http://pubsonline.informs.org/journal/opre/

OPERATIONS RESEARCH
Vol. 66, No. 4, July–August 2018, pp. 1040–1057
ISSN 0030-364X (print), ISSN 1526-5463 (online)

The Benefits of State Aggregation with Extreme-Point Weighting for Assemble-to-Order Systems

Emre Nadar,[a] **Alp Akcay,**[b] **Mustafa Akan,**[c] **Alan Scheller-Wolf**[c]

[a] Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey; [b] School of Industrial Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, Netherlands; [c] Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Contact: emre.nadar@bilkent.edu.tr, http://orcid.org/0000-0002-9904-4243 (EN); a.e.akcay@tue.nl, http://orcid.org/0000-0003-2000-6816 (AA); akan@andrew.cmu.edu, http://orcid.org/0000-0002-1664-4113 (MA); awolf@andrew.cmu.edu, http://orcid.org/0000-0001-6871-2360 (AS-W)

Received: June 13, 2015
Revised: September 5, 2016; April 1, 2017
Accepted: November 15, 2017
Published Online in Articles in Advance: July 24, 2018

Subject Classifications: inventory/production: multi-item/echelon/stage; dynamic programming/optimal control: Markov; inventory/production: approximations
Area of Review: Operations and Supply Chains

https://doi.org/10.1287/opre.2017.1710

Abstract. We provide a new method for solving a very general model of an assemble-to-order system: multiple products, multiple components that may be demanded in different quantities by different products, batch production, random lead times, and lost sales, modeled as a Markov decision process under the discounted cost criterion. A *control policy* specifies when a batch of components should be produced and whether an arriving demand for each product should be satisfied. As optimal solutions for our model are computationally intractable for even moderately sized systems, we approximate the optimal cost function by reformulating it on an *aggregate* state space and restricting each aggregate state to be represented by its extreme original states. Our aggregation drastically reduces the value iteration computational burden. We derive an upper bound on the distance between aggregate and optimal solutions. This guarantees that the value iteration algorithm for the original problem initialized with the aggregate solution converges to the optimal solution. We also establish the optimality of a *lattice-dependent base-stock and rationing* policy in the aggregate problem when certain product and component characteristics are incorporated into the aggregation/disaggregation schemes. This enables us to further alleviate the value iteration computational burden in the aggregate problem by eliminating suboptimal actions. Leveraging all of our results, we can solve the aggregate problem for systems of up to 22 components, with an average distance of 11.09% from the optimal cost in systems of up to 4 components (for which we could solve the original problem to optimality).

Funding: The authors thank the National Science Foundation [Grants CMMI 1351821 and CMMI 1334194], Carnegie Mellon University, Bilkent University, and Eindhoven University of Technology for financial support.
Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2017.1710.

Keywords: assemble-to-order systems • Markov decision processes • approximate dynamic programming • aggregation

1. Introduction

Assemble-to-order (ATO) systems appear in many industries where rapid delivery of customized products or multi-item orders plays a vital role; they are particularly popular in the automotive, consumer electronics, and online retailing industries. (See Kapuscinski et al. 2004, Xu et al. 2009, and Lu et al. 2015, for examples.) ATO production holds inventory at the component level so that a product may be assembled from its components—if sufficient inventory exists—immediately after customer demand for this product occurs. This strategy allows the producer to offer greater product variety by providing flexibility in the use of potentially scarce components.

Despite the popularity of ATO systems in industry and a vast literature (see survey papers by Song and Zipkin 2003 and Atan et al. 2017, for example), knowledge of ATO inventory management is largely limited to (i) optimal policies for *specific* restricted product structures such as the "*M*" or "*W*" systems and (ii) performance evaluation and optimization techniques for *heuristic* policies for general problems. In this study we present a new approach to optimizing *general* ATO systems by approximating the problem via *aggregation* of the state space. Numerical experiments reveal the practicality of our aggregation method, extending current knowledge of (ii). Our aggregation of the state space also facilitates analytical treatment of the general ATO problem, which is notoriously difficult when defined on the original state space. Structural results that we obtain from the aggregate problem appear to extend current knowledge of (i).

We model the problem as an infinite-horizon Markov decision process (MDP) under the expected total discounted cost criterion. Each component is produced in batches of fixed size in a make-to-stock

fashion; production times are independent and have an Erlang distribution. Demand for each product arrives as an independent Poisson process; if not satisfied immediately, these demands are lost. The state space of the problem consists of the on-hand inventory level and production status for each component. A control policy specifies when to produce a batch of any component and whether to fulfill an arriving demand if sufficient inventory exists. Solving this ATO problem to optimality is extremely problematic even under exponential production times since the state space is unmanageably large; see Nadar et al. (2016). We develop an effective, and computationally efficient, aggregation method to reduce the problem's state space. This enables us to solve instances with up to 22 components. The average distance from the optimal cost is 11.09% in instances with up to four components (for which we could solve the original problem to optimality).

Our aggregation method first partitions the original state space into disjoint subsets such that each subset consists of the same number of adjacent original states; see Definition 1. Each subset forms an "aggregate" state in the aggregate problem. We then formulate the optimality equation for the aggregate problem. Since an aggregate state can be represented by its original states with varying degrees of importance, the controller disaggregates the aggregate state into its original states according to a specific probability distribution. The disaggregated system is then again subject to the admissible action space of the original problem and moves between original states as in the original problem. However, the controller determines the optimal actions based on the cost function evaluated not in those original states but in the corresponding *aggregate* states.

The probability distribution used to disaggregate an aggregate state has the potential to greatly influence the performance of the aggregate problem. We introduce a rule that disaggregates each aggregate state into its two extreme original states (the smallest and the largest): our *extreme point* policy (see Definition 2). This rule proves very effective in our numerical experiments: if we take the average percentage deviation from optimal cost across all states as our performance criterion, the average distance of our aggregate solution from the optimal solution is 6.32% under the extreme point policy, on a test bed of 180 instances. Inspired by Rogers and Plante (1993), we also consider an alternative rule that assigns equal disaggregation probabilities to all original states of an aggregate state: the *uniform* policy (see Definition 3). The average distance of our aggregate solution from the optimal solution is 5.97% under the uniform policy, on the same test bed. However, if we take the percentage deviations from optimal cost across all states *weighted by optimal stationary distribution*, the average distances are 10.3% and 12.86%, respectively.

While our disaggregation rule is comparable to the uniform rule with respect to optimal cost, it is significantly better computationally: on our test bed, the computation time of the aggregate problem under our rule is, on average, 96.4% lower than that of the original problem when solved by value iteration. The corresponding reduction under the uniform rule is, on average, 11.2%.

We then derive a finite error bound for the cost function of our aggregate problem, regardless of the disaggregation rule; see Theorem 1. This bound enables us to prove that a value iteration algorithm that starts with the cost function of our aggregate problem converges to the optimal cost function of the original problem. If such an algorithm is implemented on our test bed under the extreme point policy, the *maximum* distance from the optimal cost across all states drops to 10%, 5%, and 1% with, on average, 76%, 64%, and 45% fewer iterations, respectively, compared with the standard value iteration algorithm.

We also establish the optimality of a lattice-dependent base-stock and lattice-dependent rationing (LBLR) policy (defined on the original states with positive disaggregation probabilities) in the aggregate problem when certain component and product characteristics are incorporated into the aggregation/disaggregation schemes (see Theorem 2): for any configuration of component orders, the original state space of the *on-hand* inventory levels (with positive disaggregation probabilities) can be partitioned into disjoint lattices such that, on each lattice, (a) it is optimal to produce a component if and only if the current original state is less than the base-stock level of that component on the current lattice; and (b) it is optimal to fulfill a demand for a product if and only if the current original state is no less than the rationing level for that product on the current lattice. We use this optimal policy structure to further reduce the computational burden of the value iteration algorithm in the aggregate problem via action elimination. This procedure reduces the already-low computation time of the aggregate problem under the extreme point policy by a further 56.8%, on average, on a different test bed of 90 instances (each satisfying the conditions that ensure the optimality of LBLR).

We thus contribute to the ATO literature in several ways:

- To our knowledge, we are the first to study state aggregation in the optimization of ATO systems, highlighting the practicality of our aggregate solution with respect to both accuracy and computation time under the *extreme point* policy.
- We find a finite error bound for our aggregate solution, under *general* disaggregation, in *general* ATO

systems. This allows us to validate the use of a value iteration method that starts with our aggregate solution in the original problem. This method converges faster than the standard value iteration method according to our numerical experiments.

• We identify the optimal policy structure in the aggregate problem of *general* ATO systems by incorporating certain product and component characteristics into aggregation/disaggregation schemes. Our aggregation method renders the state space separable into multiple disjoint lattices so that well-known threshold policies (base-stock and rationing) are optimal on each of those lattices. This allows us to restrict the search for an optimal policy to the class of LBLR policies in the aggregate problem, dramatically reducing the computational burden.

• Our aggregate problem includes the original problem of ATO *assembly* systems as a special case. Prior research established the optimal policy structure for ATO assembly systems with *exponential* production times. We characterize the optimal policy structure for these systems by allowing for less variable and thus often more realistic Erlang production times.

The rest of the paper is organized as follows: Section 2 reviews the related literature. Section 3 describes the original problem. Section 4 describes the aggregate problem along with two possible disaggregation rules. Section 5 offers an error bound for the aggregate problem and proves the convergence of the value iteration algorithm that starts with the aggregate solution. Section 6 establishes the optimal policy structure in the aggregate problem under certain aggregation/disaggregation rules. Section 7 presents our numerical results. Section 8 offers a summary and concludes. All proofs are contained in an online appendix.

2. Related Literature

As mentioned in the introduction, ATO systems have received much attention in the literature; Song and Zipkin (2003) and Atan et al. (2017) provide comprehensive reviews. Several authors have identified the optimal and/or asymptotically optimal policy structure for managing inventory in very specific ATO systems: Lu et al. (2010) establish that no-holdback (NHB) allocation rules are optimal for generalized W-systems operating under an independent base-stock (IBS) replenishment policy when the "symmetric cost" condition holds. Doğru et al. (2010) prove that an IBS policy and an NHB allocation policy with a priority-based backorder clearing (PBC) rule are optimal for W-systems with identical component lead times when the "symmetric cost" or "balanced capacity" condition holds. Reiman and Wang (2012) extend the results of Doğru et al. (2010) to generalized W-systems with symmetric costs when all unique components have the same lead time, which is longer than that of the common component. Lu et al. (2015) show that a coordinated base-stock (CBS) policy and NHB allocation rules are optimal for N- and W-systems with nonidentical replenishment lead times and symmetric costs. They also reveal the asymptotic optimality of a CBS policy and an NHB policy with a PBC rule under high demand volume and asymmetric costs.

For ATO systems with identical component lead times, Reiman and Wang (2015) establish the asymptotic optimality of a stochastic program–based allocation rule and an IBS policy as the lead time goes to infinity. Wan and Wang (2015) prove the asymptotic optimality of the allocation rule in Reiman and Wang (2015) under high demand volume, showing that stock reservation is necessary for asymptotic optimality in many systems. Doğru et al. (2017) leverage the allocation rule of Reiman and Wang (2015) to generate results for M-systems: stock reservation is asymptotically optimal if the inventory cost of the assembled product exceeds the sum of those of individual products, and an NHB policy with a myopic priority rule is asymptotically optimal otherwise.

For ATO systems with lost sales, optimality results exist under Markovian assumptions on production and demand: Benjaafar and ElHafsi (2006) establish the optimality of a state-dependent base-stock and state-dependent rationing (SBSR) policy for an ATO assembly system with multiple demand classes. ElHafsi (2009) extends this result to customer orders that arrive as a compound Poisson process. ElHafsi et al. (2008) prove that an SBSR policy is optimal for an ATO system with a nested product structure. Benjaafar et al. (2011) show that an SBSR policy is optimal for an ATO assembly system with multiple stages, each producing a different item in batches of variable sizes. And Nadar et al. (2014) establish that an LBLR policy is optimal for generalized M-systems. Nadar et al. (2016) *numerically* demonstrate the optimal performance of LBLR for ATO systems with general product structures.

Despite the great interest from both academia and industry, and despite the suggestive results in Nadar et al. (2016), the optimal policy structure is still unknown in general ATO systems. For example, for systems with general product structures, even under assumptions of binary component requirements and identical (finite) replenishment lead times, *no* optimal policy structure has been proved. One reason for this is that such an optimal policy needs to address both replenishment and allocation issues for arbitrary numbers of interdependent components and products.

Consequently, several papers focus on heuristic policies: Akçay and Xu (2004) demonstrate the practicality of an order-based component allocation rule in a periodic-review ATO system with response time windows and the IBS policy. They optimize the base-stock

levels based on sample average approximation (SAA). Huang (2014) evaluates the use of last-come, first-served (within one period) and product-based priority (within time windows) rules for component allocation in a periodic-review ATO system with the IBS policy. Several other papers study the optimization of IBS or independent (s, S) policies in continuous-review ATO systems with the first-come, first-served (FCFS) allocation rule (Lu and Song 2005, Lu et al. 2005, Zhao and Simchi-Levi 2006, van Jaarsveld and Dollevoet 2011). Finally, van Jaarsveld and Scheller-Wolf (2015) develop an SAA algorithm that computes near-optimal base-stock levels in large-scale ATO systems with the FCFS allocation rule. They also explore the performance of IBS and FCFS policies. All of these papers assume that any unmet demand is backlogged. Lost sales models have proven notoriously difficult to optimize; for recent developments, see Goldberg et al. (2016) and Xin and Goldberg (2016) for single-item systems and Zipkin (2015) for ATO systems with zero lead time.

Aggregation is an approximate dynamic programming method used to provide a good approximation to a value function using a smaller state space. We refer the reader to Tsitsiklis and Van Roy (1996), chapter 6 in Heyman and Sobel (2003), chapter 8 in Powell (2011), and chapter 6 in Bertsekas (2012) for discussions of value function approximations via aggregation. See also Rogers et al. (1991) for a comprehensive survey of aggregation methods in optimization.

A few researchers have studied the aggregation technique in the ATO literature: Vliegen and van Houtum (2009) use this technique on the so-called service tool problem with joint returns and partial order service. They propose an approximation in which all states with the same number of tools in the return pipeline are aggregated, ignoring the sets the tools were demanded in. Bušić et al. (2012) use aggregation with an appropriate modification of the transition probabilities to construct bounding chains with a common state space of reduced cardinality. They also apply their method in the service tool problem. Bušić and Coupechoux (2014) take a similar approach to derive bounding chains for an original Markov chain, and then they use them as inputs in a perfect simulation algorithm for the purpose of drawing samples from the exact stationary distribution of the original Markov chain. The above papers focus on the performance evaluation of a given policy, while we use state aggregation in search of an *optimal* policy in a general ATO system. We refer the reader to Kemeny and Snell (1976) for further details on state aggregation in Markov chains.

3. The Original Problem

We consider an ATO system with m components ($i = 1, 2, \ldots, m$) and n products ($j = 1, 2, \ldots, n$). Define $\mathbf{A}$ as an $m \times n$ nonnegative resource-consumption matrix; a_{ij} is the number of units of component i needed to assemble one unit of product j, and $\mathbf{a}_j$ is the jth column of $\mathbf{A}$. Each component i is produced in batches of a fixed size q_i in a make-to-stock fashion. Define $\mathbf{q} = (q_1, q_2, \ldots, q_m)$ as the vector of production batch sizes. The production time for a batch of component i is independent of the number and status of outstanding replenishment orders of any type, and it has a k_i-Erlang distribution consisting of k_i stages, each exponentially distributed with the same rate $k_i\mu_i$. Thus, production times with coefficients of variation (CVs) greater than 0 and no larger than 1 can be modeled. (If CVs larger than 1 were desired, hyperexponential production times could be used.) Production for a batch of component i can be initiated only if there is no other batch of component i under production. Assembly lead times are negligible so that assembly operations can be postponed until demand is realized. Demand for each product j arrives as an independent Poisson process with finite rate λ_j. Demand for product j can be fulfilled only if all the required components are available. Demand may also be rejected in the presence of all the necessary components. Unfilled demand of product j incurs a unit lost sale cost c_j that includes the lost profit margin and the potential loss of goodwill.

More restricted versions of our problem have been studied in the literature on Markovian inventory systems with lost sales; see, for example, Ha (1997a, 2000), Benjaafar and ElHafsi (2006), ElHafsi et al. (2008), and Nadar et al. (2014, 2016). More restricted assumptions also appear in the literature on Markovian inventory systems with backorders; see, for example, Ha (1997b), de Véricourt et al. (2002), and Gayon et al. (2009b). Key features of our model that generalize these papers are multiple components demanded by different products according to a general product structure (as opposed to single-component, assembly, nested, and M-system product structures) and Erlang production times (as opposed to exponential production times).

Let $K = \max_i k_i$. The state of the system $\mathbf{X}(t) = \{X_{ki}(t)\}$ is a $K \times m$ matrix that consists of component inventory levels and information regarding the status of current production for each component: $X_{1i}(t)$ is a nonnegative integer denoting the on-hand inventory for component i at time t. For $2 \leqslant k \leqslant k_i$, $X_{ki}(t)$ is a binary integer such that $\sum_{k=2}^{k_i} X_{ki}(t) \leqslant 1$, and $X_{ki}(t) = 1$ means $k_i - k + 1$ Erlang stages have been completed for a batch of component i at time t ($X_{ki}(t) = 0$ if $K \geqslant k > k_i$). Note that if $X_{ki}(t) = 1$, the remaining time to complete the production of a batch of component i has a $(k-1)$-Erlang distribution with mean $(k-1)/k_i\mu_i$. Component i held in stock incurs a unit holding cost per unit time $h_i > 0$; the total inventory holding cost rate in state $\mathbf{X}(t)$ is $h(\mathbf{X}(t)) = \sum_i h_i X_{1i}(t)$. Define t_κ as the time of occurrence of the

κth state transition. Also let $t_0 = 0$. Since all interevent times are exponentially distributed, the state of the system is constant, and the optimality equations remain the same for all t such that $t_\kappa \leqslant t < t_{\kappa+1}$. This implies that decision epochs can be restricted to times when the state changes.

Thus, we formulate the problem as an MDP and focus on Markovian policies for which actions at each decision epoch depend solely on the current state. A control policy l specifies, for each state $\mathbf{x} = \{x_{ki}\}$, the action $\mathbf{u}^l(\mathbf{x}) = (u^{(1)}, \ldots, u^{(m)}, u_1, \ldots, u_n)$, $u^{(i)}, u_j \in \{0,1\}$, $\forall\, i, j$, where if $u^{(i)} = 1$, then production of component i is initiated; if $u^{(i)} = 0$, then component i is not produced; if $u_j = 1$, then demand for product j is satisfied; and if $u_j = 0$, then demand for product j is rejected. Denote by $\mathbb{U}(\mathbf{x})$ the set of admissible actions in state $\mathbf{x}$. For any action $\mathbf{u} = (u^{(1)}, \ldots, u^{(m)}, u_1, \ldots, u_n) \in \mathbb{U}(\mathbf{x})$, (1) $u^{(i)} = 0$ if $\exists k$ s.t. $2 \leqslant k \leqslant k_i$ and $x_{ki} = 1$; and (2) $u_j = 0$ if $\exists i$ s.t. $x_{1i} < a_{ij}$. As component orders are not part of our system state until the first Erlang stage is complete, these can, in effect, be cancelled upon state transition. This assumption is standard (again see, for example, Ha 1997a, b, 2000; de Véricourt et al. 2002; Benjaafar and ElHafsi 2006; ElHafsi et al. 2008; Gayon et al. 2009b; Nadar et al. 2014). Nadar et al. (2016) showed that this assumption is benign in their numerical experiments for ATO systems with exponential production times.

Let v denote a real-valued function defined on $\mathbb{N}_0^{K\times m}$, where $\mathbb{N}_0$ is the set of nonnegative integers. Also define $0 < \alpha < 1$ as the discount parameter. For a given policy $l = \tilde{l}$ and a starting state $\mathbf{X}(0) = \mathbf{x}$, the expected discounted cost over an infinite planning horizon, $v^{\tilde{l}}(\mathbf{x})$, can be written as

$$v^{\tilde{l}}(\mathbf{x}) = E\left[\int_0^\infty e^{-\alpha t} h(\mathbf{X}(t))\, dt + \sum_{j=1}^{n} \int_0^\infty e^{-\alpha t} c_j\, dN_j(t) \,\middle|\, \mathbf{X}(0) = \mathbf{x},\, l = \tilde{l}\right], \quad (1)$$

where $N_j(t)$ is the cumulative number of demands for product j that have not been fulfilled from on-hand inventory up to time t.

The time between the transition to state $\mathbf{x}$ and the transition to the next state is exponentially distributed with rate $\nu_x(\mathbf{u})$ if action $\mathbf{u} = (u^{(1)}, \ldots, u^{(m)}, u_1, \ldots, u_n) \in \mathbb{U}(\mathbf{x})$ is selected in state $\mathbf{x}$. Following Lippman (1975), we consider a uniformized version of the problem where the rate of transition ν is an upper bound for all states and controls; we take $\nu = \sum_i k_i\mu_i + \sum_j \lambda_j$. Thus, the time interval length $(t_{\kappa+1} - t_\kappa)$ is exponentially distributed with rate ν, $\forall\, \kappa$. This transforms the continuous-time control problem into an equivalent discrete-time control problem.

If action $\mathbf{u} = (u^{(1)}, \ldots, u^{(m)}, u_1, \ldots, u_n) \in \mathbb{U}(\mathbf{x})$ is selected in state $\mathbf{x}$, the next state is $\tilde{\mathbf{x}}$ with probability $p_{x,\tilde{x}}(\mathbf{u})$. Thus,

$$p_{x,\tilde{x}}(\mathbf{u}) = \begin{cases} k_i\mu_i u^{(i)}/\nu & \text{if } \tilde{\mathbf{x}} = \mathbf{x} + \mathbf{e}_{k_i i} \text{ (initiate production)},\\ k_i\mu_i/\nu & \text{if } \tilde{\mathbf{x}} = \mathbf{x} + \mathbf{e}_{ki} - \mathbf{e}_{k+1,i} \text{ and } 2 \leqslant k < k_i \\ & \text{(order progresses)},\\ k_i\mu_i/\nu & \text{if } \tilde{\mathbf{x}} = \mathbf{x} + q_i\mathbf{e}_{1i} - \mathbf{e}_{2i} \text{ (order arrives)},\\ \lambda_j u_j/\nu & \text{if } \tilde{\mathbf{x}} = \mathbf{x} - \sum_i a_{ij}\mathbf{e}_{1i} \text{ (satisfy demand)},\\ \left[\nu - \sum_i k_i\mu_i u^{(i)} - \sum_{i\in I_x} k_i\mu_i - \sum_j \lambda_j u_j\right]/\nu & \text{if } \tilde{\mathbf{x}} = \mathbf{x},\\ 0 & \text{otherwise}, \end{cases}$$

where I_x is the set of components i for which $\exists k$ such that $2 \leqslant k \leqslant k_i$ and $x_{ki} = 1$, and $\mathbf{e}_{ki}$ is an $K \times m$ matrix with 1 in the kth row and ith column and 0 in every other entry. In this discrete-time framework, $N_j(t_\kappa)$ is the cumulative number of unsatisfied demands for product j at the time of the κth transition, and $h(\mathbf{X}(t_\kappa))$ is the total inventory holding cost rate during the time interval $[t_\kappa, t_{\kappa+1})$. Then, $v^{\tilde{l}}(\mathbf{x})$ in (1) can be rewritten as follows (see Benjaafar and ElHafsi 2006 and Nadar et al. 2014 for similar formulations in the ATO literature):

$$v^{\tilde{l}}(\mathbf{x}) = E\left[\sum_{\kappa=0}^{\infty}\left(\frac{\nu}{\alpha+\nu}\right)^\kappa \frac{h(\mathbf{X}(t_\kappa))}{\alpha+\nu} + \sum_{\kappa=1}^{\infty}\left(\frac{\nu}{\alpha+\nu}\right)^\kappa \cdot \sum_{j=1}^{n} c_j(N_j(t_\kappa) - N_j(t_{\kappa-1})) \,\middle|\, \mathbf{X}(0) = \mathbf{x},\, l = \tilde{l}\right]. \quad (2)$$

Our objective is to identify a policy l^* that minimizes the expected discounted cost. Below we formulate the optimality equation that holds for the optimal cost function $v^* = v^{l^*}$:

$$v^*(\mathbf{x}) = \min_{u\in\mathbb{U}(x)}\left\{\frac{h(\mathbf{x})}{\alpha+\nu} + \frac{\nu}{\alpha+\nu}\sum_{j=1}^{n}\frac{\lambda_j c_j(1-u_j)}{\nu} + \frac{\nu}{\alpha+\nu}\sum_{\tilde{x}} p_{x,\tilde{x}}(\mathbf{u})v^*(\tilde{\mathbf{x}})\right\}. \quad (3)$$

Therefore, our continuous-time control problem is equivalent to a discrete-time control problem with discount factor $\nu/(\alpha+\nu)$ and cost per stage given by

$$\frac{h(\mathbf{x})}{\alpha+\nu} + \frac{\nu}{\alpha+\nu}\sum_{j=1}^{n}\frac{\lambda_j c_j(1-u_j)}{\nu}.$$

As it is always possible to redefine the timescale, without loss of generality, we assume $\alpha + \nu = 1$. Then the optimality equation in (3) can be simplified as follows:

$$v^*(\mathbf{x}) = h(\mathbf{x}) + \sum_i k_i\mu_i T^{(i)}v^*(\mathbf{x}) + \sum_j \lambda_j T_j v^*(\mathbf{x}), \quad (4)$$

where the operators $T^{(i)}$ for component i and T_j for product j are defined as

$$T^{(i)}v(\mathbf{x}) = \begin{cases} \min\{v(\mathbf{x}+\mathbf{e}_{k_i i}), v(\mathbf{x})\} & \text{if } x_{ki}=0,\ \forall k\in\{2,\dots,k_i\}, \\ v(\mathbf{x}+\mathbf{e}_{ki}-\mathbf{e}_{k+1,i}) & \text{if } x_{k+1,i}=1 \text{ and } 2\leqslant k<k_i, \\ v(\mathbf{x}+q_i\mathbf{e}_{1i}-\mathbf{e}_{2i}) & \text{otherwise, i.e., } x_{2i}=1; \end{cases} \quad (5)$$

$$T_j v(\mathbf{x}) = \begin{cases} \min\{v(\mathbf{x})+c_j, v(\mathbf{x}-\textstyle\sum_i a_{ij}\mathbf{e}_{1i})\} & \text{if } x_{1i}\geqslant a_{ij},\ \forall i, \\ v(\mathbf{x})+c_j & \text{otherwise.} \end{cases} \quad (6)$$

For a given state $\mathbf{x}$, the operator $T^{(i)}$ specifies whether or not to initiate production of a batch of component i if there is no batch of component i under production, and the operator T_j specifies whether or not to fulfill an arriving demand for product j if sufficient inventory exists.

In Section 7, as a computational requirement, we restrict the state space to be finite; define $\bar{\mathbf{x}}_1 = (\bar{x}_{11}, \dots, \bar{x}_{1m})$ as a vector of upper bounds for component inventory levels and $\mathscr{X}$ as the set of system states. Thus, for any state $\mathbf{x}\in\mathscr{X}$ at any time, we must have $0\leqslant x_{1i}\leqslant \bar{x}_{1i}$, $\forall i$. Also, for any action $\mathbf{u} = (u^{(1)}, \dots, u^{(m)}, u_1, \dots, u_n)\in\mathbb{U}(\mathbf{x})$, we must have $u^{(i)} = 0$ if $x_{1i} + q_i > \bar{x}_{1i}$. Note that the upper bounds should be sufficiently high so that they are never visited at optimality.

4. The Aggregate Problem

In this section we use "hard aggregation" to approximate the value function of our ATO problem in Section 3. We group the original system states into disjoint nonempty subsets; each subset forms an "aggregate" state, and each original state belongs to only one aggregate state. (In "soft aggregation," each original state is associated with a convex combination of aggregate states; see Bertsekas 2012, chap. 6.) Define $\mathscr{Y}$ as the set of aggregate states. Denote $\mathbf{x}\in\mathbf{y}$ if the original state $\mathbf{x}$ belongs to aggregate state $\mathbf{y}$, and for every $\mathbf{x}$, denote by $\mathbf{y}(\mathbf{x})$ the aggregate state $\mathbf{y}$ with $\mathbf{x}\in\mathbf{y}$. We introduce the disaggregation probability d_{yx} as the degree to which $\mathbf{y}$ is represented by $\mathbf{x}$. In a hard aggregation scheme,

$$\sum_{x\in y} d_{yx} = 1, \quad \forall\, \mathbf{y}\in\mathscr{Y}.$$

We thus restrict the disaggregation probabilities d_{yx} to be zero for states $\mathbf{x}$ that are not in state $\mathbf{y}$.

Letting $\mathbf{b} = (b_1, \dots, b_m)$ denote a vector of positive integers, below we define the aggregation scheme that we consider in this and subsequent sections.

Definition 1 (On-Hand Inventory Aggregation). The aggregate state $\mathbf{y}(\mathbf{x}) = \{y_{ki}\}$ is a $K\times m$ matrix that is constructed as follows:

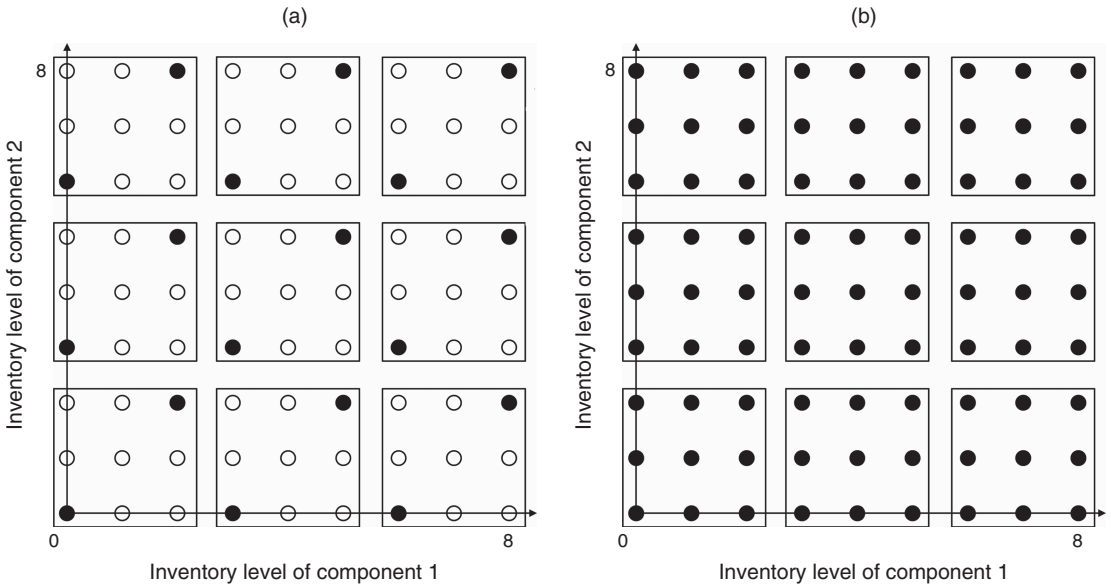
$$y_{1i} = \lfloor x_{1i}/b_i \rfloor, \quad \forall\, i,$$

i.e., $x_{1i} = b_i y_{1i} + z_i$ where $z_i\in\{0,1,\dots,b_i-1\}$, $\forall\, i$, and

$$y_{ki} = x_{ki}, \quad \forall\, i \text{ and } \forall\, k\in\{2,\dots,k_i\}.$$

See Figure 1 for an illustration. The size of the state space decreases as b_i increases; if $\mathbf{b}$ is defined minimally (i.e., $\mathbf{b} = (1,\dots,1)$), the aggregate problem reduces to the original problem. In Section 7, again as a computational requirement, we restrict the aggregate state space to be finite. Define $\bar{\mathbf{y}}_1 = (\bar{y}_{11}, \dots, \bar{y}_{1m})$ as a

Figure 1. Illustration of Our Aggregation Scheme with $\mathbf{b} = (3,3)$ and Two Possible Disaggregation Rules for a 2×2 System with $K = 1$



Notes. Each circle (filled or unfilled) is an original state. Each square is an aggregate state and has $3\times 3 = 9$ original states. For instance, the upper right square in each graph is $\mathbf{y} = (2,2)$. Disaggregation probability of each state shown in a filled circle is 1/2 in (a) and is 1/9 in (b).

vector of upper bounds for aggregate state variables that correspond to on-hand inventory levels in the original problem. Thus, for any aggregate state $\mathbf{y} \in \mathcal{Y}$ at any time, we must have $0 \leqslant y_{1i} \leqslant \bar{y}_{1i}$, $\forall\, i$. We choose our upper bounds for component inventory levels in the original problem as follows:

$$\bar{x}_{1i} + 1 = b_i(\bar{y}_{1i} + 1), \quad \forall\, i.$$

Thus, each aggregate state has $\prod_i b_i$ original states; we reduce the state space by a factor of $\prod_i b_i$.

Below we formulate the Bellman equations that hold for the optimal cost function approximation r^* under our aggregation scheme (see Bertsekas 2012, chap. 6 for a detailed explanation):

$$r^*(\mathbf{y}) = \sum_{x \in y} d_{yx}\tilde{v}(\mathbf{x}), \quad \mathbf{y} \in \mathcal{Y}, \tag{7}$$

where $\tilde{v}(\mathbf{x})$ is the optimal cost-to-go from original state $\mathbf{x}$ that is generated from aggregate state $\mathbf{y}$ as in Definition 1. The function $\tilde{v}$ is defined as

$$\tilde{v}(\mathbf{x}) = h(\mathbf{x}) + \sum_i k_i\mu_i\tilde{v}^{(i)}(\mathbf{x}) + \sum_j \lambda_j\tilde{v}_j(\mathbf{x}), \tag{8}$$

where the functions $\tilde{v}^{(i)}$ for component i and $\tilde{v}_j$ for product j are given by

$$\tilde{v}^{(i)}(\mathbf{x}) = \begin{cases} \min\{r^*(\mathbf{y}(\mathbf{x}+\mathbf{e}_{k_i i})), r^*(\mathbf{y}(\mathbf{x}))\} \\ \quad \text{if } x_{ki} = 0,\ \forall\, k \in \{2,\ldots,k_i\}, \\ r^*(\mathbf{y}(\mathbf{x}+\mathbf{e}_{ki} - \mathbf{e}_{k+1,i})) \\ \quad \text{if } x_{k+1,i} = 1 \text{ and } 2 \leqslant k < k_i, \\ r^*(\mathbf{y}(\mathbf{x}+q_i\mathbf{e}_{1i} - \mathbf{e}_{2i})) \\ \quad \text{otherwise, i.e., } x_{2i} = 1; \end{cases} \tag{9}$$

and

$$\tilde{v}_j(\mathbf{x}) = \begin{cases} \min\left\{r^*(\mathbf{y}(\mathbf{x})) + c_j, r^*(\mathbf{y}(\mathbf{x} - \textstyle\sum_i a_{ij}\mathbf{e}_{1i}))\right\} \\ \qquad\qquad \text{if } x_{1i} \geqslant a_{ij},\ \forall\, i, \\ r^*(\mathbf{y}(\mathbf{x})) + c_j \quad \text{otherwise.} \end{cases} \tag{10}$$

Once r^* is computed, a heuristic policy (for the original problem) can be found through the minimizations in (9) and (10). We use the terms "aggregate-optimal cost function" and "aggregate-optimal policy" to denote the function r^* and the heuristic policy obtained from the function r^*, respectively. We now introduce our rule for disaggregation probabilities in (7).

Definition 2 (Extreme Point Disaggregation). The lowest and highest original states that belong to aggregate state $\mathbf{y}$ are equally representative (see Figure 1(a) for an example); that is,

$$d_{yx} = \begin{cases} 1/2 & \text{if } x_{1i} = b_i y_{1i},\ \forall\, i, \\ 1/2 & \text{if } x_{1i} = b_i y_{1i} + b_i - 1,\ \forall\, i, \\ 0 & \text{otherwise.} \end{cases}$$

The extreme point policy has several desirable characteristics. First, it may significantly reduce the error introduced by our aggregation in Definition 1: Suppose that the current original state is in the middle portion of any aggregate state—for example, $\mathbf{x} = (1,1)$ in Figure 1(a). If a batch of components is ordered in (9) or a demand is satisfied in (10), the system will likely stay in the same aggregate state. Assigning positive disaggregation probabilities to such substates may lead to many such self-transitions, causing r^* to remain the same in many minimizations in (9) and (10) and worsening our approximation. The extreme point policy reduces the number of self-transitions: the system moves from the lowest original state of a particular aggregate state to a different aggregate state whenever a demand for any product is satisfied and from the highest original state of a particular aggregate state to a different aggregate state whenever a batch of any component is produced. Section 7.1 compares the cost performance of this approach with that of an alternative *uniform* disaggregation rule from the literature that assigns equal disaggregation weights to all original states of an aggregate state (Definition 3). The results obtained tend to confirm our intuition.

Definition 3 (Uniform Disaggregation). All states that belong to aggregate state $\mathbf{y}$ are equally representative (see Figure 1(b) for an example); that is, $d_{yx} = (\prod_i b_i)^{-1}$ if $\mathbf{x} \in \mathbf{y}$ and $d_{yx} = 0$ otherwise.

We label the aggregate problems under the disaggregation rules in Definitions 2 and 3 as AP-Ex and AP-Un, respectively. The acronym "Ex" stands for extreme and "Un" for uniform.

Rogers and Plante (1993) evaluate the use of uniform disaggregation in estimating the limiting probabilities of band diagonal Markov chains, comparing it to assigning the disaggregation weights by solving a group of subsystems in the original problem. They find that equal weighting yields approximate limiting probabilities that are often as good as those obtained from the alternative method, motivating us to use the uniform policy as a natural benchmark for our rule.

The extreme point policy also substantially reduces the per-iteration computational complexity of the value iteration algorithm we employ to solve the aggregate problem: since states with zero disaggregation probabilities will never be visited in the aggregate problem, there is no need to execute computations (8)–(10) for those states. This reduces the per-iteration computational complexity of the original problem by a factor of $\Pi_i b_i/2$. Our numerical experiments indicate that the iterations required for convergence of the value iteration algorithm are similar in both the original and aggregate problems. Thus, the per-iteration computational savings from our disaggregation scheme translate into significant savings in value function computation; see Sections 7.1 and 7.2.

5. Error Bounds for the Aggregate Problem

In this section we prove that value iteration, initialized with the aggregate solution, is guaranteed to converge in the original problem. We prove this in three steps: First, we establish upper bounds on the difference of the optimal cost function in (4) evaluated at any two original states having the same configuration of component orders (see Theorem 1(a)). Second, we use these upper bounds to construct a finite error bound for the aggregate-optimal cost function, regardless of the disaggregation rule, based on certain problem parameters (α, h_i, c_j, and a_{ij}) and the vector $\mathbf{b}$ (see Theorem 1(b)). Finally, this finite error bound allows us to validate the use of the value iteration method that starts with the aggregate-optimal cost function in the original problem (see Theorem 1(c)).

We define C_i as the maximum possible inventory cost savings (including both lost sales and holding costs) from serving a demand that consumes component i, *minus* the maximum possible holding cost saving from having one fewer unit of component i (in the long run); that is, $C_i = \max_{j:a_{ij}>0}(c_j + \sum_{\hat{i}} a_{\hat{i}j} h_{\hat{i}}/\alpha) - h_i/\alpha$. Also, let $\geq$ denote componentwise inequality; that is, $\tilde{\mathbf{x}} \geq \hat{\mathbf{x}} \Leftrightarrow \tilde{x}_{ki} \geqslant \hat{x}_{ki}, \forall k, i$. With these, we establish the following structural properties of our optimal cost function. (The proofs of Lemma 1 and all other subsequent results appear in the online appendix.)

Lemma 1. *The optimal cost function satisfies the following inequalities*:

(a) $v^*(\hat{\mathbf{x}}) + (h(\tilde{\mathbf{x}}) - h(\hat{\mathbf{x}}))/\alpha \geqslant v^*(\tilde{\mathbf{x}}), \forall \tilde{\mathbf{x}} \geq \hat{\mathbf{x}}$ s.t. $\tilde{x}_{ki} = \hat{x}_{ki}$, $\forall k > 1, \forall i$.

(b) $v^*(\mathbf{x} + \mathbf{e}_{1i}) + C_i \geqslant v^*(\mathbf{x}), \forall \mathbf{x}, \forall i$.

Lemma 1(a) states that when the system moves to a higher inventory level, the optimal cost increases by no more than the maximum possible increase in holding costs, which occurs when the additional $\tilde{x}_{1i} - \hat{x}_{1i}$ units of each component i remain in inventory for a very long time. Lemma 1(b) states that when the inventory level of component i is reduced by one, the optimal cost increases by no more than C_i. Lemma 1(b) also implies that $v^*(\mathbf{x} + \sum_{i=1}^m \mathbf{e}_{1i}) + \sum_{i=1}^m C_i \geqslant v^*(\mathbf{x}), \forall \mathbf{x}$.

Next we define the operator T on the set of real-valued functions v: $Tv(\mathbf{x}) = h(\mathbf{x}) + \sum_i k_i \mu_i T^{(i)} v(\mathbf{x}) + \sum_j \lambda_j T_j v(\mathbf{x})$ where the operators $T^{(i)}$ and T_j are given in (5) and (6). Also, let $T^1 v = Tv$ and $T^k v = T(T^{k-1} v)$, $\forall k > 1$. We are now ready to state the main result of this section based on Lemma 1.

Theorem 1. (a) *The following inequality holds for any two original states* $\tilde{\mathbf{x}}$ *and* $\hat{\mathbf{x}}$ *having the same configuration of component orders* $\{x_{ki}\}_{k>1}$:

$$v^*(\tilde{\mathbf{x}}) - v^*(\hat{\mathbf{x}}) \leqslant \frac{h(\tilde{\mathbf{x}} - \hat{\mathbf{x}} + \tau \sum_{i=1}^m \mathbf{e}_{1i}) + \alpha\tau \sum_{i=1}^m C_i}{\alpha},$$

where $\tau = \min\{k \in \mathbb{N}_0 : k \geqslant \hat{x}_{1i} - \tilde{x}_{1i},\ \forall i\}$.

(b) *There exists a finite error bound for the aggregate-optimal cost function that can be specified as follows*:

$$r^*(\mathbf{y}) - \epsilon \leqslant v^*(\mathbf{x}) \leqslant r^*(\mathbf{y}) + \epsilon, \quad \forall \mathbf{y} \in \mathcal{Y}, \mathbf{x} \in \mathbf{y},$$

where

$$\epsilon = \max_{\substack{\tilde{x},\hat{x} \\ \text{s.t. } 0 \leqslant \tilde{x}_{1i}, \hat{x}_{1i} < b_i, \forall i}} \frac{h(\tilde{\mathbf{x}} - \hat{\mathbf{x}} + \tau \sum_{i=1}^m \mathbf{e}_{1i}) + \alpha\tau \sum_{i=1}^m C_i}{\alpha^2}.$$

(c) *The value iteration algorithm starting with the aggregate-optimal cost function converges to the optimal cost function of the original problem; that is, if* $v(\mathbf{x}) = r^*(\mathbf{y}(\mathbf{x}))$, $\forall \mathbf{x}$, *then* $\lim_{k\to\infty}(T^k v)(\mathbf{x}) = v^*(\mathbf{x})$, $\forall \mathbf{x}$.

The upper bounds in Theorem 1(a) build on the structural properties in Lemma 1. In the ATO literature, several authors have derived upper bounds on the optimal cost function difference between *certain* inventory levels for more restricted versions of our general ATO problem to identify the product or demand class that should have the highest fulfillment priority. See, for instance Ha (1997a, 2000), Benjaafar and ElHafsi (2006), ElHafsi et al. (2008), and ElHafsi (2009). However, to the best of our knowledge, we are the first to introduce upper bounds on the optimal cost function difference between *arbitrary* inventory levels.

The upper bounds in Theorem 1(a) allow us to construct the error bound ϵ in Theorem 1(b). Note that ϵ increases with h_i, C_i, and $1 - \alpha$ (the discount factor). Numerical experiments indicate that ϵ is typically multiple orders of magnitude larger than the actual errors for the instances in Section 7. Nevertheless, in Theorem 1(c), our finite error bound guarantees the convergence of the value iteration algorithm in the original problem (with infinite state space) initialized with the (unbounded) aggregate-optimal cost function (defined on an infinite state space). Numerical experiments in Section 7.3 demonstrate the usefulness of this value iteration algorithm.

We also establish a different error bound in Corollary 1, which is *tighter* than the error bound in Theorem 1. But the error bound in Corollary 1 is only available under the following assumption, of which there are several examples in the literature.

Assumption 1. *There exists a product* (*denoted by* j^*) *such that* (i) *one unit of this product consumes at least one unit from each component, and* (ii) *it is always optimal to satisfy demands for this product if sufficient inventory exists.*

Below are four specific ATO product structures that satisfy Assumption 1, which are special cases of our general ATO problem in Section 3.

1. *An assembly product structure:* Suppose that $K = a_{ij} = 1$, $\forall i, j$. In such systems, Benjaafar and ElHafsi (2006) show that it is always optimal to satisfy demands for the product with the highest lost sale cost if sufficient inventory is available.

2. *A nested product structure with unitary component requirements:* Suppose that $K = 1$, $m = n$, $a_{ij} = 1$ if $i \geqslant j$, $a_{ij} = 0$ otherwise, and $c_1 > \cdots > c_m$. In such systems, ElHafsi et al. (2008) show that it is always optimal to satisfy demands for product 1 if sufficient inventory is available.

3. *A nested product structure with nonunitary component requirements:* Suppose that $K = 1$. Also, letting τ_j denote a positive integer, $\forall\, j$, suppose that $a_{i1} = 1$, $a_{ij} = \tau_1 \times \cdots \times \tau_{j-1}$, $\forall\, j > 1$, $\forall\, i$, $c_{j+1} \geqslant \tau_j c_j$, $\forall\, j < n$, and $\mathbf{q} = \mathbf{a}_n$. Then it is always optimal to satisfy demands for product n if sufficient inventory is available; see the online appendix for a proof. Such systems may appear in semiconductor manufacturing. For instance, the price of an Intel processor with 8 cores and 16 threads is more than twice that with 4 cores and 8 threads, which is again more than twice that with 2 cores and 4 threads (https://www.intc.com/investor-relations/investor-education-and-news/cpu-price-list/default.aspx). Our nested structure holds in this example if the goodwill loss has a slight effect on the lost sale cost.

4. *An M-system product structure:* Suppose that $m = 2$, $n = 3$, $K = 1$, $a_{11} = a_{22} = a_{13} = a_{23} = 1$, and $a_{12} = a_{21} = 0$. Also, suppose that $c_1 + c_2 \leqslant c_3$. Then it can be shown that it is always optimal to satisfy demands for product 3 if sufficient inventory is available. Such systems may appear when a vendor sells a bundled product at a premium price since the bundling process itself requires substantial technical knowledge. For instance, the price of a computer is often higher than the sum of the prices of its components.

Assumption 1 fails to hold if there is no product that uses all the components or if every product that uses all the components has at least one state in which its demand is rejected at optimality.

Corollary 1. *Under Assumption 1, the following inequality holds for any two original states $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ having the same configuration of component orders $\{x_{ki}\}_{k>1}$:*

$$v^*(\tilde{\mathbf{x}}) - v^*(\hat{\mathbf{x}}) \leqslant \frac{h(\tilde{\mathbf{x}} - \hat{\mathbf{x}} + \tau \sum_{i=1}^m a_{ij^*}\mathbf{e}_{1i}) + \alpha\tau c_{j^*}}{\alpha},$$

where $\tau = \min\{k \in \mathbb{N}_0 : ka_{ij^*} \geqslant \hat{x}_{1i} - \tilde{x}_{1i},\ \forall\, i\}$. Furthermore, there exists a finite error bound for the aggregate-optimal cost function that can be specified as follows:

$$r^*(\mathbf{y}) - \epsilon \leqslant v^*(\mathbf{x}) \leqslant r^*(\mathbf{y}) + \epsilon, \quad \forall\, \mathbf{y} \in \mathcal{Y},\ \mathbf{x} \in \mathbf{y},$$

where

$$\epsilon = \max_{\substack{\tilde{x},\hat{x} \\ \text{s.t. } 0 \leqslant \tilde{x}_{1i}, \hat{x}_{1i} < b_i, \forall\, i}} \frac{h(\tilde{\mathbf{x}} - \hat{\mathbf{x}} + \tau \sum_{i=1}^m a_{ij^*}\mathbf{e}_{1i}) + \alpha\tau c_{j^*}}{\alpha^2}.$$

Corollary 1 follows from Theorem 1 if $\sum_{i=1}^m \mathbf{e}_{1i}$ and $\sum_{i=1}^m C_i$ are replaced with $\sum_{i=1}^m a_{ij^*}\mathbf{e}_{1i}$ and c_{j^*}, respectively. Unlike Theorem 1, Corollary 1 builds on the inequality $v^*(\mathbf{x} + \sum_{i=1}^m a_{ij^*}\mathbf{e}_{1i}) + c_{j^*} \geqslant v^*(\mathbf{x})$, $\forall\, \mathbf{x}$, rather than the structural property in Lemma 1(b). This inequality holds under Assumption 1: when the system moves to a lower inventory level upon fulfillment of a demand for product j^*, the optimal cost function increases by no more than c_{j^*}, because it is always optimal to satisfy demands for product j^* if sufficient inventory exists.

6. Characterization of the Aggregate-Optimal Policy

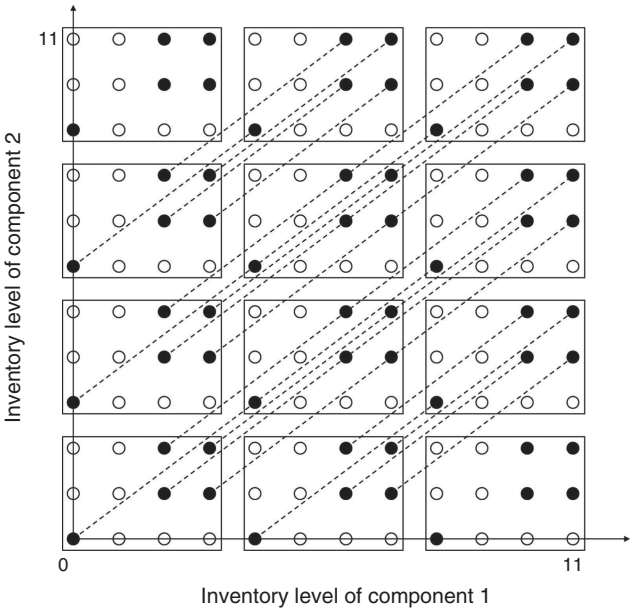
We are able to establish the aggregate-optimal policy structure under the following assumption.

Assumption 2. (i) *If the production time of each component has an exponential distribution (i.e., $K = 1$), $b_i \geqslant \max\{q_i, \max_j\{a_{ij}\}\}$, $\forall\, i$. If the production time of at least one component has an Erlang distribution (i.e., $K > 1$), $b_i = q_i \geqslant \max_j\{a_{ij}\}$, $\forall\, i$.*

(ii) *For $\mathbf{x} \in \mathbf{y}$, the disaggregation probability d_{yx} can be strictly positive only if either $x_{1i} < \min_j\{a_{ij}\} + b_i\lfloor x_{1i}/b_i \rfloor$, $\forall\, i$ or $x_{1i} \geqslant \max_j\{a_{ij}\} + b_i\lfloor x_{1i}/b_i \rfloor$, $\forall\, i$. In addition, for $\tilde{\mathbf{x}} \in \tilde{\mathbf{y}}$ and $\hat{\mathbf{x}} \in \hat{\mathbf{y}}$, $d_{\tilde{y}\hat{x}} = d_{\hat{y}\hat{x}}$ if $\tilde{x}_{1i} - b_i\tilde{y}_{1i} = \hat{x}_{1i} - b_i\hat{y}_{1i}$, $\forall\, i$, or, equivalently, $d_{yx} = d_z$, where $x_{1i} = b_i y_{1i} + z_i$, $0 \leqslant z_i < b_i$, $\forall\, i$, and $\mathbf{z} = (z_1, \ldots, z_m)$, $\forall\, \mathbf{y}$, and $\forall\, \mathbf{x} \in \mathbf{y}$.*

See Figure 2 for an illustration of aggregation and disaggregation schemes under Assumption 2.

Figure 2. Illustration of Aggregation and Disaggregation Schemes Under Assumption 2 When $K = 1$, $\mathbf{A} = ((2,1),(1,1))$, $\mathbf{q} = (1,1)$, and $\mathbf{b} = (4,3)$



Notes. Each circle is an original state, and each rectangle is an aggregate state. Disaggregation probabilities can be positive for original states shown in filled circles, but they are zero for other original states (e.g., $d_{(0,0),(2,1)} \geqslant d_{(0,0),(1,1)} = 0$). Disaggregation probabilities for original states on the same location of different aggregate states are the same (e.g., $d_{(0,0),(2,1)} = d_{(1,2),(6,7)}$). The original states on each dashed line form a different lattice used in Theorem 2.

If $K = 1$ and $b_i = \max\{q_i, \max_j\{a_{ij}\}\}, \forall i$, two original states map to different aggregate states if either the inventory levels of some component i in these original states are at least one replenishment batch apart, or if the same number of units of some product j cannot be made from on-hand inventory of some component i in these original states (assuming an ample supply for all the other components). If $K > 1$, b_i should be equal to the replenishment batch size of component i, which is no smaller than the number of units of component i required by any product.

Assumption 2(ii) implies that the disaggregation probability of the state $\mathbf{x} \in \mathbf{y}$ can be positive only if either the aggregate state $\mathbf{y}(\mathbf{x})$ remains the same upon fulfillment of a demand for product j, $\forall j$, or if the aggregate state $\mathbf{y}(\mathbf{x})$ drops to $\mathbf{y}(\mathbf{x}) - \mathbf{e}$ upon fulfillment of a demand for product j, $\forall j$. The extreme point policy satisfies Assumption 2(ii) when $a_{ij} \geqslant 1$ and $b_i > \max_j a_{ij}, \forall i, j$.

We establish the aggregate-optimal inventory replenishment and allocation policies through the structural properties of our aggregate-optimal cost function. Let $\mathbf{e}_{ki}$ denote a zero matrix if $k > k_i$. Define $\mathcal{V}$ as the set of real-valued functions f on $\mathbb{N}_0^{K\times m}$ that satisfy the following properties.

Property 1. $f(\mathbf{y} + \sum_{j=1}^m \mathbf{e}_{1j} + \mathbf{e}_{ki}) - f(\mathbf{y} + \sum_{j=1}^m \mathbf{e}_{1j} + \mathbf{e}_{k+1,i}) \geqslant f(\mathbf{y} + \mathbf{e}_{ki}) - f(\mathbf{y} + \mathbf{e}_{k+1,i}), \forall \mathbf{y}, \forall i,$ *and* $\forall k \in \{1, \ldots, k_i\}$,

Property 2. $f(\mathbf{y} + \mathbf{e}_{ki} + \mathbf{e}_{\tilde{k}+1,\tilde{i}}) - f(\mathbf{y} + \mathbf{e}_{k+1,i} + \mathbf{e}_{\tilde{k}+1,\tilde{i}}) \geqslant f(\mathbf{y} + \mathbf{e}_{ki} + \mathbf{e}_{\tilde{k}\tilde{i}}) - f(\mathbf{y} + \mathbf{e}_{k+1,i} + \mathbf{e}_{\tilde{k}\tilde{i}}), \forall \mathbf{y}, \forall i \neq \tilde{i}, \forall k \in \{1, \ldots, k_i\},$ *and* $\forall \tilde{k} \in \{1, \ldots, k_{\tilde{i}}\}$,

Property 3. $f(\mathbf{y} + \mathbf{e}_{1i} + \mathbf{e}_{k_i i}) - f(\mathbf{y} + \mathbf{e}_{1i}) \geqslant f(\mathbf{y} + \mathbf{e}_{k_i i}) - f(\mathbf{y}), \forall \mathbf{y}, \forall i.$

Property 1 states that the difference $f(\mathbf{y} + \mathbf{e}_{ki}) - f(\mathbf{y} + \mathbf{e}_{k+1,i})$ weakly increases if each of the variables y_{1i} is increased by one. Property 2 states that the difference $f(\mathbf{y} + \mathbf{e}_{ki} + \mathbf{e}_{\tilde{k}+1,\tilde{i}}) - f(\mathbf{y} + \mathbf{e}_{k+1,i} + \mathbf{e}_{\tilde{k}+1,\tilde{i}})$ weakly decreases if the batch of component $\tilde{i}$ under production proceeds to the next Erlang stage. When $K = 1$, Property 2 reduces to Topkis' (1978, 1998) submodularity property on $\mathbb{N}_0^m$. Property 3 states that the difference $f(\mathbf{y} + \mathbf{e}_{k_i i}) - f(\mathbf{y})$ weakly increases as the variable y_{1i} increases. When $K = 1$, Property 3 reduces to discrete convexity in each of the variables y_{1i}. We show in the proof of Lemma 2 that Properties 1 and 2 together imply Property 3.

Several papers characterize the optimal policy for inventory systems with exponential production times by proving that the optimal cost function satisfies Properties 1–3 when $K = 1$. See, for instance, Benjaafar and ElHafsi (2006), ElHafsi et al. (2008), ElHafsi (2009), and Gayon et al. (2009a). Unlike these papers, Ha (2000) considers a single-item inventory system with Erlang production times; it can be shown that the optimal structural property in Ha (2000) is equivalent to Property 1 when $m = 1$. However, to the best of our knowledge, no one has studied the above properties when both $K > 1$ and $m > 1$.

Define the operator F on the set of real-valued functions r:

$$Fr(\mathbf{y}) = \sum_{x\in y} d_{yx}\left(h(\mathbf{x}) + \sum_i k_i \mu_i v^{(i)}(\mathbf{x}) + \sum_j \lambda_j v_j(\mathbf{x})\right), \quad \mathbf{y} \in \mathcal{Y}, \tag{11}$$

where the functions $v^{(i)}$ for component i and v_j for product j are given by

$$v^{(i)}(\mathbf{x}) = \begin{cases} \min\{r(\mathbf{y}(\mathbf{x} + \mathbf{e}_{k_i i})), r(\mathbf{y}(\mathbf{x}))\} \\ \quad \text{if } x_{ki} = 0, \ \forall k \in \{2, \ldots, k_i\}, \\ r(\mathbf{y}(\mathbf{x} + \mathbf{e}_{ki} - \mathbf{e}_{k+1,i})) \\ \quad \text{if } x_{k+1,i} = 1 \text{ and } 2 \leqslant k < k_i, \\ r(\mathbf{y}(\mathbf{x} + q_i \mathbf{e}_{1i} - \mathbf{e}_{2i})) \\ \quad \text{otherwise, i.e., } x_{2i} = 1; \text{ and} \end{cases} \tag{12}$$

$$v_j(\mathbf{x}) = \begin{cases} \min\{r(\mathbf{y}(\mathbf{x})) + c_j, r(\mathbf{y}(\mathbf{x} - \sum_i a_{ij}\mathbf{e}_{1i}))\} \\ \quad \text{if } x_{1i} \geqslant a_{ij}, \ \forall i, \\ r(\mathbf{y}(\mathbf{x})) + c_j \quad \text{otherwise.} \end{cases} \tag{13}$$

Lemma 2 shows that $\mathcal{V}$ propagates through the operator F, and that our aggregate-optimal cost function is an element of $\mathcal{V}$.

Lemma 2. *Under Assumption 2, if $r \in \mathcal{V}$, then $Fr \in \mathcal{V}$. Furthermore, the aggregate-optimal cost function r^* is an element of $\mathcal{V}$.*

Thus, our aggregate-optimal cost function satisfies Properties 1–3 under Assumption 2. The aggregate-optimal policy form in Theorem 2 builds on Property 1, and the comparative statics of the aggregate-optimal policy parameters build on both Properties 1 and 2. It is possible to construct numerical examples showing that Property 1 may fail to hold if Assumption 2 is violated.

We introduce the notation $\mathbb{L}(\mathbf{p}, \mathbf{b}) = \{\mathbf{p} + k\mathbf{b}\colon k \in \mathbb{N}_0\}$ to denote an m-dimensional lattice with initial vector $\mathbf{p} \in \mathbb{N}_0^m$ and common difference $\mathbf{b}$, where $\exists i$ s.t. $p_i < b_i$. Thus, for any $\mathbf{b}$, $\mathbb{N}_0^m = \bigcup_p \mathbb{L}(\mathbf{p}, \mathbf{b})$ and $\mathbb{L}(\mathbf{p}_1, \mathbf{b}) \cap \mathbb{L}(\mathbf{p}_2, \mathbf{b}) = \varnothing, \forall \mathbf{p}_1 \neq \mathbf{p}_2$. We partition the original state space of the *on-hand* inventory levels into multiple disjoint lattices with common difference $\mathbf{b}$. Each original state in a particular aggregate state corresponds to a different lattice. Application of Lemma 2 allows us to characterize the form of the aggregate-optimal policy over such lattices of the original state space, restricted to the states with positive disaggregation probabilities. See Figure 2 for an example.

Recall that $\mathbf{e}_{ki}$ is an $K \times m$ matrix with 1 in the kth row and ith column and 0 in every other entry. We also

introduce the notation $\mathbf{e}_i$ to denote the ith unit vector of dimension m. With these, we are now ready to establish the aggregate-optimal policy structure.

Theorem 2. *Under Assumption 2, there exists an aggregate-optimal stationary policy such that we have the following*:

(1) *For any configuration of component orders* $\mathbf{x}_{-1} = \{x_{ki}\}_{k>1}$, *the aggregate-optimal inventory replenishment policy for each component* i *is a* lattice-dependent base-stock policy *with lattice-dependent base-stock levels* $S_i^*(\mathbf{p}, \mathbf{x}_{-1}) \in \mathbb{L}(\mathbf{p}, \mathbf{b})$, $\forall\, \mathbf{p}$. *It is aggregate-optimal to produce a batch of component* i *if and only if* $\{x_{1i}\} \in \mathbb{L}(\mathbf{p}, \mathbf{b})$ *is less than* $S_i^*(\mathbf{p}, \mathbf{x}_{-1})$.

(2) *For any configuration of component orders* $\mathbf{x}_{-1} = \{x_{ki}\}_{k>1}$, *the aggregate-optimal inventory allocation policy for each product* j *is a* lattice-dependent rationing policy *with lattice-dependent rationing levels* $R_j^*(\mathbf{p}, \mathbf{x}_{-1}) \in \mathbb{L}(\mathbf{p}, \mathbf{b})$, $\forall\, \mathbf{p}$. *It is aggregate-optimal to fulfill a demand for product* j *if and only if* $\{x_{1i}\} \in \mathbb{L}(\mathbf{p}, \mathbf{b})$ *is greater than or equal to* $R_j^*(\mathbf{p}, \mathbf{x}_{-1})$.

The aggregate-optimal policy has the following additional properties:

(i) *The controller is indifferent between producing and not producing a batch of component* i *at aggregate-optimality if* $\{x_{1j}\} \in \mathbb{L}(\mathbf{p}, \mathbf{b})$ *and* $\mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i}) = \mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j})$.

(ii) *The base-stock levels* $S_i^*(\mathbf{p}, \mathbf{x}_{-1})$ *and* $S_i^*(\mathbf{r}, \mathbf{x}_{-1})$ *map to the same aggregate state if* $\mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j}) + \mathbf{e}_{1i} = \mathbf{y}(\sum_{j=1}^m r_j \mathbf{e}_{1j}) + \mathbf{e}_{1i} = \mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i}) = \mathbf{y}(\sum_{j=1}^m r_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i})$.

(iii) *The base-stock levels obey* $S_i^*(\mathbf{p}, \tilde{\mathbf{x}}_{-1}) \geqslant S_i^*(\mathbf{p}, \mathbf{x}_{-1})$ *if* $\mathbf{x}_{-1}$ *becomes* $\tilde{\mathbf{x}}_{-1}$ *upon completion of any Erlang stage* $k < k_{\tilde{\imath}}$ *for any component* $\tilde{\imath} \neq i$ *and* $\mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j}) + \mathbf{e}_{1i} = \mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i})$.

(iv) *The base-stock levels obey* $S_i^*(\mathbf{p} + b_{\tilde{\imath}} \mathbf{e}_{\tilde{\imath}}, \tilde{\mathbf{x}}_{-1}) \geqslant S_i^*(\mathbf{p}, \mathbf{x}_{-1}) + b_{\tilde{\imath}} \mathbf{e}_{\tilde{\imath}}$ *if* $\mathbf{x}_{-1}$ *becomes* $\tilde{\mathbf{x}}_{-1}$ *upon completion of the last Erlang stage for any component* $\tilde{\imath} \neq i$ *and* $\mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j}) + \mathbf{e}_{1i} = \mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i})$.

(v) *The base-stock levels obey* $S_i^*(\mathbf{p}, \mathbf{x}_{-1}) + \sum_{\tilde{\imath} \in I_1} b_{\tilde{\imath}} \mathbf{e}_{\tilde{\imath}} \geqslant S_i^*(\mathbf{p} + \sum_{\tilde{\imath} \in I_1} b_{\tilde{\imath}} \mathbf{e}_{\tilde{\imath}}, \tilde{\mathbf{x}}_{-1})$ *if* $i \in I_1$; $x_{k\tilde{\imath}} = \tilde{x}_{k\tilde{\imath}} - 1$, $\forall\, \tilde{\imath} \in I_k$, $\forall\, k > 1$; $I_{\hat{k}} \cap I_{\tilde{k}} = \varnothing$, $\forall\, \hat{k} \neq \tilde{k}$; *and* $\mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j}) + \mathbf{e}_{1i} = \mathbf{y}(\sum_{j=1}^m p_j \mathbf{e}_{1j} + q_i \mathbf{e}_{1i})$.

(vi) *It is aggregate-optimal to fulfill a demand for product* j *if* $\{x_{1i}\} \in \mathbb{L}(\mathbf{p}, \mathbf{b})$, *where* $\mathbf{a}_j \preceq \mathbf{p}$ *and* $\mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i} - \sum_{i=1}^m a_{ij} \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i})$.

(vii) *The rationing levels obey* $R_j^*(\mathbf{p}, \mathbf{x}_{-1})$ *and* $R_j^*(\mathbf{r}, \mathbf{x}_{-1})$ *map to the same aggregate state if* $\mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m r_i \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i} + \sum_{i=1}^m b_i \mathbf{e}_{1i} - \sum_{i=1}^m a_{ij} \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m r_i \mathbf{e}_{1i} + \sum_{i=1}^m b_i \mathbf{e}_{1i} - \sum_{i=1}^m a_{ij} \mathbf{e}_{1i})$.

(viii) *The rationing levels obey* $R_j^*(\mathbf{p}, \mathbf{x}_{-1}) \geqslant R_j^*(\mathbf{p}, \tilde{\mathbf{x}}_{-1})$ *if* $\mathbf{x}_{-1}$ *becomes* $\tilde{\mathbf{x}}_{-1}$ *upon completion of any Erlang stage* $k < k_i$ *for any component* i *and* $\mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i} + \sum_{i=1}^m b_i \mathbf{e}_{1i} - \sum_{i=1}^m a_{ij} \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i})$.

(ix) *The rationing levels obey* $R_j^*(\mathbf{p}, \mathbf{x}_{-1}) + b_i \mathbf{e}_i \geqslant R_j^*(\mathbf{p} + b_i \mathbf{e}_i, \tilde{\mathbf{x}}_{-1})$ *if* $\mathbf{x}_{-1}$ *becomes* $\tilde{\mathbf{x}}_{-1}$ *upon completion of the last Erlang stage for any component* i *and* $\mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i} + \sum_{i=1}^m b_i \mathbf{e}_{1i} - \sum_{i=1}^m a_{ij} \mathbf{e}_{1i}) = \mathbf{y}(\sum_{i=1}^m p_i \mathbf{e}_{1i})$.

Using Property 1, for a given configuration of component orders, Theorem 2 shows that a *lattice-dependent base-stock and lattice-dependent rationing* (LBLR) policy is the aggregate-optimal policy under Assumption 2: Property 1 implies that, as the system moves to a higher inventory level on the lattice $\mathbb{L}(\mathbf{p}, \mathbf{b})$, the desirability of producing a batch of component i decreases in a nonstrict sense (aggregate-optimality of base-stock policies, point 1), and the desirability of satisfying a demand for product j increases in a nonstrict sense (aggregate-optimality of rationing policies, point 2).

Theorem 2 also proves the following properties of the aggregate-optimal replenishment policy:

- Point (i) says that if the current aggregate state stays the same when the inventory level of component i is increased by its batch size, which is possible only if $K = 1$ (see Assumption 2), no difference exists between producing and not producing component i at aggregate-optimality.
- For a given configuration of component orders, point (ii) shows that if increasing the inventory level of component i by its batch size moves the system from two different points of two disjoint lattices in an aggregate state $\mathbf{y}$ to the same aggregate state $\mathbf{y} + \mathbf{e}_{1i}$, the aggregate-optimal base-stock levels of component i on these two lattices must be in the same aggregate state.
- Based on Property 2, point (iii) says that the aggregate-optimal base-stock level of component i on a particular lattice weakly increases when a batch of any component $\tilde{\imath} \neq i$ under production proceeds to the next Erlang stage. Likewise, point (iv) states that the aggregate-optimal base-stock level of component i weakly increases when the system moves to a different lattice with an increment of $b_{\tilde{\imath}}$ in the inventory level of component $\tilde{\imath} \neq i$ and $x_{2\tilde{\imath}}$ decreases by one.
- Conversely, based on Properties 1 and 2, point (v) says that, when the system moves to a different lattice with an increment of b_i in the inventory level of component i and an increment of $b_{\tilde{\imath}}$ in the inventory level of component $\tilde{\imath} \neq i$, the aggregate-optimal base-stock level of component i increases by no more than $b_i \mathbf{e}_i + b_{\tilde{\imath}} \mathbf{e}_{\tilde{\imath}}$. This result continues to hold when the system moves to a different lattice with increments of b_i and $b_{\tilde{\imath}}$ but *also* a batch of some component $\hat{\imath} \notin \{i, \tilde{\imath}\}$ under production proceeds to the next Erlang stage.

Theorem 2 proves the following additional properties of the aggregate-optimal allocation policy:

- Point (vi) shows that it is aggregate-optimal to satisfy a demand for product j if the current aggregate state remains the same upon fulfillment of this demand.
- For a given configuration of component orders, point (vii) states that if the system moves from two different points of two disjoint lattices in a particular aggregate state to a lower aggregate state upon fulfillment of a demand for product j, the aggregate-optimal

rationing levels for product j on these two lattices must be in the same aggregate state.

- Based on Property 1, points (viii) and (ix) say that, upon fulfillment of a demand for product j, if the system moves from one particular lattice in any aggregate state to a lower aggregate state, then the following hold: The aggregate-optimal rationing level for product j on this particular lattice weakly decreases when a batch of any component i under production proceeds to the next Erlang stage. However, when the system moves from this lattice to a different lattice with an increment of b_i in the inventory level of component i and x_{2i} decreases by 1, the aggregate-optimal rationing level for product j increases by $b_i\mathbf{e}_i$ or decreases.

The notion of LBLR was first introduced by Nadar et al. (2014) to characterize the optimal policy structure for ATO systems with generalized M-system product structure and exponential production times. Nadar et al. (2014) established the comparative statics of the optimal policy parameters for their ATO problem, which are in line with points (iv) and (ix) of Theorem 2. In their computational work, Nadar et al. (2016) reveal the optimal performance of LBLR for ATO systems with general product structures and exponential production times, but they present no method for its proof. Unlike Nadar et al. (2014, 2016), we use the notion of LBLR to analytically characterize the *aggregate-optimal* policy structure for ATO systems with *general* product structures and *Erlang* production times. Thus, the models in Nadar et al. (2014, 2016) are special cases of our model in this paper. For exponential production times (i.e., when $K = 1$), our partitioning of the state space into disjoint lattices is based on the vector $\mathbf{b}$ defined by our aggregation scheme; that in Nadar et al. (2014, 2016) is based on certain assumed product characteristics that we do not require. When $K > 1$, Theorem 2 also establishes the comparative statics of the aggregate-optimal policy parameters with respect to the configuration of component orders.

When $K = b_i = q_i = a_{ij} = 1$, $\forall\, i, j$, and thus Assumption 2 is satisfied, our aggregate problem reduces to the *original* problem in Benjaafar and ElHafsi (2006), who characterize the optimal policy structure for an ATO assembly system with *exponential* production times. When $K > b_i = q_i = a_{ij} = 1$, $\forall\, i, j$, we characterize the optimal policy structure for an ATO assembly system with *Erlang* production times, generalizing Benjaafar and ElHafsi (2006).

Section 7.4 implements the above structural results to accelerate the value iteration algorithm, by restricting the action space to only generate LBLR policies in each iteration step.

7. Numerical Experiments

In Section 7.1 we numerically investigate the performance of aggregate-optimal cost functions in AP-Ex and AP-Un as approximations to the optimal cost function in the original problem (OP). After evaluating the scalability of AP-Ex in Section 7.2, we evaluate the use of the value iteration algorithm initialized with the aggregate-optimal cost function in OP, comparing it to the standard value iteration algorithm in Section 7.3. Finally, in Section 7.4, we exploit the aggregate-optimal policy structure (see Theorem 2) in computation of the aggregate-optimal cost function. All computations have been executed on a system with 2.9 GHz CPU and 16 GB of RAM.

7.1. Performance Evaluation of AP-Ex and AP-Un

In this subsection we investigate the performance of the optimal solutions of AP-Ex and AP-Un, in comparison with the optimal solution of OP. We generate 30 instances with 3 components and 3 products by randomly selecting a_{ij} and q_i from the set $\{1,2,3,4,5\}$, h_i from the set $\{1,3,5\}$, c_j from the set $\{50,100,150,200\}$, μ_i from the set $\{1,2\}$, and λ_j from the set $\{0.2,0.4\}$, $\forall\, i, j$. We solve each of the 30 instances for exponential and 2-Erlang production times; for discount factors $\nu/(\alpha+\nu)$ of 0.95, 0.97, and 0.99; and for aggregation schemes in which $\mathbf{b} \in \{(3,3,3),(4,4,4),(6,6,6)\}$. We impose $\bar{\mathbf{x}}_1 = (35,35,35)$ in all instances. (Because $\bar{x}_{1i} + 1 = 36$ is an integer multiple of 3, 4, and 6, each aggregate state contains the same number of original states under each aggregation scheme. Also, note that our instances in this subsection do not necessarily satisfy Assumptions 1 and 2.) We evaluate the performance of AP-Ex and AP-Un in terms of the following measures:

(i) *APD*: The average of percentage deviation from the optimal cost across all states; that is,

$$\frac{100}{|\mathscr{X}|} \times \sum_{\mathbf{x}\in\mathscr{X}} \frac{|r^*(\mathbf{y}(\mathbf{x})) - v^*(\mathbf{x})|}{v^*(\mathbf{x})},$$

where $|\mathscr{X}|$ is the number of original states in the set $\mathscr{X}$.

(ii) *WAPD*: The weighted average of percentage deviation from the optimal cost across all states based on the optimal stationary distribution; that is,

$$100 \times \sum_{\mathbf{x}\in\mathscr{X}} \frac{\pi_x |r^*(\mathbf{y}(\mathbf{x})) - v^*(\mathbf{x})|}{v^*(\mathbf{x})},$$

where π_x is the limiting probability the system is in state $\mathbf{x}$ at the optimal solution to OP.

(iii) *CTR*: The computation time ratio of the aggregate problem to the OP.

After calculating each of these performance measures in each instance, we construct 95% confidence intervals on these measures from our 30 instances; see Table 1.

Table 1 indicates that AP-Ex is computationally much less demanding than both OP and AP-Un. As mentioned in Section 4, this is primarily due to the

Table 1. Confidence Intervals on Performance Measures for AP-Ex and AP-Un

		Exponential production times ($k_i = 1, \forall i$)					
		AP-Ex			AP-Un		
b_i	$\frac{\nu}{\alpha+\nu}$	APD	WAPD	CTR	APD	WAPD	CTR
3	0.95	3.49 ± 0.34	4.75 ± 1.09	0.063 ± 0.000	2.97 ± 0.15	6.14 ± 1.18	0.814 ± 0.005
—	0.97	3.58 ± 0.42	5.06 ± 1.67	0.063 ± 0.001	3.13 ± 0.21	7.19 ± 1.49	0.811 ± 0.006
—	0.99	3.68 ± 0.68	4.86 ± 1.60	0.063 ± 0.000	4.23 ± 0.46	7.38 ± 1.38	0.812 ± 0.006
4	0.95	6.01 ± 0.49	8.38 ± 2.05	0.028 ± 0.000	4.20 ± 0.25	9.14 ± 1.52	0.807 ± 0.004
—	0.97	6.53 ± 0.61	9.33 ± 2.12	0.028 ± 0.000	4.59 ± 0.33	10.53 ± 2.02	0.809 ± 0.007
—	0.99	7.31 ± 1.01	9.46 ± 2.06	0.027 ± 0.000	6.35 ± 0.67	10.65 ± 1.91	0.807 ± 0.005
6	0.95	8.37 ± 0.75	17.56 ± 2.57	0.009 ± 0.000	7.65 ± 0.40	18.24 ± 2.43	0.818 ± 0.009
—	0.97	8.65 ± 0.86	15.20 ± 2.69	0.009 ± 0.000	9.00 ± 0.54	19.79 ± 2.59	0.815 ± 0.007
—	0.99	9.08 ± 1.50	14.04 ± 2.61	0.009 ± 0.000	13.16 ± 1.15	20.03 ± 2.43	0.812 ± 0.005
Average		6.30	9.85	0.033	6.14	12.12	0.812
		2-Erlang production times ($k_i = 2, \forall i$)					
3	0.95	3.59 ± 0.33	5.36 ± 1.19	0.075 ± 0.001	3.11 ± 0.16	7.63 ± 1.04	0.943 ± 0.012
—	0.97	3.65 ± 0.38	5.87 ± 1.22	0.075 ± 0.002	3.13 ± 0.15	8.24 ± 1.07	0.959 ± 0.007
—	0.99	3.87 ± 0.53	4.83 ± 1.49	0.075 ± 0.001	4.10 ± 0.29	8.21 ± 1.21	0.967 ± 0.021
4	0.95	5.94 ± 0.44	10.04 ± 2.48	0.031 ± 0.001	4.22 ± 0.25	10.77 ± 1.49	0.952 ± 0.011
—	0.97	6.43 ± 0.50	10.90 ± 2.29	0.031 ± 0.001	4.40 ± 0.26	11.88 ± 1.43	0.966 ± 0.007
—	0.99	7.91 ± 0.84	10.89 ± 2.20	0.031 ± 0.001	6.04 ± 0.47	11.58 ± 1.55	0.967 ± 0.010
6	0.95	8.32 ± 0.75	20.07 ± 2.59	0.010 ± 0.000	7.18 ± 0.39	20.10 ± 2.86	0.967 ± 0.012
—	0.97	8.60 ± 0.78	16.00 ± 2.31	0.010 ± 0.000	8.00 ± 0.42	22.13 ± 2.32	0.982 ± 0.007
—	0.99	8.65 ± 1.04	12.79 ± 2.52	0.010 ± 0.000	12.02 ± 0.79	21.78 ± 2.19	0.985 ± 0.007
Average		6.33	10.75	0.039	5.80	13.59	0.965

existence of many original states with zero disaggregation probability, yielding fewer original states to evaluate per iteration. The computational advantage of AP-Ex over OP increases with $\mathbf{b}$, whereas it is not affected much by discount factor and production time variability.

Table 1 also shows that although AP-Ex performs slightly worse than AP-Un with respect to APD, it has a distinct advantage with respect to a more reliable measure, WAPD: As discussed in Section 4, disaggregating the aggregate state into its two extreme original states better captures the dynamics of replenishment and fulfillment decisions by allowing for more transitions between aggregate states, and thus it tends to improve the performance of our aggregation scheme.

We intuitively expect the aggregate problem to perform worse at higher discount factors: the original problem becomes more complicated when the future costs get less discounted, and thus it is critical to treat the problem in a sophisticated manner. Our analytical error bounds in Section 5 are in line with this intuition. But, contrary to our expectations, for AP-Ex with $\mathbf{b} = (6,6,6)$, Table 1 indicates that WAPD significantly *decreases* as the discount factor increases. This could be because the optimal cost increases with the discount factor, and thus even if the aggregate-optimal cost function deviates more from the optimal, its percentage deviation can still decrease. Table 1 also shows that as expected, at a fixed discount factor, both APD and WAPD increase with $\mathbf{b}$. Last, we note that AP-Ex performs slightly better when the production times are more variable.

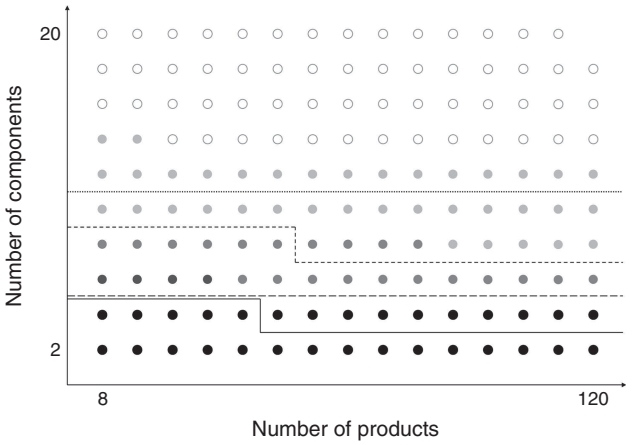
7.2. Selected Larger Instances

To evaluate the scalability of OP and AP-Ex, we analyze the computation times of OP and AP-Ex for several instances of various sizes. We generate instances in which $m \in \{2,4,\ldots,20\}$ and $n \in \{8,16,\ldots,120\}$ by randomly selecting problem parameters as in Section 7.1. We consider each instance for exponential and 2-Erlang production times and for aggregation schemes in which $\mathbf{b} \in \{(2,\ldots,2),(4,\ldots,4),(8,\ldots,8),(12,\ldots,12)\}$. We impose $\bar{\mathbf{x}}_1 = (23,\ldots,23)$ and $\nu/(\alpha+\nu) = 0.99$ in all instances. (Again, note that our instances in this subsection do not necessarily satisfy Assumptions 1 and 2.) Figure 3 exhibits the instances we could solve within five hours.

(i) *Exponential production times:* For AP-Ex, we could solve instances with 6 components when $\mathbf{b} = (2,\ldots,2)$, 8 components when $\mathbf{b} = (4,\ldots,4)$, 14 components when $\mathbf{b} = (8,\ldots,8)$, and 20 components when $\mathbf{b} = (12,\ldots,12)$. For OP, we could solve instances with up to only 4 components.

(ii) 2-*Erlang production times:* For AP-Ex, we could solve instances with 4 components when $\mathbf{b} \in \{(2,\ldots,2),(4,\ldots,4)\}$, 8 components when $\mathbf{b} = (8,\ldots,8)$, and 10 components when $\mathbf{b} = (12,\ldots,12)$. For OP, we could solve instances with 4 components but with many

Figure 3. Instances That Could Be Solved Within Five Hours



Notes. (i) When $k_i = 1$, $\forall i$, an unfilled circle is an instance solved in AP-Ex for $\mathbf{b} = (12, \ldots, 12)$, a light gray–filled circle for $\mathbf{b} \geq (8, \ldots, 8)$, a gray-filled circle for $\mathbf{b} \geq (4, \ldots, 4)$, and a dark gray–filled circle for $\mathbf{b} \geq (2, \ldots, 2)$. A black-filled circle is an instance solved in both OP and AP-Ex. (ii) When $k_i = 2$, $\forall i$, a circle below the dotted line is an instance solved in AP-Ex for $\mathbf{b} = (12, \ldots, 12)$, a circle below the dashed line for $\mathbf{b} \geq (8, \ldots, 8)$, and a circle below the long-dashed line for $\mathbf{b} \geq (2, \ldots, 2)$. A circle below the solid line is an instance solved in both OP and AP-Ex.

fewer products than those solved in AP-Ex when $\mathbf{b} \in \{(2, \ldots, 2), (4, \ldots, 4)\}$.

Figure 3 indicates that increasing the number of components has a much greater impact on the scalability of both OP and AP-Ex than increasing the number of products. This is because an increment in the number of products only has the effect of increasing the number of allocation operators in a given state by one in both OP and AP-Ex. However, an increment in the number of components leads to an exponential increase in the state space of OP at a growth rate of $k_i(\bar{x}_{1i} + 1)$ and in the state space of AP-Ex at a growth rate of $k_i(\bar{x}_{1i} + 1)/b_i$.

For the instances we could solve in OP, Tables 2 and 3 exhibit the average CTR and APD values of AP-Ex, respectively: AP-Ex outperforms OP with respect to computation times by up to one order of magnitude when $\mathbf{b} = (2, \ldots, 2)$, by up to three orders of magnitude when $\mathbf{b} = (4, \ldots, 4)$, and by up to four orders of magnitude when $\mathbf{b} \in \{(8, \ldots, 8), (12, \ldots, 12)\}$. The computational advantage of AP-Ex tends to increase with the number of components. AP-Ex deviates from OP with respect to cost by only a couple of percent when $\mathbf{b} = (2, \ldots, 2)$, by 12.2% when $\mathbf{b} = (4, \ldots, 4)$, by 41.8% when $\mathbf{b} = (8, \ldots, 8)$, and by 39.4% when $\mathbf{b} = (12, \ldots, 12)$, on average.

Our experimental setup is similar to the one proposed by Nadar et al. (2016), who develop a linear program to find the optimal average cost for Markovian ATO systems. Taking $K = 1$ and $\bar{\mathbf{x}}_1 = (10, \ldots, 10)$, they could solve instances with up to 2 components and 15 products and instances with up to 4 components and 2 products. With the help of our aggregation method, when $K = 1$, we are able to solve significantly larger instances (with up to 20 components and 112 products) with much larger bounds on inventory levels ($\bar{\mathbf{x}}_1 = (23, \ldots, 23)$ in our instances). Our aggregation method also enables us to solve many different test problems of various sizes that have been studied by many other authors under more restricted policies. See, for instance, Zhang (1997), Cheng et al. (2002), Akçay and Xu (2004), Lu and Song (2005), and Lu et al. (2005).

7.3. Initialization of the Value Iteration Algorithm via the Aggregate Solution

We now compare initializing the value iteration algorithm for the *original* problem with the aggregate-optimal cost function of AP-Ex versus initializing it with the zero function. For each instance in Section 7.1 (not necessarily satisfying Assumptions 1 and 2), we perform the following:

(i) We solve AP-Ex by the value iteration algorithm and denote the computation time by t.

(ii) We run the value iteration algorithm that starts with the zero function in OP until time t.

(iii) Denote the value iteration algorithm in OP that starts with the aggregate-optimal cost function obtained in (i) by AP-VI. We run AP-VI until the *maximum* percentage deviation from the optimal cost across all states (MAXPD) reaches D. Let I_1 denote the number of iterations performed by AP-VI.

(iv) Denote the value iteration algorithm in OP that starts with the cost function obtained at time t in (ii)

Table 2. Average CTR Values for AP-Ex in Selected Larger Instances

	$n \in \{8,16,24,32,40\}$				$n \in \{48,56,64,72,80\}$				$n \in \{88,96,104,112,120\}$			
m	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$
$k_i = 1$												
2	0.4630	0.1276	0.0447	0.0304	0.4511	0.1109	0.0303	0.0167	0.4457	0.1066	0.0278	0.0145
4	0.1160	0.0057	0.0004	0.0001	0.1206	0.0059	0.0003	0.0001	0.1098	0.0060	0.0003	0.0001
$k_i = 2$												
2	0.1038	0.0279	0.0080	0.0039	0.2018	0.0464	0.0131	0.0057	0.3102	0.0639	0.0155	0.0069
4	0.2407	0.0189	0.0012	0.0002								

Note. The instances with $k_i = 2$, $m = 4$, and $n > 40$ could not be solved in OP within five hours.

Table 3. Average APD Values for AP-Ex in Selected Larger Instances

	$n \in \{8,16,24,32,40\}$				$n \in \{48,56,64,72,80\}$				$n \in \{88,96,104,112,120\}$			
m	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$	$b_i = 2$	$b_i = 4$	$b_i = 8$	$b_i = 12$
$k_i = 1$												
2	2.745	8.493	32.297	29.175	3.357	16.600	47.464	45.402	2.658	12.980	48.531	47.231
4	1.646	11.424	35.577	30.807	1.787	11.454	45.152	43.514	1.752	10.952	46.982	45.832
$k_i = 2$												
2	2.752	8.864	31.267	29.826	3.344	16.536	47.384	45.315	2.579	12.828	48.482	47.216
4	2.429	11.859	34.647	29.468								

Note. The instances with $k_i = 2$, $m = 4$, and $n > 40$ could not be solved in OP within five hours.

by Z-VI. We run Z-VI until MAXPD again reaches D. Let I_2 denote the number of iterations performed by Z-VI.

We then take the average of I_1/I_2 from our 30 instances for $D \in \{1,5,10\}$; see Table 4.

Table 4 indicates that AP-VI is computationally better than Z-VI in all cases. Table 4 also shows that the average iteration ratio increases as MAXPD decreases: AP-VI is less beneficial percentage-wise when a more accurate cost function is desired. But AP-VI can still reduce the number of iterations by 45%, on average, even when MAXPD is 1. Another important observation from Table 4 is that AP-VI has the greatest computational benefit over Z-VI when the discount factor is 0.99 (i.e., when the number of iterations required for convergence of the algorithm is largest) and when $\mathbf{b} = (3,3,3)$ (i.e., when the distance of the aggregate solution from the optimal solution is smallest). Last, we note that AP-VI is more beneficial under more variable production times.

7.4. The Value of Implementing the Aggregate-Optimal Policy Structure

Our structural results in Section 6 can be exploited to further reduce computation for AP-Ex: the value iteration (VI) algorithm employed in Section 7.1 to solve AP-Ex starts with zero function r_0 and successively computes $Fr_0, F^2r_0, \ldots$, where $F^k r_0$ refers to k compositions of the operator F defined in Section 6. As r_0 satisfies Properties 1–3, $F^k r_0$ satisfies Properties 1–3 for all k; see Lemma 2. Thus, the policy that attains the minimum in each iteration k takes the form of LBLR. As the algorithm progresses in the kth iteration by computing $F^k r_0$ across states, extra information on the lattice-dependent base-stock and rationing levels becomes available. For instance, for a given configuration of component orders, if it is not optimal to produce a batch of component i in inventory level $\{x_{1i}\} \in \mathbb{L}(\mathbf{p},\mathbf{b})$, then the base-stock level for component i on the lattice $\mathbb{L}(\mathbf{p},\mathbf{b})$ should be no greater than $\{x_{1i}\}$. We therefore need not consider inventory levels $\{z_{1i}\} \in \mathbb{L}(\mathbf{p},\mathbf{b})$ that are greater than $\{x_{1i}\}$ that have not been visited yet in the current iteration.

We thus modify the VI algorithm: once the optimal actions are found in any state in any iteration, we eliminate actions that are inconsistent with LBLR from the action space of those states to be visited in that iteration. We also consider a variant of this modified VI algorithm where we eliminate actions that are inconsistent with not only LBLR but also properties (i)–(ix) in Theorem 2; we label these algorithms as LBLR-VI and LBLRI-VI, respectively. LBLRI-VI incorporates the interdependent thresholds across different lattices into the action elimination procedure.

We consider systems in which $m = n$, $K = 1$, $\bar{\mathbf{x}}_1 = (35,\ldots,35)$, and $\nu/(\alpha+\nu) \in \{0.95, 0.97, 0.99\}$ and aggregation schemes in which $\mathbf{b} \in \{(3,\ldots,3),(4,\ldots,4),$

Table 4. Average Iteration Number Ratios of AP-VI to Z-VI

		Exponential production times			2-Erlang production times		
b_i	$\nu/\alpha+\nu$	MAXPD = 10	MAXPD = 5	MAXPD = 1	MAXPD = 10	MAXPD = 5	MAXPD = 1
3	0.95	0.157	0.288	0.496	0.194	0.328	0.512
—	0.97	0.115	0.247	0.465	0.148	0.279	0.485
—	0.99	0.076	0.176	0.383	0.093	0.214	0.430
4	0.95	0.246	0.376	0.563	0.306	0.423	0.592
—	0.97	0.210	0.351	0.545	0.259	0.388	0.574
—	0.99	0.163	0.296	0.510	0.209	0.351	0.550
6	0.95	0.390	0.503	0.646	0.421	0.524	0.661
—	0.97	0.348	0.464	0.622	0.369	0.479	0.627
—	0.99	0.259	0.399	0.584	0.279	0.401	0.576
Average		0.218	0.345	0.535	0.253	0.376	0.556

Table 5. Average Computation Time Ratios of LBLR-VI and LBLRI-VI to VI

				$\nu/(\alpha+\nu)=0.95$		$\nu/(\alpha+\nu)=0.97$		$\nu/(\alpha+\nu)=0.99$	
				LBLR-VI	LBLRI-VI	LBLR-VI	LBLRI-VI	LBLR-VI	LBLRI-VI
(m,n)	**b**	ALL	NOL	VI	VI	VI	VI	VI	VI
(2, 2)	(3, 3)	6.26	46	0.451	0.405	0.439	0.408	0.455	0.418
—	(4, 4)	4.76	34	0.477	0.443	0.473	0.443	0.483	0.439
—	(6,6)	3.27	22	0.558	0.493	0.563	0.514	0.569	0.506
(3, 3)	(3, 3, 3)	4.35	794	0.487	0.384	0.492	0.375	0.515	0.394
—	(4, 4, 4)	3.36	434	0.525	0.422	0.524	0.421	0.551	0.429
—	(6, 6, 6)	2.37	182	0.613	0.475	0.609	0.485	0.617	0.486
(4, 4)	(3, 3, 3, 3)	3.40	12,190	0.513	0.361	0.536	0.368	0.561	0.371
—	(4, 4, 4, 4)	2.66	4,930	0.545	0.402	0.558	0.406	0.579	0.412
—	(6, 6, 6, 6)	1.93	1,342	0.625	0.465	0.636	0.467	0.662	0.481
Average				0.533	0.428	0.537	0.432	0.555	0.437

Note. ALL, average lattice length; NOL, number of lattices.

$(6,\ldots,6)\}$. We generate 10 instances for each $m \in \{2,3,4\}$, by randomly selecting a_{ij} from the set $\{1,2\}$, q_i from the set $\{1,2,3\}$, h_i from the set $\{1,3,5\}$, c_j from the set $\{50,100,150,200\}$, μ_i from the set $\{1,2\}$, and λ_j from the set $\{0.2,0.4\}$, $\forall\, i,j$. Unlike Sections 7.1–7.3, we restricted a_{ij} to be less than 3 and q_i to be less than 4 in order to implement our extreme point policy, without violating Assumption 2, in *all* instances. We solve each instance in AP-Ex using each of the VI, LBLR-VI, and LBLRI-VI algorithms, and we calculate the computation time ratios of LBLR-VI and LBLRI-VI to VI.

Table 5 exhibits the average computation time ratios for LBLR-VI and LBLRI-VI. Notice that LBLRI-VI performs no worse than LBLR-VI with respect to computation times. Also, for AP-Ex, all of the VI, LBLR-VI, and LBLRI-VI algorithms yield the same cost function in each iteration, and thus the total number of iterations remains the same across all these algorithms. Ultimately, LBLRI-VI reduces the already-low computation time of VI by 56.8% on average.

We observe from Table 5 that as **b** decreases (potentially improving the accuracy of AP-Ex) at a fixed discount factor and a fixed value of m, LBLR-VI becomes computationally more desirable. As m decreases at a fixed discount factor and a fixed value of b_i, LBLR-VI again becomes more desirable. The increasing benefit of LBLR-VI in these two cases comes from the increasing average lattice length (see Table 5): this growing number of states along any lattice provides more opportunities to eliminate those actions that are inconsistent with the threshold policies. Unlike LBLR-VI, as m increases at a fixed discount factor and a fixed value of b_i, LBLRI-VI becomes more desirable: as m increases, although the average lattice length drops, the number of lattices significantly increases (see Table 5), enabling more opportunities to eliminate those actions that are inconsistent with the interdependent thresholds across different lattices. Last, we note that the discount factor has no significant impact on the computational benefits of LBLR-VI and LBLRI-VI.

8. Concluding Remarks

We have studied state aggregation for value function approximation applied to general ATO systems. The aggregate model under our extreme point policy provides significant computational savings over the original problem while still maintaining satisfactory accuracy: the computation time of the aggregate problem is on average 96.4% lower than that of the original problem, and the average distance of the aggregate solution from the optimal solution is 6.32% on our test bed of three-component, three-product instances. Although the average distances from the optimal solution are similar under the extreme point and uniform policies, the computation time of the aggregate problem is on average only 11.2% lower than that of the original problem under the uniform policy. With the extreme point policy, we could solve instances with up to 20 components in the aggregate problem, whereas we could solve instances with up to 4 components in the original problem.

We also establish a finite error bound for the aggregate solution, which allows us to prove that the value iteration algorithm on the original problem converges to the optimal solution if it starts with the aggregate solution. Numerical experiments in the original problem reveal that the numbers of iterations performed until the maximum distance from the optimal cost across all states drops to 5% and 1% are, on average, 64% and 45% lower, respectively, in the value iteration algorithm initialized with the aggregate solution, in comparison with the standard value iteration algorithm. We last show the optimality of the LBLR policy for the aggregate problem that uses knowledge of key component and product characteristics, which allows

us to further reduce the computation time of the aggregate problem. We could solve instances with 22 components by implementing the optimal policy structure.

The structural knowledge derived from the aggregate problem may inspire and facilitate future research on characterizing the optimal policy form and/or developing near-optimal heuristic policies in the *original* problem for general ATO systems. Future extensions of this paper could also focus on large-scale computational methods for the aggregate problem, such as simulation-based policy iteration and value iteration methods. Another direction for future research is to study aggregation in more complex ATO problems that can still be modeled as MDPs with potentially larger state and/or action spaces. Examples include ATO systems with variable replenishment quantities and/or backorders. Although we considered Erlang production times in this paper, our structural results continue to hold when production times follow a Coxian$^+$ distribution having (weakly) greater termination rates in later phases, including the Erlang distribution as a special case. See Osogami and Harchol-Balter (2006) for a definition of the Coxian$^+$ distribution.

Acknowledgments

The authors thank the area editor, associate editor, and two referees for their constructive comments and suggestions. The authors are particularly grateful to the associate editor for inspiring them to construct the error bound for the aggregate problem of general ATO systems in Theorem 1.

References

Akçay Y, Xu SH (2004) Joint inventory replenishment and component allocation optimization in an assemble-to-order system. *Management Sci.* 50(1):99–116.

Atan Z, Ahmadi T, Stegehuis C, de Kok T, Adan I (2017) Assemble-to-order systems: A review. *Eur. J. Oper. Res.* 261(3):866–879.

Benjaafar S, ElHafsi M (2006) Production and inventory control of a single product assemble-to-order system with multiple customer classes. *Management Sci.* 52(12):1896–1912.

Benjaafar S, ElHafsi M, Lee CY, Zhou W (2011) Optimal control of an assembly system with multiple stages and multiple demand classes. *Oper. Res.* 59(2):522–529.

Bertsekas DP (2012) *Dynamic Programming and Optimal Control*, Vol. 2 (Athena Scientific, Nashua, NH).

Bušić A, Coupechoux E (2014) Exact simulation for assemble-to-order systems. Working paper, École normale supérieure , Paris.

Busic A, Vliegen I, Scheller-Wolf A (2012) Comparing Markov chains: Aggregation and precedence relations applied to sets of states, with applications to assemble-to-order systems. *Math. Oper. Res.* 37(2):259–287.

Cheng F, Ettl M, Lin G, Yao DD (2002) Inventory-service optimization in configure-to-order systems. *Manufacturing Service Oper. Management* 4(2):114–132.

de Véricourt F, Karaesmen F, Dallery Y (2002) Optimal stock allocation for a capacitated supply system. *Management Sci.* 48(11):1486–1501.

Doğru MK, Reiman MI, Wang Q (2010) A stochastic programming based inventory policy for assemble-to-order systems with application to the W model. *Oper. Res.* 58(4, Part 1):849–864.

Doğru MK, Reiman MI, Wang Q (2017) Assemble-to-order inventory management via stochastic programming: Chained BOMs and the *M*-system. *Production Oper. Management* 26(3):446–468.

ElHafsi M (2009) Optimal integrated production and inventory control of an assemble-to-order system with multiple non-unitary demand classes. *Eur. J. Oper. Res.* 194(1):127–142.

ElHafsi M, Camus H, Craye E (2008) Optimal control of a nested-multiple-product assemble-to-order system. *Internat. J. Production Res.* 46(19):5367–5392.

Gayon JP, Benjaafar S, de Véricourt F (2009a) Using imperfect advance demand information in production-inventory systems with multiple customer classes. *Manufacturing Service Oper. Management* 11(1):128–143.

Gayon JP, de Véricourt F, Karaesmen F (2009b) Stock rationing in an $M/E_r/1$ multi-class make-to-stock queue with backorders. *IIE Trans.* 41(12):1096–1109.

Goldberg DA, Katz-Rogozhnikov DA, Lu Y, Sharma M, Squillante MS (2016) Asymptotic optimality of constant-order policies for lost sales inventory models with large lead times. *Math. Oper. Res.* 41(3):898–913.

Ha A (1997a) Inventory rationing in a make-to-stock production system with several demand classes and lost sales. *Management Sci.* 43(8):1093–1103.

Ha A (1997b) Stock rationing policy for a make-to-stock production system with two priority classes and backordering. *Naval Res. Logist.* 44(5):457–472.

Ha A (2000) Stock rationing in an $M/E_k/1$ make-to-stock queue. *Management Sci.* 46(1):77–87.

Heyman DP, Sobel MJ (2003) *Stochastic Models in Operations Research, Volume* II*: Stochastic Optimization* (Dover Publications, Mineola, NwY).

Huang K (2014) Benchmarking non-first-come-first-served component allocation in an assemble-to-order system. *Ann. Oper. Res.* 223(1):217–237.

Kapuscinski R, Zhang RQ, Carbonneau P, Moore R, Reeves B (2004) Inventory decisions in Dell's supply chain. *Interfaces* 34(3):191–205.

Kemeny JG, Snell JL (1976) *Finite Markov Chains* (Springer-Verlag, New York).

Lippman S (1975) Applying a new device in the optimization of exponential queueing systems. *Oper. Res.* 23(4):687–710.

Lu Y, Song JS (2005) Order-based cost optimization in assemble-to-order systems. *Oper. Res.* 53(1):151–169.

Lu Y, Song JS, Yao DD (2005) Backorder minimization in multiproduct assemble-to-order systems. *IIE Trans.* 37(8):763–774.

Lu L, Song JS, Zhang H (2015) Optimal and asymptotically optimal policies for assemble-to-order n- and W-systems. *Naval Res. Logist.* 62(8):617–645.

Lu Y, Song JS, Zhao Y (2010) No-holdback allocation rules for continuous-time assemble-to-order systems. *Oper. Res.* 58(3):691–705.

Nadar E, Akan M, Scheller-Wolf A (2014) Optimal structural results for assemble-to-order generalized *M*-systems. *Oper. Res.* 62(3):571–579.

Nadar E, Akan M, Scheller-Wolf A (2016) Experimental results indicating lattice-dependent policies may be optimal for general assemble-to-order systems. *Production Oper. Management* 25(4):647–661.

Osogami T, Harchol-Balter M (2006) Closed form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation* 63(6):524–552.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (John Wiley & Sons, Hoboken, NJ).

Reiman MI, Wang Q (2012) A stochastic program based lower bound for assemble-to-order inventory systems. *Oper. Res. Lett.* 40(2):89–95.

Reiman MI, Wang Q (2015) Asymptotically optimal inventory control of assemble-to-order systems with identical lead times. *Oper. Res.* 63(3):716–732.

Rogers DF, Plante RD (1993) Estimating equilibrium probabilities for band diagonal Markov chains using aggregation and disaggregation techniques. *Comput. Oper. Res.* 20(8):857–877.

Rogers DF, Plante RD, Wong RT, Evans JR (1991) Aggregation and disaggregation techniques and methodology in optimization. *Oper. Res.* 39(4):553–582.
Song JS, Zipkin P (2003) Supply chain operations: Assemble-to-order systems. De Kok T, Graves S, eds. *Supply Chain Management: Design, Coordination, and Operation*, Handbooks in Operations Research and Management Science, Vol. 11 (Elsevier, Amsterdam), 561–596.
Topkis DM (1978) Minimizing a submodular function on a lattice. *Oper. Res.* 26(4):305–321.
Topkis DM (1998) *Supermodularity and Complementarity* (Princeton University Press, Princeton, NJ).
Tsitsiklis JN, Van Roy B (1996) Feature-based methods for large scale dynamic programming. *Machine Learning* 22(1–3):59–94.
van Jaarsveld W, Dollevoet T (2011) Spare parts inventory control for an aircraft component repair shop. Econometric Institute Report EI 2011-24, Erasmus University Rotterdam, Rotterdam, Netherlands.
van Jaarsveld W, Scheller-Wolf A (2015) Optimization of industrial-scale assemble-to-order systems. *INFORMS J. Comput.* 27(3): 544–560.
Vliegen IMH, van Houtum GJ (2009) Approximate evaluation of order fill rates for an inventory system of service tools. *Internat. J. Production Econom.* 118(1):339–351.
Wan H, Wang Q (2015) Asymptotically-optimal component allocation for assemble-to-order production-inventory systems. *Oper. Res. Lett.* 43(3):304–310.
Xin L, Goldberg DA (2016) Optimality gap of constant-order policies decays exponentially in the lead time for lost sales models. *Oper. Res.* 64(6):1556–1565.
Xu PJ, Allgor R, Graves SC (2009) Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing Service Oper. Management* 11(2):340–355.
Zhang AX (1997) Demand fulfillment rates in an assemble-to-order system with multiple products and dependent demands. *Production Oper. Management* 6(3):309–324.
Zhao Y, Simchi-Levi D (2006) Performance analysis and evaluation of assemble-to-order systems with stochastic sequential lead times. *Oper. Res.* 54(4):706–724.
Zipkin P (2015) Some specially structured assemble-to-order systems. *Oper. Res. Lett.* 44(1):136–142.

Emre Nadar is an assistant professor in the Department of Industrial Engineering at Bilkent University. His research interests include supply chain management, inventory control, and problems that lie at the interface of operations and sustainability. He won the Gerald L. Thompson Doctoral Dissertation Award in Management Science at Carnegie Mellon University in 2012. He received first prize in the POMS College of Supply Chain Management 2012 Student Paper Competition.

Alp Akcay is an assistant professor in the School of Industrial Engineering at the Eindhoven University of Technology. His research interests include aproximate dynamic programming, simulation-based optimization, and data analytics with applications in manufacturing, maintenance, and supply chain management.

Mustafa Akan is an associate professor of operations management at the Tepper School of Business, CMU. He received his PhD in managerial economics and strategy from the Kellogg School of Management in 2008. His thesis received the Best Dissertation Award of the Aviation Applications Section. He won the Gerald L. Thompson Teaching Award and the NSF CAREER Award in 2014. His research interests include pricing and revenue management, healthcare operations, matching markets, and queueing theory.

Alan Scheller-Wolf is the Richard M. Cyert Professor of Operations Management at the Tepper School of Business, where he also serves as the Senior Associate Dean of Research. His research interests include inventory theory (especially ATO systems, systems with capacities, alternative supply options, and/or perishable products), energy, service systems, computer science, stochastic processes, and queueing theory.