), X(t)) + \sum_{k=1}^{K} Z_k(t) D_k(t) (P_k(t) - \alpha_k) \tag{11}$$

The value of $\phi(t)$ represents the total instantaneous profit due to material purchasing and product sales on slot t, under the assumption that all demands are fulfilled (so that $\tilde{D}_k(t) = D_k(t)$ for all k). Define $\phi_{actual}(t)$ as the *actual* instantaneous profit, defined by replacing the $D_k(t)$ values in the right hand side of (11) with $\tilde{D}_k(t)$ values. Note that $\phi(t)$ can be either positive, negative, or zero, as can $\phi_{actual}(t)$.

Define time average expectations $\overline{\phi}$ and $\overline{\phi}_{actual}$ as follows:

$$\overline{\phi} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\phi(\tau)\}, \ \overline{\phi}_{actual} \triangleq \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\phi_{actual}(\tau)\}$$

Every slot t, the plant observes the current queue vector $\boldsymbol{Q}(t)$, the current demand state $Y(t)$, and the current supply state $X(t)$, and chooses a purchase vector $\boldsymbol{A}(t) \in \mathcal{A}(X(t))$ and pricing vectors $\boldsymbol{Z}(t)$, $\boldsymbol{P}(t)$ (with $Z_k(t) \in \{0, 1\}$ and $P_k(t) \in \mathcal{P}_k$ for all $k \in \{1, \ldots, K\}$). The consumers then react by generating a random demand vector $\boldsymbol{D}(t)$ with expectations given by (3). The actual number of products filled is scheduled by choosing the $\tilde{\boldsymbol{D}}(t)$ vector according to the scheduling constraints (5)-(6), and the resulting queueing update is given by (1). The goal of the plant is to maximize the time average expected profit $\overline{\phi}_{actual}$. For convenience, a table of notation is given in Table I.

III. Characterizing Maximum Time Average Profit

Assume infinite buffer capacity (so that $Q_{m,max} = \infty$ for all $m \in \{1, \ldots, M\}$). Consider any control algorithm that makes decisions for $\boldsymbol{Z}(t)$, $\boldsymbol{P}(t)$, $\boldsymbol{A}(t)$, and also makes scheduling decisions for $\tilde{\boldsymbol{D}}(t)$, according to the system structure as described in the previous section. Define ϕ^{opt} as the

TABLE I
TABLE OF NOTATION

Notation	Definition
$X(t)$	Supply state, $\pi(x) = Pr[X(t) = x]$
$\boldsymbol{A}(t) = (A_1(t), \ldots, A_M(t))$	Raw materials purchased on slot t
$c(\boldsymbol{A}(t), X(t))$	Raw material cost function
$\mathcal{A}(X(t))$	Constraint set for variables $\boldsymbol{A}(t)$
$Y(t)$	Consumer demand state
$\boldsymbol{Z}(t) = (Z_1(t), \ldots, Z_K(t))$	0/1 sale vector
$\boldsymbol{P}(t) = (P_1(t), \ldots, P_K(t))$	Price vector, $P_k(t) \in \mathcal{P}_k$
$\boldsymbol{D}(t) = (D_1(t), \ldots, D_K(t))$	Random demand vector
$F_k(p, y)$	Demand function
$\boldsymbol{Q}(t) = (Q_1(t), \ldots, Q_M(t))$	Queue vector of raw materials
$Q_{m,max}$	Maximum buffer size for queue m
α_k	Cost of assembly for product type k
β_{mk}	# materials m used for product k
$\phi(t)$	Instantaneous profit (given by (11))
$\boldsymbol{\mu}(t) = (\mu_1(t), \ldots, \mu_M(t))$	Departures of raw materials (see (15))
$\tilde{\boldsymbol{D}}(t), \tilde{\boldsymbol{\mu}}(t)$	Actual demands and departures
$\phi_{actual}(t)$	Actual instantaneous profit for slot t

maximum time average profit over all such algorithms, so that all algorithms must satisfy $\overline{\phi}_{actual} \leq \phi^{opt}$, but there exist algorithms that can yield profit arbitrarily close to ϕ^{opt}. The value of ϕ^{opt} is determined by the steady state distributions $\pi(x)$ and $\pi(y)$, the cost function $c(\boldsymbol{A}(t), X(t))$, and the demand functions $F_k(p, y)$.

It turns out that ϕ^{opt} can be characterized by a sub-class of stationary randomized algorithms. Specifically, we say that a policy is (X, Y)-*only* if it chooses $\boldsymbol{P}(t)$, $\boldsymbol{Z}(t)$, $\boldsymbol{A}(t)$ values as a stationary and randomized function only of the current observed $X(t)$ and $Y(t)$ states.

Theorem 1: There exists an (X, Y)-only policy $\boldsymbol{P}^*(t)$, $\boldsymbol{Z}^*(t)$, $\boldsymbol{A}^*(t)$ such that:

$$\mathbb{E}\{\phi^*(t)\} = \phi^{opt} \quad (12)$$

$$\mathbb{E}\{A_m^*(t)\} = \mathbb{E}\{\mu_m^*(t)\} \, \forall m \in \{1, \ldots, M\} \quad (13)$$

where ϕ^{opt} is the optimal time average profit, and where $\mathbb{E}\{\phi^*(t)\}$ and $\mathbb{E}\{\mu_m^*(t)\}$ are given by:

$$\mathbb{E}\{\phi^*(t)\} = -\mathbb{E}\{c(\boldsymbol{A}^*(t), X(t))\} + \sum_{k=1}^{K} \mathbb{E}\{Z_k^*(t)(P_k^*(t) - \alpha_k)F_k(P_k^*(t), Y(t))\}$$

$$\mathbb{E}\{\mu_m^*(t)\} = \sum_{k=1}^{K} \beta_{mk}\mathbb{E}\{Z_k^*(t)F_k(P_k^*(t), Y(t))\} \ \forall m$$

where the expectations are with respect to the stationary probability distributions $\pi(x)$ and $\pi(y)$ for $X(t)$ and $Y(t)$, and the (potentially randomized) decisions for $\boldsymbol{A}^*(t), \boldsymbol{Z}^*(t), \boldsymbol{P}^*(t)$ that depend on $X(t)$ and $Y(t)$.

Proof: See [13]. □

Note that Theorem 1 contains no variables for the scheduling decisions for $\tilde{\boldsymbol{D}}(t)$, made subject to (5)-(6). Such scheduling decisions allow choosing $\tilde{\boldsymbol{D}}(t)$ in reaction to the demands $\boldsymbol{D}(t)$, and hence allow more flexibility beyond the choice of the $\boldsymbol{Z}(t)$ and $\boldsymbol{P}(t)$ variables alone (which must be chosen before the demands $\boldsymbol{D}(t)$ are observed). That such additional scheduling options cannot be exploited to increase time average profit is a consequence of our proof of Theorem 1. It can be shown that the expectations in Theorem 1 can be achieved by algorithms that use at most 2 prices per demand state $Y(t)$ (such as a "regular" price and a reduced "sale" price) [13]. This is similar to the 2-price result observed in [12] for the case of a service provider with a single queue. Simple examples can be given to show that two prices are often *necessary* to achieve maximum time average profit, even when consumer demand states $Y(t)$ are the same for all slots (see [12] for a simple example for the related service-provider problem).

IV. A DYNAMIC PRICING AND PURCHASING ALGORITHM

Here we construct a dynamic algorithm that makes purchasing and pricing decisions in reaction to the current queue sizes and the observed $X(t)$ and $Y(t)$ states, without knowledge of the $\pi(x)$ and $\pi(y)$ probabilities that govern the evolution of these states. We begin with the assumption that $X(t)$ is i.i.d. over slots with probabilities $\pi(x) = Pr[X(t) = x]$, and $Y(t)$ is i.i.d. over slots with $\pi(y) = Pr[Y(t) = y]$. Section V extends to more general non-i.i.d. processes.

Define $1_k(t)$ as an indicator variable that is 1 if and only if $Q_m(t) < \mu_{m,max}$ for some queue m such that $\beta_{mk} > 0$ (so that type m raw material is used to create product k). To begin, let us choose an algorithm from the restricted class of algorithms that choose $\boldsymbol{Z}(t)$ values to satisfy the following *edge constraint* every slot t:

$$\forall k, \text{ we have: } Z_k(t) = 0 \text{ whenever } 1_k(t) = 1 \quad (14)$$

That is, the edge constraint (14) ensures that no type k product is sold unless inventory in all of its corresponding raw material queues m is at least $\mu_{m,max}$. Under this restriction, we always have enough raw material for any generated demand vector, and so $\tilde{D}_k(t) = D_k(t)$ for all products k and all slots t. Thus, from (11) we have $\phi_{actual}(t) = \phi(t)$. Define $\mu_m(t)$ as the number of material queue m departures on slot t:

$$\mu_m(t) \triangleq \sum_{k=1}^{K} \beta_{mk} Z_k(t) D_k(t) \quad (15)$$

The queueing dynamics of (1) thus become:

$$Q_m(t+1) = Q_m(t) - \mu_m(t) + A_m(t) \quad (16)$$

The above equation continues to assume we have infinite buffer space, but we soon show that we need only a finite buffer to implement our solution.

A. Lyapunov Drift

For a given set of non-negative parameters $\{\theta_m\}$ for $m \in \{1, \ldots, M\}$, define the non-negative *Lyapunov function* $L(\boldsymbol{Q}(t))$ as follows:

$$L(\boldsymbol{Q}(t)) \triangleq \frac{1}{2} \sum_{m=1}^{M} (Q_m(t) - \theta_m)^2 \quad (17)$$

This Lyapunov function is similar to that used for stock trading problems in [14][15], and has the flavor of keeping queue

backlog near a non-zero value θ_m, as in [16]. Define the conditional Lyapunov drift $\Delta(\boldsymbol{Q}(t))$ as follows:[3]

$$\Delta(\boldsymbol{Q}(t)) \triangleq \mathbb{E}\left\{L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t)) \mid \boldsymbol{Q}(t)\right\} \quad (18)$$

Define a constant $V > 0$, to be used to affect the revenue-storage tradeoff. Using the stochastic optimization technique of [8], our approach is to design a strategy that, every slot t, observes current system conditions $\boldsymbol{Q}(t)$, $X(t)$, $Y(t)$ and makes pricing and purchasing decisions to minimize a bound on the following "drift-plus-penalty" expression:

$$\Delta(\boldsymbol{Q}(t)) - V\mathbb{E}\left\{\phi(t) \mid \boldsymbol{Q}(t)\right\}$$

where $\phi(t)$ is the instantaneous profit function defined in (11).

B. Computing the Drift

We have the following lemma.

Lemma 1: (Drift Computation) Under any algorithm that satisfies the edge constraint (14), and for any constants $V \geq 0$, $\theta_m \geq 0$ for $m \in \{1, \ldots, M\}$, the Lyapunov drift $\Delta(\boldsymbol{Q}(t))$ satisfies:

$$\Delta(\boldsymbol{Q}(t)) - V\mathbb{E}\left\{\phi(t)|\boldsymbol{Q}(t)\right\} \leq B - V\mathbb{E}\left\{\phi(t)|\boldsymbol{Q}(t)\right\} + \sum_{m=1}^{M}(Q_m(t) - \theta_m)\mathbb{E}\left\{A_m(t) - \mu_m(t) \mid \boldsymbol{Q}(t)\right\} \quad (19)$$

where the constant B is defined:

$$B \triangleq \frac{1}{2}\sum_{m=1}^{M}\max[A_{m,max}^2, \mu_{m,max}^2] \quad (20)$$

Proof: The edge constraint (14) ensures that the dynamics (16) hold for all t. By subtracting θ_m from (16) and squaring the result, we have:

$$(Q_m(t+1) - \theta_m)^2 = (Q_m(t) - \theta_m)^2 + (A_m(t) - \mu_m(t))^2 + 2(Q_m(t) - \theta_m)(A_m(t) - \mu_m(t))$$

Dividing by 2, summing over $m \in \{1, \ldots, M\}$, and taking conditional expectations yields:

$$\Delta(\boldsymbol{Q}(t)) = \mathbb{E}\left\{B(t)|\boldsymbol{Q}(t)\right\} + \sum_{m=1}^{M}(Q_m(t) - \theta_m)\mathbb{E}\left\{A_m(t) - \mu_m(t)|\boldsymbol{Q}(t)\right\}$$

where $B(t)$ is defined:

$$B(t) \triangleq \frac{1}{2}\sum_{m=1}^{M}(A_m(t) - \mu_m(t))^2 \quad (21)$$

By the finite bounds on $A_m(t)$ and $\mu_m(t)$, we clearly have $B(t) \leq B$ for all t. □

[3]Strictly speaking, we should use the notation $\Delta(\boldsymbol{Q}(t), t)$, as the drift may be non-stationary. However, we use the simpler notation $\Delta(\boldsymbol{Q}(t))$ as a formal representation of the right hand side of (18).

Now note from the definition of $\mu_m(t)$ in (15) that:

$$\begin{aligned}
&\mathbb{E}\left\{\mu_m(t) \mid \boldsymbol{Q}(t)\right\} \\
&= \mathbb{E}\left\{\sum_{k=1}^{K}\beta_{mk}Z_k(t)D_k(t) \mid \boldsymbol{Q}(t)\right\} \\
&= \sum_{k=1}^{K}\beta_{mk}\mathbb{E}\left\{\mathbb{E}\left\{Z_k(t)D_k(t) \mid \boldsymbol{Q}(t), P_k(t), Y(t)\right\} \mid \boldsymbol{Q}(t)\right\} \\
&= \sum_{k=1}^{K}\beta_{mk}\mathbb{E}\left\{Z_k(t)F_k(P_k(t), Y(t)) \mid \boldsymbol{Q}(t)\right\} \quad (22)
\end{aligned}$$

where we have used the law of iterated expectations in the final equality. Similarly, we have from the definition of $\phi(t)$ in (11):

$$\mathbb{E}\left\{\phi(t) \mid \boldsymbol{Q}(t)\right\} = -\mathbb{E}\left\{c(\boldsymbol{A}(t), X(t)) \mid \boldsymbol{Q}(t)\right\} + \sum_{k=1}^{K}\mathbb{E}\left\{Z_k(t)(P_k(t) - \alpha_k)F_k(P_k(t), Y(t)) \mid \boldsymbol{Q}(t)\right\} \quad (23)$$

Plugging (22) and (23) into (19) yields:

$$\begin{aligned}
&\Delta(\boldsymbol{Q}(t)) - V\mathbb{E}\left\{\phi(t) \mid \boldsymbol{Q}(t)\right\} \leq B \\
&+ \sum_{m=1}^{M}(Q_m(t) - \theta_m)\Bigg[\mathbb{E}\left\{A_m(t)|\boldsymbol{Q}(t)\right\} \\
&- \sum_{k=1}^{K}\beta_{mk}\mathbb{E}\left\{Z_k(t)F_k(P_k(t), Y(t)) \mid \boldsymbol{Q}(t)\right\}\Bigg] \\
&+ V\mathbb{E}\left\{c(\boldsymbol{A}(t), X(t)) \mid \boldsymbol{Q}(t)\right\} \\
&- V\mathbb{E}\left\{\sum_{k=1}^{K}Z_k(t)(P_k(t) - \alpha_k)F_k(P_k(t), Y(t)) \mid \boldsymbol{Q}(t)\right\} \quad (24)
\end{aligned}$$

In particular, the right hand side of (24) is identical to the right hand side of (19).

C. The Dynamic Purchasing and Pricing Algorithm

Minimizing the right hand side of (24) over all feasible choices of $\boldsymbol{A}(t)$, $\boldsymbol{Z}(t)$, $\boldsymbol{P}(t)$ (given the observed $X(t)$ and $Y(t)$ states and the observed queue values $\boldsymbol{Q}(t)$) yields the following algorithm:

Joint Purchasing and Pricing Algorithm (JPP): Every slot t, perform the following actions:

1) *Purchasing:* Observe $\boldsymbol{Q}(t)$ and $X(t)$, and choose $\boldsymbol{A}(t) = (A_1(t), \ldots, A_M(t)) \in \mathcal{A}(X(t))$ as the minimizer of the following expression:

$$Vc(\boldsymbol{A}(t), X(t)) + \sum_{m=1}^{M}A_m(t)(Q_m(t) - \theta_m) \quad (25)$$

2) *Pricing:* Observe $\boldsymbol{Q}(t)$ and $Y(t)$. For each product $k \in \{1, \ldots, K\}$, if $1_k(t) = 1$, choose $Z_k(t) = 0$ and do not offer product k for sale. If $1_k(t) = 0$, choose $P_k(t) \in \mathcal{P}_k$ as the maximizer of the following expression:

$$V(P_k(t) - \alpha_k)F_k(P_k(t), Y(t)) + F_k(P_k(t), Y(t))\sum_{m=1}^{M}\beta_{mk}(Q_m(t) - \theta_m) \quad (26)$$

If the above maximized expression is positive, set $Z_k(t) = 1$ and keep $P_k(t)$ as the maximizing value. Else, set $Z_k(t) = 0$ and do not offer product k for sale.

3) *Queue Update:* Fulfill all demands $D_k(t)$, and update the queues $Q_m(t)$ according to (16) (noting by construction of this algorithm that $\tilde{\boldsymbol{D}}(t) = \boldsymbol{D}(t)$ for all t, so that the dynamics (16) are equivalent to (1)).

The above JPP algorithm does not require knowledge of probability distributions $\pi(x)$ or $\pi(y)$, and is decoupled into separate policies for pricing and purchasing. The pricing policy is quite simple and involves maximizing a (possibly non-concave) function of one variable $P_k(t)$ over the 1-dimensional set $\mathcal{P}_k$. For example, if $\mathcal{P}_k$ is a discrete set of 1000 price options, this involves evaluating the function over each option and choosing the maximizing price. The purchasing policy is more complex, as $\boldsymbol{A}(t)$ is an integer vector that must satisfy (8)-(10). This is a *knapsack problem* due to the constraint (10). However, the decision is trivial in the case when $c_{max} = \sum_{m=1}^{M} x_{m,max} A_{m,max}$, in which case the constraint (10) is redundant.

D. Deterministic Queue Bounds

We have the following simple lemma that shows the above policy can be implemented on a finite buffer system.

Lemma 2: (Finite Buffer Implementation) If initial inventory satisfies $Q_m(0) \leq Q_{m,max}$ for all $m \in \{1, \ldots, M\}$, where $Q_{m,max} \triangleq \theta_m + A_{m,max}$, then JPP yields $Q_m(t) \leq Q_{m,max}$ for all slots $t \geq 0$ and all queues $m \in \{1, \ldots, M\}$.

Proof: Fix a queue m, and suppose that $Q_m(t) \leq Q_{m,max}$ for some slot t (this holds by assumption at $t = 0$). We show that $Q_m(t+1) \leq Q_{m,max}$. To see this, note that because the function $c(\boldsymbol{A}(t), X(t))$ in (7) is non-decreasing in every entry of $\boldsymbol{A}(t)$ for each $X(t) \in \mathcal{X}$, the purchasing policy (25) yields $A_m(t) = 0$ for any queue m that satisfies $Q_m(t) > \theta_m$. It follows that $Q_m(t)$ cannot increase if it is greater than θ_m, and so $Q_m(t+1) \leq \theta_m + A_{m,max}$ (because $A_{m,max}$ is the maximum amount of increase for queue m on any slot). □

The following important related lemma shows that queue sizes $Q_m(t)$ are always above $\mu_{m,max}$, provided that they start with at least this much raw material and that the θ_m values are chosen to be sufficiently large. Specifically, define θ_m as follows:

$$\theta_m \triangleq \max_{\{k \in \{1,\ldots,K\} | \beta_{mk} > 0\}} \nu_{mk} \tag{27}$$

where

$$\nu_{mk} \triangleq \frac{V(P_{k,max} - \alpha_k)}{\beta_{mk}} + 2\mu_{max} + \sum_{i \in \{1,\ldots,M\}, i \neq m} \frac{\beta_{ik} A_{i,max}}{\beta_{mk}}$$

Lemma 3: Suppose that θ_m is defined by (27), and that $Q_m(0) \geq \mu_{m,max}$ for all $m \in \{1, \ldots, M\}$. Then for all slots $t \geq 0$ we have:

$$Q_m(t) \geq \mu_{m,max} \ \ \forall m \in \{1, \ldots, M\}$$

Proof: Fix $m \in \{1, \ldots, M\}$, and suppose that $Q_m(t) \geq \mu_{m,max}$ for some slot t (this holds by assumption for $t = 0$). We prove it also holds for slot $t+1$. If $Q_m(t) \geq 2\mu_{m,max}$, then $Q_m(t+1) \geq \mu_{m,max}$ (because at most $\mu_{m,max}$ units can depart queue m on any slot), and hence we are done. Suppose now that $\mu_{m,max} \leq Q_m(t) < 2\mu_{m,max}$. In this case the pricing functional in (26) satisfies the following for any product k such that $\beta_{mk} > 0$:

$$\begin{aligned} & V(P_k(t) - \alpha_k) F_k(P_k(t), Y(t)) \\ & \quad + F_k(P_k(t), Y(t)) \sum_{i=1}^{M} \beta_{ik}(Q_i(t) - \theta_i) \\ \leq \ & F_k(P_k(t), Y(t)) \times \Big[V(P_{k,max} - \alpha_k) \\ & \quad + \sum_{i \in \{1,\ldots,M\}, i \neq m} \beta_{ik} A_{i,max} + \beta_{mk}(2\mu_{m,max} - \theta_m) \Big] \\ \leq \ & 0 \end{aligned}$$

where the first inequality uses the fact that $(Q_i(t) - \theta_i) \leq A_{i,max}$ for all queues i (and in particular for all $i \neq m$) by Lemma 2. The final inequality uses the definition of θ_m in (27). It follows that the pricing rule (26) sets $Z_k(t) = 0$ for all products k that use raw material m, and so no departures can take place from $Q_m(t)$ on the current slot. Thus: $\mu_{m,max} \leq Q_m(t) \leq Q_m(t+1)$, and we are done. □

E. Performance Analysis of JPP

Theorem 2: (Performance of JPP) Suppose that θ_m is defined by (27) for all $m \in \{1, \ldots, M\}$, and that $\mu_{m,max} \leq Q_m(0) \leq \theta_m + A_{m,max}$ for all $m \in \{1, \ldots, M\}$. Suppose $X(t)$ and $Y(t)$ are i.i.d. over slots. Then under the JPP algorithm implemented with any parameter $V > 0$, we have:

(a) For all $m \in \{1, \ldots, M\}$ and all slots $t \geq 0$:

$$\mu_{m,max} \leq Q_m(t) \leq Q_{m,max} \triangleq \theta_m + A_{m,max}$$

where $Q_{m,max} = O(V)$.

(b) For all slots $t > 0$ we have:

$$\frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\phi_{actual}(\tau)\} \geq \phi^{opt} - \frac{B}{V} - \frac{\mathbb{E}\{L(\boldsymbol{Q}(0))\}}{Vt} \tag{28}$$

where the constant B is defined in (20) and is independent of V, and where ϕ^{opt} is the optimal time average profit defined in Theorem 1.

(c) The time average profit converges with probability 1, and satisfies:

$$\lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \phi_{actual}(\tau) \geq \phi^{opt} - \frac{B}{V} \ \text{(with probability 1)} \tag{29}$$

Thus, the time average profit is within $O(1/V)$ of optimal, and hence can be pushed arbitrarily close to optimal by increasing V, with a tradeoff in the maximum buffer size that is $O(V)$. Defining $\epsilon = B/V$ yields the desired $[O(\epsilon), O(1/\epsilon)]$ profit-buffer size tradeoff.

Proof: (Theorem 2 parts (a) and (b)) Part (a) follows immediately from Lemmas 2 and 3. To prove part (b), note that JPP observes $\boldsymbol{Q}(t)$ and makes control decisions for $Z_k(t)$, $\boldsymbol{A}(t)$, $P_k(t)$ that minimize the right hand side of (19) under any alternative choices. Thus:

$$\begin{aligned} \Delta(\boldsymbol{Q}(t)) - V\mathbb{E}\{\phi(t) \mid \boldsymbol{Q}(t)\} \leq B - V\mathbb{E}\{\phi^*(t) | \boldsymbol{Q}(t)\} \\ + \sum_{m=1}^{M} (Q_m(t) - \theta_m)\mathbb{E}\{A_m^*(t) - \mu_m^*(t) | \boldsymbol{Q}(t)\} \end{aligned} \tag{30}$$

where $\mathbb{E}\{\phi^*(t)|\boldsymbol{Q}(t)\}$, and $\mathbb{E}\{\mu_m^*(t)|\boldsymbol{Q}(t)\}$ correspond to any alternative choices for the decision variables $Z_k^*(t)$, $P_k^*(t)$, $A_m^*(t)$ subject to the same constraints, being that $P_k^*(t) \in \mathcal{P}_k$, $A_m^*(t)$ satisfies (8)-(10), and $Z_k^*(t) \in \{0,1\}$, and $Z_k^*(t) = 0$ whenever $1_k(t) = 1$. Because $Q_m(t) \geq \mu_{m,max}$ for all m, we have $1_k(t) = 0$ for all $k \in \{1,\ldots,K\}$. Thus, the (X,Y)-only policy of Theorem 1 satisfies the desired constraints. Further, this policy makes decisions based only on $(X(t),Y(t))$, which are i.i.d. over slots and hence independent of the current queue state $\boldsymbol{Q}(t)$. Thus, from (12)-(13) we have:

$$\begin{aligned}\mathbb{E}\{\phi^*(t)|\boldsymbol{Q}(t)\} &= \mathbb{E}\{\phi^*(t)\} = \phi^{opt} \\ \mathbb{E}\{A_m^*(t) - \mu_m^*(t)|\boldsymbol{Q}(t)\} &= \mathbb{E}\{A_m^*(t) - \mu_m^*(t)\} = 0 \ \forall m\end{aligned}$$

Plugging the above two identities into the right hand side of (30) yields:

$$\Delta(\boldsymbol{Q}(t)) - V\mathbb{E}\{\phi(t) \mid \boldsymbol{Q}(t)\} \leq B - V\phi^{opt} \quad (31)$$

Taking expectations of the above and using the definition of $\Delta(\boldsymbol{Q}(t))$ yields:

$$\mathbb{E}\{L(\boldsymbol{Q}(t+1))\} - \mathbb{E}\{L(\boldsymbol{Q}(t))\} - V\mathbb{E}\{\phi(t)\} \leq B - V\phi^{opt}$$

The above holds for all slots $t \geq 0$. Summing over $\tau \in \{0,1,\ldots,t-1\}$ for some integer $t > 0$ yields:

$$\mathbb{E}\{L(\boldsymbol{Q}(t))\} - \mathbb{E}\{L(\boldsymbol{Q}(0))\} - V\sum_{\tau=0}^{t-1}\mathbb{E}\{\phi(\tau)\} \leq Bt - Vt\phi^{opt}$$

Dividing by tV, rearranging terms, and using non-negativity of $L(\boldsymbol{Q}(t))$ yields:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{\phi(\tau)\} \geq \phi^{opt} - \frac{B}{V} - \frac{\mathbb{E}\{L(\boldsymbol{Q}(0))\}}{Vt} \quad (32)$$

Because $1_k(t) = 0$ for all k and all τ, we have $\phi(\tau) = \phi_{actual}(\tau)$ for all τ, and we are done. □

Proof: (Theorem 2 part (c)) Taking a limit of (32) proves that:

$$\liminf_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{\phi(\tau)\} \geq \phi^{opt} - B/V$$

The above result made no assumption on the initial distribution of $\boldsymbol{Q}(0)$. Thus, letting $\boldsymbol{Q}(0)$ be any particular initial state, we have:

$$\liminf_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{\phi(\tau)|\boldsymbol{Q}(0)\} \geq \phi^{opt} - B/V \quad (33)$$

However, under the JPP algorithm the process $\boldsymbol{Q}(t)$ is a discrete time Markov chain with finite state space (because it is an integer valued vector with finite bounds given in part (a)). Suppose now the initial condition $\boldsymbol{Q}(0)$ is a recurrent state. It follows that the time average of $\phi(t)$ must converge to a well defined constant $\overline{\phi}(\boldsymbol{Q}(0))$ with probability 1 (where the constant may depend on the initial recurrent state $\boldsymbol{Q}(0)$ that is chosen). Further, because $\phi(\tau)$ is bounded above and below by finite constants for all τ, by the Lebesgue dominated convergence theorem we have:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{\phi(\tau)|\boldsymbol{Q}(0)\} = \overline{\phi}(\boldsymbol{Q}(0))$$

Using this in (33) yields:

$$\overline{\phi}(\boldsymbol{Q}(0)) \geq \phi^{opt} - B/V$$

This is true for all initial states that are recurrent. If $\boldsymbol{Q}(0)$ is a transient state, then $\boldsymbol{Q}(t)$ eventually reaches a recurrent state and hence achieves a time average that is, with probability 1, greater than or equal to $\phi^{opt} - B/V$. □

F. Place-Holder Values

Theorem 2 seems to require the initial queue values to satisfy $Q_m(t) \geq \mu_{m,max}$ for all t. This suggests that we need to purchase that many raw materials before start-up. Here we use the *place holder backlog* technique of [17] to show that we can achieve the same performance without this initial start-up cost, without loss of optimality. The technique also allows us to reduce our maximum buffer size requirement $Q_{m,max}$ by an amount $\mu_{m,max}$ for all $m \in \{1,\ldots,M\}$, with no loss in performance.

To do this, we start the system off with exactly $\mu_{m,max}$ units of *fake raw material* in each queue $m \in \{1,\ldots,M\}$. Let $Q_m(t)$ be the total raw material in queue m, including both the actual and fake material. Let $Q_m^{actual}(t)$ be the amount of actual raw material in queue m. Then for slot 0 we have:

$$Q_m(0) = Q_m^{actual}(0) + \mu_{m,max} \ \ \forall m \in \{1,\ldots,M\}$$

Assume that $\mu_{m,max} \leq Q_m(0) \leq \theta_m + A_{m,max}$ for all $m \in \{1,\ldots,M\}$, as needed for Theorem 2. Thus, $0 \leq Q_m^{actual}(0) \leq \theta_m + A_{m,max} - \mu_{m,max}$ (so that the actual initial condition can be 0). We run the JPP algorithm as before, using the $\boldsymbol{Q}(t)$ values (not the actual queue values). However, if we are ever asked to assemble a product, we use *actual* raw materials whenever possible. The only problem comes if we are asked to assemble a product for which there are not enough actual raw materials available. However, we know from Theorem 2 that the queue value $Q_m(t)$ never decreases below $\mu_{m,max}$ for any $m \in \{1,\ldots,M\}$. It follows that we are *never* asked to use any of our fake raw material. Therefore, the fake raw material stays untouched in each queue for all time, and we have:

$$Q_m(t) = Q_m^{actual}(t) + \mu_{m,max} \ \ \forall m \in \{1,\ldots,M\}, \forall t \geq 0 \quad (34)$$

The sample path of the system is equivalent to a sample path that does not use fake raw material, but starts the system in the non-zero initial condition $\boldsymbol{Q}(0)$. Hence, the resulting profit achieved is the same. Because the limiting time average profit does not depend on the initial condition, the time average profit is still at least $\phi^{opt} - B/V$. However, by (34) the actual amount of raw material held is reduced by exactly $\mu_{m,max}$ on each slot, which also reduces the maximum buffer size $Q_{m,max}$ by exactly this amount.

G. Demand-Blind Pricing

As in [12] for the service provider problem, here we consider the special case when the demand function $F_k(P_k(t),Y(t))$ has the form:

$$F_k(P_k(t),Y(t)) = h_k(Y(t))\hat{F}_k(P_k(t)) \quad (35)$$

for some non-negative functions $h_k(Y(t))$ and $\hat{F}_k(P_k(t))$. This holds, for example, when $Y(t)$ represents the integer number of customers at time t, and $\hat{F}_k(p)$ is the expected demand at price p for each customer, so that $F_k(P_k(t), Y(t)) = Y(t)\hat{F}_k(P_k(t))$. Under the structure (35), the JPP pricing algorithm (26) reduces to choosing $P_k(t) \in \mathcal{P}_k$ to maximize:

$$V(P_k(t) - \alpha_k)\hat{F}_k(P_k(t)) + \hat{F}_k(P_k(t)) \sum_{m=1}^{M} \beta_{mk}(Q_m(t) - \theta_m)$$

Thus, the pricing can be done without knowledge of the demand state $Y(t)$.

V. Arbitrary Supply and Demand Processes

The JPP algorithm provides similar performance for non-i.i.d. processes with a mild "decaying memory" property [13]. Here we consider its performance without any probabilistic assumptions on $X(t)$ and $Y(t)$, where efficiency is measured against a class of ideal T-slot lookahead policies that have perfect knowledge of the future over frames of T slots. Let T be a positive integer and let $t_0 \geq 0$. Define $\phi_T(t_0)$ as the optimal expected profit achievable over the interval $\{t_0, t_0 + 1, ..., t_0 + T - 1\}$ by any policy that has complete knowledge of the $X(t)$ and $Y(t)$ values over this interval and that ensures the quantity of raw materials purchased over the interval is equal to the expected amount consumed. Note here that although the future $X(t)$, $Y(t)$ values are assumed to be known, the random demands $D_k(t)$ are still unknown. Mathematically, $\phi_T(t_0)$ can be defined as the solution to the following optimization problem (with variables $A_m(\tau)$, $Z_k(\tau), P_k(\tau)$):

(**P1**)

$$\max: \quad \phi_T(t_0) = \sum_{\tau=t_0}^{t_0+T-1} \mathbb{E}\left\{\phi(\tau) \mid \hat{H}_T(t_0)\right\}$$

$$s.t.: \quad \sum_{\tau=t_0}^{t_0+T-1} \left[A_m(\tau) - \sum_{k=1}^{K} \beta_{mk} Z_k(\tau) F_k(P_k(\tau), Y(\tau))\right] = 0 \ \forall m \in \{1, \ldots, M\}$$

Constraints $(2), (8), (9), (10)$.

Here $\hat{H}_T(t_0) \triangleq [(X(\tau), Y(\tau))]_{\tau=t_0}^{t_0+T-1}$; $\phi(\tau)$ is defined in (11) as the instantaneous profit obtained at time τ; $A_m(\tau)$ and $\sum_{k=1}^{K} \beta_{mk} Z_k(\tau) F_k(P_k(\tau), Y(\tau))$ are the number of newly ordered parts and the expected number of consumed parts in time τ, respectively; and the expectation is taken over the randomness in $D_k(\tau)$ due to the fact that the demand at time τ is a random variable. Since purchasing no materials and selling no products over the entire interval is a valid policy, we see that the value $\phi_T(t_0) \geq 0$ for all t_0 and all T.

In the following, we look at the performance of JPP over the interval from 0 to $JT - 1$, which is divided into a total of J frames with length T each. We show that for any $J > 0$, the JPP algorithm yields an average profit over $\{0, 1, ..., JT - 1\}$ that is close to the profit obtained with an optimal T-slot lookahead policy implemented on each T-slot frame.

A. Performance of JPP under arbitrary supply and demand

Theorem 3: Suppose the Joint Purchasing and Pricing Algorithm (JPP) is implemented, with θ_m satisfying condition (27), and that $\mu_{m,max} \leq Q_m(0) \leq \theta_m + A_{m,max}$ for all $m \in \{1, 2, ..., M\}$. Then for any arbitrary $X(t)$ and $Y(t)$ processes, the queue backlog values satisfies part (a) of Theorem 2. Moreover, for any positive integers J and T, and any $\hat{H}_{JT}(0)$ (which specifies all $X(\tau)$, $Y(\tau)$ values for $\tau \in \{0, 1, \ldots, JT - 1\}$), the time average profit over the interval $\{0, 1, ..., JT - 1\}$ satisfies:

$$\frac{1}{JT} \sum_{\tau=0}^{JT-1} \mathbb{E}\left\{\phi(\tau) \mid \hat{H}_{JT}(0)\right\} \geq$$

$$\frac{1}{JT} \sum_{j=0}^{J-1} \phi_T(jT) - \frac{BT}{V} - \frac{\mathbb{E}\left\{L(\boldsymbol{Q}(0)) | \hat{H}_{JT}(0)\right\}}{VJT}.$$

where $\phi_T(jT)$ is defined to be the optimal value of the problem (**P1**) over the interval $\{jT, ..., (j+1)T - 1\}$. The constant B is defined in (20).

Proof: See [13]. □

References

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control, vols. 1 and 2*. Athena Scientific, Belmont, Mass, 1995.

[2] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis. A neuro-dynamic programming approach to retailer inventory management. *Proc. of 36th Conf. on Decision and Control, San Diego*, Dec. 1997.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Mass, 1996.

[4] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007.

[5] Y. Aviv and A. Pazgal. Pricing of short life-cycle products through active learning. *working paper*, October 2002.

[6] S. Benjaafar and M. ElHafsi. Production and inventory control of a single product assemble-to-order system with multiple customer classes. *Management Science*, vol. 52, no. 12, pp. 1896-1912, December 2006.

[7] E. L. Plambeck and A. R. Ward. Optimal control of a high-volume assemble-to-order system. *Mathematics of Operations Research*, vol. 31, no. 3, pp. 453-477, August 2006.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1-149, 2006.

[9] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proc. IEEE INFOCOM*, March 2005.

[10] M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915-2934, July 2006.

[11] L. Jiang and J. Walrand. Stable and utility-maximizing scheduling for stochastic processing networks. *Allerton Conference on Communication, Control, and Computing*, 2009.

[12] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic communication system. *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 406-419, April 2010.

[13] M. J. Neely and L. Huang. Dynamic product assembly and inventory control for maximum profit. *ArXiv Technical Report, arXiv:1004.0479v1*, April 2010.

[14] M. J. Neely. Stock market trading via stochastic network optimization. *ArXiv Technical Report*, arXiv:0909.3891v1, Sept. 2009.

[15] M. J. Neely. Stock market trading via stochastic network optimization. *Proc. IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010.

[16] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *IEEE Transactions on Information Theory*, vol. 53, no. 9, pp. 3095-3113, Sept. 2007.

[17] M. J. Neely and R. Urgaonkar. Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage. *Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA*, Oct. 2008.