

# Online Appendix to: To Wave Or Not To Wave? Order Release Policies for Warehouses with an Automated Sorter

September 25, 2009

## A.1. Detailed Process Description

We first provide a detailed description of the specific process in which most of our field observations have been performed, which is typical of the generic process type described in section §1 of the paper. Figure A.1 provides a schematic layout representation.

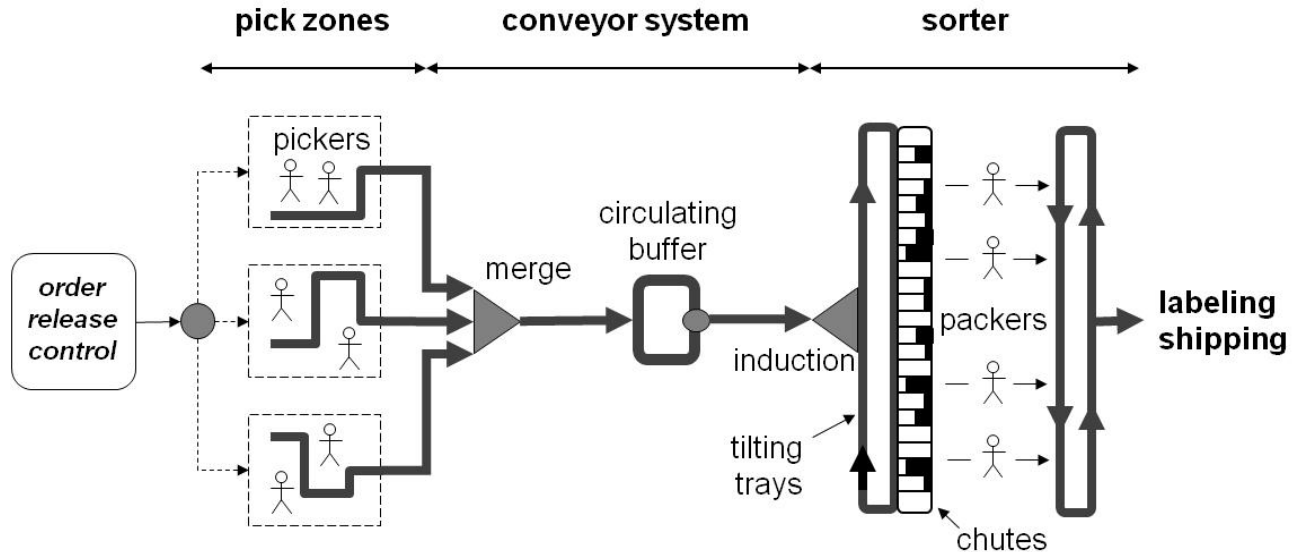


Figure A.1: Flow Diagram of the Pick-to-Ship Process in an Online Retailing Warehouse.

This process is used by an online retailer shipping directly to customers from its warehouses, so that a first key feature is the use of split-case picking and sorting, i.e., the entities being picked and circulating through the process are individual items, as opposed to cases. Also, the very large selection of items included in that firm's offering (a marketing advantage often leveraged by online retailers, who are not limited by the page and shelf space restrictions of mail order catalogs and physical stores, respectively) results in a relatively large

picking area and long item travel times to the sorter. As in many other large warehouses, the picking area is subdivided into several zones, each having its dedicated team of pickers. These workers carry portable digital 2-way wireless communication devices with a bar code scanner and an LCD screen showing the nature and location of items to be picked. Picking an item involves scanning its bar-code and placing it into a plastic tote carried by an individual rolling cart. Totes are offloaded when full onto a conveyor belt spreading through their pick zone, which relieves the pickers from unloading travel, as described in Owyong and Yih (2006). Conveyor belts carrying totes coming out of all the pick zones lead after a merge point to a *recirculating buffer* where selected totes may be temporarily held for the purpose of reducing the accumulation time of orders in sorter chutes (as in Le-Duc and de Koster 2005), or time between the arrivals of the first and last item of each order in a chute (*chute-dwell time*). The induction stations have automated coordinated induction belts and were designed using realistic throughput models of the type described in Johnson and Meller (2002), resulting in relatively high capacity and low labor costs. In this setting, packers thus constitute the second largest labor category of the outbound process after pickers. They are tasked with putting the items from any completed chute into a cardboard box of appropriate size and place it onto a conveyor leading to automated stuffing and labeling stations. Their work is guided by a light system signaling every chute as complete (green), incomplete (orange), or unassigned (no light). Finally, we point out that our partner’s warehouses use a sophisticated data collection system involving bar-code scanners carried by pickers and packers and also placed in many locations in the conveyor system, induction stations and sorter chutes. This system generates a database of detailed flow timing information for individual orders which provided many insights about the actual behavior of this process, as discussed in the next section.

## **A.2. Flow Data Analysis**

The database of order flow event timing mentioned in the previous section enables a quantitative empirical analysis of warehouse dynamics, and ultimately provided us with the distributional input data required by the quantitative models and simulation experiments discussed in sections §4-6 of the paper. A first quantity of interest that we analyzed is the empirical distribution of *transit time*, or time necessary for a given item to travel from the pick zone where it is collected to its assigned chute in the sorter. As an illustration, Figure A.2 shows

the empirical p.d.f. of transit times for items picked from a given picking zone over a 24 hour period during the peak of the 2003 season, which constitutes a representative example of the many other such distributions we have constructed.

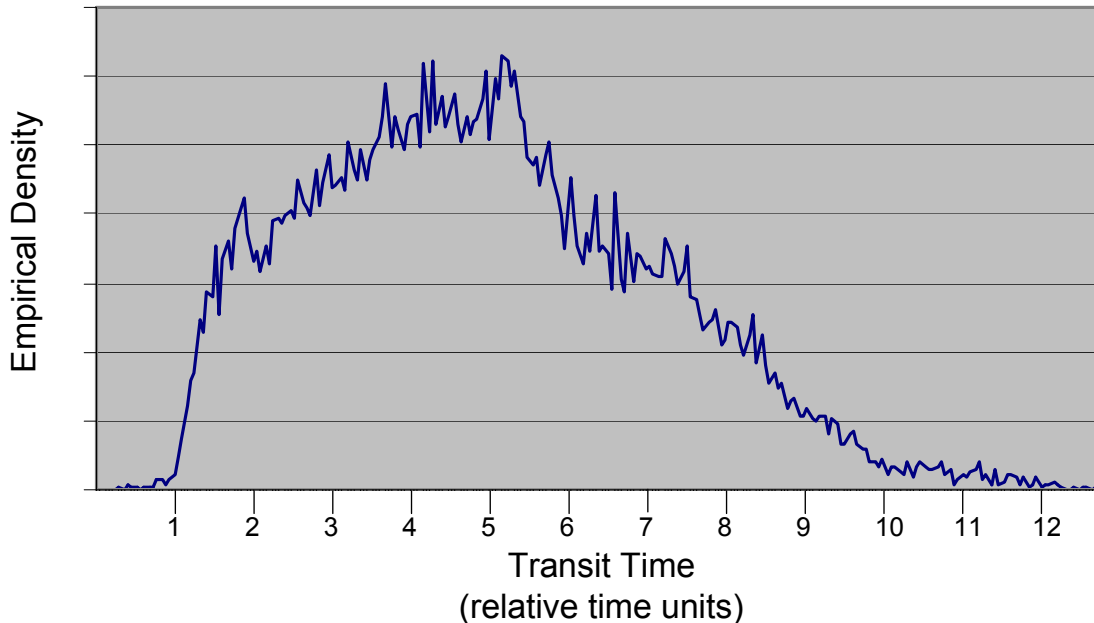


Figure A.2: Empirical Density of Transit Times from the Same Picking Zone Over 24 Hours

Note that the relative time units used for the x-axis in Figure A.2 to disguise some confidential information show a variation from 1 to 12 of the item transit times, which was typical across all picking zones over that time period. Another typical feature is that the distribution shown in Figure A.2 is multi-modal, suggesting the superposition of several heterogeneous behavioral modes. We hypothesized that this was primarily driven by conveyor congestion, and that the overall behavior or the pick-to-ship process could be characterized fairly accurately using a limited number of congestion levels, each corresponding to a range of values for the total number of items on the conveyor system between the picking area and the sorter. To verify that hypothesis, we constructed and plotted the data series representing the number of items on the entire conveyor system over time during the same 24h period, and we defined a limited number of congestion levels based on the amount of data available (indexed as in the paper as  $g \in \{1, \dots, \bar{g}\}$ ). Figure A.3 illustrates this process on a dataset which led us to define 7 congestion levels.

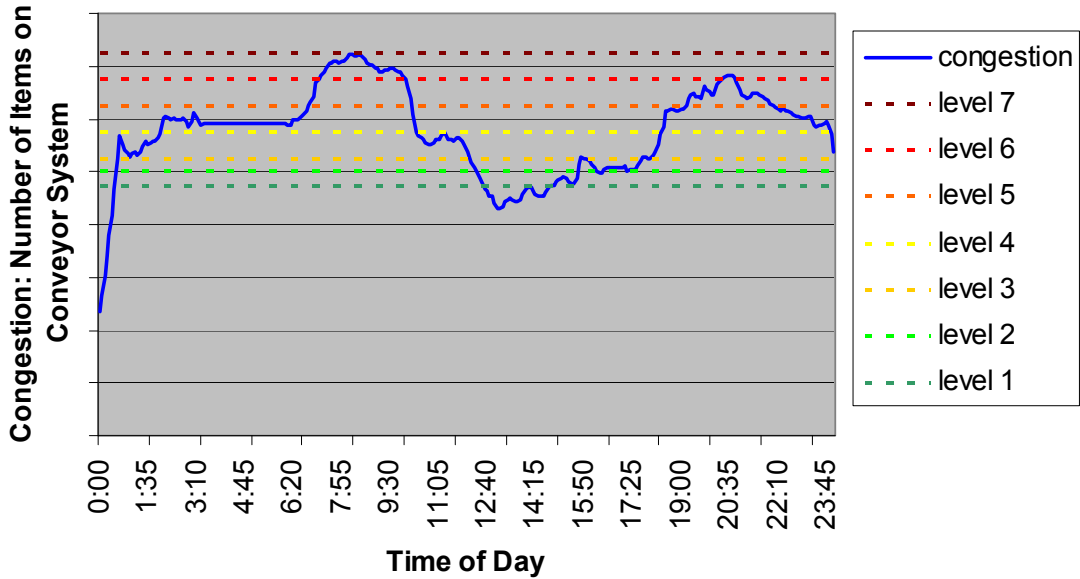


Figure A.3: Number of Items on Conveyor System Over 24 Hours

The next step is to construct the empirical item transit time distributions for each congestion level separately. That is, instead of considering all the items picked from a given picking zone over 24 hours as before, we only consider the items that were picked during the times when the system was in a given congestion level. Figure A.4 shows two such transit time distributions for the same conveyor zone as Figure A.2, and corresponding to congestion levels  $g = 2$  and 6 respectively. They also show the Gumbel (or *CMT1*) distributions with the same first two moments as the empirical distributions just defined.

These figures illustrate the following features, which we found typical of all other picking zones and congestion levels:

- These empirical transit time distributions seem (mostly) unimodal, seemingly validating at a qualitative level the hypothesis formulated earlier that the heterogeneous behaviors of transit times when observed over long periods of time can be satisfactorily explained by the variations of the system congestion level. Observe that the modes of the distributions represented in Figures A.4 (a) and (b) correspond exactly to the peaks observed on the distribution represented in Figure A.2 around the relative time values 2 and 4.5 respectively.
- The empirical transit time distributions seem to be very well fitted by *CMT1* distribu-

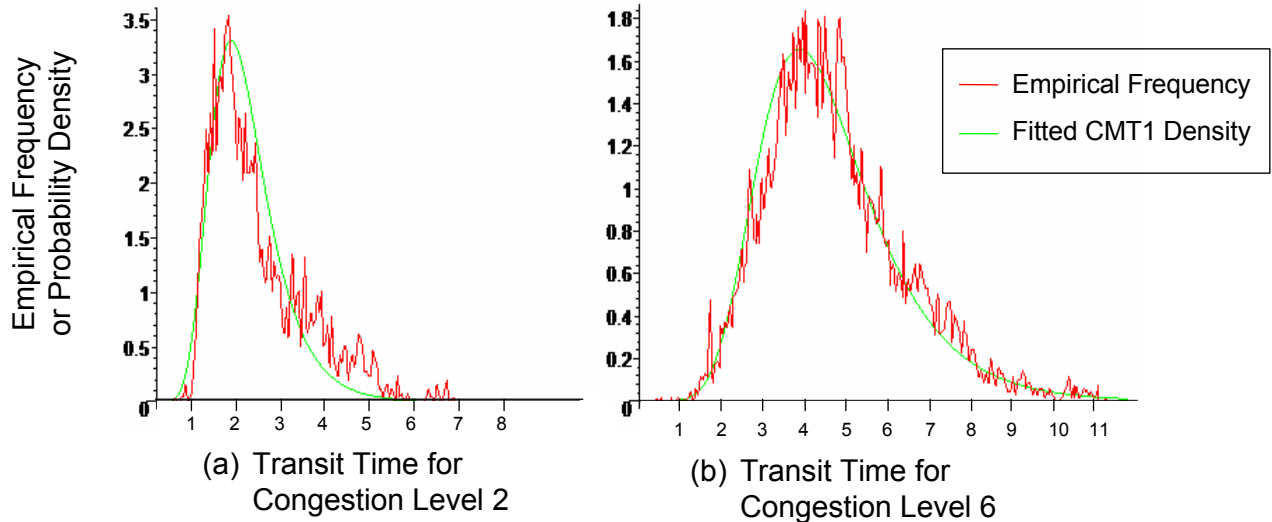


Figure A.4: Empirical Frequency and Fitted *CMT1* Density for Transit Times from a Conveyor Zone Over 24 Hours for Congestion Levels  $g = 2$  and  $g = 6$

tions. It has already been observed (Gallien and Wein 2001) that *CMT1* distributions are suitable for modeling transportation times, as their sharp left tail represents typical physical limitations of the transportation means (in the present setting, the conveyor belt speed), while their heavier right tail accounts for all the potential problems encountered along the way (here, congestion at the merge points and delays at the circulation buffer for example).

Besides its predictive validity, the notion of congestion levels just defined also generated interesting new insights about the behavior of the pick-to-ship process, as shown by examining the dependence of the empirical time-to-chute and chute-dwell time distributions on the congestion levels  $g \in \{1, \dots, \bar{g}\}$ . As illustrated by Figure A.5 (a representative example constructed with 5 congestion levels), the mean time-to-chute  $\mathbb{E}[A(g)]$  follows an unsurprising overall increasing trend with  $g$ , however the mean chute-dwell time  $\mathbb{E}[B(g)]$  exhibits a noticeable drop at an intermediary congestion level, and increases beyond that. This phenomenon (which we consistently observed on several disjoint data sets and with various congestion level definitions) can be explained by the circulating buffer (see §A.1). Specifically, this buffer includes an active tote release logic allowing to dynamically delay the arrival of selected totes to the sorter, with the goal of reducing chute-dwell time for the orders containing items in those totes. The proprietary tote delaying logic implemented (which is based on a dynamic

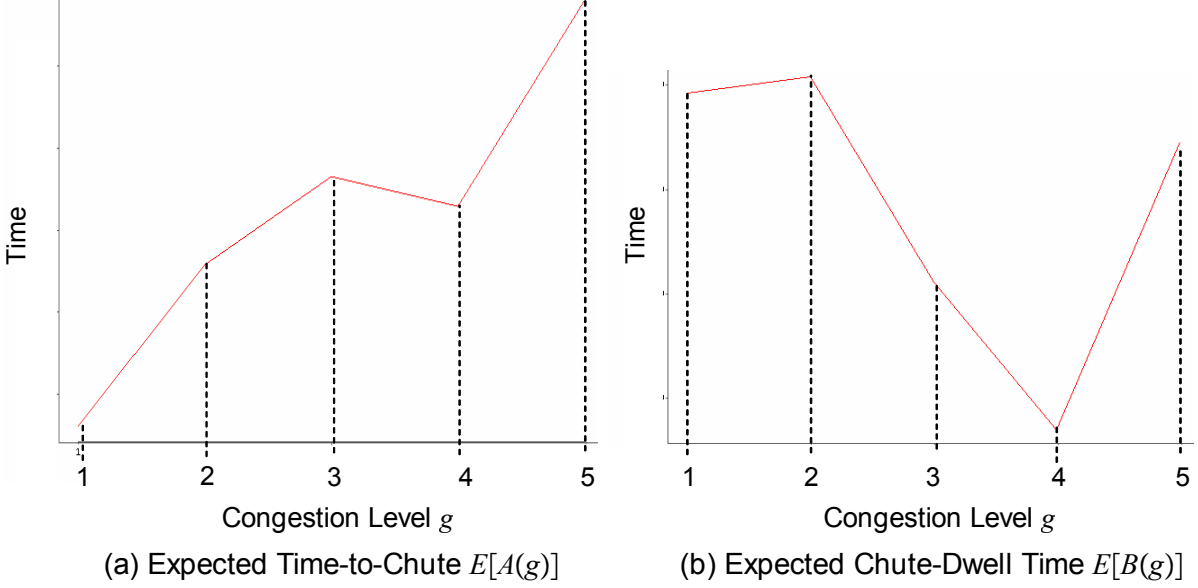


Figure A.5: Variation of Expected Time-to-Chute  $E[A(g)]$  and Chute-Dwell Time  $E[B(g)]$  with Congestion Level  $g$

priority ranking of totes) does not have a large impact for low congestion levels (such as 1 and 2 on Figure A.5 (b)). For medium to high congestion levels (3 and 4 on Figure A.5 (b)) however, the circulating buffer performs its function adequately and the active control logic implemented results in a significant reduction of the average order chute-dwell time. Finally, this buffer does have a limited capacity, so that when congestion increases further (to 5 on Figure A.5 (b)), it becomes full and, in order to preserve throughput, loses then its ability to increase the sojourn time of selected totes (think of Little’s law). The role played by this buffer also explains the slight drop of  $\mathbb{E}[A(g)]$  observed in Figure A.5 (a) at congestion level 4, although this is not nearly as significant. In summary, this data analysis uncovered the existence of a non-trivial nominal operating regime (congestion level 4 in Figure A.5) resulting from the design of this process, further motivating the development of order release control policies able to stabilize the process around it (see section §4.6 in the paper).

### A.3. Statement and Discussion of Search Algorithm.

We now describe the search algorithm that we implemented in order to compute approximations of the multiplier  $\theta$  and policy  $\lambda^\theta$  solving both  $UDP[\theta]$  and  $CDP[\beta]$ , as described in section §4.5 of the paper:

**Algorithm** SEARCH[ $\epsilon$ ]     **input:** Input data for problem  $CDP[\beta]$ , numbers  $\underline{\theta}, \bar{\theta} \geq 0$  such that  $\underline{\theta} \leq \theta^* \leq \bar{\theta}$  with  $\theta^*$  defined as in Lemma 1 in the paper.

**output:** A number  $\theta$  and policy  $\lambda^\theta$  that is near optimal for  $CDP[\beta]$ .

1. Set  $k = 1$ ,  $\underline{\theta}^k = \underline{\theta}$ ,  $\bar{\theta}^k = \bar{\theta}$ ;
2. Set  $\theta^k = \frac{\underline{\theta}^k + \bar{\theta}^k}{2}$ ; compute an optimal solution  $\lambda^{\theta^k}$  to  $UDP[\theta^k]$ , and  $\mathbf{c}^{\theta^k}(x, y, z)$ ;
3. If  $\mathbf{c}^{\theta^k}(x, y, z) > \beta$  set  $\underline{\theta}^{k+1} = \theta^k$  and  $\bar{\theta}^{k+1} = \bar{\theta}^k$ ; otherwise set  $\underline{\theta}^{k+1} = \underline{\theta}^k$  and  $\bar{\theta}^{k+1} = \theta^k$ ;
4. If  $(\bar{\theta}^{k+1} - \underline{\theta}^{k+1}) < \epsilon$ , stop, set  $\theta^f = \bar{\theta}^{k+1}$ , compute an optimal solution  $\lambda^f$  to  $UDP[\theta^f]$ , and return  $(\theta^f, \lambda^f)$ ; otherwise set  $k = k + 1$  and go to step 2.

We initialized that algorithm by setting  $\underline{\theta}$  to 0 and  $\bar{\theta}$  to the first value of a geometric sequence  $(\theta^k)_{k \in \mathbb{N}}$  such that  $\mathbf{c}^{\theta^k}(x, y, z) > \beta$ . Note that, as described in the paper, we only compute approximate solutions to the unconstrained DPs  $UDP[\theta^k]$  stated in the algorithm definition. While we were not able to develop a theoretical characterization of the convergence properties of SEARCH[ $\epsilon$ ], the following Lemma provides an a posteriori bound for the suboptimality of any policy to which it converges:

**Lemma 1** *Let  $(x, y, z)$  be the initial system state, let  $(\lambda, \theta)$  be the output of algorithm SEARCH[ $\epsilon$ ], and let  $\lambda^*$  an optimal policy for  $CDP[\beta]$ . Then  $c_\lambda(x, y, z) \leq \beta$ , i.e.  $\lambda$  is feasible for  $CDP[\beta]$ , and*

$$r_{\lambda^*}(x, y, z) - \theta(\beta - c_\lambda(x, y, z)) \leq r_\lambda(x, y, z) \leq r_{\lambda^*}(x, y, z). \quad (\text{A.1})$$

**Proof:** Let  $(\theta^*, \lambda^*)$  be as defined in the statement of Lemma 2 in the paper, and let  $(\theta, \lambda)$  be the algorithm output. Because  $\theta > \theta^*$ , from Lemma 3 in the paper we have  $r_\lambda(x, y, z) \leq r_{\lambda^*}(x, y, z)$  and  $c_\lambda(x, y, z) \leq c_{\lambda^*}(x, y, z) = \beta$ , hence  $\lambda$  is feasible. Furthermore, since the policy  $\lambda^*$  is suboptimal for  $UDP[\theta]$ ,

$$r_{\lambda^*}(x, y, z) - \theta.c_{\lambda^*}(x, y, z) \leq r_\lambda(x, y, z) - \theta.c_\lambda(x, y, z).$$

Substitution yields (A.1), completing the proof.

The intuitive interpretation for the suboptimality gap  $\theta(\beta - c_\lambda(x, y, z))$  appearing in (A.1) is that suboptimality increases when the final policy  $\lambda$  does not use all the allowed risk

provided by the model formulation, and this effect is all the more sensitive as the penalty for violating the constraint is high. That gap however can indeed only be evaluated a posteriori, since neither the final value of  $\theta$  nor the difference  $\beta - c_\lambda(x, y, z)$  are known in advance. The missing link in this characterization of convergence properties is a relationship showing that (and how)  $\theta(\beta - c_\lambda(x, y, z))$  decreases as  $\epsilon$  goes to zero, which we have unfortunately not been able to establish theoretically. In practice however, the final value of  $\theta(\beta - c_\lambda(x, y, z))$  obtained for our choice of  $\epsilon$  was always less than 2% (and in most cases, less than 1%) of  $r_{\lambda^*}(x, y, z)$ . This is not surprising because from Lemma 1 in the paper, there exists an optimal policy for  $CDP[\beta]$  which only randomizes between two actions in one state; given the very high number of states, randomization in a single one has little impact, and deterministic policies can match the desired risk value for all practical purposes.

#### A.4. Description of Approximate DP Algorithm.

We now describe the algorithm that we have implemented in order to solve approximately each instance of the dynamic program  $UDP[\theta]$  described in section §4.4 of the paper and the previous section of this Appendix<sup>1</sup>; additional background on the corresponding approximate dynamic programming methods and concepts may be found in Bertsekas and Tsitsiklis (1996).

Our first approximation consists of discretizing the state and control spaces. Specifically, we consider increasing finite sequences  $\hat{\mathbf{x}} = \{\hat{x}_i\}_{i=0}^m$ ,  $\hat{\mathbf{y}} = \{\hat{y}_j\}_{j=0}^m$ ,  $\hat{\mathbf{z}} = \{\hat{z}_k\}_{k=0}^m$ ,  $\hat{\boldsymbol{\lambda}} = \{\hat{\lambda}_i\}_{i=0}^\ell$  and the projection  $\mathcal{P} : \mathbb{N}^3 \rightarrow \hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}}$  defined such that  $\mathcal{P}(x, y, z)$  minimizes within  $\hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}}$  the rectangular distance to state  $(x, y, z) \in \mathbb{N}^3$ . The control space  $\hat{\boldsymbol{\lambda}}$  is obtained by a regular discretization  $\hat{\lambda}_i \triangleq i\bar{\lambda}/\ell$ , but our state space discretization is denser around the states that are more likely to be visited often. That is,  $\hat{\mathbf{x}}$ ,  $\hat{\mathbf{y}}$  and  $\hat{\mathbf{z}}$  are constructed such that the simulated steady state occupancy measure  $\mathbb{P}(\mathcal{P}^{-1}(\hat{x}_i, \hat{y}_j, \hat{z}_k))$  under the best constant solution to the optimization problem  $CDP[\beta]$  defined in the paper is approximately constant over  $(i, j, k)$ , subject to a maximum value constraint for the discretization step sizes  $\hat{x}_{i+1} - \hat{x}_i$ ,  $\hat{y}_{j+1} - \hat{y}_j$  and  $\hat{z}_{k+1} - \hat{z}_k$ .

Secondly, we implement a policy iteration algorithm relying on an approximate Robbins-Monro stochastic approximation scheme for the evaluation step, and Monte-Carlo simulations

---

<sup>1</sup> For notational simplicity, we omit any dependence on  $\theta$  of the functions and variables mentioned in this subsection.

for the improvement step. Starting with a policy  $\lambda^q : \hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}} \rightarrow \hat{\lambda}$  and initial value function estimates  $j_q^0, r_q^0$  and  $c_q^0$  defined over  $\hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}}$ , the evaluation step implements the recursion

$$\begin{cases} r_q^{s+1}(\hat{x}, \hat{y}, \hat{z}) = (1 - \gamma_s)r_q^s(\hat{x}, \hat{y}, \hat{z}) + \gamma_s [\lambda^q(\hat{x}, \hat{y}, \hat{z}) + \alpha.r_q^s(\mathcal{P}(x', y', z'))] \\ c_q^{s+1}(\hat{x}, \hat{y}, \hat{z}) = (1 - \gamma_s)c_q^s(\hat{x}, \hat{y}, \hat{z}) + \gamma_s [1_{\{\hat{y}+\hat{z}>n\}} + \alpha.c_q^s(\mathcal{P}(x', y', z'))] \\ j_q^{s+1}(\hat{x}, \hat{y}, \hat{z}) = r_q^{s+1}(\hat{x}, \hat{y}, \hat{z}) - \theta.c_q^{s+1}(\hat{x}, \hat{y}, \hat{z}), \end{cases} \quad (\text{A.2})$$

for all  $(\hat{x}, \hat{y}, \hat{z}) \in \hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}}$ , where  $\gamma_s \triangleq \frac{a}{b+s}$  is a diminishing step function ( $a$  and  $b$  are constant), and  $(x', y', z')$  denote a simulated realization under system (4) in the paper of variables  $(X_{t+1}, Y_{t+1}, Z_{t+1})$  given  $(X_t, Y_t, Z_t) = (\hat{x}, \hat{y}, \hat{z})$  and  $\lambda_t = \lambda^q(\hat{x}, \hat{y}, \hat{z})$ . Termination for recursion (A.2) is triggered by either  $s + 1 = n_{\text{eval}}$  or

$$\sup_{(\hat{x}, \hat{y}, \hat{z})} |j^{s+1}(\hat{x}, \hat{y}, \hat{z}) - j^s(\hat{x}, \hat{y}, \hat{z})| \leq \epsilon_1,$$

where  $n_{\text{eval}}$  is a specified maximum number of policy evaluation steps and  $\epsilon_1 > 0$  is a specified accuracy parameter. At that point,  $\mathbf{j}_q^{s+1}$  is considered an estimate for the value function  $\mathbf{j}_q$  of policy  $\lambda^q$ . The ensuing policy improvement step consists of computing

$$\lambda^{q+1}(\hat{x}, \hat{y}, \hat{z}) = \arg \max_{\lambda \in \hat{\lambda}} \left( \lambda - \theta.1_{\{\hat{y}+\hat{z}>n\}} + \alpha \frac{1}{n_{\text{mc}}} \sum_{\omega=1}^{n_{\text{mc}}} \mathbf{j}_q(\mathcal{P}(x'_\omega, y'_\omega, z'_\omega)) \right) \quad (\text{A.3})$$

for all  $(\hat{x}, \hat{y}, \hat{z}) \in \hat{\mathbf{x}} \times \hat{\mathbf{y}} \times \hat{\mathbf{z}}$ , where  $(x'_\omega, y'_\omega, z'_\omega)_{\omega=1}^{n_{\text{mc}}}$  are  $n_{\text{mc}}$  simulated realizations under system (1) in the paper of variables  $(X_{t+1}, Y_{t+1}, Z_{t+1})$  given  $(X_t, Y_t, Z_t) = (\hat{x}, \hat{y}, \hat{z})$  and  $\lambda_t = \lambda$ . The evaluation step (A.2) applied to policy  $\lambda^{q+1}$  provides then an estimate for its value function  $\mathbf{j}_{q+1}$ . At that point the main recursion loop just described is repeated (and the algorithm proceeds to another policy improvement step), unless  $q + 1 = n_{\text{improv}}$  or

$$\sup_{(\hat{x}, \hat{y}, \hat{z})} |j_{q+1}(\hat{x}, \hat{y}, \hat{z}) - j_q(\hat{x}, \hat{y}, \hat{z})| \leq \epsilon_2, \quad (\text{A.4})$$

where  $n_{\text{improv}}$  is a specified maximum number of policy improvement steps and  $\epsilon_2 > 0$  is a specified accuracy parameter.

The computational time of the algorithm just described is primarily driven by the number of simulations of system (1) in the paper that it performs, which is bounded from above by

$$n_{\text{improv}}.m^3(\ell n_{\text{mc}} + n_{\text{eval}}).$$

In our numerical experiments, we have found that with about 22,000 states ( $m^3$ ) and 100 control values ( $\ell$ ), the maximum number of improvement steps  $n_{\text{improv}}$ , evaluation steps  $n_{\text{eval}}$

and Monte-Carlo estimations  $n_{\text{mc}}$  could be chosen so that the final value of the l.h.s of (A.4), also known as the Bellman error, was no larger than 2% of the average value function upon algorithm termination. This required a computational time of about 30 minutes on a modern computer. Longer computations did lower the Bellman error further, but we observed that the resulting policy remained almost identical beyond that point. Finally, when implementing the dichotomic search over the multiplier  $\theta$  described in section §4.5 of the paper and §A.3, we found that after solving 8 to 10 instances of  $UDP[\theta]$  (or about 4 to 5 hours of computations) our suboptimality bound for the resulting policy relative to problem  $CDP[\beta]$  was always below 2% (see Lemma 1).

## References

Bertsekas, D. P. and J. N. Tsitsiklis (1996). *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific.