Management Science

Dynamic Inventory Repositioning in On-Demand Rental Networks

Saif Benjaafar, Daniel Jiang, Xiang Li, Xiaobo Li

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

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–18
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Dynamic Inventory Repositioning in On-Demand Rental Networks

Saif Benjaafar,[a],* Daniel Jiang,[b] Xiang Li,[c] Xiaobo Li[d]

[a] Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, Minnesota 55455; [b] Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15261; [c] Research and Development (R&D), Amazon, Seattle, Washington, 98109; [d] Department of Industrial Systems Engineering and Management, National University of Singapore, 117576, Singapore

Contact: saif@umn.edu, https://orcid.org/0000-0002-1949-0105 (SB); drjiang@pitt.edu, https://orcid.org/0000-0002-5388-8061 (DJ); xiang.li2@target.com (XL); iselix@nus.edu.sg, https://orcid.org/0000-0002-1909-628X (XL)

Received: October 26, 2019
Revised: January 3, 2021; June 3, 2021
Accepted: July 19, 2021
Published Online in Articles in Advance: February 10, 2022

https://doi.org/10.1287/mnsc.2021.4286

Copyright: © 2022 INFORMS

Abstract. We consider a rental service with a fixed number of rental units distributed across multiple locations. The units are accessed by customers without prior reservation and on an on-demand basis. Customers can decide on how long to keep a unit and where to return it. Because of the randomness in demand and in returns, there is a need to periodically reposition inventory away from some locations and into others. In deciding on how much inventory to reposition and where, the system manager balances potential lost sales with repositioning costs. Although the problem is increasingly common in applications involving on-demand rental services, not much is known about the nature of the optimal policy for systems with a general network structure or about effective approaches to solving the problem. In this paper, first, we show that the optimal policy in each period can be described in terms of a well-specified region over the state space. Within this region, it is optimal not to reposition any inventory, whereas, outside the region, it is optimal to reposition but only such that the system moves to a new state that is on the boundary of the no-repositioning region. We also provide a simple check for when a state is in the no-repositioning region. Second, we leverage the features of the optimal policy, along with properties of the optimal cost function, to propose a provably convergent approximate dynamic programming algorithm to tackle problems with a large number of dimensions.

History: Accepted by David Simchi-Levi, optimization.
Funding: The research of S. Benjaafar is in part funded by the US National Science Foundation (NS) [Grant SCC-1831140]. The research of X. Li is in part funded by the Singapore Ministry of Education [Grant R-266-000-128-133]. This work used the Extreme Science and Engineering Discovery Environment [NSF Grant ACI-1548562], the Bridges-2 system [NSF Grant ACI-1445606], and resources from the University of Pittsburgh Center for Research Computing.
Supplemental Material: Data and the electronic companion are available at https://doi.org/10.1287/mnsc.2021.4286.

Keywords: rental networks • inventory repositioning • optimal policies • approximate dynamic programming algorithms • stochastic dual dynamic programming

1. Introduction

We consider a rental service with a fixed number of rental units distributed across multiple locations. Customers are able to rent a unit for one or more periods without a prior reservation and without specifying, at the time of initiating the rental, the duration of the rental or the return location. That is, customers are allowed to return a unit rented at one location to any other location. We refer to such a service as being *on-demand* and *one-way*. Demand that cannot be fulfilled at the location at which it arises is considered lost and incurs a lost sales penalty (or is fulfilled through other means at an additional cost). Because of the randomness in demand, the length of the rental periods, and return locations, there is a need to periodically reposition inventory away from some locations and into others. Inventory repositioning is costly, and the cost depends on both the origin and destination of the repositioning. The service provider is interested in minimizing the lost revenue from unfulfilled demand (lost sales) and the cost incurred from repositioning inventory (repositioning cost). Note that more aggressive inventory repositioning can reduce lost sales but leads to higher repositioning cost. Hence, the firm must carefully mitigate the trade-off between demand fulfillment and inventory repositioning.

Problems with the above features are increasingly common in applications involving on-demand rental services.[1] We are particularly motivated by a variety of one-way car sharing services that allow customers to rent from one location and return to another location. Examples include the car sharing service ShareNow (a merger of formerly Car2Go and DriveNow) and similar recently launched services, such as LimePod, BlueLA, and BlueSG. These services are on-demand in the sense that they do not require an ahead-of-time reservation or the specification of a return duration ahead of use. They are one-way in the sense that customers are not required to return a vehicle to the same location from which it was rented and instead are allowed to decide on a location, among those in the network, that is most convenient. In other words, these services let customers decide on when and where to return a vehicle, with this information not necessarily shared with the rental service until the rental terminates. For example, LimePod advertises its service as being *free-floating*, offering customers the ability to walk up to any parked vehicle, initiate a rental via its mobile app, and then return the vehicle to any legal parking location within the service region at any time. Customers are charged based on the duration of the rental, with charges calculated at the time the rental terminates.

A challenge in managing these services (and other examples of on-demand rental systems) is the spatial mismatch between vehicle supply and demand that arises from the uncertainty in rental origins, duration, and destinations. Unless adequately mitigated with the periodic repositioning of inventory, the spatial mismatch between supply and demand can lead to a significant loss in revenue.

Although the problem is increasingly common in applications involving on-demand rental services,[2] not much is known about the nature of the optimal policy for systems with a general network structure or about effective approaches to solving the problem. This lack of results appears to be due to the multidimensional nature of the problem (i.e., more than one inventory location) compounded by the presence of randomness in demand, rental periods, and return locations as well as lost sales. In this paper, we address these limitations through two main contributions. The first contribution is theoretical and the second is computational:

• On the theoretical side, we offer a characterization of the optimal policy for the dynamic inventory repositioning problem in a general network setting, accounting for randomness in trip volumes, duration, origin, and destination as well as spatial and temporal dependencies (e.g., likelihood of a trip terminating somewhere being dependent on its origin as well as trip volumes that are dependent on time and location).

• On the computational side, we describe a new cutting-plane-based approximate dynamic programming (ADP) algorithm that can effectively solve the dynamic repositioning problem to near optimality. We provide a proof of convergence for our algorithm that takes a fundamentally different view from existing cutting-plane-based approaches (this can be viewed as a theoretical contribution to the ADP literature, independent from the repositioning application). We also propose a clustering-based extension to our ADP method that scales to large-scale systems and illustrates its effectiveness on problems with up to 100 locations.

Specifically, we formulate the repositioning problem as a multiperiod stochastic dynamic program. We show that the optimal policy in each period can be described in terms of two well-specified regions over the state space. If the system is in a state that falls within one region, it is optimal not to reposition any inventory (we refer to this region as "the no-repositioning" region). If the system is in a state that is outside this region, then it is optimal to reposition some inventory but only such that the system moves to a new state that is on the boundary of the no-repositioning region. Moreover, we provide a simple check for when a state is in the no-repositioning region, which also allows us to compute the optimal policy more efficiently.

One of the distinctive features of the problem considered lies in its nonlinear state update function. This nonlinearity introduces difficulties in showing the convexity of the problem that must be solved in each period. To address this difficulty, we leverage the fact that the state update function is piecewise-affine and derive properties for the directional derivatives of the value function. This approach has potential applicability to other systems with piecewise-affine state update functions.

Because of the curse of dimensionality, the optimal policy (and the value function) can be difficult to compute for problems with more than a small number of dimensions. To address this issue, we leverage the results obtained regarding the structure of both the value function and the optimal policy to construct an approximate dynamic programming algorithm. The algorithm combines aspects of approximate value iteration (see, for example, De Farias and Van Roy 2000 and Munos and Szepesvári 2008) and stochastic dual dynamic programming (see Pereira and Pinto 1991). We conduct numerical experiments to illustrate the effectiveness of jointly utilizing value function and policy structure, which, to our knowledge, has not yet been explored by related methods in the literature.

The rest of the paper is organized as follows. In Section 2, we review related literature. In Section 3, we describe and formulate the problem. In Sections 4 and 5, we give our structural results for the optimal value function and the optimal policy, respectively. Next, in

Section 6, we describe our ADP approach, along with several numerical studies. In Section 7, we address the problem of scaling the ADP algorithm to large systems via the clustering-based extension. In Section 8, we provide concluding comments.

1.1. Notation

Throughout the paper, the following notation will be used. We use $\boldsymbol{e}$ to denote a vector of all ones, $\boldsymbol{e}_i$ to denote a vector of zeros except one at the ith entry, and $\mathbf{0}$ to denote a vector of all zeros (the dimension of these vectors will be clear from the context). Also, we write $\Delta_{n-1}(M)$ to denote the $(n-1)$-dimensional simplex, that is, $\Delta_{n-1}(M) = \{(x_1, \ldots, x_n) \mid \sum_{i=1}^n x_i = M, \boldsymbol{x} \geq \mathbf{0}\}$. Similarly, we use $S_n(M)$ to denote the n-dimensional simplex with interior, that is, $S_n(M) = \{(x_1, \ldots, x_n) \mid \sum_{i=1}^n x_i \leq M, \boldsymbol{x} \geq \mathbf{0}\}$. Throughout, we use ordinary lowercase letters (e.g., x) to denote scalars and boldfaced lowercase letters (e.g., $\boldsymbol{x}$) to denote vectors. The Euclidean norm is denoted $\|\cdot\|_2$. For functions f_1 and f_2 with domain $\mathcal{X}$, let $\|f_1\|_\infty = \sup_{x\in\mathcal{X}}|f_1(x)|$ and let $f_1 \leq f_2$ denote $f_1(x) \leq f_2(x)$ for all $x \in \mathcal{X}$. We denote the boundary of a set E by $\mathcal{B}(E)$ and the interior of E by E°.

2. Literature Review

There is growing literature on inventory repositioning in car and bike sharing systems; see, for example, Nair and Miller-Hooks (2011), Shu et al. (2013), Bruglieri et al. (2014), O'Mahony and Shmoys (2015), Freund et al. (2020), Liu et al. (2016), Ghosh et al. (2017), Schuijbroek et al. (2017), Li et al. (2018), Shui and Szeto (2018), and Nyotta et al. (2019) and the references therein. Most of this literature focuses on the static repositioning problem, where the goal is to find the optimal placement of vehicles before demand arises, with no more repositioning being made afterward (e.g., repositioning overnight for the next day). The objective function in the associated optimization problem typically accounts for repositioning and user dissatisfaction costs (e.g., lost sales). Much of this work employs mixed integer programming formulations and focuses on the development of algorithms and heuristics. Similarly, the papers that focus on *dynamic* repositioning generally consider heuristic solution techniques and do not offer structural results regarding the optimal policy (see, for example, Ghosh et al. 2017 and Li et al. 2018).

A notable exception is Li and Tao (2010) who study a finite horizon problem with two locations. They show that the optimal policy in the last period is characterized by a two-limit control policy (a single dimensional version of the policy we show is true in general). They conjecture that the optimal policy has a similar structure in other periods but do not provide a proof. In this paper, we prove that this conjecture is indeed true. Moreover, we prove that a generalized version of the policy holds for problems with more than two locations.

A related stream of literature models vehicle sharing systems as closed queueing networks; see, for example, George and Xia (2011), Banerjee et al. (2021), Waserhole and Jost (2016), Banerjee et al. (2018), Braverman et al. (2019), and Benjaafar et al. (2021). This literature treats time as being continuous with demand (in the form of an arrival process) that is typically stationary. A common objective in this literature is to identify control policies that maximize a function of the amount of demand satisfied. Control levers include demand throttling (e.g., via pricing), vehicle dispatching (deciding on how to allocate available vehicles to demand as it arises), and empty vehicle repositioning. In general, optimal dynamic policies, such as the ones we consider in this paper, are difficult to characterize. Instead much of this literature relies on analyzing asymptotic regimes, when either the number of vehicles goes to infinity or both the number of vehicles and demand go to infinity. Braverman et al. (2019) consider the optimal repositioning problem in the asymptotic regime where both demand and number of vehicles are allowed to go to infinity. The resulting static repositioning policy is shown to provide an upper bound on the optimal objective function for the finite problem. The static repositioning problem is also discussed in Benjaafar et al. (2021) under a demand balance assumption and using an approximation for vehicle availability at each location.

Waserhole and Jost (2016) and Banerjee et al. (2021) consider the optimal pricing problem in the asymptotic regime when the number of vehicles goes to infinity. The resulting static pricing policy is shown to provide guaranteed bounds for the finite system. Banerjee et al. (2018) consider the optimal dispatching problem for a stylized setting under the infinite number of vehicles regime and show that the resulting policy provides strong performance bounds.

In this paper, we take a different approach by studying optimal dynamic policies that are generally state dependent. We consider a setting where time is discretized and allow for demand to be nonstationary. However, we do treat the inventory of vehicles as continuous (this is also the case under the asymptotic regimes considered in the queueing-based literature). We suspect, though we do not prove it, that the optimal dynamic policy for vehicle repositioning under a queueing model may share similar features to the optimal policy we show for our setting.

Other related papers include Chung et al. (2018), who analyze incentive-based repositioning policies for bike sharing; Bimpikis et al. (2019) and Balseiro et al. (2021), who consider the spatial aspect of pricing in ride-sharing networks, a related problem to ours; and Ma et al. (2020), who consider a setting with full information to design a spatial-temporal mechanism for prices and wages with desirable properties. Other work considers strategic issues such as fleet sizing, service region design, infrastructure planning, and user dissatisfaction; see, for example, Raviv and Kolka (2013), Jian et al. (2016), He et al. (2017), Kaspi et al. (2017), Lu et al. (2017), Freund et al. (2017), and Kabra et al. (2020). Comprehensive reviews of the literature on vehicle and bike sharing can be found in Freund et al. (2019) and He et al. (2019).

There is literature that addresses inventory repositioning that arises in other settings, including in the context of repositioning of empty containers in the shipping industry, empty railcars in railroad operations, cars in traditional car rentals, and emergency vehicles; see, for example, Lee and Meng (2015) for a comprehensive review. The literature on empty container repositioning is particularly extensive. However, that literature focuses on simple networks and relies on heuristics when considering more general problems; see, for example, Song (2005) and Li et al. (2007). To our knowledge, there are no results regarding the optimal policy for a general network. There is also extensive literature on emergency vehicle repositioning when the demand from different locations is random. Berman (1981) introduces a dynamic programming formulation of the problem. More recent work includes Maxwell et al. (2010) who describe an ADP approach and Maxwell et al. (2014) who provide a lower bound on the performance of repositioning policies; see Bélanger et al. (2019) for a comprehensive review of this literature.

The paper that is closest to ours is He et al. (2020), which was subsequent to an earlier version of this paper[3] and which considers a problem similar to ours and solves it using a robust optimization approach. Further discussion of this paper can be found in Section 4. Another subsequent paper that is similar to both our work and He et al. (2020) is Zhao et al. (2020): they derive a structural result for the special case of two locations, two periods, and no ongoing rentals when there is a fixed cost to repositioning. Zhao et al. (2020) build upon the models first introduced in our paper[4] and the work of He et al. (2020).

The problem we consider in this paper shares features with the well-studied dynamic portfolio optimization problem when there are transaction costs. The dynamic portfolio optimization problem involves periodically reallocating funds among different assets, taking into account the stochastic nature of how the value of these assets evolves over time. Constantinides (1979) shows that the structure of the optimal policy resembles that of the optimal policy we describe in this paper for the vehicle sharing problem; see also Leland (1999). That is, it is optimal to do nothing if the system state (defined by the current values of the assets) is within a specified region; otherwise, it is optimal to reallocate funds so that the system state after reallocation lies on the boundary of the do-nothing region. Eberly and Van Mieghem (1997) consider a broader class of resource/capacity allocation problems and prove a similar structure for the optimal policy; see Van Mieghem (2003) for a review of related literature. A discussion of computational approaches, bounds on the optimal solution, and heuristics can be found in Muthuraman and Kumar (2006) and Brown and Smith (2011) and the references therein. The dynamics of the problem we consider are different (e.g., the amount of demand at one location in one period affects the distribution of vehicles at other locations in future periods). Moreover, in our case, the total capacity in the system must be held constant. Our problem is neither a special case nor a generalization of the dynamic portfolio optimization problem. However, these similarities hint that both problems may belong to a more general class of problems whose optimal solution has such a feature.

Finally, there is related literature on computational methods that can solve problems with convex value functions. Some well-known cutting-plane-based approaches are the *stochastic decomposition* algorithm of Higle and Sen (1991), the *stochastic dual dynamic programming* (SDDP) method introduced in Pereira and Pinto (1991), and the *cutting plane and partial sampling* approach of Chen and Powell (1999). Our method is most closely related to SDDP, where full expectations are computed at each iteration. Linowsky and Philpott (2005), Philpott and Guan (2008), Shapiro (2011), and Girardeau et al. (2014) provide convergence analyses of SDDP; but these analyses are designed for finite-horizon problems (or two-stage stochastic programs) and rely on an exact terminal value function and/or that there only exists a finite number of cuts.

Our algorithm is most closely related to the cutting-plane methods for the infinite-horizon setting proposed in Birge and Zhao (2007) and Warrington et al. (2019). Birge and Zhao (2007) prove uniform convergence of the value function approximations to optimal value for the case of linear dynamics, given a strong condition that the cut in each iteration is computed at a state where a Bellman error criterion is approximately maximized. Computation of such a state is a *difference of convex functions* optimization problem (or a suitable approximation). Warrington et al. (2019) focus on the deterministic setting, use a fixed set of sampled states at which cuts are computed, and do not show consistency of their algorithm. Our algorithm removes these restrictions, yet we are still able to show uniform convergence to the optimal value function. In particular, our analysis allows for nonlinear dynamics and cuts to be computed at states sampled from a distribution. Furthermore, our use of policy structure (i.e., the no-repositioning region characterization) in an SDDP-like algorithm is new.

As an alternative to cutting-plane algorithms, Godfrey and Powell (2001) and Powell et al. (2004) propose methods based on stochastic approximation (see Kushner and Yin 2003) to estimate scalar or separable convex functions, where a piecewise-linear approximation is updated iteratively via noisy samples while ensuring that convexity is maintained. Nascimento and Powell (2009) extend the technique to a finite-horizon ADP setting for the problem of lagged asset acquisition (single inventory state) and provide a convergence analysis; see also Nascimento and Powell (2010). However, these methods are not immediately applicable to our situation, where the value function is multidimensional.

3. Problem Formulation

We consider a product rental network consisting of n locations and N rental units. Inventory levels are reviewed periodically; in each period, a decision is made on how much inventory to reposition away from one location to another. Inventory repositioning is costly, and the cost depends on both the origins and destinations of the repositioning. The review periods are of equal length; decisions are made over a specified planning horizon, either finite or infinite.

Demand in each period is positive and random, with each unit of demand requiring the usage of one rental unit for one or more periods, with the rental period being also random. Demand that cannot be satisfied at the location at which it arises is considered lost and incurs a lost sales penalty. A location in the context of a free-floating car sharing system may correspond to a specified geographic area (e.g., a zip code area, a neighborhood, or a set of city blocks). Units rented at one location can be returned to another. Hence, not only are rental durations random but so are return destinations. At any time, a rental unit can be either at one of the locations, available for rent, or in an "ongoing rental" state with a customer.

The sequence of events in each period is as follows. At the beginning of the period, inventory level at each location is observed. A decision is then made on how much inventory to reposition away from one location to another. Subsequently, demand is realized at each location followed by the realization of product returns. Our model assumes, for tractability, that repositioning occurs within a single review period.[5] Note that the solution we obtain is still implementable (feasible) even if the assumption regarding the repositioning time does not hold.

We index the periods by $t \in \mathbb{N}$, with $t = 1$ indicating the first period in the planning horizon. We let $\boldsymbol{x}_t = (x_{t,1}, \dots, x_{t,n}) \in \mathbb{R}^n$ denote the vector of inventory levels before repositioning in period t, where $x_{t,i}$ denotes the corresponding inventory level at location i. Our model uses continuous inventory levels for tractability. This means that the resulting repositioning decisions produced by the model will be continuous. To implement them in a discrete system, one would need to perform a rounding step.[6] Similarly, we let $\boldsymbol{y}_t = (y_{t,1}, \dots, y_{t,n}) \in \mathbb{R}^n$ denote the vector of inventory levels after repositioning in period t, where $y_{t,i}$ denotes the corresponding inventory level at location i. Note that inventory repositioning should always preserve the total on-hand inventory. Therefore, we require $\sum_{i=1}^n y_{t,i} = \sum_{i=1}^n x_{t,i}$. As we will make clear later, $\boldsymbol{x}_t$ is only a part of the state in our dynamic system. The second part of the state is the vector of ongoing rentals, defined below.

Inventory repositioning is costly and, for each unit of inventory repositioned away from location i to location j, a cost of c_{ij} is incurred. Consistent with our motivating application of a car sharing system, we assume there is a cost associated with the repositioning of each unit; see He et al. (2020) for similar treatment. Let $\boldsymbol{c} = (c_{ij})$ denote the cost matrix, and let w_{ij} denote the amount of inventory to be repositioned away from location i to location j. Then, the minimum cost associated with repositioning from an inventory level $\boldsymbol{x}$ to another inventory level $\boldsymbol{y}$ is given by the solution to the following linear program:

$$\begin{aligned} \min \quad & \boldsymbol{c} \cdot \boldsymbol{w} \\ \text{subject to} \quad & \sum_{i=1}^n w_{ij} - \sum_{k=1}^n w_{jk} = y_j - x_j \quad \forall j = 1, \dots, n \\ & \boldsymbol{w} \geq 0. \end{aligned}$$

The first constraint ensures that the change in inventory level at each location is consistent with the amounts of inventory being moved into ($\sum_i w_{ij}$) and out of ($\sum_k w_{jk}$) that location. The second constraint ensures that the amount of inventory being repositioned away from one location to another is always nonnegative so that the associated cost is accounted for in the objective. It is clear that the value of the linear program depends only on $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{x}$. Define

$$\begin{aligned} C(\boldsymbol{z}) = \min \quad & \boldsymbol{c} \cdot \boldsymbol{w} \\ \text{subject to} \quad & \sum_{i=1}^n w_{ij} - \sum_{k=1}^n w_{jk} = z_j \quad \forall j = 1, \dots, n \\ & \boldsymbol{w} \geq 0, \end{aligned} \tag{1}$$

for any $\boldsymbol{z} \in \mathcal{H}$ where $\mathcal{H} := \{\boldsymbol{z} \in \mathbb{R}^n : \sum_{i=1}^n z_i = 0\}$. Then the inventory repositioning cost from $\boldsymbol{x}$ to $\boldsymbol{y}$ is $C(\boldsymbol{y} - \boldsymbol{x})$. Without loss of generality, we assume that $c_{ij} \geq 0$

satisfy the triangle inequality (i.e., $c_{ik} \le c_{ij} + c_{jk}$ for all i, j, k).

We let $\boldsymbol{d}_t = (d_{t,1}, \ldots, d_{t,n})$ denote the vector of random demands in period t, with $d_{t,i}$ corresponding to the demand at location i. The amount of demand that cannot be fulfilled is given by $(d_{t,i} - y_{t,i})^+ = \max(0, d_{t,i} - y_{t,i})$. Let β_i denote the per unit lost sales penalty incurred in location i. Then, the total lost sales penalty incurred in period t across all locations is given by $L(\boldsymbol{y}_t, \boldsymbol{d}_t) = \sum_{i=1}^n \beta_i (d_{t,i} - y_{t,i})^+$. We assume that each product can be rented at most once within a review period, that is, rental periods are longer than review periods.

To model the randomness in the rental return process, we assume that, at the end of each period t, a random fraction $p_{t,ij}$ of products rented from location i is returned to location j for all $i, j \in \{1, 2, \ldots, n\}$, with the rest continuing to be rented. We let $\boldsymbol{P}_t$ denote the matrix of random fractions, that is,

$$\boldsymbol{P}_t = \begin{pmatrix} p_{t,11} & \cdots & p_{t,1n} \\ \vdots & \ddots & \vdots \\ p_{t,n1} & \cdots & p_{t,nn} \end{pmatrix}.$$

The ith row of $\boldsymbol{P}_t$ must satisfy $\sum_{j=1}^n p_{t,ij} \le 1$. The case where $\sum_{j=1}^n p_{t,ij} < 1$ corresponds to a setting where rentals are not immediately returned, whereas the case where $\sum_{j=1}^n p_{t,ij} = 1$ corresponds to a setting where rental periods are exactly equal to one. Let μ_t denote the joint distribution of $\boldsymbol{d}_t$ and $\boldsymbol{P}_t, i = 1, 2, \ldots, n$. We assume that the random sequence $(\boldsymbol{d}_t, \boldsymbol{P}_t)$ is independent over time, and the expected aggregate demand in each period is finite (i.e., $\int_0^\infty \sum_{i=1}^n d_{t,i}\, d\mu_t < +\infty$). However, we allow $\boldsymbol{d}_t$ and $\boldsymbol{P}_t$ to be dependent. The randomness of $\boldsymbol{P}_t$ is consistent with the on-demand nature of many rental services, where the provider does not have information regarding the return destination.

Finally, let $\gamma_{t,i}$ for $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots, T$ denote the quantity of the product rented from location i that remains outstanding at the beginning of period t. Let $\rho \in [0, 1)$ be the rate at which future costs are discounted.

The model we described above can be formulated as a Markov decision process. Fix a time period t. The system states correspond to the on-hand inventory levels $\boldsymbol{x}_t$ and the outstanding inventory levels $\boldsymbol{\gamma}_t$. The state space is specified by the $(2n-1)$-dimensional simplex, that is, $(\boldsymbol{x}_t, \boldsymbol{\gamma}_t) \in \Delta_{2n-1}(N)$. Throughout the paper, we denote $S := S_n(N)$ and $\Delta := \Delta_{2n-1}(N)$ because these notations are frequently used. Actions correspond to the vector of target inventory levels $\boldsymbol{y}_t$.

Given state $(\boldsymbol{x}_t, \boldsymbol{\gamma}_t)$, the action space is an $(n-1)$-dimensional simplex, that is, $\boldsymbol{y}_t \in \Delta_{n-1}(\boldsymbol{e}^T \boldsymbol{x}_t)$. The transition probabilities are induced by the state update function:

$$x_{t+1,i} = (y_{t,i} - d_{t,i})^+ + \sum_{j=1}^n (\gamma_{t,j} + \min(y_{t,j}, d_{t,j}))\, p_{t,ji} \quad \forall i = 1, 2, \ldots, n,\ t = 1, 2, \ldots, T$$

$$\gamma_{t+1,i} = (\gamma_{t,i} + \min(y_{t,i}, d_{t,i}))\left(1 - \sum_{j=1}^n p_{t,ij}\right) \quad \forall i = 1, 2, \ldots, n,\ t = 1, 2, \ldots, T.$$

Given a state $(\boldsymbol{x}_t, \boldsymbol{\gamma}_t)$ and an action $\boldsymbol{y}_t$, the repositioning cost is given by $C(\boldsymbol{y}_t - \boldsymbol{x}_t)$, and the expected lost sales penalty is given by $l_t(\boldsymbol{y}_t) = \int L_t(\boldsymbol{y}_t, \boldsymbol{d}_t)\, d\mu_t = \int \sum_i \beta_i (d_{t,i} - y_{t,i})^+ \, d\mu_t$. The single-period cost is the sum of the inventory repositioning cost and lost sales penalty $r_t(\boldsymbol{x}_t, \boldsymbol{\gamma}_t, \boldsymbol{y}_t) = C(\boldsymbol{y}_t - \boldsymbol{x}_t) + l_t(\boldsymbol{y}_t)$. The objective is to minimize the expected discounted cost over a specified planning horizon. In the case of a finite planning horizon with T periods, the optimality equations are given by

$$v_t(\boldsymbol{x}_t, \boldsymbol{\gamma}_t) = \min_{\boldsymbol{y}_t \in \Delta_{n-1}(\boldsymbol{e}^T \boldsymbol{x}_t)} r_t(\boldsymbol{x}_t, \boldsymbol{\gamma}_t, \boldsymbol{y}_t) + \rho \int v_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1})\, d\mu_t \tag{2}$$

for $t = 1, 2, \ldots, T$ and $v_{T+1}(\boldsymbol{x}_{T+1}, \boldsymbol{\gamma}_{T+1}) = 0$, where ρ is the discount factor introduced above.

It is useful to note that the problem to be solved in each period can be expressed in the following form:

$$v_t(\boldsymbol{x}_t, \boldsymbol{\gamma}_t) = \min_{\boldsymbol{y}_t \in \Delta_{n-1}(\boldsymbol{e}^T \boldsymbol{x}_t)} C(\boldsymbol{y}_t - \boldsymbol{x}_t) + u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t), \tag{3}$$

where

$$u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t) = \int U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)\, d\mu_t, \tag{4}$$

and

$$U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t) = L_t(\boldsymbol{y}_t, \boldsymbol{d}_t) + \rho v_{t+1}(\tau_x(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t), \tau_\gamma(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)), \tag{5}$$

where

$$\begin{aligned} \tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) &= (\boldsymbol{y} - \boldsymbol{d})^+ + \boldsymbol{P}^T(\boldsymbol{\gamma} + \min(\boldsymbol{y}, \boldsymbol{d})), \\ \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) &= (\boldsymbol{\gamma} + \min(\boldsymbol{y}, \boldsymbol{d})) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}), \end{aligned} \tag{6}$$

where $\circ$ denotes the Hadamard product (or the entrywise product), that is, $(a_1, a_2, \ldots, a_n) \circ (b_1, b_2, \ldots, b_n) = (a_1 b_1, a_2 b_2, \ldots, a_n b_n)$. The next two assumptions state some useful conditions on the return fractions $\boldsymbol{P}_t$ and the repositioning costs c_{ij}.

Assumption 1. *Let $p_{\min} \in (0, 1]$ be a constant. For every period t, there exists a random variable $p_t \in [p_{\min}, 1]$ such that $\sum_{j=1}^n p_{t,ij} = \sum_{j=1}^n p_{t,kj} = p_t, \quad \forall i, k = 1, 2, \ldots, n$. An* *equivalent statement is that* $p_{t,ij} = p_t \tilde{q}_{t,ij}$ *for some* $\tilde{q}_{t,ij}$ *where* $\sum_{j=1}^{n} \tilde{q}_{t,ij} = 1$ *for all i.*

Assumption 1 implies that the probability of a vehicle being returned in a given period does not depend on the location at which the vehicle is rented, but the distribution of the return locations does depend on the origin[7]

Assumption 2. *The repositioning costs satisfy* $\rho c_{\max} - c_{\min} \le p_{\min}(\beta_i - c_{\min})$ *for all* $i = 1, \ldots, n$, *where* $c_{\max} = \max_{i,j} c_{ij}$ *and* $c_{\min} = \min_{i,j;i \ne j} c_{ij}$.

The second assumption enforces boundedness in the difference of cost parameters, with the upper bound depending on $p_{\min}$. If $p_{\min} = 1$, where the rental duration is always one period (corresponding to the setting of He et al. 2020), the restriction reduces to $\rho c_{\max} \le \beta_i$ for all i. This means that the cost of lost sales outweighs the cost of inventory repositioning in the next period.[8] If $p_{\min} < 1$, the assumption prevents the unpleasant situation where one might want to deliberately "hide" the inventory because of the difference in the repositioning cost. It is clear from the assumption that $\rho c_{\max} - c_{\min} \le p_{\min}(\beta_i - c_{\min}) \le p_t(\beta_i - c_{\min})$ for all i.

Under Assumptions 1 and 2, we are able to show (see Section 4 and 5) that the value function in each period, consisting of the lost sales and the cost-to-go as defined next, is always convex, which is perhaps surprising given the nonlinear state update and the lost sales feature.

4. Convexity of $u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t)$

The main purpose of this section is to establish the convexity of $u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t)$ defined in (5) for all periods t. We will also show that a similar result holds for the infinite-horizon case. These results will allow us later on (see Section 5) to characterize the structure of the optimal policy. They will also be useful in developing an efficient solution procedure (see Section 6).

4.1. The Finite-Horizon Problem

In this section, we consider the finite-horizon problem discussed in Section 3. For the last period (period T), $u_T(\boldsymbol{y}_T, \boldsymbol{\gamma}_T) = l_T(\boldsymbol{y}_T)$ is clearly convex. A natural question is whether the convexity of $u_t(\cdot)$ is preserved when we consider previous periods. The main difficulty is that the state update in (6) is nonlinear. However, if we introduce an auxiliary variable $\boldsymbol{\omega} = \min\{\boldsymbol{d}, \boldsymbol{y}\}$, the state update function could be written in the following linear form:

$$\tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) = \boldsymbol{y} - \boldsymbol{\omega} + \boldsymbol{P}^T(\boldsymbol{\gamma} + \boldsymbol{\omega}),\ \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) = (\boldsymbol{\gamma} + \boldsymbol{\omega}) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}).$$

We show that we can replace the constraint $\boldsymbol{\omega} = \min\{\boldsymbol{d}, \boldsymbol{y}\}$ with $\boldsymbol{\omega} \le \min\{\boldsymbol{d}, \boldsymbol{y}\}$, which would then imply the convexity of $u_t(\cdot)$ (see Electronic Companion EC.1.3 for the analysis, given as a series of technical lemmas). Let

$$u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{\eta}) = \lim_{t \downarrow 0} \frac{u(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma} + t\boldsymbol{\eta}) - u(\boldsymbol{x}, \boldsymbol{\gamma})}{t} \tag{7}$$

denote the directional derivative of $u(\cdot)$ at $(\boldsymbol{x}, \boldsymbol{\gamma})$ along the direction $(\boldsymbol{z}, \boldsymbol{\eta})$. We call $(\boldsymbol{z}, \boldsymbol{\eta})$ a *feasible direction* at $(\boldsymbol{x}, \boldsymbol{\gamma})$ if $(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma} + t\boldsymbol{\eta}) \in \Delta$ for small enough $t > 0$. The main results are summarized in the following theorem.

Theorem 1. *Suppose Assumptions 1 and 2 hold. For* $t = 1, \ldots, T$, *both* $u_t(\cdot)$ *defined in (4) and* $v_t(\cdot)$ *defined in (2) are convex and continuous in* Δ. *Moreover, the following properties of* $u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t)$ *hold for* $t = 1, \ldots, T$:

a. $u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t) = \mathbb{E}_{\boldsymbol{d}_t, \boldsymbol{P}_t}[U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)]$ *where* $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ *can be reformulated as the following convex optimization program*

$$\begin{aligned} U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t) = \min_{\boldsymbol{\omega}, \boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}} & \sum_{i=1}^{n} \beta_i (d_{t,i} - \omega_i) + \rho v_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}) \\ \textit{subject to}\ & \boldsymbol{x}_{t+1} = \boldsymbol{y}_t - \boldsymbol{\omega} + \boldsymbol{P}_t^T(\gamma_t + \boldsymbol{\omega}), \\ & \boldsymbol{\gamma}_{t+1} = (\gamma_t + \boldsymbol{\omega}) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}), \\ & \boldsymbol{\omega} \le \boldsymbol{y}_t, \textit{ and} \\ & \boldsymbol{\omega} \le \boldsymbol{d}_t; \end{aligned} \tag{8}$$

b. $|u_t'(\boldsymbol{y}_t, \boldsymbol{\gamma}_t; \pm\boldsymbol{\eta}, \mp\boldsymbol{\eta})| \le \sum_{i=1}^{n} \beta_i \eta_i$ *for all* $(\boldsymbol{y}_t, \boldsymbol{\gamma}_t) \in \Delta$ *and any feasible direction* $(\pm\boldsymbol{\eta}, \mp\boldsymbol{\eta})$ *with* $\boldsymbol{\eta} \ge \boldsymbol{0}$;

c. $u_t'(\boldsymbol{y}_t, \boldsymbol{\gamma}_t; \boldsymbol{0}, \boldsymbol{z}) \le (\rho/2) c_{\max} \sum_{i=1}^{n} |z_i|$ *for all* $(\boldsymbol{x}, \boldsymbol{\gamma}) \in \Delta$ *and any feasible direction* $(\boldsymbol{0}, \boldsymbol{z})$ *with* $\boldsymbol{e}^T \boldsymbol{z} = 0$;

d. $u_t(\cdot)$ *is Lipschitz continuous on* Δ° *with Lipschitz constant* $(3/2)\sqrt{2n}\,\beta_{\max}$, *where* $\beta_{\max} = \max_i \beta_i$.

A comprehensive proof of Theorem 1 can be found in Electronic Companion EC.1.3. Here, we give an outline of the approach. We apply induction, starting from $v_{T+1}(\boldsymbol{y}, \boldsymbol{\gamma}) = 0$. We show in Proposition EC.1 and Proposition EC.2 in the electronic companion that if $v_{t+1}(\cdot)$ is convex and satisfies certain bounds on its directional derivatives, then for any realization of $\boldsymbol{d}_t, \boldsymbol{P}_t$, the function $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ can be reformulated as the convex program (8) and satisfies two types of bounds on its directional derivatives. The first type (item 1) shows that if we turn some of the available inventory into ongoing rentals, the reduced or enlarged cost can be upper bounded by the lost sales cost of these products. The same bound holds if we remove some of the ongoing rentals and make them available at the locations from which they were rented. The second type bound states that if we change the origin of some of the ongoing rentals (i.e., we change $\boldsymbol{\gamma}$ only), the difference in cost can be upper bounded by the product of $(\rho c_{\max}/2)$ and the one-norm of the difference in $\boldsymbol{\gamma}$. The primary reason is that the total return fraction for period t, p_t, does not depend on the origin. Therefore, the difference of costs is at most the repositioning cost in the next period. To complete the induction, we show in Proposition EC.3 that given the convexity of $u_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t)$ and aforementioned bounds on its directional derivatives, $v_t(\boldsymbol{x}_t, \boldsymbol{\gamma}_t)$ is convex and satisfies the directional derivative bounds required by Proposition EC.1 and EC.2. Finally, we show that item (b) and (c) imply item (d).

We note that although the formulation in (8) has similarities to the result in lemma 1 in He et al. (2020), our proof technique is fundamentally different. First, to show that the reformulation is exact, we need to show that if $\boldsymbol{\omega} \neq \min(\boldsymbol{d}_t, \boldsymbol{y}_t)$, we can increase some components of $\boldsymbol{\omega}$ so that the objective function is not worse while keeping it feasible. For the problem in He et al. (2020) (which corresponds to our problem with $p_{\min} = 1$), because all the outstanding cars are be returned at the end of the period, we can arrive at any state $\boldsymbol{x}_{t+1}$ by adjusting the decision variable y_t. However, when $p_{\min} < 1$, a change in $\boldsymbol{\omega}$ would result in a change in γ_{t+1}, which cannot be rebalanced by changing $\boldsymbol{y}_t$. To show the monotonicity of $\boldsymbol{\omega}$, we need the directional derivative of $v_{t+1}(\cdot)$ to satisfy delicate bounds as required in Proposition EC.1. Second, though the aforementioned bound of $v_{t+1}(\cdot)$ is clearly true for the last period, for general t, we require that the directional derivatives of $u_t(\cdot)$ satisfy the two types of bounds in Theorem 1. To show that these bounds indeed hold, we carry out careful convex analysis and induction per Proposition EC.1, Proposition EC.2, and Proposition EC.3. Above all, it is highly nontrivial to show the exactness of the reformulation in (8). Our proof technique might be of independent interest for high-dimensional inventory problems with lost sales.

An immediate consequence of the reformulation is that, if all random variables satisfy discrete distributions, the problem can be written as a large-scale linear program.

Corollary 1. *Suppose Assumptions 1 and 2 hold and $(\boldsymbol{d}_t, \boldsymbol{P}_t)$ follows a discrete distribution for all t, then the optimal policy $\boldsymbol{\pi}^* = (\pi_1^*, \dots, \pi_T^*)$ can be computed as the optimal solution to a large-scale linear program.*

Corollary 1 allows us to approximate the optimal policy by replacing each expectation with the finite sum of a few samples and solve the large-scale linear program (LP). However, because the size of the LP grows exponentially in the number of samples and the number of periods, for reasonable values of T, we can only afford to solve the problem with a single sample path. In Section 6, we use one sample (the mean demand) for each period to approximate the optimal policy and compare it with our ADP solution procedure.

4.2. The Infinite-Horizon Problem

We have shown that $u_t(\cdot)$ is convex for each period for the finite-horizon problem. Next we show that the same can be said about the stationary problem with infinitely many periods. In such a problem, we denote the common distribution for $(\boldsymbol{d}_t, \boldsymbol{P}_t)$ by μ. Similarly, we denote the common values of $L_t(\cdot)$, $l_t(\cdot)$ and $r_t(\cdot)$ by $L(\cdot)$, $l(\cdot)$ and $r(\cdot)$, respectively. We use $\boldsymbol{\pi}$ to denote a stationary policy that uses the same decision rule π in each period. Under $\boldsymbol{\pi}$, the state of the process is a Markov random sequence $\{(X_t, \Gamma_t), t = 1, 2, \dots\}$. The optimization problem can be written as a Markov decision process (MDP):

$$v(\boldsymbol{x}, \boldsymbol{\gamma}) = \min_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{x}}^{\boldsymbol{\pi}} \left\{ \sum_{t=1}^{\infty} \rho^{t-1} r(X_t, \Gamma_t, \pi(X_t, \Gamma_t)) \right\}, \tag{9}$$

where $X_1 = \boldsymbol{x}$ is the initial state of the process. Let $\tilde{v}_T(\boldsymbol{x}, \boldsymbol{\gamma}) = \min_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{x}}^{\boldsymbol{\pi}} \{\sum_{t=1}^{T} \rho^{t-1} r(X_t, \Gamma_t, \pi_t(X_t, \Gamma_t))\}$ denote the value function of a stationary problem with T periods. It is well known that the functions $\tilde{v}_T(\cdot)$ converge uniformly to $v(\cdot)$ and $v(\cdot)$ is the unique solution to

$$v(\boldsymbol{x}, \boldsymbol{\gamma}) = \min_{\boldsymbol{y} \in \Delta_{n-1}(\boldsymbol{e}^T \boldsymbol{x})} r(\boldsymbol{x}, \boldsymbol{\gamma}, \boldsymbol{y}) + \rho \int v(\tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}), \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}))\, d\mu, \tag{10}$$

where $\tau_x(\cdot)$ and $\tau_\gamma(\cdot)$ correspond to the state update functions defined in (6), that is,

$$\begin{aligned} \tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) &= (\boldsymbol{y} - \boldsymbol{d})^+ + \boldsymbol{P}^T(\boldsymbol{\gamma} + \min(\boldsymbol{y}, \boldsymbol{d})) \\ &\qquad \forall t = 1, 2, \dots, T, \text{ and} \\ \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) &= (\boldsymbol{\gamma} + \min(\boldsymbol{y}, \boldsymbol{d})) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}) \\ &\qquad \forall t = 1, 2, \dots, T. \end{aligned} \tag{11}$$

For details, the reader may refer to chapter 6 of Puterman (1994).

As in the finite-horizon version, the problem to be solved can be written in the following form:

$$v(\boldsymbol{x}, \boldsymbol{\gamma}) = \min_{\boldsymbol{y} \in \Delta_{n-1}(\boldsymbol{e}^T \boldsymbol{x})} C(\boldsymbol{y} - \boldsymbol{x}) + u(\boldsymbol{y}, \boldsymbol{\gamma}), \tag{12}$$

where

$$u(\boldsymbol{y}, \boldsymbol{\gamma}) = \int U(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P})\, d\mu, \tag{13}$$

and

$$U(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) = L(\boldsymbol{y}, \boldsymbol{d}) + \rho v(\tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}), \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P})). \tag{14}$$

Theorem 2. *Suppose Assumptions 1 and 2 hold. Both $u(\cdot)$ defined in (13) and $v(\cdot)$ defined in (12) are convex and continuous in Δ. Moreover, we have*

a. *$U(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P})$ defined in (14) can be reformulated as the following convex optimization program*

$$\begin{aligned} U(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}) = \min_{\boldsymbol{\omega}, \boldsymbol{\tau_x}, \boldsymbol{\tau_\gamma}} \quad & \sum_{i=1}^{n} \beta_i (d_i - \omega_i) + \rho v(\boldsymbol{\tau_x}, \boldsymbol{\tau_\gamma}) \\ \text{subject to} \quad & \boldsymbol{\tau_x} = \boldsymbol{y} - \boldsymbol{\omega} + \boldsymbol{P}^T(\gamma + \boldsymbol{\omega}), \\ & \boldsymbol{\tau_\gamma} = (\gamma + \boldsymbol{\omega}) \circ (\boldsymbol{e} - \boldsymbol{P}\boldsymbol{e}), \\ & \boldsymbol{\omega} \le \boldsymbol{y}, \text{and} \\ & \boldsymbol{\omega} \le \boldsymbol{d}; \end{aligned} \tag{15}$$

b. $|u'(\boldsymbol{y},\boldsymbol{\gamma};\mp\boldsymbol{\eta},\pm\boldsymbol{\eta})| \le \sum_{i=1}^{n}\beta_i\eta_i$ *for all* $(\boldsymbol{x},\boldsymbol{\gamma})\in\Delta$ *and any feasible direction* $(\mp\boldsymbol{\eta},\pm\boldsymbol{\eta})$ *with* $\boldsymbol{\eta}\ge\boldsymbol{0}$;

c. $u'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z}) \le (\rho c_{\max}/2)\sum_{i=1}^{n}|z_i|$ *for all* $(\boldsymbol{x},\boldsymbol{\gamma})\in\Delta$ *and any feasible direction* $(\boldsymbol{0},\boldsymbol{z})$ *with* $\boldsymbol{e}^T\boldsymbol{z}=0$; *and*

d. $u(\cdot)$ *is Lipschitz continuous on* Δ° *with Lipschitz constant* $(3/2)\sqrt{2n}\,\beta_{\max}$, *where* $\beta_{\max}=\max_i\beta_i$.

5. The Optimal Repositioning Policy

In this section, we characterize the structure of the optimal policy. We do so for both the finite and infinite-horizon cases. Recall that, for both cases, the repositioning problem can be stated as

$$v(\boldsymbol{x},\boldsymbol{\gamma}) = \min_{\boldsymbol{y}\in\Delta_{n-1}(\boldsymbol{e}^T\boldsymbol{x})} C(\boldsymbol{y}-\boldsymbol{x}) + u(\boldsymbol{y},\boldsymbol{\gamma}) \text{ for } (\boldsymbol{x},\boldsymbol{\gamma})\in\Delta, \tag{16}$$

where $C(\cdot)$ is the repositioning cost specified by (1) and $u(\cdot)$ is a convex and continuous function that maps Δ to $\mathbb{R}\cup\{-\infty,\infty\}$. The principal result of this section is the characterization of the optimal policy through the *no-repositioning set*, the collection of inventory levels from which no repositioning should be made. The no-repositioning set for a function $u(\cdot)$ when the outstanding inventory level is $\boldsymbol{\gamma}$ can be defined as follows:

$$\Omega_u(\boldsymbol{\gamma}) = \{\boldsymbol{x}\in\Delta_{n-1}(I): u(\boldsymbol{x},\boldsymbol{\gamma}) \le C(\boldsymbol{y}-\boldsymbol{x}) + u(\boldsymbol{y},\boldsymbol{\gamma}) \quad \forall\boldsymbol{y}\in\Delta_{n-1}(I)\},\ \forall\boldsymbol{\gamma}\in S, \tag{17}$$

where $I = N-\sum_{i=1}^{n}\gamma_i$. Note that I is a function of $\boldsymbol{\gamma}$ (or equivalently $\boldsymbol{x}$). For notational simplicity, we suppress the dependency of I on $\boldsymbol{\gamma}$ (or $\boldsymbol{x}$). By definition, no repositioning should be made from inventory levels inside $\Omega_u(\boldsymbol{\gamma})$. In the following theorem, we show that $\Omega_u(\boldsymbol{\gamma})$ is nonempty, connected, and compact; and, for inventory levels outside $\Omega_u(\boldsymbol{\gamma})$, it is optimal to reposition to some point on the boundary of $\Omega_u(\boldsymbol{\gamma})$. Recall that we denote the boundary of a set E by $\mathcal{B}(E)$ and the interior of E by E°.

Theorem 3. *The no-repositioning set* $\Omega_u(\boldsymbol{\gamma})$ *is nonempty, connected, and compact for all* $\boldsymbol{\gamma}\in S$. *An optimal policy* π^* *to* (16) *satisfies*

$$\begin{aligned} \pi^*(\boldsymbol{x},\boldsymbol{\gamma}) &= \boldsymbol{x} && \text{if } \boldsymbol{x}\in\Omega_u(\boldsymbol{\gamma});\\ \pi^*(\boldsymbol{x},\boldsymbol{\gamma}) &\in \mathcal{B}(\Omega_u(\boldsymbol{\gamma})) && \text{otherwise.}\end{aligned} \tag{18}$$

Solving a nondifferentiable convex program such as (16) usually involves some computational effort. One way to reduce this effort, suggested by Theorem 3, is to characterize the no-repositioning set $\Omega_u(\boldsymbol{\gamma})$. Characterizing the no-repositioning region can help us identify when a state is inside $\Omega_u(\boldsymbol{\gamma})$, which allows our ADP algorithm to more easily compute the value iteration step; see Section 6. Recall that $u'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{z},\boldsymbol{\eta}) = \lim_{t\downarrow 0}\frac{u(\boldsymbol{x}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta})-u(\boldsymbol{x},\boldsymbol{\gamma})}{t}$ denotes the directional derivative of $u(\cdot)$ at $(\boldsymbol{x},\boldsymbol{\gamma})$ along the direction $(\boldsymbol{z},\boldsymbol{\eta})$. Because $u(\cdot)$ is assumed to be convex and continuous in Δ, $u'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{z},\boldsymbol{\eta})$ is well defined for $(\boldsymbol{x},\boldsymbol{\gamma})\in\Delta$. Recall also that $(\boldsymbol{z},\boldsymbol{\eta})$ is a *feasible direction* at $(\boldsymbol{x},\boldsymbol{\gamma})$ if $(\boldsymbol{x}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta})\in\Delta$ for small enough $t>0$. In what follows, we provide a series of first-order characterizations of $\Omega_u(\boldsymbol{\gamma})$, the first of which relies on the directional derivatives.

Proposition 1. *The inventory level* $\boldsymbol{x}\in\Omega_u(\boldsymbol{\gamma})$ *if and only if*

$$u'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{z},\boldsymbol{0}) \ge -C(\boldsymbol{z}) \tag{19}$$

for any feasible direction $(\boldsymbol{z},\boldsymbol{0})$ *at* $(\boldsymbol{x},\boldsymbol{\gamma})$.

Proposition 1 is essential for several subsequent results. However, using Proposition 1 to verify whether a point lies inside the no-repositioning set is computationally impractical, as it involves checking an infinite number of inequalities in the form of (19). In the following proposition, we provide a second characterization of $\Omega_u(\boldsymbol{\gamma})$ using the subdifferentials. Before we proceed, we introduce the following notations. $\boldsymbol{g}$ is said to be a subgradient of $u(\cdot,\boldsymbol{\gamma})$ at $\boldsymbol{x}$ if $u(\boldsymbol{y},\boldsymbol{\gamma})\ge u(\boldsymbol{x},\boldsymbol{\gamma})+\boldsymbol{g}^T(\boldsymbol{y}-\boldsymbol{x})$ for all $\boldsymbol{y}$. The set of all subgradients of $u(\cdot,\boldsymbol{\gamma})$ at $\boldsymbol{x}$ is denoted by $\partial_x u(\boldsymbol{x},\boldsymbol{\gamma})$. It is well known that $\partial_x u(\boldsymbol{x},\boldsymbol{\gamma})$ is nonempty, closed, and convex for $\boldsymbol{x}$ in the interior, which is equivalent to $\mathbf{x}>\boldsymbol{0}$ in our setting.

Proposition 2. *The inventory level* $\boldsymbol{x}\in\Omega_u(\boldsymbol{\gamma})$ *if*

$$\partial_x u(\boldsymbol{x},\boldsymbol{\gamma})\cap\mathcal{G}\ne\emptyset, \tag{20}$$

where $\mathcal{G}=\{(g_1,\dots,g_n): g_i-g_j\le c_{ij}\ \forall i,j\}$. *If* $\boldsymbol{x}>\boldsymbol{0}$, *then the converse is also true.*

Proposition 2 suggests whether a point lies inside the no-repositioning set depends on whether $u(\cdot,\boldsymbol{\gamma})$ has certain subgradients at this point. Such a characterization is useful if we can compute the subdifferential $\partial_x u(\boldsymbol{x},\boldsymbol{\gamma})$. In particular, if $u(\cdot,\boldsymbol{\gamma})$ is differentiable at $\boldsymbol{x}$, then $\partial_x u(\boldsymbol{x},\boldsymbol{\gamma})$ consists of a single point $\nabla_x u(\boldsymbol{x},\boldsymbol{\gamma})$. In this case, determining its optimality only involves checking $n(n-1)$ inequalities.

Corollary 2. *Suppose* $u(\cdot,\boldsymbol{\gamma})$ *is differentiable at* $\boldsymbol{x}\in\Delta_{n-1}(I)$. *Then,* $\boldsymbol{x}\in\Omega_u(\boldsymbol{\gamma})$ *if and only if*

$$\frac{\partial u(\boldsymbol{x},\boldsymbol{\gamma})}{\partial x_i} - \frac{\partial u(\boldsymbol{x},\boldsymbol{\gamma})}{\partial x_j} \le c_{ij} \tag{21}$$

for all i, j.

The no-repositioning set $\Omega_u(\boldsymbol{\gamma})$ can take on many forms. We first discuss the case where there are only two locations. In this case, the no-repositioning set corresponds to a closed line segment with the boundary being the two end points. The optimal policy reduces to a state-dependent two-threshold policy.

Corollary 3. *Suppose* $n=2$. *For* $\boldsymbol{\gamma}\in S$, *let* $I=N-\gamma_1-\gamma_2$. *Then* $\Omega_u(\boldsymbol{\gamma})=\{(x,I-x): x\in[s_1(\boldsymbol{\gamma}),s_2(\boldsymbol{\gamma})]\}$, *where* $s_1(\boldsymbol{\gamma})$

$= \inf\{x : u'((x, I-x, \gamma_1, \gamma_2); (1,-1,0,0)) \geq -c_{21}\}$ *and* $s_2(\boldsymbol{\gamma}) = \sup\{x : -u'((x, I-x, \gamma_1, \gamma_2); (-1,1,0,0)) \leq c_{12}\}$. *An optimal policy* π^* *to* (16) *satisfies*

$$\begin{aligned} &\pi^*(x, I-x, \gamma_1, \gamma_2) = (s_1(\boldsymbol{\gamma}), I - s_1(\boldsymbol{\gamma})) && \textit{if } x < s_1(\boldsymbol{\gamma}),\\ &\pi^*(x, I-x, \gamma_1, \gamma_2) = (x, I-x) && \textit{if } s_1(\boldsymbol{\gamma}) \leq x < s_2(\boldsymbol{\gamma}),\\ &\pi^*(x, I-x, \gamma_1, \gamma_2) = (s_2(\boldsymbol{\gamma}), I - s_2(\boldsymbol{\gamma})) && \textit{otherwise}. \end{aligned}$$

Corollary 3 is a direct consequence of Theorem 3, Proposition 1, and the fact that there are only two feasible directions. It shows that the optimal policy to Problem (16) in the two-dimensional case is described by two thresholds $s_1(\boldsymbol{\gamma}) < s_2(\boldsymbol{\gamma})$ on the on-hand inventory level x at location 1. If x is lower than s_1, it is optimal to bring the inventory level up to s_1 by repositioning inventory from location 2 to location 1. On the other hand, if x is greater than s_2, it is optimal to bring the inventory level at location 1 down to s_2. When x falls between s_1 and s_2, it is optimal not to reposition as the benefit of inventory repositioning cannot offset the cost.

When there are more than two locations, a threshold policy is not naturally defined because of the total inventory constraint. In what follows, we characterize the no-repositioning set for two important special cases, the first of which corresponds to when $u(\cdot, \boldsymbol{\gamma})$ is a convex quadratic function. If the demands are uniformly distributed, then for the last period, $u(\cdot, \boldsymbol{\gamma})$ is a quadratic function because only the lost sales cost is involved. In this case, the no-repositioning set is a polyhedron defined by $n(n-1)$ linear inequalities.

Example 1. For a fixed γ, suppose $u(\boldsymbol{y}, \boldsymbol{\gamma}) = \boldsymbol{y}^T \boldsymbol{B}(\boldsymbol{\gamma})\boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{b}(\boldsymbol{\gamma}) + b_0(\boldsymbol{\gamma})$ and $\boldsymbol{B}(\boldsymbol{\gamma})$ is positive semidefinite. By Corollary 2, $\Omega_u(\boldsymbol{\gamma}) = \{\boldsymbol{y} \in \Delta_{n-1}(I) : 2\boldsymbol{y}^T B_i(\boldsymbol{\gamma}) + b_i(\boldsymbol{\gamma}) - 2\boldsymbol{y}^T B_j(\boldsymbol{\gamma}) - b_j(\boldsymbol{\gamma}) \leq c_{ij}\ \forall i, j\}$, where $B_i(\boldsymbol{\gamma})$ is the i-th row of $\boldsymbol{B}(\boldsymbol{\gamma})$.

We point out that, in general, the no-repositioning set can be nonconvex. The following example illustrates that even if $u(\cdot)$ is smooth, $\Omega_u(\boldsymbol{\gamma})$ might still be nonconvex.

Example 2. Suppose $\boldsymbol{\gamma} = \boldsymbol{0}$, $u(\boldsymbol{y}) = y_1^3 + y_2^2 + y_3^2$, and $c_{ij} = 0.5$ (note that the inventory state $\boldsymbol{y}$ is always nonnegative so u is convex). Then, the no-repositioning set is characterized by $\Omega_u = \{\boldsymbol{y} \in \Delta_{n-1} : -0.5 \leq 3y_1^2 - 2y_3 \leq 0.5,\ -0.5 \leq 3y_1^2 - 2y_2 \leq 0.5,\ -0.5 \leq 2y_2 - 2y_3 \leq 0.5\}$.

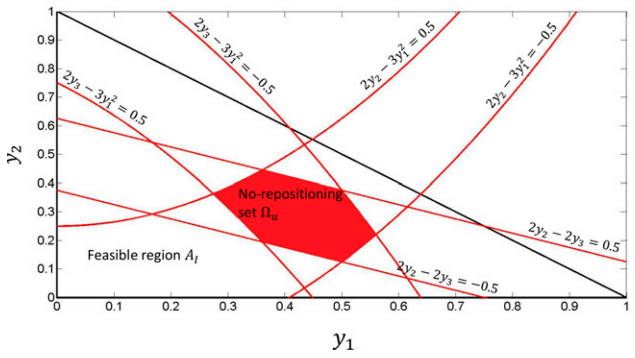
Note that in Example 2, $u(\boldsymbol{y})$ is a convex function but the no-repositioning set is not convex because the region under the parabolas $2y_2 - 3y_1^2 = 0.5$ and $2y_3 - 3y_1^2 = 0.5$ is not convex. See Figure 1 for the case where $N = y_1 + y_2 + y_3 = 1$.

6. An Approximate Dynamic Programming Approach

So far, we have studied the theoretical properties of the repositioning problem. In this section, we propose an approximate dynamic programming (ADP) algorithm which we call Repositioning-ADP (R-ADP) that exploits the structure of both the value function and the optimal policy under a sampled demand and return model. Although Theorems 1 and 2 allow for the use of convex optimization to help resolve the issue of a multidimensional continuous action space, the difficulty of a multidimensional and continuous state space remains. We refer readers to Bertsekas and Tsitsiklis (1996) and Powell (2007) for a detailed discussion of the computational challenges and solution methods associated with large MDPs. In particular, we note that when the problem size (the number of locations or the number of time periods) is large, simple approximations of continuous problems, such as discretization or aggregation, will usually fail. In addition, discretization can cause our structural properties to break down, which means the convexity result and characterization of the optimal policy given Theorems 1 and 2 can no longer be readily used. Informal numerical experiments show that even if we do not consider the ongoing rentals (rental period is always one), approximating the dynamic program via discretization within a reasonable accuracy is already a formidable task for a three-location problem.

It is thus necessary for us to consider more scalable techniques. A key feature of the algorithm we describe next is that each iteration involves solving one or more linear programs, allowing it to leverage the scalability and computational advantages of off-the-shelf solvers. We show via numerical experiments that the algorithm can produce high-quality solutions on problems with states up to 19 dimensions (10 locations) within a reasonable amount of time. The algorithm also possesses the important theoretical property of asymptotically optimal value function approximations; see Theorem 4. In the rest of this section, we motivate and describe the algorithm, prove its convergence, discuss some practical considerations, and present the numerical results.

Figure 1. (Color online) An Illustration of a Nonconvex No-Repositioning Set



6.1. The R-ADP Algorithm

Theorems 1 and 2 describe the most important feature of our dynamic program, that $u(\cdot)$, the summation of current period lost sales and the cost-to-go, is convex and continuous. Moreover, Proposition 2 provides a characterization of when it is optimal not to reposition. Our algorithm takes advantage of these two structural results. It is well known that a convex function can be written as the point-wise supremum of its tangent hyperplanes, that is,

$$u(\boldsymbol{y},\boldsymbol{\gamma}) = \sup_{\hat{\boldsymbol{y}},\hat{\boldsymbol{\gamma}}} u(\hat{\boldsymbol{y}},\hat{\boldsymbol{\gamma}}) + (\boldsymbol{y}-\hat{\boldsymbol{y}})^T \nabla_{\boldsymbol{y}} u(\hat{\boldsymbol{y}},\hat{\boldsymbol{\gamma}}) + (\boldsymbol{\gamma}-\hat{\boldsymbol{\gamma}})^T \nabla_{\boldsymbol{\gamma}} u(\hat{\boldsymbol{y}},\hat{\boldsymbol{\gamma}}).$$

This suggests that we can build an approximation to $u(\cdot)$ by iteratively adding lower-bounding hyperplanes, with the hope that the approximation becomes arbitrarily good when enough hyperplanes are considered. This is the main idea of the algorithm, with special considerations made to account for the complicated structure of the state update functions. Using a lower, piecewise-affine approximation of a convex function is a commonly used idea in stochastic programming; see, for example, figure 1 of Philpott and Guan (2008) for an illustration.

Our algorithm is motivated by various aspects of approximate value iteration (see De Farias and Van Roy 2000 and Munos and Szepesvári 2008) and stochastic dual dynamic programming (see Pereira and Pinto 1991). The features and analysis that distinguish our ADP algorithm from previous work in the literature are summarized below.

1. Our algorithm has the ability to skip the optimization step when a sampled state is detected as being in the no-repositioning region. This step uses Proposition 2, and it is applied to the value function approximation at every iteration.

2. The underlying model of SDDP and other cutting-plane methods (see, e.g., Higle and Sen 1991, Pereira and Pinto 1991, Birge and Zhao 2007) is typically a two-stage or multistage stochastic *linear* program. In our case, we have nonlinear state updates, which makes the optimization step of the algorithm difficult. To sidestep this difficulty, in our algorithm, we approximate $u(\cdot)$ instead of $v(\cdot)$, while computing the state updates outside of the optimization step.

3. Our algorithm is designed for the infinite-horizon setting, where each approximation "bootstraps" from the previous approximation and convergence is achieved despite the absence of a terminal condition, such as "$v_{T+1} \equiv 0$" used in the finite-horizon case. As such, the convergence analyses used in Chen and Powell (1999), Linowsky and Philpott (2005), Philpott and Guan (2008), Shapiro (2011), and Girardeau et al. (2014) do not apply.[9] Moreover, we remove a strong condition used in a previous convergence result by Birge and Zhao (2007) for the infinite-horizon setting, where cuts are computed at states that approximately maximize a Bellman error criterion. Selecting such a state requires solving a difference of convex functions optimization problem. Our algorithm and proof technique do not require this costly step.

Throughout this section, suppose that we are given M samples of the demand and the return fraction matrix $(\boldsymbol{d}_1,\boldsymbol{P}_1),(\boldsymbol{d}_2,\boldsymbol{P}_2),\ldots,(\boldsymbol{d}_M,\boldsymbol{P}_M)$. Our goal is to optimize the sampled model. The idea is to start with an initial piecewise-affine lower approximation $u_0(\boldsymbol{y},\boldsymbol{\gamma})$ (such as $u_0(\boldsymbol{y},\boldsymbol{\gamma})=0$) and then dynamically add linear functions (referred to as cuts in our discussion) into consideration. Suppose we currently have $u_J(\boldsymbol{y},\boldsymbol{\gamma}) = \max_{k=1,\ldots,N_J} g_k(\boldsymbol{y},\boldsymbol{\gamma})$, where $g_k(\boldsymbol{y},\boldsymbol{\gamma}) = (\boldsymbol{y}-\boldsymbol{y}_k)^T \boldsymbol{a}_k + (\boldsymbol{\gamma}-\boldsymbol{\gamma}_k)^T \boldsymbol{b}_k + \boldsymbol{\iota}_k$, and N_J is the total number of cuts in the approximation after iteration J. We then need to evaluate the functional value and the gradient of the following function: $\bar{u}_J(\boldsymbol{y},\boldsymbol{\gamma}) = \frac{1}{M}\sum_{s=1}^{M} \{L(\boldsymbol{y},\boldsymbol{d}_s) + \rho\bar{v}_J(\tau_{\boldsymbol{x}}(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d}_s,\boldsymbol{P}_s),\tau_{\boldsymbol{\gamma}}(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d}_s,\boldsymbol{P}_s))\}$, where

$$\bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta}) = \min_{z\in\Delta_{n-1}(e^T x)} C(z-x) + u_J(z,\boldsymbol{\zeta}) \tag{22}$$

at a sample point $(\tilde{\boldsymbol{y}},\tilde{\boldsymbol{\gamma}})$. Note that $\bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta})$ is a linear program. To find out the derivatives of $\bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta})$, we write down the dual formulation for $\bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta})$ as follows:

$$\begin{aligned} \bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta}) = \max\ & (\lambda_0\boldsymbol{e}+\boldsymbol{\lambda})^T\boldsymbol{x} + \sum_{k=1}^{J}\mu_k\Big(-\boldsymbol{a}_k^T\boldsymbol{y}_k + \boldsymbol{b}_k^T(\boldsymbol{\zeta}-\boldsymbol{\gamma}_k)+\boldsymbol{\iota}_k\Big)\\ \text{s.t. } & \sum_{k=1}^{J}\mu_k = 1,\\ & \lambda_i-\lambda_j \le c_{ij},\ \ \forall i,j=1,2,\ldots,n,\\ & -\lambda_i + \sum_{k=1}^{J}\mu_k a_{ki} - \lambda_0 \ge 0,\ \ \forall i=1,2,\ldots,n,\\ & \mu_k \ge 0,\ \ \forall k=1,2,\ldots,N_J. \end{aligned} \tag{23}$$

From (23), we understand that $\nabla_x \bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta}) = \lambda_0^*\boldsymbol{e}+\boldsymbol{\lambda}^*$ and $\nabla_\zeta \bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta}) = \sum_{k=1}^{J}\mu_k^*\boldsymbol{b}_k$, where $(\lambda_0^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)$ is an optimal solution for Problem (23). The Jacobian matrix for the state update function is

$$\nabla_{\bar{x},y} = \mathrm{Diag}(1_{y_t>d_k}) + \boldsymbol{P}_k \circ (1_{y_t\le d_k}\boldsymbol{e}^T),\ \nabla_{\bar{x},\gamma} = \boldsymbol{P}_k,$$
$$\nabla_{\bar{\gamma},\gamma} = \mathrm{Diag}(\boldsymbol{e}-\boldsymbol{P}_k\boldsymbol{e}),\ \text{and}\ \nabla_{\bar{\gamma},y} = \mathrm{Diag}((\boldsymbol{e}-\boldsymbol{P}_k\boldsymbol{e})\circ 1_{y_t\le d_k}),$$

where $\bar{x}$ and $\bar{\gamma}$ stand for $\tau_{\boldsymbol{x}}$ and $\tau_{\boldsymbol{\gamma}}$, respectively. By computing Equation (23) for all pairs of $(\tau_{\boldsymbol{x}}(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d}_i,\boldsymbol{P}_i),\tau_{\boldsymbol{\gamma}}(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d}_i,\boldsymbol{P}_i))$, we can find the tangent hyperplane of $\bar{u}_J(\boldsymbol{y},\boldsymbol{\gamma})$ at $(\tilde{\boldsymbol{y}},\tilde{\boldsymbol{\gamma}})$.

Although (23) can always be solved, we can apply Proposition 2, a characterization of the no-repositioning region, to reduce the computational load. We first define some terms for $u_J(\boldsymbol{y},\boldsymbol{\gamma})$. Let $\mathcal{K}=\{k\,|\,a_{ki}-a_{kj}\le c_{ij}\ \ \forall i,j\}$ denote the set of cuts that satisfy the no-reposition condition. We also let

$$\begin{aligned}D^k=\{(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta\,|\,&(\boldsymbol{y}-\boldsymbol{y}_k)^T\boldsymbol{a}_k+(\boldsymbol{\gamma}-\boldsymbol{\gamma}_k)^T\boldsymbol{b}_k+\boldsymbol{\iota}_k\\ &\ge(\boldsymbol{y}-\boldsymbol{y}_l)^T\boldsymbol{a}_l+(\boldsymbol{\gamma}-\boldsymbol{\gamma}_l)^T\boldsymbol{b}_l\\ &+c_l\ \ \forall l=1,2,\dots,K\}\end{aligned}\tag{24}$$

denote a subset of the feasible region that is dominated by the k-th cut. Then we have the following lemma.

Lemma 1. *If $\boldsymbol{x}\in D^k$ with $k\in\mathcal{K}$, we have $\boldsymbol{x}\in\Omega_{u_J}(\boldsymbol{\zeta})$[10] and one optimal solution for Problem (23) is $\boldsymbol{\lambda}=\boldsymbol{a}_k$, $\mu_k=1,\mu_l=0,\ \forall l\ne k$.*

The R-ADP algorithm is described in Table 1, whereas the procedure for adding new cuts is described in Table 6 in Electronic Companion EC.1.5. The essential idea is to iterate the following steps: (1) sample a set of states, (2) compute the appropriate supporting hyperplanes at each state, and (3) add the hyperplanes to the convex approximation of $u(\boldsymbol{y},\boldsymbol{\gamma})$. If $u_J(\boldsymbol{y},\boldsymbol{\gamma})\le u(\boldsymbol{y},\boldsymbol{\gamma})$, we have $\bar{u}_J(\boldsymbol{y},\boldsymbol{\gamma})\le u(\boldsymbol{y},\boldsymbol{\gamma})$. Therefore, $g_{s+N_J}(\boldsymbol{y},\boldsymbol{\gamma})$, a tangent hyperplane for $\bar{u}_J(\boldsymbol{y},\boldsymbol{\gamma})$, is a lower bound for $u(\boldsymbol{y},\boldsymbol{\gamma})$, which means that $u_{J+1}(\boldsymbol{y},\boldsymbol{\gamma})$ is also a lower bound for $u(\boldsymbol{y},\boldsymbol{\gamma})$. Through the course of R-ADP, we obtain an improving sequence of lower approximations to the true $u(\boldsymbol{y},\boldsymbol{\gamma})$ function. Hence, if u_0 is a uniform underestimate of u, we know that $u_J(\boldsymbol{y},\boldsymbol{\gamma})$ is a bounded monotone sequence and thus its limit exists.

There are several reasonable strategies for sampling the set $\mathcal{S}_{J+1}$. The easiest way is to set $|\mathcal{S}_J|=1$ (i.e., only add a single cut[11] per iteration) and then sample one state according to some distribution over Δ— this is the approach taken in the numerical experiments of this paper. Our implementation of R-ADP also uses an iteration-dependent state sampling distribution to improve the practical performance (see Electronic Companion EC.1.6); therefore, we introduce the following assumption to support the convergence analysis.

Assumption 3. *On any iteration J, the sampling distribution produces a set $\mathcal{S}_J$ of states from Δ°. The sampled sets $\{\mathcal{S}_J\}_{J=1}^\infty$ satisfy $\sum_{J=1}^\infty\mathbf{P}(\mathcal{S}_J\cap A\ne\emptyset)=\infty$ for any set $A\subseteq\Delta^\circ$ with positive volume.*

This is not a particularly restrictive assumption and should be interpreted simply as requiring an adequate exploration of the state space, a common requirement for ADP and reinforcement learning algorithms (Bertsekas and Tsitsiklis 1996). As an example, for the case of one sample per iteration, one might consider the following sampling strategy, parameterized by a deterministic sequence $\{\epsilon_J\}$: with probability $1-\epsilon_J$, choose the state in any manner and, with probability ϵ_J, select a state uniformly at random over Δ°. In this case, we have that $\mathbf{P}(\mathcal{S}_J\cap A\ne\emptyset)\ge\epsilon_J\cdot\text{volume}(A)$. As long as $\sum_J\epsilon_J=\infty$, Assumption 3 is satisfied.

6.2. Convergence of the R-ADP Algorithm

We are now ready to discuss the convergence of the R-ADP algorithm. For simplicity, we consider the case where $|\mathcal{S}_{J+1}|=1$ for all iterations J. The extension to the batch case, $|\mathcal{S}_{J+1}|>1$, follows the same idea and is merely a matter of more complicated notation (note, however, that we will nevertheless make use of a simple special case of batch algorithm as an analysis tool within the proof).

Theorem 4. *Suppose Assumptions 1–3 hold and that R-ADP samples one state per iteration. Suppose the initial value function approximation u_0 is a lower bound on the optimal value function u and satisfies properties (b) and (c) stated in Theorem 1, namely, that*

- *$|u_0'(\boldsymbol{y},\boldsymbol{\gamma};\pm\boldsymbol{\eta},\mp\boldsymbol{\eta})|\le\sum_{i=1}^n\beta_i\eta_i$ for all $(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta$ and any feasible direction $(\pm\boldsymbol{\eta},\mp\boldsymbol{\eta})$ with $\boldsymbol{\eta}\ge\boldsymbol{0}$; and*
- *$u_0'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z})\le(\rho c_{\max}/2)\sum_{i=1}^n|z_i|$ for all $(\boldsymbol{x},\boldsymbol{\gamma})\in\Delta$ and any feasible direction $(\boldsymbol{0},\boldsymbol{z})$ with $\boldsymbol{e}^T\boldsymbol{z}=0$.*

Then, the sequence $\{u_J\}$ converges uniformly and almost surely to the optimal value function u, that is, it holds that $\|u_J-u\|_\infty\to0$ almost surely.

The proof of Theorem 4 relies on relating each sample path of the algorithm to an auxiliary algorithm

Table 1. R-ADP Algorithm

```
Parameters: $l, c_{ij}, \rho > 0$
Data: $\boldsymbol{d}_1,\boldsymbol{d}_2,\dots,\boldsymbol{d}_M,\boldsymbol{P}_1,\boldsymbol{P}_2,\dots,\boldsymbol{P}_M$
Input: Initial approximation $u_0(\boldsymbol{y},\boldsymbol{\gamma})=\max_{k=1,\dots,N_0}g_k(\boldsymbol{y},\boldsymbol{\gamma})$
for $J=0,1,2,\dots$
  Sample a finite set of states $\mathcal{S}_{J+1}$ from $\Delta$ according to some distribution
  for $s=1,2,\dots,|\mathcal{S}_{J+1}|$
      Let $(\tilde{\boldsymbol{y}}_s,\tilde{\boldsymbol{\gamma}}_s)$ be the $s$th sampled state in $\mathcal{S}_{J+1}$
      Run Subroutine in Table EC.2 in Electronic Companion EC.1.5
         with input $(\tilde{\boldsymbol{y}}_s,\tilde{\boldsymbol{\gamma}}_s)$ and let the result be $g_{s+N_J}(\boldsymbol{y},\boldsymbol{\gamma})$
  end for
  Set $u_{J+1}(\boldsymbol{y},\boldsymbol{\gamma})=\max_{k=1,\dots,N_{J+1}}g_k(\boldsymbol{y},\boldsymbol{\gamma})$, where $N_{J+1}=N_J+|\mathcal{S}_{J+1}|$
end for
```

where the cuts are added in "batches" rather than one by one. We show that, after accounting for the different timescales, the value function approximations generated by R-ADP are close to the approximations generated by the auxiliary algorithm. By noticing that the auxiliary algorithm is an approximate value iteration algorithm whose per-iteration error can be bounded in $\|\cdot\|_\infty$ because of Lemma EC.7 in the electronic companion, we quantify its error against exact value iteration, which in turn allows us to quantify the error between R-ADP and exact value iteration. We make use of ϵ-covers of the state space (for arbitrarily small ϵ) along with Assumption 3 to argue that this error converges to zero. Note that one can satisfy the conditions for u_0 by taking it to be a constant function that is a lower bound of u; for example, $u_0(\cdot) = 0$ is a suitable choice. In Electronic Companion EC.1.6, we discuss two practical aspects associated with implementing R-ADP: (1) checking for and removing redundant cuts and (2) specifying an effective state-sampling distribution.

6.3. Benchmarking R-ADP on Random Problem Instances

We first present some benchmarking results of running R-ADP on a set of randomly generated problems ranging from $n = 2$ to $n = 10$ locations, the largest of which corresponds to a dynamic program with a 19-dimensional continuous state space. We set the discount factor as $\rho = 0.95$, the repositioning costs to be $c_{\min} = c_{\max} = 1$, and the lost sales cost as $\beta_i = 2$. We consider normalized total inventory of $n = 1$; and for each problem instance, we take $M = 50$ demand and return probability samples as follows. With each location i, we associate a truncated normal demand distribution (so that it is nonnegative) with mean ν_i and standard deviation σ_i. The ν_i are drawn from a uniform distribution and then normalized so that $\sum_i \nu_i = 0.3$. We then set $\sigma_i = \nu_i$ so that locations with higher mean demand are also more volatile. Next, we follow Assumption 1 and sample one outcome of a matrix $(\tilde{q}_{ij})$ such that each row is chosen uniformly from a standard simplex. Each of the M samples of the return probability matrix consists of $(\tilde{q}_{ij})$ multiplied by a random scaling factor drawn from Uniform$(0.7, 0.9)$. Hence, we have $p_{\min} = 0.7$. We compare the performance of the R-ADP policy to the performance of several baselines approaches:

• *Myopic policy*: The myopic policy (Myo.) minimizes the single-period lost sales and repositioning costs, that is, the policy associated with $v(\cdot) = 0$.

• *Rolling-horizon deterministic lookahead policy (Mean)*: This policy considers a k-period rolling horizon lookahead, obtained by solving the large-scale LP described in Corollary 1 taking $(\boldsymbol{d}_t, \boldsymbol{P}_t)$ to be their means. We use the abbreviation "k-RH-M" to refer to this policy.

• *No-repositioning policy*: The no-repositioning policy (No-R) does not reposition any inventory.

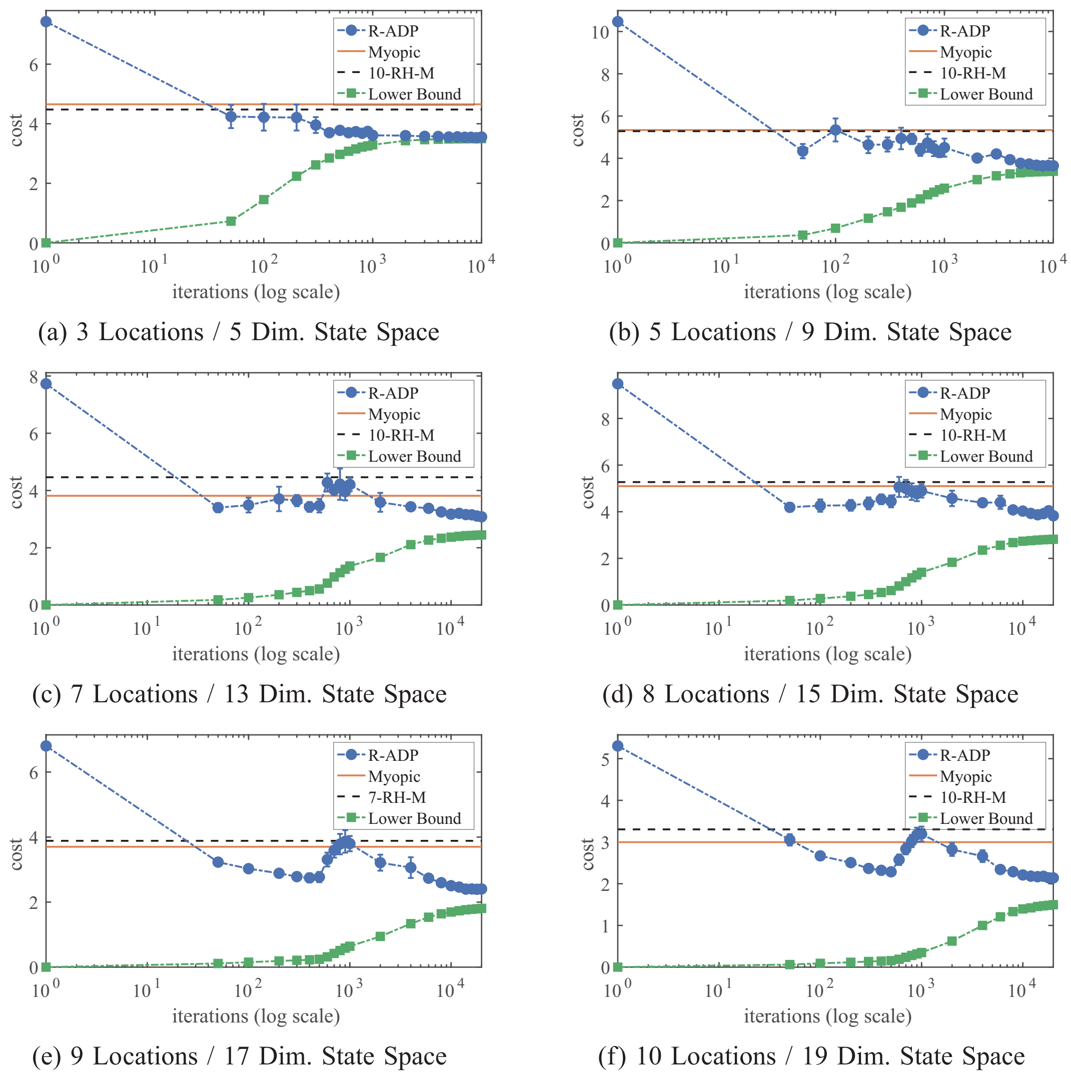
We use a maximum of 1,000 cuts for all problem instances and we run the R-ADP algorithm for 10,000 iterations for $n \le 6$ and for 20,000 iterations for $n = 7, 8, 9, 10$. We initially sample 80% of states randomly[12] and 20% of states from the replay buffer of the myopic policy. As the algorithm progresses, we transition toward a distribution of 20% randomly, 0% from the myopic replay buffer, and 80% from the current ADP replay buffer. Note that Assumption 3 is satisfied for this sampling scheme. Redundancy checks are performed every 250 iterations. The performance of the ADP algorithm is evaluated using Monte-Carlo simulation over 500 sample paths (across 20 initial states, randomly sampled subject to zero outstanding rentals) at various times during the training process. Because the ADP algorithm itself is random, we repeat the training process 10 times for each problem instance in order to obtain confidence intervals (which are shown in Figure 2).

The results[13] are summarized in Table 2. The first column "n" shows the number of locations (note that $2n - 1$ is the dimension of the state space). The second column 'Sec./iter.' shows the CPU time for training the R-ADP policy on a 4 GHz Intel Core i7 processor using four cores, which includes the time needed to remove cuts and generate the replay buffer. Figure 3 shows the amount of computational savings when using the no-repositioning region structure (Lemma 1 and Proposition 2), with 95% confidence intervals. We roughly attain 5%–8% CPU savings by making use of the policy structure derived. The remaining columns give a percentage optimality relative to the ADP lower bound for each of the policies, computed as the percentage of the lower bound achieved when the baseline no-repositioning policy is set as "0% optimal." (Note that this is a *lower bound* on the percentage optimality relative to the optimal policy.) This is done via

$$\%\,\text{optimality to lower bound} = \frac{\text{cost of No-R policy} - \text{cost of R-ADP policy}}{\text{cost of No-R policy} - \text{highest lower bound}}. \quad (25)$$

In terms of wall clock time, we observe that our ADP algorithm produces near-optimal results for $n \le 6$ within an hour (for $n = 6$, we are using $0.29 \cdot 10000$ seconds or 48 minutes). For the larger problems of $n \ge 7$, when provided a limited amount of computation — around three hours for 20,000 iterations —the estimated optimality gap is slightly larger, between 12% and 17%. Note that these are *offline* computation times; the learned policy is implemented by solving a linear program (in roughly the same amount of time as a single iteration of R-ADP, typically less than one second).

Figure 2. (Color online) Performance of R-ADP on Randomly Generated Problems



Notes. (a) Three locations/5 dim. state space; (b) 5 locations/9 dim. state space; (c) 7 locations/13 dim. state space; (d) 8 locations/15 dim. state space; (e) 9 locations/17 dim. state space; (f) 10 locations/19 dim. state space. dim, dimensional.

Figure 2 shows the performance of the R-ADP policy as the algorithm progresses, along with 95% confidence intervals and lower bounds. We also show the best performing variant of the rolling-horizon deterministic lookahead approach and the myopic policy. In all cases, the cost of the R-ADP policy eventually becomes lower than that of the baselines and approaches the accompanying lower bound. Note that the minor upticks in cost around iteration 1,000 appear to be due to the value function approximation hitting the limit of 1,000 cuts for the first time.

Based on the average performance across the different n (last row of Table 1), the myopic policy comes closest to matching the performance of R-ADP. We note that the total amount of inventory repositioned by the R-ADP policy is considerably higher than that of the myopic policy, between 37% higher (for $n = 5$) and 79% (for $n = 9$) higher. This suggests that the improvement upon the myopic policy can be attributed to a more aggressive repositioning strategy. Because the myopic policy does not take into account customers' return behaviors, the additional repositioning activity observed in the ADP policy can be explained by its attempt to plan for the future by counteracting the effects of $\boldsymbol{P}$. In Electronic Companion EC.1.7, we vary parameter settings and provide comparative statics regarding the impact of (1) total demand, (2) demand volatility, (3) return fraction (i.e., fraction of vehicles returned per period), and (4) uniformity of return locations.

7. Scaling to Large Systems via Clustering

In Section 6.3, we provided computational results for R-ADP on 19-dimensional MDPs ($n = 10$ locations).

Table 2. Performance Comparison of Repositioning Policies

n	Sec./iter.	R-ADP	Myo.	3-RH-M	5-RH-M	7-RH-M	10-RH-M
2	0.06	99.2%	60.1%	64.9%	65.8%	66.0%	65.7%
3	0.21	98.7%	70.7%	71.8%	73.9%	73.5%	75.2%
4	0.27	95.9%	78.1%	69.2%	69.1%	70.1%	69.1%
5	0.22	96.4%	72.5%	70.7%	71.6%	72.9%	73.3%
6	0.29	94.1%	75.3%	74.4%	75.5%	76.1%	76.7%
7	0.42	88.1%	74.2%	59.3%	60.3%	61.3%	61.9%
8	0.44	85.0%	66.0%	61.6%	62.4%	62.1%	63.3%
9	0.48	88.2%	62.2%	57.1%	57.5%	58.5%	58.0%
10	0.54	83.4%	60.8%	49.7%	51.0%	51.6%	52.7%
avg.	—	92.1%	68.9%	64.3%	65.2%	65.8%	66.2%

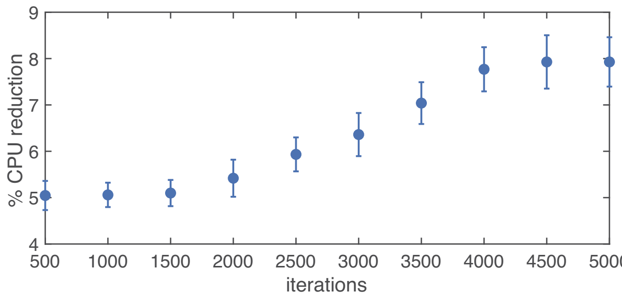
Note. avg., average; sec., seconds; iter., iterations.

Approximating MDPs of larger dimensions is well known to be challenging due to the curse of dimensionality. For example, Lu et al. (2017) approximate a nine-location problem using a two-stage stochastic integer program and He et al. (2020) approximate a five-location problem using a robust approach within an MDP model without a convergence guarantee. However, practical instances of the repositioning problem may involve much larger values of n. In this section, we show that a surprisingly simple extension of R-ADP via a clustering approach allows it to scale to very large systems (for example, systems with $n = 100$ locations). The approach outperforms the rolling-horizon deterministic lookahead baseline, a commonly used "scalable" strategy for large-scale MDPs (Powell 2007). Additionally, we show that this method also produces good results when the effective horizon[14] of the MDP is long (i.e., when $\rho = 0.99$).

7.1. Clustered Repositioning-ADP

We propose the following simple framework for applying R-ADP to problems with a large number of locations n. We execute the R-ADP algorithm on an auxiliary MDP obtained by clustering locations together so that there is a manageable number of them and then heuristically deconstruct the "clustered policy" into a policy for the original MDP. This policy is referred to as Clustered Repositioning-ADP

Figure 3. (Color online) Computational Savings Using Policy Structure ($n = 5$)



(CR-ADP). Specifically, the algorithm consists of the following steps:

1. First, we partition the n locations into $\tilde{n}$ clusters. Let $\mathcal{C}_k \subseteq \{1, 2, \ldots, n\}$ be the k-th cluster for $k = 1, 2, \ldots, \tilde{n}$.

2. Using these clusters, we define a transformation of the problem parameters from the n-location instance to the $\tilde{n}$-location instance. Denote the transformed demand, return fraction, repositioning costs, and lost sales cost by $\tilde{\boldsymbol{d}}_t$, $\tilde{\boldsymbol{P}}_t$, $\tilde{\boldsymbol{c}}$, and $\tilde{\beta}_i$, respectively. We then solve the more tractable $\tilde{n}$-location problem using R-ADP and obtain a policy $\tilde{\pi}$.

3. To implement the policy: given an n-location state, we obtain an $\tilde{n}$-location state by summing inventory in each cluster $\mathcal{C}_k$ and produce an $\tilde{n}$-location repositioning-decision $\tilde{\boldsymbol{y}}$ using $\tilde{\pi}$. We then use a "splitting" heuristic to construct a n-location repositioning-decision $\boldsymbol{y}$ from $\tilde{\boldsymbol{y}}$.

There are many reasonable ways to design the clusters, transform the problem parameters, and to split inventory. We show results for a straightforward instantiation of this framework, where adjacent locations are clustered, demand is summed up within clusters, repositioning costs and return fractions are appropriately averaged, and inventory is split according to the demand proportion of each location relative to the cluster's total demand. Details are given in Electronic Companion EC.1.8.

Tables 3 and 4 show the results of the clustered approach CR-ADP when compared with the same baseline policies used in Section 6.3, for $\rho = 0.95$ and $\rho = 0.99$. Note that these results are given in terms of expected cost, rather than an optimality percentage, because we do not obtain lower bounds of the original MDP when using CR-ADP. We consider problems with locations ranging from $n = 20$ to $n = 100$, with $\tilde{n} = 10$ clusters in each case. Note that the baselines do not use clustering; in fact, for large values of n, the LP used in the rolling-horizon deterministic lookahead approach becomes computationally intractable for larger values of the lookahead horizon. We see that despite solving a clustered, approximate problem, the ADP approach outperforms all of the baseline policies.

These results point to the benefit of jointly considering downstream values and the stochasticity of

Table 3. Summary of Results on Large-Scale Instances ($\tilde{n} = 10$, $\rho = 0.95$)

n	CR-ADP	Myo.	No-R	3-RH-M	5-RH-M	7-RH-M	10-RH-M
20	4.04	4.73	9.40	4.51	4.43	4.38	4.31
30	2.91	3.17	6.10	3.54	3.41	3.30	3.24
40	3.35	3.96	8.15	4.22	4.14	4.04	4.05
50	3.77	4.16	8.48	4.33	4.19	4.14	—
60	3.80	4.20	9.01	4.36	4.24	—	—
70	3.77	4.13	8.84	4.31	4.18	—	—
80	3.34	3.83	7.93	3.96	3.87	—	—
90	3.83	4.16	8.72	4.33	—	—	—
100	3.41	3.90	7.69	3.89	—	—	—

Note. No-R, No-Repositioning.

Table 4. Summary of Results on Large-Scale Instances ($\tilde{n} = 10$, $\rho = 0.99$)

n	CR-ADP	Myo.	No-R	3-RH-M	5-RH-M	7-RH-M	10-RH-M
20	21.50	26.09	61.98	24.83	24.01	23.65	23.50
30	14.55	17.04	51.10	20.12	19.36	19.21	18.83
40	16.18	20.63	52.46	22.42	21.97	21.59	21.53
50	18.88	21.81	59.24	23.22	22.54	22.16	—
60	19.97	22.17	60.70	23.37	23.17	—	—
70	18.56	21.81	59.90	22.89	22.63	—	—
80	15.67	20.24	55.75	21.23	21.17	—	—
90	19.00	21.85	59.86	23.16	—	—	—
100	16.65	20.48	53.06	20.76	—	—	—

Note. No-R, No-Repositioning.

demand/return fractions, as neither of these features alone is able to produce high-performing policies (as evidenced by the myopic and deterministic lookahead approaches). Lastly, we remark that many aspects of our clustering heuristic design could be further refined and the most appropriate strategy might be highly problem dependent. Our goal here is to show that even a naive approach to clustering can bring value, rather than to perform a systematic study of clustering heuristics, which we leave to future work. In Electronic Companion EC.1.9, we illustrate an application of the CR-ADP approach to a real-world system with 200 locations.

8. Conclusion

In this paper, we consider the problem of optimal repositioning of inventory in a product rental network with multiple locations and where demand, rental periods, and return locations are stochastic. We show that the optimal policy is specified in terms of a region in the state space, inside of which it is optimal not to carry out any repositioning and outside of which it is optimal to reposition inventory. We also prove that when repositioning, it is always optimal to do so such that the system moves to a new state that is on the boundary of the no-repositioning region and provide a simple check for when a state is in the no-repositioning region. We then propose a provably convergent approximate dynamic programming algorithm, R-ADP, that builds a lower approximation of the convex value function by iteratively adding hyperplanes. We also propose a clustering-based approach that allows our ADP algorithm to solve large-scale problems. Numerical experiments on problems with up to 100 locations support the effectiveness of the algorithmic approach.

Acknowledgments

The authors thank Editor-in-Chief David Simchi-Levi, the anonymous associate editor, and the three referees for their constructive comments and suggestions, which have significantly improved the paper.

Endnotes

[1] Renting may become even more prevalent as the economy shifts away from a model built on the exclusive ownership of resources to one based on on-demand access and resource sharing (Sundararajan 2016, Benjaafar and Hu 2020).

[2] Other applications where the periodic repositioning of inventory is important include bike share systems where customers can pick up a bike from one location and return it to any other location within the service region; shipping container rentals in the freight industry where containers can be rented in one location and returned to a different location, with locations corresponding in some cases to ports in different countries; and the use of certain medical equipment, such as IV pumps and wheelchairs, in large hospitals by different departments located in various areas of the hospital.

[3] The first version of our paper appeared online ahead of the first version of He et al. (2020). He et al. (2020) refer to that version of our paper.

[4] Zhao et al. (2020) cite the working version of this paper.

[5] Similar assumptions on relocation/travel times have been made in much of the existing literature on this topic. He et al. (2020) assume that both customer trips and repositioning trips can be completed within a period. Balseiro et al. (2021) and Waserhole and Jost (2016) assume that travel times are instantaneous. Bimpikis et al. (2019) assumes that going from one location to another takes one period.

[6] This is reasonable when the number of rental units N is large and is consistent with treatment elsewhere in the literature (see, for example, Li and Tao 2010, He et al. 2020, and Zhao et al. 2020) and in much of the literature on stochastic inventory control (see, for example, Zipkin 2000).

[7] We make this assumption for tractability of the theoretical analysis in Section 4 but note that it is plausible because whether to return and where to return are usually two separate decisions for customers. Furthermore, in the case of rental networks located in dense urban regions, we expect many rental locations to have similar properties in terms of customers' rental/return behaviors. In Electronic Companion EC.1.1, we provide empirical support for this assumption based on real data obtained from the one-way car sharing service Car2Go.

[8] A similar but slightly weaker condition is assumed in He et al. (2020). In this sense, the convexity in our paper can include their results as a special case except that in their model, lost sales costs depend on both origin and destination and the return destinations are known by the platform at the time of rental. In our case, we assume, consistent with the reality of many one-way vehicle sharing systems, that the destination of a rental is not revealed until a realized trip is completed. In settings where the lost sales cost depends on both the origin and destination of a trip, β_i has the interpretation of the expected lost sales cost over all destinations.

[9] For example, we do not make use of a property that there are only a finite number of distinct cuts; see lemma 1 of Philpott and Guan (2008). We remark, however, that our algorithm has a natural adaptation for finite-horizon problems.

[10] Note that, in general, $\Omega_u(\boldsymbol{\gamma}) \neq \bigcup_{k \in \mathcal{K}} D^k$. The reason is that even if two cuts are both not in $\mathcal{K}$, the intersection of these two cuts could still include the subgradient that satisfies the no-reposition condition.

[11] If parallel computing is available, one might consider the "batch" version of the algorithm (i.e., $|\mathcal{S}_{J+1}| > 1$) by performing the inner for-loop of the R-ADP algorithm in Table 1 on multiple processors (or workers). In this case, each worker receives u_J, samples a state, and computes the appropriate supporting hyperplane. The main

processor would then aggregate the results into u_{J+1} and start the next iteration by broadcasting u_{J+1} to each worker.

[12] Each sampled state is given by $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{\gamma}}) = (\xi\,\boldsymbol{y}', (1-\xi)\,\boldsymbol{\gamma}') \in \Delta$ where $\boldsymbol{y}'$ and $\boldsymbol{\gamma}'$ are independent uniform samples from $\Delta_{n-1}(N)$ and $\xi \sim \text{Uniform}(p_{\min}(M), p_{\max}(M))$ where $p_{\min}(M)$ and $p_{\max}(M)$ are the minimum and maximum row sums of the return fraction matrix over the M samples. This sampling scheme can be considered a nearly uniform sample over the state space, except with the two parts of the state rescaled by relevant problem parameters so that they are more likely to fall in important regions.

[13] The same random seed is used in all instances (i.e., all n) to generate the problem parameters.

[14] One way to determine an appropriate discount factor is to connect it with the *effective planning horizon* of the MDP, commonly taken to be $\mathcal{O}(1/(1-\rho))$; see, for example, Jiang et al. (2015).

References

Balseiro SR, Brown DB, Chen C (2021) Dynamic pricing of relocating resources in large networks. *Management Sci.* 67(7):4075–4094.

Banerjee S, Freund D, Lykouris T (2021) Pricing and optimization in shared vehicle systems: An approximation framework. *Oper. Res.*, ePub ahead of November 18, https://doi.org/10.1287/opre.2021.2165.

Banerjee S, Kanoria Y, Qian P (2018) Dynamic assignment control of a closed queueing network under complete resource pooling. Preprint, submitted March 13, https://arxiv.org/abs/1803.04959.

Bélanger V, Ruiz A, Soriano P (2019) Recent optimization models and trends in location, relocation, and dispatching of emergency medical vehicles. *Eur. J. Oper. Res.* 272(1):1–23.

Benjaafar S, Hu M (2020) Operations management in the age of the sharing economy: What is old and what is new? *Manufacturing Service Oper. Management*, 22(1):93–101.

Benjaafar S, Wu S, Liu H, Gunnarsson EB (2021) Dimensioning on-demand vehicle sharing systems. *Management Sci.* ePub ahead of print March 15, https://doi.org/10.1287/mnsc.2021.3957.

Berman O (1981) Dynamic repositioning of indistinguishable service units on transportation networks. *Transportation Sci.* 15(2):115–136.

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA).

Bimpikis K, Candogan O, Saban D (2019) Spatial pricing in ride-sharing networks. *Oper. Res.* 67(3):744–769.

Birge JR, Zhao G (2007) Successive linear approximation solution of infinite-horizon dynamic stochastic programs. *SIAM J. Optim.* 18(4):1165–1186.

Braverman A, Dai JG, Liu X, Ying L (2019) Empty-car routing in ridesharing systems. *Oper. Res.* 67(5):1437–1452.

Brown DB, Smith JE (2011) Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Sci.* 57(10):1752–1770.

Bruglieri M, Colorni A, Luè A (2014) The vehicle relocation problem for the one-way electric vehicle sharing: An application to the Milan case. *Procedia Soc. Behav. Sci.* 111:18–27.

Chen ZL, Powell WB (1999) Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *J. Optim. Theory Appl.* 102(3):497–524.

Chung H, Freund D, Shmoys DB (2018) Bike angels: An analysis of Citi Bike's incentive program. *Proc. 1st ACM SIGCAS Conf. Comput. Sustainable Soc.* (ACM, New York).

Constantinides GM (1979) Multiperiod consumption and investment behavior with convex transactions costs. *Management Sci.* 25(11):1127–1137.

De Farias DP, Van Roy B (2000) On the existence of fixed points for approximate value iteration and temporal-difference learning. *J. Optim. Theory Appl.* 105(3):589–608.

Eberly JC, Van Mieghem JA (1997) Multi-factor dynamic investment under uncertainty. *J. Econom. Theory* 75(2):345–387.

Freund D, Henderson SG, Shmoys DB (2017) Minimizing multimodular functions and allocating capacity in bike-sharing systems. *Internat. Conf. Integer Programming Combinatorial Optim.* (Springer, Berlin), 186–198.

Freund D, Henderson SG, Shmoys DB (2019) Bike sharing. Hu M, ed. *Sharing Economy: Making Supply Meet Demand* (Springer, Berlin), 435–459.

Freund D, Norouzi-Fard A, Paul A, Wang C, Henderson SG, Shmoys DB (2020) Data-driven rebalancing methods for bike-share systems. Crisostomi E, et al., eds. *Analytics for the Sharing Economy: Mathematics, Engineering and Business Perspectives* (Springer Nature, Switzerland), 255–278.

George DK, Xia CH (2011) Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *Eur. J. Oper. Res.* 211:198–207.

Ghosh S, Varakantham P, Adulyasak Y, Jaillet P (2017) Dynamic repositioning to reduce lost demand in bike sharing systems. *J. Artificial Intelligence Res.* 58:387–430.

Girardeau P, Leclere V, Philpott AB (2014) On the convergence of decomposition methods for multistage stochastic convex programs. *Math. Oper. Res.* 40(1):130–145.

Godfrey GA, Powell WB (2001) An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Sci.* 47(8):1101–1112.

He L, Hu Z, Zhang M (2020) Robust repositioning for vehicle sharing. *Manufacturing Service Oper. Management* 22(2):241–256.

He L, Mak HY, Rong Y (2019) Operations management of vehicle sharing systems. Hu M, ed. *Sharing Economy: Making Supply Meet Demand* (Springer, Berlin).

He L, Mak HY, Rong Y, Shen ZJM (2017) Service region design for urban electric vehicle sharing systems. *Manufacturing Service Oper. Management* 19(2):309–327.

Higle JL, Sen S (1991) Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Math. Oper. Res.* 16(3):650–669.

Jian N, Freund D, Wiberg HM, Henderson SG (2016) Simulation optimization for a large-scale bike-sharing system. *Proc. 2016 Winter Simulation Conf.* (IEEE Press, Piscataway, NJ), 602–613.

Jiang N, Kulesza A, Singh S, Lewis R (2015) The dependence of effective planning horizon on model accuracy. *Proc. 2015 Internat. Conf. Autonomous Agents Multiagent Systems* (International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC), 1181–1189.

Kabra A, Belavina E, Girotra K (2020) Bike-share systems: Accessibility and availability. *Management Sci.* 66(9):3803–3824.

Kaspi M, Raviv T, Tzur M (2017) Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. *IISE Trans.* 49(2):144–158.

Kushner HJ, Yin GG (2003) *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35 (Springer, Berlin).

Lee CY, Meng Q (2015) *Handbook of Ocean Container Transport Logistics* (Springer, Berlin).

Leland HE (1999) Optimal portfolio management with transactions costs and capital gains taxes. Working Paper RPF-290, IBER, University of California Berkeley, Berkeley.

Li Z, Tao F (2010) On determining optimal fleet size and vehicle transfer policy for a car rental company. *Comput. Oper. Res.* 37(2):341–350.

Li Y, Zheng Y, Yang Q (2018) Dynamic bike reposition: A spatio-temporal reinforcement learning approach. *Proc. 24th ACM SIGKDD Conf. Knowledge Discovery Data Mining* (ACM, New York).

Li J, Leung CS, Wu Y, Liu K (2007) Allocation of empty containers between multi-ports. *Eur. J. Oper. Res.* 182(1):400–412.

Linowsky K, Philpott AB (2005) On the convergence of sampling-based decomposition algorithms for multistage stochastic programs. *J. Optim. Theory Appl.* 125(2):349–366.

Liu J, Sun L, Chen W, Xiong H (2016) Rebalancing bike sharing systems: A multi-source data smart optimization. *Proc. 22nd ACM SIGKDD Conf. Knowledge Discovery Data Mining* (ACM, New York), 1005–1014.

Lu M, Chen Z, Shen S (2017) Optimizing the profitability and quality of service in carshare systems under demand uncertainty. *Manufacturing Service Oper. Management* 20(2):162–180.

Ma H, Fang F, Parkes DC (2020) Spatio-temporal pricing for ride-sharing platforms. *ACM SIGecom Exchanges* 18(2):53–57.

Maxwell MS, Restrepo M, Henderson SG, Topaloglu H (2010) Approximate dynamic programming for ambulance redeployment. *INFORMS J. Comput.* 22(2):266–281.

Maxwell MS, Ni EC, Tong C, Henderson SG, Topaloglu H, Hunter SR (2014) A bound on the performance of an optimal ambulance redeployment policy. *Oper. Res.* 62(5):1014–1027.

Munos R, Szepesvári C (2008) Finite-time bounds for fitted value iteration. *J. Machine Learn. Res.* 9(May):815–857.

Muthuraman K, Kumar S (2006) Multidimensional portfolio optimization with proportional transaction costs. *Math. Finance* 16(2):301–335.

Nair R, Miller-Hooks E (2011) Fleet management for vehicle sharing operations. *Transportation Sci.* 45(4):524–540.

Nascimento JM, Powell WB (2009) An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Math. Oper. Res.* 34(1):210–237.

Nascimento JM, Powell WB (2010) Dynamic programming models and algorithms for the mutual fund cash balance problem. *Management Sci.* 56(5):801–815.

Nyotta B, Bravo F, Feldman J (2019) Free rides in dockless, electric vehicle sharing systems. Preprint, submitted June 7, https://dx.doi.org/10.2139/ssrn.3391937.

O'Mahony E, Shmoys DB (2015) Data analysis and optimization for (citi) bike sharing. *Proc. AAAI Conf. Artificial Intelligence* 29(1): 687–694.

Pereira MVF, Pinto LMVG (1991) Multi-stage stochastic optimization applied to energy planning. *Math. Programming* 52:359–375.

Philpott AB, Guan Z (2008) On the convergence of stochastic dual dynamic programming and related methods. *Oper. Res. Lett.* 36(4):450–455.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. (John Wiley & Sons, New York).

Powell WB, Ruszczyński A, Topaloglu H (2004) Learning algorithms for separable approximations of discrete stochastic optimization problems. *Math. Oper. Res.* 29(4):814–836.

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. (John Wiley & Sons, Inc., New York)

Raviv T, Kolka O (2013) Optimal inventory management of a bike-sharing station. *IIE Trans.* 45(10):1077–1093.

Schuijbroek J, Hampshire RC, Van Hoeve WJ (2017) Inventory rebalancing and vehicle routing in bike sharing systems. *Eur. J. Oper. Res.* 257(3):992–1004.

Shapiro A (2011) Analysis of stochastic dual dynamic programming method. *Eur. J. Oper. Res.* 209(1):63–72.

Shu J, Chou MC, Liu Q, Teo CP, Wang IL (2013) Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Oper. Res.* 61(6):1346–1359.

Shui C, Szeto W (2018) Dynamic green bike repositioning problem–A hybrid rolling horizon artificial bee colony algorithm approach. *Transportation Res. Part D Transportation Environ.* 60: 119–136.

Song DP (2005) Optimal threshold control of empty vehicle redistribution in two depot service systems. *IEEE Trans. Automatic Control* 50(1):87–90.

Sundararajan A (2016) *The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism* (MIT Press, Cambridge, MA).

Van Mieghem JA (2003) Commissioned paper: Capacity management, investment, and hedging: Review and recent developments. *Manufacturing Service Oper. Management* 5(4):269–302.

Warrington J, Beuchat PN, Lygeros J (2019) Generalized dual dynamic programming for infinite horizon problems in continuous state and action spaces. *IEEE Trans. Automat. Control* 64(12):5012–5023.

Waserhole A, Jost V (2016) Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO J. Transporation Logist.* 5(3):293–320.

Zhao L, Liu Z, Hu P (2020) Dynamic repositioning for vehicle sharing with setup costs. *Oper. Res. Lett.* 48(6):792–797.

Zipkin PH (2000) *Foundations of Inventory Management* (McGraw-Hill, New York).

Electronic Companion to “Dynamic Inventory Repositioning in On-DemandRental Networks”

EC.1. Appendix

EC.1.1. Empirical Justification of Assumption 1 and Estimation of $\boldsymbol{P}_t$

In this section, we describe our analysis of a publicly available Car2Go dataset[1], which contains GPS locations of Car2Go rental vehicles, at roughly 15-minute intervals, between June 2012 and December 2013 in Portland, Oregon. The goal is to understand whether Assumption 1 is realistic in practice. We use the following procedure to obtain estimates of the return fraction p_t specified in Assumption 1, calculated using data from the entire year of 2013.

1. We assume hourly rental periods (i.e., repositioning occurs every hour) and that the city is divided into n regions, $1, 2, \ldots, n$.
2. For each vehicle, we extract all of its trips throughout the year under the assumption that trips end when the vehicle remains at the same GPS coordinates between successive measurements (15 minutes). This process yielded a total of 367,736 trips across 311 vehicles. Each trip is associated with a start time, return time, rental origin, and return location.
3. For each hour t, we directly estimate the return fraction for each region i by computing the set of rentals that are ongoing at hour t originating from region i (therefore, this includes any rental from i initiated at or before t and returned strictly after t). The return fraction from region i is then the fraction of these trips that are returned at hour $t+1$ (to any return location).

Table EC.1 shows the results of this estimation procedure for $n = 9$. These $n = 9$ regions are constructed using a 3x3 grid, defined by latitude regions $(45.35°, 45.50°]$, $(45.50°, 45.55°]$, and $(45.55°, 45.82°]$, and the longitude regions $(-122.86°, -122.65°]$, $(-122.65°, -122.60°]$, and $(-122.60°, -121.65°]$.

Assumption 1 states that the return fraction p_t does not depend on the rental origin, but we compute these quantities as if the dependency on rental origin is allowed. However, it can be seen that the variation of the return fraction across regions is minor (i.e., the return fractions in each row are similar), thereby providing empirical evidence that Assumption 1 is not unreasonable.

We also note that the return fraction matrix $\boldsymbol{P}_t$ can be estimated using the same procedure that is outlined above, assuming that GPS data is available. To obtain an estimate of $p_{t,ij}$, we would simply alter Step 3 above to calculate the fraction of ongoing trips from region i that are returned specifically to region j at hour $t+1$.

[1] `https://aaronparecki.com/car2go/`

Time Period | Region	1	2	3	4	5	6	7	8	9
8am-9am	0.770	0.765	0.776	0.787	0.780	0.773	0.798	0.794	0.777
9am-10am	0.743	0.724	0.791	0.740	0.739	0.721	0.731	0.742	0.728
10am-11am	0.657	0.674	0.626	0.677	0.671	0.652	0.705	0.703	0.665
11am-12pm	0.644	0.681	0.642	0.671	0.681	0.660	0.668	0.697	0.671
12pm-1pm	0.693	0.659	0.664	0.661	0.684	0.676	0.655	0.705	0.655
1pm-2pm	0.642	0.664	0.714	0.642	0.682	0.636	0.650	0.696	0.686
2pm-3pm	0.697	0.650	0.611	0.655	0.688	0.668	0.684	0.693	0.630
3pm-4pm	0.656	0.647	0.651	0.680	0.671	0.644	0.660	0.668	0.650
4pm-5pm	0.667	0.654	0.616	0.664	0.676	0.651	0.633	0.632	0.630
5pm-6pm	0.664	0.696	0.644	0.700	0.727	0.690	0.687	0.706	0.698

Table EC.1 Estimates of Return Fraction p_t (see Assumption 1) from 2013 Car2Go GPS Data

EC.1.2. Properties of the Repositioning Cost

In this subsection, We describe some properties of the repositioning cost $C(\cdot)$ defined in (1).

LEMMA EC.1. *$C(\cdot)$ satisfies the following properties:*

1. *(Positive Homogeneity): $C(t\boldsymbol{z}) = tC(\boldsymbol{z})$ for all $t \geq 0$.*
2. *(Convexity): $C(\lambda \boldsymbol{z}_1 + (1-\lambda)\boldsymbol{z}_2) \leq \lambda C(\boldsymbol{z}_1) + (1-\lambda)C(\boldsymbol{z}_2)$ for all $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{H}$ and $\lambda \in [0,1]$.*
3. *(Sub-Additivity): $C(\boldsymbol{z}_1 + \boldsymbol{z}_2) \leq C(\boldsymbol{z}_1) + C(\boldsymbol{z}_2)$ for all $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{H}$.*
4. *(Continuity): $C(\boldsymbol{z})$ is continuous in $\boldsymbol{z} \in \mathcal{H}$.*

Proof. It is clear that the linear program (1) is bounded feasible. Therefore, an optimal solution to (1) exists and the strong duality holds. The dual linear program can be written as

$$C(\boldsymbol{z}) = \begin{array}{rl} \max & \boldsymbol{\lambda}^T \boldsymbol{z} \\ \text{subject to} & \lambda_j - \lambda_i \leq c_{ij} \quad \forall \quad i,j. \end{array} \tag{EC.1}$$

From (EC.1), we have

$$C(t\boldsymbol{z}) = \max\left\{t\boldsymbol{\lambda}^T\boldsymbol{z} : \lambda_j - \lambda_i \leq c_{ij}, \forall\ i,j\right\} = t\max\left\{\boldsymbol{\lambda}^T\boldsymbol{z} : \lambda_j - \lambda_i \leq c_{ij}, \forall\ i,j\right\} = tC(\boldsymbol{z}).$$

Therefore, $C(\cdot)$ is positively homogeneous. As the pointwise supremum of a collection of convex and lower semicontinuous functions ($\boldsymbol{\lambda}^T\boldsymbol{z}$ for each $\boldsymbol{\lambda}$), $C(\cdot)$ is also convex and lower semicontinuous. It is well known that a convex function on a locally simplicial convex set is upper semicontinuous (Rockafellar (1970) Theorem 10.2). Therefore, as $\mathcal{H}$ is a polyhedron, $C(\cdot)$ must be continuous. From the positive homogeneity and the convexity, we have

$$C(\boldsymbol{z}_1 + \boldsymbol{z}_2) = 2C\left(\frac{1}{2}\boldsymbol{z}_1 + \frac{1}{2}\boldsymbol{z}_2\right) \leq 2\left(\frac{1}{2}C(\boldsymbol{z}_1) + \frac{1}{2}C(\boldsymbol{z}_2)\right) = C(\boldsymbol{z}_1) + C(\boldsymbol{z}_2).$$

Therefore, $C(\cdot)$ is sub-additive.

Moreover, due to the triangle inequality, it is not optimal to simultaneously move inventory into and out of the same location. This property can be stated as follows.

LEMMA EC.2. *There exists an optimal solution $\boldsymbol{w}$ to (1) such that*

$$\sum_{i=1}^{n} w_{ij} = z_j^+ \quad \textit{and} \quad \sum_{k=1}^{n} w_{jk} = z_j^- \quad \textit{for all } j = 1, \dots, n.$$

Proof. It is easy to see that an equivalent condition is $w_{i,j}w_{j,k} = 0$ for all i, j, k. To show this is true, suppose $\boldsymbol{w}$ is an optimal solution and there exists i, j, k such that $w_{i,j}, w_{j,k} > 0$. If $i = k$, we can set at least one of $w_{i,j}$ and $w_{j,i}$ to 0 without violating the constraints. If $i \neq k$, we can set at least one of $w_{i,j}$ and $w_{j,k}$ to 0, and increase $w_{i,k}$ accordingly. In both cases, the resulting objective is at least as good. Repeating this for all i, k and j can enforce this condition for all i, k and j. Lemma EC.2 leads to the following bound for the repositioning cost $C(\boldsymbol{z})$.

LEMMA EC.3.

$$\frac{c_{\min}}{2} \sum_{i=1}^{n} |z_i| \le C(\boldsymbol{z}) \le \frac{c_{\max}}{2} \sum_{i=1}^{n} |z_i|. \tag{EC.2}$$

The proof follows from Lemma EC.2. There exists an optimal solution $\boldsymbol{w}$ to (1) such that

$$\begin{aligned} C(\boldsymbol{z}) = \sum_{i,j} c_{ij} w_{ij} &= \frac{1}{2} \sum_{j} \sum_{i} c_{ij} w_{ij} + \frac{1}{2} \sum_{j} \sum_{i} c_{ji} w_{ji} \\ &\le \frac{c_{\max}}{2} \sum_{j} z_j^+ + \frac{c_{\max}}{2} \sum_{j} z_j^- = \frac{c_{\max}}{2} \sum_{j} |z_j|. \end{aligned}$$

It is easy to see that, in (EC.2), the equality holds if $c_{ij} = c_{\max}$ for all i, j. Therefore, the bound is tight. The lower bound follows the same logic.

EC.1.3. Proof of Theorem 1

In this subsection, we provide a complete and self-contained proof for our main result, Theorem 1. We first provide some basic properties for directional derivatives. Recall the definition of the directional derivative:

$$u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{\eta}) = \lim_{t \downarrow 0} \frac{u(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma} + t\boldsymbol{\eta}) - u(\boldsymbol{x}, \boldsymbol{\gamma})}{t}.$$

LEMMA EC.4. *If $u(\boldsymbol{x}, \boldsymbol{\gamma})$ is jointly convex in $\boldsymbol{x}$ and $\boldsymbol{\gamma}$, then $u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{\eta})$ satisfies the following properties:*

- *(Positive Homogeneity)* $u'(\boldsymbol{x}, \boldsymbol{\gamma}; t\boldsymbol{z}, t\boldsymbol{\eta}) = t u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{\eta})$
- *(Sub-Additivity)* $u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}_1 + \boldsymbol{z}_2, \boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) \le u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}_1, \boldsymbol{\eta}_1) + u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}_2, \boldsymbol{\eta}_2)$.

Proof. These are well-known important properties for directional derivatives (see, for example, Rockafellar (1970)).

Now, we establish the fact that if $v_{t+1}(\cdot)$ is convex and additionally satisfies certain bounds on its directional derivatives, then for any realization of $\boldsymbol{d}_t, \boldsymbol{P}_t$, the function $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ can be reformulated as the convex program (8).

Proposition EC.1. *Suppose Assumptions 1 and 2 hold and*

1. $v_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1})$ *is continuous and jointly convex in* $\boldsymbol{x}_{t+1}$ *and* $\boldsymbol{\gamma}_{t+1}$,
2. $v'_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}; \boldsymbol{z}-\boldsymbol{\eta}, \boldsymbol{\eta}) \le C(-\boldsymbol{z}) + \sum_{i=1}^n (\beta_i + \rho c_{\max} - c_{\min})\eta_i$ *for any feasible direction* $(\boldsymbol{z}-\boldsymbol{\eta}, \boldsymbol{\eta})$ *with* $\boldsymbol{\eta} \ge \mathbf{0}$.

Then for all realizations $(\boldsymbol{d}, \boldsymbol{p})$, $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ *defined in (4) can be written as the following optimization problem:*

$$\begin{array}{rll} U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t) = & \min_{\boldsymbol{w}, \boldsymbol{x}_{t+1}, \gamma_{t+1}} & \sum_{i=1}^n \beta_i (d_{t,i} - w_i) + \rho v_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}) \\ & s.t. & \boldsymbol{x}_{t+1} = \boldsymbol{y}_t - \boldsymbol{w} + \boldsymbol{P}_t^T(\gamma_t + \boldsymbol{w}); \\ & & \boldsymbol{\gamma}_{t+1} = (\gamma_t + \boldsymbol{w}) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}); \\ & & \boldsymbol{w} \le \boldsymbol{y}_t; \\ & & \boldsymbol{w} \le \boldsymbol{d}_t. \end{array} \tag{EC.3}$$

Moreover, $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ *is continuous and jointly convex in* $\boldsymbol{y}_t$ *and* $\boldsymbol{\gamma}_t$.

Proof. Let $(\boldsymbol{w}^*, \boldsymbol{x}^*_{t+1}, \boldsymbol{\gamma}^*_{t+1})$ be the optimal solution to problem (EC.3). Clearly, if $\boldsymbol{w}^* = \min\{\boldsymbol{y}_t, \boldsymbol{d}_t\}$, the reformulation is correct. Now suppose that there exist an index i such that $\boldsymbol{w}^* < \min\{\boldsymbol{y}_t, \boldsymbol{d}_t\}$. Let $\hat{\boldsymbol{w}} = \boldsymbol{w}^* + \epsilon \boldsymbol{e}_i$ for a small $\epsilon > 0$. Then we have

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} &= \boldsymbol{y}_t - (\boldsymbol{w}^* + \epsilon \boldsymbol{e}_i) + \boldsymbol{P}_t^T(\gamma_t + \boldsymbol{w}^* + \epsilon \boldsymbol{e}_i) = \boldsymbol{x}^*_{t+1} + \epsilon(-\boldsymbol{e}_i + \boldsymbol{P}_t^T \boldsymbol{e}_i) \\ \hat{\boldsymbol{\gamma}}_{t+1} &= (\gamma_t + \boldsymbol{w}^* + \epsilon \boldsymbol{e}_i) \circ (\boldsymbol{e} - \boldsymbol{P}_t \boldsymbol{e}) = \boldsymbol{\gamma}^*_{t+1} + \epsilon \boldsymbol{e}_i \left(1 - \sum_{j=1}^n P_{tij}\right) = \boldsymbol{\gamma}^*_{t+1} + \epsilon \boldsymbol{e}_i (1 - p_t), \end{aligned}$$

where the last inequality is based on Assumption 1. Let

$$\hat{\boldsymbol{z}} = \boldsymbol{P}_t^T \boldsymbol{e}_i - \sum_{j=1}^n P_{tij} \boldsymbol{e}_i = \boldsymbol{P}_t^T \boldsymbol{e}_i - p_t \boldsymbol{e}_i, \quad \hat{\boldsymbol{\eta}} = \boldsymbol{e}_i (1 - p_t).$$

From the problem assumption, we have:

$$\begin{aligned} & v'_{t+1}\left(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}; -\boldsymbol{e}_i + \boldsymbol{P}_t^T \boldsymbol{e}_i, \boldsymbol{e}_i (1 - p_t)\right) \\ = {} & v'_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}; \hat{\boldsymbol{z}} - \hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\eta}}) \\ \le {} & C(-\hat{\boldsymbol{z}}) + \sum_{j=1}^n (\beta_i + \rho c_{\max} - c_{\min}) \hat{\eta}_j \\ \le {} & \frac{c_{\max}}{2} \|\hat{\boldsymbol{z}}\|_1 + (\beta_i + \rho c_{\max} - c_{\min})(1 - p_t) \qquad \text{(by Lemma EC.3)} \\ \le {} & \frac{c_{\max}}{2} \left(\|\boldsymbol{P}^T \boldsymbol{e}_i\|_1 + \left\| \sum_{j=1}^n P_{ij} \boldsymbol{e}_i \right\|_1 \right) + (\beta_i + \rho c_{\max} - c_{\min})(1 - p_t) \qquad \text{(by the triangle inequality)} \\ = {} & c_{\max} p_t + (\beta_i + \rho c_{\max} - c_{\min})(1 - p_t). \end{aligned}$$

Let

$$f(\boldsymbol{w}) = \sum_{i=1}^{n} \beta_i (d_{t,i} - w_i) + \rho v_{t+1}(\boldsymbol{x}_{t+1}(\boldsymbol{w}), \boldsymbol{\gamma}_{t+1}(\boldsymbol{w})),$$

where $\boldsymbol{x}_{t+1}(\boldsymbol{w}), \boldsymbol{\gamma}_{t+1}(\boldsymbol{w})$ are defined from the first two equalities. Then

$$\begin{aligned}
\frac{\partial f(\boldsymbol{w})}{\partial w_i} &= -\beta_i + \rho v'_{t+1}\left(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}; -\boldsymbol{e}_i + \boldsymbol{P}_t^T \boldsymbol{e}_i, \boldsymbol{e}_i (1-p_t)\right) \\
&\le -\beta_i + \rho c_{\max} p_t + \rho(\beta_i + \rho c_{\max} - c_{\min})(1-p_t) \\
&\le -\beta_i + \rho c_{\max} p_t + (\beta_i + \rho c_{\max} - c_{\min})(1-p_t) \qquad (\text{since } \rho \le 1) \\
&= \rho c_{\max} - c_{\min} - p_t(\beta_i - c_{\min}) \le 0 \qquad (\text{by Assumption 2}).
\end{aligned}$$

Therefore, it is always optimal to increase w_i for all $\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t$. This prove the first part of the proposition. The continuity of $u(\cdot)$ follows from the Dominated Convergence Theorem, as $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t) \le \sum_i \beta_i d_i + \rho \|v\|_\infty$. For the convexity, since the constraint of the optimization problem is linear, thus the feasible region is convex. Also $v_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1})$ is jointly convex in $\boldsymbol{x}_{t+1}$ and $\boldsymbol{\gamma}_{t+1}$, we have $U_t(\boldsymbol{y}_t, \boldsymbol{\gamma}_t, \boldsymbol{d}_t, \boldsymbol{P}_t)$ is jointly convex in $\boldsymbol{y}_t$ and $\boldsymbol{\gamma}_t$ (see, e.g., section 3.2.5 of Boyd et al. (2004)).

From Proposition EC.1, to show the convexity of $u_t(\cdot)$, we require $v_{t+1}(\cdot)$ to be convex and satisfy the bounds on directional derivative as presented in item 2. If the convexity of $u_t(\cdot)$ could imply the convexity and the aforementioned bounds of directional derivation for $v_t(\cdot)$, then the induction is complete. Unfortunately, the convexity of $u_t(\cdot)$ does not imply that

$$v'_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\gamma}_{t+1}; \boldsymbol{z} - \boldsymbol{\eta}, \boldsymbol{\eta}) \le C(-\boldsymbol{z}) + \sum_{i=1}^{n} (\beta_i + \rho c_{\max} - c_{\min}) \eta_i.$$

To overcome this difficulty, in Proposition EC.2, we assume a stronger condition on $v_{t+1}(\cdot)$ that implies that $u_t(\cdot)$ satisfies two types of bounds on partial derivative as stated in the Theorem 1. Then in Proposition EC.3, we show that if $u_t(\cdot)$ is convex and satisfies these two types of bounds, $v_t(\cdot)$ would be convex and satisfies the stronger condition assumed in Proposition EC.2. This would complete the induction step.

Before we present Proposition EC.2, we first show how to decompose the directional derivative of $U_{t,\boldsymbol{d},\boldsymbol{p}}(\cdot,\cdot) \triangleq U_t(\cdot,\cdot,\boldsymbol{d},\boldsymbol{p})$. Through this Lemma, we connect the directional derivatives of $u_t(\cdot)$ and those of $v_{t+1}(\cdot)$. We define some notation for indices sets. For any vector $\boldsymbol{y} \in \mathbb{R}^n$, we let $J^-(\boldsymbol{y}) = \{i \,|\, y_i < 0\}$, $J^0(\boldsymbol{y}) = \{i \,|\, y_i = 0\}$, $J^+(\boldsymbol{y}) = \{i \,|\, y_i > 0\}$, $J^{0+}(\boldsymbol{y}) = \{i \,|\, y_i \ge 0\}$ and $J^{0-}(\boldsymbol{y}) = \{i \,|\, y_i \le 0\}$.

Lemma EC.5. *For any realization $(\boldsymbol{d}, \boldsymbol{P})$, we have*

$$U'_{t,\boldsymbol{d},\boldsymbol{P}}(\boldsymbol{y}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{\eta}) = -\sum_{i \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^-(\boldsymbol{z}))} \beta_i z_i + \rho\, v'_{t+1}(\boldsymbol{x}, \boldsymbol{\zeta}; \boldsymbol{w}^+, \boldsymbol{\delta}^+), \tag{EC.4}$$

where

$$w_i^+ = \begin{cases} z_i + \iota_i + \sum_{j \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^-(\boldsymbol{z}))} z_j p_{ji} & \textit{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^+(\boldsymbol{z})), \\ \iota_i + \sum_{j \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^-(\boldsymbol{z}))} z_j p_{ji} & \textit{for } i \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^{0-}(\boldsymbol{z})), \end{cases}$$

with $\iota_i \triangleq \sum_{j=1}^n \eta_j p_{ji}$ *and*

$$\delta_i^+ = \begin{cases} \eta_i (1 - \sum_{j=1}^n p_{ij}) & \textit{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^+(\boldsymbol{z})), \\ (\eta_i + z_i)(1 - \sum_{j=1}^n p_{ij}) & \textit{for } i \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^{0-}(\boldsymbol{z})), \end{cases}$$

and

$$\boldsymbol{x} = \tau_x(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}); \;\; \boldsymbol{\zeta} = \tau_\gamma(\boldsymbol{y}, \boldsymbol{\gamma}, \boldsymbol{d}, \boldsymbol{P}).$$

Proof. Let

$$\vartheta_i = \sum_{j=1}^n \gamma_j p_{ji}, \quad \iota_i = \sum_{j=1}^n \eta_j p_{ji}$$

Note that

$$L(\boldsymbol{y}, \boldsymbol{d}) = \sum_{i \in J^-(\boldsymbol{y}-\boldsymbol{d})} \beta_i (d_i - y_i), \tag{EC.5}$$

and let the next state, under $(\boldsymbol{d}, \boldsymbol{P})$, be defined by $\boldsymbol{x}(\boldsymbol{y}, \boldsymbol{\gamma}), \boldsymbol{\zeta}(\boldsymbol{y}, \boldsymbol{\gamma})$, with components

$$x_i(\boldsymbol{y}, \boldsymbol{\gamma}) = \begin{cases} (y_i - d_i) + \vartheta_i + \sum_{j \in J^+(\boldsymbol{y}-\boldsymbol{d})} d_j p_{ji} + \sum_{j \in J^{0-}(\boldsymbol{y}-\boldsymbol{d})} y_j p_{ji} & \text{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}), \\ \vartheta_i + \sum_{j \in J^+(\boldsymbol{y}-\boldsymbol{d})} d_j p_{ji} + \sum_{j \in J^{0-}(\boldsymbol{y}-\boldsymbol{d})} y_j p_{ji} & \text{for } i \in J^{0-}(\boldsymbol{y}-\boldsymbol{d}), \end{cases} \tag{EC.6}$$

$$\zeta_i(\boldsymbol{y}, \boldsymbol{\gamma}) = \begin{cases} (\gamma_i + d_i)(1 - \sum_{j=1}^n p_{ij}) & \text{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}), \\ (\gamma_i + y_i)(1 - \sum_{j=1}^n p_{ij}) & \text{for } i \in J^{0-}(\boldsymbol{y}-\boldsymbol{d}). \end{cases} \tag{EC.7}$$

Choose t small enough so that the following hold:

$$\begin{aligned} J^-(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}) &= J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^-(\boldsymbol{z})), \\ J^0(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}) &= J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^0(\boldsymbol{z}), \\ J^+(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}) &= J^+(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^+(\boldsymbol{z})), \\ J^{0-}(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}) &= J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d}) \cap J^{0-}(\boldsymbol{z})), \end{aligned} \tag{EC.8}$$

where the last equation follows directly from the first and second. For $\boldsymbol{y} + t\boldsymbol{z}$, we have, directly by (EC.5), that

$$L(\boldsymbol{y} + t\boldsymbol{z}, \boldsymbol{d}) = \sum_{i \in J^-(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d})} \beta_i (d_i - y_i - t z_i),$$

and directly from (EC.6) and (EC.7),

$$\begin{aligned} & x_i(\boldsymbol{y} + t\boldsymbol{z}, \boldsymbol{\gamma} + t\boldsymbol{\eta}) \\ &= \begin{cases} \begin{aligned} & y_i + t z_i - d_i + \vartheta_i + t\iota_i + \textstyle\sum_{j \in J^+(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d})} d_j p_{ji} \\ & + \textstyle\sum_{j \in J^{0-}(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d})} (y_j + t z_j) p_{ji} \end{aligned} & \text{for } i \in J^+(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}), \\ \\ \begin{aligned} & \vartheta_i + t\iota_i + \textstyle\sum_{j \in J^+(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d})} d_j p_{ji} \\ & + \textstyle\sum_{j \in J^{0-}(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d})} (y_j + t z_j) p_{ji} \end{aligned} & \text{for } i \in J^{0-}(\boldsymbol{y} + t\boldsymbol{z} - \boldsymbol{d}), \end{cases} \end{aligned}$$

and

$$\zeta_i(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}) = \begin{cases} (\gamma_i + t\eta_i + d_i)(1-\sum_{j=1}^n p_{i,j}) & \text{for } i \in J^+(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d}), \\ (\gamma_i + t\eta_i + y_i + tz_i)(1-\sum_{j=1}^n p_{i,j}) & \text{for } i \in J^{0-}(\boldsymbol{y}+t\boldsymbol{z}-\boldsymbol{d}). \end{cases}$$

It follows by (EC.8) that

$$L(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{d}) - L(\boldsymbol{y},\boldsymbol{d}) = -t \sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^-(\boldsymbol{z}))} \beta_i z_i.$$

For the state update equations, we have

$$\begin{aligned} &x_i(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}) - x_i(\boldsymbol{y},\boldsymbol{\gamma}) \\ &= \begin{cases} tz_i + t\iota_i + \sum_{j\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^-(\boldsymbol{z}))} tz_j p_{ji} & \text{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^+(\boldsymbol{z})), \\ t\iota_i + \sum_{j\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^-(\boldsymbol{z}))} tz_j p_{ji} & \text{for } i \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^{0-}(\boldsymbol{z})), \end{cases} \end{aligned}$$

and

$$\begin{aligned} &\zeta_i(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}) - \zeta_i(\boldsymbol{y},\boldsymbol{\gamma}) \\ &\qquad = \begin{cases} t\eta_i(1-\sum_{j=1}^n p_{ij}) & \text{for } i \in J^+(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^+(\boldsymbol{z})), \\ t(\eta_i+z_i)(1-\sum_{j=1}^n p_{ij}) & \text{for } i \in J^-(\boldsymbol{y}-\boldsymbol{d}) \cup (J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^{0-}(\boldsymbol{z})). \end{cases} \end{aligned}$$

Set

$$\boldsymbol{w}^+ = \frac{\boldsymbol{x}(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}) - \boldsymbol{x}(\boldsymbol{y},\boldsymbol{\gamma})}{t}, \quad \boldsymbol{\delta}^+ = \frac{\boldsymbol{\zeta}(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}) - \boldsymbol{\zeta}(\boldsymbol{y},\boldsymbol{\gamma})}{t}.$$

Then

$$\lim_{t\to 0} \frac{v_{t+1}(\boldsymbol{x}(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta}),\boldsymbol{\zeta}(\boldsymbol{y}+t\boldsymbol{z},\boldsymbol{\gamma}+t\boldsymbol{\eta})) - v_{t+1}(\boldsymbol{x}(\boldsymbol{y},\boldsymbol{\gamma}),\boldsymbol{\zeta}(\boldsymbol{y},\boldsymbol{\gamma}))}{t} = v'_{t+1}(\boldsymbol{x},\boldsymbol{\zeta};\boldsymbol{w}^+,\boldsymbol{\delta}^+).$$

It follows that

$$U'_{t,\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{z},\boldsymbol{\eta}) = -\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d})\cap J^-(\boldsymbol{z}))} \beta_i z_i + \rho v'_{t+1}(\boldsymbol{x},\boldsymbol{\zeta};\boldsymbol{w}^+,\boldsymbol{\delta}^+). \tag{EC.9}$$

This concludes the proof.

Now we are ready to present Proposition EC.2, which shows that given the convexity of $v_{t+1}(\boldsymbol{x}_{t+1},\boldsymbol{\gamma}_{t+1})$ and certain bounds on its directional derivatives, the function $u_t(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)$ satisfies two types of bounds on its directional derivatives.

PROPOSITION EC.2. *Suppose Assumptions 1 and 2 hold and*

1. $v_{t+1}(\boldsymbol{x}_{t+1},\boldsymbol{\gamma}_{t+1})$ *is continuous and jointly convex in* $\boldsymbol{x}_{t+1}$ *and* $\boldsymbol{\gamma}_{t+1}$*,*
2. $v'_{t+1}(\boldsymbol{x}_{t+1},\boldsymbol{\gamma}_{t+1};\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta}) \le C(-\boldsymbol{z}) + \sum_{i=1}^n(\beta_i + \rho c_{\max} - c_{\min})\eta_i$ *for any feasible direction* $(\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta})$ *with* $\boldsymbol{\eta}\ge\boldsymbol{0}$*,*
3. $v'_{t+1}(\boldsymbol{x}_{t+1},\boldsymbol{\gamma}_{t+1};\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta}) \le C(-\boldsymbol{z}) + \sum_{i=1}^n \beta_i\eta_i$ *for any feasible direction* $(\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta})$ *with* $\boldsymbol{\eta}\ge\boldsymbol{0}$*,*

4. *$v'_{t+1}(\boldsymbol{x}_{t+1},\boldsymbol{\gamma}_{t+1};\mathbf{0},\boldsymbol{z}) \le (\rho c_{\max}/2)\sum_{i=1}^n |z_i|$ for any feasible direction $(\mathbf{0},\boldsymbol{z})$ with $\boldsymbol{e}^T\boldsymbol{z}=\mathbf{0}$,*

then we have:

1. *$|u'_t(\boldsymbol{y}_t,\boldsymbol{\gamma}_t;\mp\boldsymbol{\xi},\pm\boldsymbol{\xi})| \le \sum_{i=1}^n \beta_i\xi_i$ for all $(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)\in\Delta$ and any feasible direction $(\mp\boldsymbol{\xi},\pm\boldsymbol{\xi})$ with $\boldsymbol{\xi}\ge\mathbf{0}$;*

2. *$u'_t(\boldsymbol{y}_t,\boldsymbol{\gamma}_t;\mathbf{0},\boldsymbol{z}) \le (\rho c_{\max}/2)\sum_{i=1}^n |z_i|$ for all $(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)\in\Delta$ and any feasible direction $(\mathbf{0},\boldsymbol{z})$ with $\boldsymbol{e}^T\boldsymbol{z}=0$.*

Proof. We omit the subscript t to reduce notation. To show the first inequality, we start with showing that $u'(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi})\le\sum_i\beta_i\xi_i$. From Lemma EC.5, noting that $-\boldsymbol{\xi}\le 0$, we have

$$U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\gamma;-\boldsymbol{\xi},\boldsymbol{\xi}) = \sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \beta_i\xi_i + \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{w}-\boldsymbol{\delta},\boldsymbol{\delta}),$$

where

$$w_i = \begin{cases} -\xi_i\sum_{j=1}^n p_{ij} + \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}\xi_j p_{ji} & \text{for } i\in J^+(\boldsymbol{y}-\boldsymbol{d}),\\ \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}\xi_j p_{ji} & \text{for } i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d}),\end{cases}$$

and

$$\delta_i = \begin{cases} \xi_i(1-\sum_{j=1}^n p_{ij}) = \xi_i(1-p) & \text{for } i\in J^+(\boldsymbol{y}-\boldsymbol{d}),\\ 0 & \text{for } i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d}).\end{cases}$$

Note that $\boldsymbol{e}^T\boldsymbol{w}=0$ and so it is clear that $(\boldsymbol{w}-\boldsymbol{\delta},\boldsymbol{\delta})$ is a feasible direction at $(\boldsymbol{x},\boldsymbol{\zeta})$. It follows that

$$\begin{aligned}
&U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\gamma;-\boldsymbol{\xi},\boldsymbol{\xi})\\
=&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})}\beta_i\xi_i + \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{w}-\boldsymbol{\delta},\boldsymbol{\delta})\\
\le&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \rho C(-\boldsymbol{w}) + \rho\sum_{i=1}^n(\beta_i+\rho c_{\max}-c_{\min})|\delta_i| \text{ (by assumption)}\\
\le&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \frac{\rho c_{\max}}{2}\sum_{i=1}^n|w_i| + \rho\sum_{i=1}^n(\beta_i+\rho c_{\max}-c_{\min})|\delta_i| \text{ (by Lemma EC.3)}\\
\le&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \frac{2\rho c_{\max}}{2}\sum_{i=1}^n\sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}|\xi_j|p_{ji}\\
&+\rho\sum_{i\in J^+(\boldsymbol{y}-\boldsymbol{d})}(\beta_i+\rho c_{\max}-c_{\min})|\xi_i|(1-p) \text{ (by the triangle inequality)}\\
=&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}(p\rho c_{\max}+\rho(\beta_i+\rho c_{\max}-c_{\min})(1-p))|\xi_j|\\
\le&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}(p\rho c_{\max}+(\beta_i+\rho c_{\max}-c_{\min})(1-p))|\xi_j|\\
\le&\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))}\beta_i|\xi_i| + \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})}(\beta_i+(\rho c_{\max}-c_{\min})-p(\beta_i-c_{\min}))|\xi_j|
\end{aligned}$$

$$
\begin{aligned}
&\le \sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})\cup(J^0(\boldsymbol{y}-\boldsymbol{d}))} \beta_i|\xi_i| + \sum_{i\in J^+(\boldsymbol{y}-\boldsymbol{d})} \beta_i|\xi_i| \text{ (by Assumption 2)}\\
&= \sum_{i=1}^n \beta_i|\xi_i|
\end{aligned}
$$

So, $U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi}) \le \sum_{i=1}^n \beta_i|\xi_i| = \sum_{i=1}^n \beta_i\xi_i$ holds for each $(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta$ and $(\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta})$ feasible. It follows that $u'(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi}) = \int U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi})\,d\mu \le \sum_i \beta_i\xi_i$. From Lemma EC.1, $u(\boldsymbol{y},\boldsymbol{\gamma})$ is convex, thus

$$
u'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) \ge -u'(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi}) \ge -\sum_i \beta_i\xi_i.
$$

Now we show that $u'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) \le \sum_{i=1}^n \beta_i\xi_i$ for all $(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta$ for all feasible direction $(\boldsymbol{\xi},-\boldsymbol{\xi})$ with $\boldsymbol{\xi}\ge\boldsymbol{0}$. From the previous analysis, we have

$$
U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) = -\sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})} \beta_i\xi_i + \rho v'(\boldsymbol{x},\boldsymbol{\zeta};\boldsymbol{w}+\boldsymbol{\delta},-\boldsymbol{\delta})
$$

where

$$
w_i = \begin{cases} \xi_i \sum_{j=1}^n p_{ij} - \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \xi_j p_{ji} & \text{for } i\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d}),\\ -\sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \xi_j p_{ji} & \text{for } i\in J^-(\boldsymbol{y}-\boldsymbol{d}), \end{cases}
$$

$$
\delta_i = \begin{cases} \xi_i(1-\sum_{j=1}^n p_{ij}) = \xi_i(1-p) & \text{for } i\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d}),\\ 0 & \text{for } i\in J^-(\boldsymbol{y}-\boldsymbol{d}). \end{cases}
$$

Clear $(\boldsymbol{w}+\boldsymbol{\delta},-\boldsymbol{\delta})$ is a feasible direction at $(\boldsymbol{x},\boldsymbol{\zeta})$. It follows that

$$
\begin{aligned}
&U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi})\\
&= -\beta_i \sum_{i\in J^-(\boldsymbol{y}-\boldsymbol{d})} \xi_i + \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{w}+\boldsymbol{\delta},-\boldsymbol{\delta})\\
&\le \frac{\rho c_{\max}}{2}\sum_{i=1}^n |w_i| + \rho\sum_{i=1}^n \beta_i|\delta_i| \quad \text{(from the third inequality of the assumption and } \boldsymbol{\xi}\ge 0)\\
&\le \frac{1}{2}\sum_{i=1}^n \beta_i|w_i| + \rho\sum_{i=1}^n \beta_i|\delta_i| \quad (\text{since } \rho c_{\max}\le\beta_i, \rho\le 1)\\
&\le \frac{\rho}{2}2\sum_{i=1}^n \sum_{j\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \beta_i|\xi_j|p_{ji} + \rho\sum_{i\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \beta_i|\xi_i|\left(1-\sum_{j=1}^n p_{ij}\right) \text{ (by the triangle inequality)}\\
&\le \rho\sum_{i\in J^+(\boldsymbol{y}-\boldsymbol{d})\cup J^0(\boldsymbol{y}-\boldsymbol{d})} \beta_i|\xi_i| \le \sum_{i=1}^n \beta_i|\xi_i|.
\end{aligned}
$$

So, $U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) \le \sum_{i=1}^n \beta_i|\xi_i| = \sum_{i=1}^n \beta_i\xi_i$ holds for each $(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta$ and $(\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta})$ feasible. It follows that $u'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) = \int U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi})\,d\mu \le \sum_i \beta_i\xi_i$. From Lemma EC.1, $u(\boldsymbol{y},\boldsymbol{\gamma})$ is convex, thus

$$
u'(\boldsymbol{y},\boldsymbol{\gamma};-\boldsymbol{\xi},\boldsymbol{\xi}) \ge -u'(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{\xi},-\boldsymbol{\xi}) \ge -\sum_i \beta_i\xi_i.
$$

Summing up all the above, we have shown the first inequality.

Now we show the second inequality. From Lemma EC.5, we have

$$U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z}) = \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\iota,\boldsymbol{z}(1-p))$$

where $\iota_i = \sum_{j=1}^n z_j p_{ji}\ \forall i$ and $p = \sum_{j=1}^n p_{ij}$ (Assumption 1). Therefore,

$$\sum_{i=1}^n \iota_i = \sum_{j=1}^n \sum_{i=1}^n z_j p_{ji} = \sum_{j=1}^n p z_j = 0,$$

and $\boldsymbol{e}^T\boldsymbol{z}(1-p) = 0$. It follows that

$$\begin{aligned}
U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z}) &= \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{\iota},\boldsymbol{z}(1-p)) \\
&\le \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{\iota},\boldsymbol{0}) + \rho v'(\boldsymbol{x},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z}(1-p)) \text{ (by subadditivity, Lemma EC.4)} \\
&\le \rho C(-\boldsymbol{\iota}) + \frac{\rho^2 c_{\max}}{2}\sum_{i=1}^n |z_i|(1-p) \text{ (by assumption)} \\
&\le \frac{\rho c_{\max}}{2}\sum_{i=1}^n |\iota_i| + \frac{\rho^2 c_{\max}}{2}\sum_{i=1}^n |z_i|(1-p) \text{ (by Lemma EC.3)} \\
&\le \frac{\rho c_{\max}}{2}\sum_{i=1}^n\sum_{j=1}^n |z_j| p_{ji} + \frac{\rho^2 c_{\max}}{2}\sum_{i=1}^n |z_i|(1-p) \text{ (by the triangle inequality)} \\
&\le \frac{\rho c_{\max}}{2}\sum_{i=1}^n |z_i|. \text{ (by the subadditivity)}
\end{aligned}$$

So, $U'_{\boldsymbol{d},\boldsymbol{p}}(\boldsymbol{y},\boldsymbol{\gamma};\boldsymbol{0},\boldsymbol{z}) \le (\rho c_{\max}/2)\sum_{i=1}^n |z_i|$ holds for all realizations $(\boldsymbol{d},\boldsymbol{p})$. It follows that the integral also satisfies the condition.

The upcoming result, Proposition EC.3, assists with the eventual induction over t by stating that if $u_t(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)$ is convex and certain bounds on its directional derivatives are satisfied, then $v_t(\boldsymbol{x}_t,\boldsymbol{\gamma}_t)$ not only is convex, but also satisfies the bounds on the directional derivatives required by Proposition EC.2. Before continuing, we introduce a technical lemma that carefully analyzes the optimal repositioning plan. The main difficulty in showing Proposition EC.3 is that there exist some situations where $x_{ti} > 0$, but $y^*_{ti} = 0$ where $\boldsymbol{y}^*_t$ is the optimal solution to $v_t(\boldsymbol{x}_t,\boldsymbol{\gamma}_t)$. In this case, if we move along the direction $\boldsymbol{z}$ with $z_i < 0$ starting from $\boldsymbol{x}_t$, we are not able to move along this direction if we start from position $\boldsymbol{y}^*$. Therefore, we would need to find another direction such that it is feasible for position $\boldsymbol{y}^*$, but the cost of repositioning can be controlled. This is the main purpose of Lemma EC.6. This lemma plays a crucial role in the proof of Proposition EC.3.

LEMMA EC.6. *Suppose $\boldsymbol{x},\boldsymbol{y} \ge 0$ and $\mathbf{e}^T\boldsymbol{x} = \mathbf{e}^T\boldsymbol{y}$. Suppose $\boldsymbol{x}-\boldsymbol{\eta} \ge 0$. Then there exists another vector $\boldsymbol{\xi}$ such that: 1) $\boldsymbol{y}-\boldsymbol{\xi} \ge 0$, 2) $\mathbf{e}^T\boldsymbol{\xi} = \mathbf{e}^T\boldsymbol{\eta}$, 3) $\sum_{j=1}^n |\xi_j - \eta_j| \le 2\sum_{j=1}^n |\eta_j|$, 4) $C(\boldsymbol{y}-\boldsymbol{\xi}-\boldsymbol{x}+\boldsymbol{\eta}) = C(\boldsymbol{y}-\boldsymbol{x}) - C(\boldsymbol{\xi}-\boldsymbol{\eta})$. Moreover, if $\boldsymbol{\eta} \ge 0$, we have $\boldsymbol{\xi} \ge 0$ and*

$$\sum_{j=1}^n \xi_i + \frac{1}{2}\sum_{j=1}^n |\xi_j - \eta_j| \le \sum_{j=1}^n \eta_j \tag{EC.10}$$

Proof. Let $\mathcal{I}=\{i\,|\,y_i-\eta_i<0\}$. If $\mathcal{I}=\emptyset$, we have $\boldsymbol{y}-\boldsymbol{\eta}\geq 0$ and we just find $\boldsymbol{\xi}=\boldsymbol{\eta}$. Thus in the followings, we assume that $\mathcal{I}\neq\emptyset$. We let $\mathcal{J}^-=\{i\,|\,y_i-x_i\leq 0\}$ and $\mathcal{J}^+=\{i\,|\,y_i-x_i>0\}$. Note that $y_i-x_i<0\;\forall i\in\mathcal{I}$, so $\mathcal{I}\subseteq\mathcal{J}^-$.

Lemma EC.2 suggests that there exists $\boldsymbol{w}\geq 0$ such that $\boldsymbol{c}\cdot\boldsymbol{w}=C(\boldsymbol{y}-\boldsymbol{x})$ and

$$y_j-x_j=\begin{cases}-\sum_{i\in\mathcal{J}^+}w_{ji}=-\sum_i w_{ji} & \text{for } j\in\mathcal{J}^-;\\ \sum_{i\in\mathcal{J}^-}w_{ij}=\sum_i w_{ij} & \text{for } j\in\mathcal{J}^+.\end{cases}$$

The interpretation of the first case above is that locations j with more inventory than the target level y_j transfer inventory to locations that are below the target level. The second case is interpreted in an analogous way. Let $\boldsymbol{\xi}$ be such that

$$\xi_j=\begin{cases} y_j & \text{for } j\in\mathcal{I};\\ \eta_j & \text{for } j\in\mathcal{J}^-\setminus\mathcal{I};\\ \sum_{i\in\mathcal{I}}\frac{\eta_i-y_i}{x_i-y_i}w_{ij}+\eta_j & \text{for } j\in\mathcal{J}^+.\end{cases}$$

We claim that $\boldsymbol{\xi}$ satisfies the desired properties. We first verify that $y_j-\xi_j\geq 0$. This is clearly true if $j\in\mathcal{J}^-$ from the construction of $\boldsymbol{\xi}$. Now let $j\in\mathcal{J}^+$, then we have

$$y_j-\xi_j=y_j-\sum_{i\in\mathcal{I}}\frac{\eta_i-y_i}{x_i-y_i}w_{ij}-\eta_j\geq y_j-\sum_{i\in\mathcal{I}}w_{ij}-\eta_j\geq y_j-\sum_i w_{ij}-\eta_j=x_j-\eta_j\geq 0,$$

where the first inequality follows from $x_i\geq\eta_i>y_i$ for $i\in\mathcal{I}$. Therefore, $\boldsymbol{y}-\boldsymbol{\xi}\geq 0$ and part (1) is complete. Also, using $x_j-y_j=\sum_{j\in\mathcal{J}^+}w_{ij}$, we have:

$$\begin{aligned}\sum_j\xi_j&=\sum_{j\in\mathcal{I}}y_j+\sum_{j\in\mathcal{J}^-\setminus\mathcal{I}}\eta_j+\sum_{j\in\mathcal{J}^+}\left(\sum_{i\in\mathcal{I}}\frac{\eta_i-y_i}{x_i-y_i}w_{ij}+\eta_j\right)\\ &=\sum_{j\in\mathcal{I}}y_j+\sum_{j\in(\mathcal{J}^-\setminus\mathcal{I})\cup\mathcal{J}^+}\eta_j+\sum_{i\in\mathcal{I}}\frac{\eta_i-y_i}{x_i-y_i}(x_i-y_i)=\sum_i\eta_i,\end{aligned}$$

verifying part (2). Note that:

$$\xi_j-\eta_j=\begin{cases} y_j-\eta_j & \text{for } j\in\mathcal{I};\\ 0 & \text{for } j\in\mathcal{J}^-\setminus\mathcal{I};\\ \sum_{i\in\mathcal{I}}\frac{\eta_i-y_i}{x_i-y_i}w_{ij} & \text{for } j\in\mathcal{J}^+.\end{cases}$$

To show part (3), we have

$$\sum_{j=1}^n|\xi_j-\eta_j|=2\sum_{j\in\mathcal{I}}(\eta_j-y_j)\leq 2\sum_{j=1}^n|\eta_j|.$$

To show that $C(\boldsymbol{y}-\boldsymbol{\xi}-\boldsymbol{x}+\boldsymbol{\eta})\leq C(\boldsymbol{y}-\boldsymbol{x})-C(\boldsymbol{\xi}-\boldsymbol{\eta})$, let $\bar{\boldsymbol{w}}$ be such that

$$\bar{w}_{ij}=\begin{cases}(1-\frac{\eta_i-y_i}{x_i-y_i})w_{ij} & \text{for } i\in\mathcal{I};\\ w_{ij} & \text{for } i\notin\mathcal{I}.\end{cases}$$

Then for all $j \in \mathcal{J}^+$, we have

$$\begin{aligned}
& y_j - \xi_j - (x_j - \eta_j) - \sum_i \bar{w}_{ij} + \sum_i \bar{w}_{ji} \\
&= (y_j - x_j) - \xi_j + \eta_j - \sum_{i \in \mathcal{I}} \bar{w}_{ij} - \sum_{i \notin \mathcal{I}} \bar{w}_{ij} \\
&= \sum_i w_{ij} - \sum_{i \in \mathcal{I}} \frac{\eta_i - y_i}{x_i - y_i} w_{ij} - \sum_i w_{ij} + \sum_{i \in \mathcal{I}} \frac{\eta_i - y_i}{x_i - y_i} w_{ij} = 0,
\end{aligned}$$

where we used $\sum_i \bar{w}_{ji} = 0$. Similarly, for all $j \in \mathcal{I}$, we have

$$\begin{aligned}
& x_j - \eta_j - y_j + \xi_j - \sum_i \bar{w}_{ji} + \sum_i \bar{w}_{ij} \\
&= x_j - \eta_j - \sum_i \left(1 - \frac{\eta_j - y_j}{x_j - y_j}\right) w_{ji} \\
&= x_j - \sum_i w_{ji} - \eta_j - \frac{\eta_j - y_j}{x_j - y_j}(y_j - x_j) = 0
\end{aligned}$$

and for all $j \in \mathcal{J}^- \backslash \mathcal{I}$, we have

$$x_j - \eta_j - y_j + \xi_j - \sum_i \bar{w}_{ji} + \sum_i \bar{w}_{ij} = x_j - y_j - \sum_i w_{ji} = 0.$$

Therefore, we have shown that $\bar{\boldsymbol{w}}$ is a feasible solution to the optimization problem for $C(\cdot)$ defined in (1) at $\boldsymbol{y} - \boldsymbol{\xi} - \boldsymbol{x} + \boldsymbol{\eta}$. Thus, we have:

$$\begin{aligned}
C(\boldsymbol{y} - \boldsymbol{\xi} - \boldsymbol{x} + \boldsymbol{\eta}) &\leq \sum_{i,j} c_{ij} \bar{w}_{ij} \\
&= \sum_{i,j} c_{ij} w_{ij} - \sum_{i \in \mathcal{I}} \sum_j c_{ij} \left(\frac{\eta_i - y_i}{x_i - y_i} w_{ij}\right) \\
&= C(\boldsymbol{y} - \boldsymbol{x}) - \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}^+} c_{ij} \left(\frac{\eta_i - y_i}{x_i - y_i} w_{ij}\right).
\end{aligned}$$

If we define $\hat{\boldsymbol{w}}$ as

$$\hat{w}_{ij} = \begin{cases} \frac{\eta_i - y_i}{x_i - y_i} w_{ij} & \text{for } i \in \mathcal{I}, j \in \mathcal{J}^+; \\ 0 & \text{otherwise,} \end{cases}$$

then we can check that $\hat{\boldsymbol{w}}$ is a feasible solution to (1) at $\boldsymbol{\xi} - \boldsymbol{\eta}$. Therefore, we have

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}^+} c_{ij} \left(\frac{\eta_i - y_i}{x_i - y_i} w_{ij}\right) = \boldsymbol{c} \cdot \hat{\boldsymbol{w}} \geq C(\boldsymbol{\xi} - \boldsymbol{\eta}).$$

Finally, we have

$$C(\boldsymbol{y} - \boldsymbol{\xi} - \boldsymbol{x} + \boldsymbol{\eta}) \leq C(\boldsymbol{y} - \boldsymbol{x}) - \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}^+} c_{ij} \left(\frac{\eta_i - y_i}{x_i - y_i} w_{ij}\right) \leq C(\boldsymbol{y} - \boldsymbol{x}) - C(\boldsymbol{\xi} - \boldsymbol{\eta}).$$

Together with the subadditivity of $C(\cdot)$, we conclude that

$$C(\boldsymbol{y} - \boldsymbol{\xi} - \boldsymbol{x} + \boldsymbol{\eta}) = C(\boldsymbol{y} - \boldsymbol{x}) - C(\boldsymbol{\xi} - \boldsymbol{\eta}),$$

proving part (4). The last claim follows directly from the construction of $\boldsymbol{\xi}$ and parts (2) and (3).

Now we are ready to prove Proposition EC.3.

PROPOSITION EC.3. *Suppose $u_t(\cdot)$ is convex and continuous in Δ and*

1. *$u_t'(\boldsymbol{y}_t,\boldsymbol{\gamma}_t;\pm\boldsymbol{\eta},\mp\boldsymbol{\eta}) \le \sum_{i=1}^n \beta_i\eta_i$ for all $(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)\in\Delta$ and for any feasible direction $(\pm\boldsymbol{\eta},\mp\boldsymbol{\eta})$ with $\boldsymbol{\eta}\ge\mathbf{0}$;*

2. *$u_t'(\boldsymbol{y}_t,\boldsymbol{\gamma}_t;\mathbf{0},\boldsymbol{z}) \le (\rho c_{\max}/2)\sum_{i=1}^n |z_i|$ for all $(\boldsymbol{y}_t,\boldsymbol{\gamma}_t)\in\Delta$ and for any feasible direction $(\mathbf{0},\boldsymbol{z})$ with $\boldsymbol{e}^T\boldsymbol{z}=0$.*

Then, the value function $v_t(\cdot)$ is convex and continuous in Δ with $\Omega_v(\boldsymbol{\gamma})=\Delta_{n-1}(I)$ for $\boldsymbol{\gamma}\in S$. For each $(\boldsymbol{x}_t,\boldsymbol{\gamma}_t)\in\Delta$ and $\boldsymbol{\eta}\ge\mathbf{0}$, the directional derivatives satisfy

$$v_t'(\boldsymbol{x}_t,\boldsymbol{\gamma}_t;\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta}) \le C(-\boldsymbol{z})+\sum_{i=1}^n(\beta_i+\rho c_{\max}-c_{\min})\eta_i \tag{EC.11}$$

for any feasible direction $(\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta})$ and

$$v_t'(\boldsymbol{x}_t,\boldsymbol{\gamma}_t;\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta}) \le C(-\boldsymbol{z})+\sum_{i=1}^n\beta_i\eta_i \tag{EC.12}$$

for any feasible direction $(\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta})$. In addition,

$$v_t'(\boldsymbol{x}_t,\boldsymbol{\gamma}_t;\mathbf{0},\boldsymbol{z}) \le \frac{\rho c_{\max}}{2}\sum_{i=1}^n|z_i| \tag{EC.13}$$

for any feasible direction $(\mathbf{0},\boldsymbol{z})$ with $\boldsymbol{e}^T\boldsymbol{z}=0$.

Proof. We omit the subscript t throughout the proof to reduce notation. To show that $v(\cdot)$ is convex, suppose $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are optimal solutions of (16) for $(\boldsymbol{x}_1,\boldsymbol{\gamma}_1)$ and $(\boldsymbol{x}_2,\boldsymbol{\gamma}_2)$, respectively. Then, $\lambda\boldsymbol{y}_1+(1-\lambda)\boldsymbol{y}_2\in\Delta_{n-1}(\lambda\boldsymbol{e}^T\boldsymbol{x}_1+(1-\lambda)\boldsymbol{e}^T\boldsymbol{x}_2)$ and thus

$$\begin{aligned} v(\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2,\lambda\boldsymbol{\gamma}_1+(1-\lambda)\boldsymbol{\gamma}_2) &\le u(\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2,\lambda\boldsymbol{\gamma}_1+(1-\lambda)\boldsymbol{\gamma}_2)\\ &\quad + C(\lambda(\boldsymbol{y}_1-\boldsymbol{x}_1)+(1-\lambda)(\boldsymbol{y}_2-\boldsymbol{x}_2))\\ &\le \lambda v(\boldsymbol{x}_1,\boldsymbol{\gamma}_1)+(1-\lambda)v(\boldsymbol{x}_2,\boldsymbol{\gamma}_2), \end{aligned}$$

by convexity of $u(\cdot)$ and Lemma EC.1. Continuity follows from Berge's Maximum Theorem, as the set-valued map $\boldsymbol{x}\mapsto\Delta_{n-1}(I)$ is continuous. To show the result in (EC.11), suppose $(\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta})$ is a feasible direction. Let $\boldsymbol{y}^*$ be an optimal solution to equation (16) at $(\boldsymbol{x},\boldsymbol{\gamma})$. Therefore,

$$v(\boldsymbol{x},\boldsymbol{\gamma})=\min_{\boldsymbol{y}\in\Delta_{n-1}(\boldsymbol{e}^T\boldsymbol{x})} C(\boldsymbol{y}-\boldsymbol{x})+u(\boldsymbol{y},\boldsymbol{\gamma})=C(\boldsymbol{y}^*-\boldsymbol{x})+u(\boldsymbol{y}^*,\boldsymbol{\gamma}).$$

Let $t>0$ be small enough such that $\boldsymbol{x}+t(\boldsymbol{z}-\boldsymbol{\eta})\ge 0$. According to Lemma EC.6, there exists a vector $\boldsymbol{\xi}\ge 0$ such that for small enough t: 1) $\boldsymbol{y}^*-t\boldsymbol{\xi}\ge 0$, 2) $\boldsymbol{e}^T\boldsymbol{\xi}=\boldsymbol{e}^T\boldsymbol{\eta}$, 3) 4) $C(\boldsymbol{y}^*-t\boldsymbol{\xi}-\boldsymbol{x}-$

$t\boldsymbol{z}+t\boldsymbol{\eta}) = C(\boldsymbol{y}^*-\boldsymbol{x}-t\boldsymbol{z}) - tC(\boldsymbol{\xi}-\boldsymbol{\eta})$. Therefore, $\boldsymbol{y}^*-t\boldsymbol{\xi}$ is a feasible solution to equation (16) at $(\boldsymbol{x}+t\boldsymbol{z}-t\boldsymbol{\eta}, \boldsymbol{\gamma}+t\boldsymbol{\eta})$ and thus we have

$$\begin{aligned}
&\frac{v(\boldsymbol{x}+t\boldsymbol{z}-t\boldsymbol{\eta},\boldsymbol{\gamma}+t\boldsymbol{\eta})-v(\boldsymbol{x},\boldsymbol{\gamma})}{t}\\
\le\ &\frac{u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\eta})-u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\xi})+u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\xi})-u(\boldsymbol{y}^*,\boldsymbol{\gamma})}{t}\\
&+\frac{C(\boldsymbol{y}^*-t\boldsymbol{\xi}-\boldsymbol{x}-t\boldsymbol{z}+t\boldsymbol{\eta})-C(\boldsymbol{y}^*-\boldsymbol{x}-t\boldsymbol{z})+C(\boldsymbol{y}^*-\boldsymbol{x}-t\boldsymbol{z})-C(\boldsymbol{y}^*-\boldsymbol{x})}{t}\\
\le\ &\frac{u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\eta})-u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\xi})+u(\boldsymbol{y}^*-t\boldsymbol{\xi},\boldsymbol{\gamma}+t\boldsymbol{\xi})-u(\boldsymbol{y}^*,\boldsymbol{\gamma})}{t}-C(\boldsymbol{\xi}-\boldsymbol{\eta})+C(-\boldsymbol{z})
\end{aligned}$$

Adding and subtracting $u(\boldsymbol{y}^*,\boldsymbol{\gamma})$ in the numerator of the first term and then taking limits on both sides, we get

$$\begin{aligned}
v'(\boldsymbol{x},\gamma;\boldsymbol{z}-\boldsymbol{\eta},\boldsymbol{\eta}) &\le u'(\boldsymbol{y}^*,\gamma;-\boldsymbol{\xi},\boldsymbol{\eta})-u'(\boldsymbol{y}^*,\gamma;-\boldsymbol{\xi},\boldsymbol{\xi})+u'(\boldsymbol{y}^*,\gamma;-\boldsymbol{\xi},\boldsymbol{\xi})-C(\boldsymbol{\xi}-\boldsymbol{\eta})+C(-\boldsymbol{z})\\
&\le u'(\boldsymbol{y}^*,\gamma;0,\boldsymbol{\eta}-\boldsymbol{\xi})+u'(\boldsymbol{y}^*,\gamma;-\boldsymbol{\xi},\boldsymbol{\xi})-C(\boldsymbol{\xi}-\boldsymbol{\eta})+C(-\boldsymbol{z})\\
&\le \frac{\rho}{2}c_{\max}\sum_{i=1}^n|\xi_i-\eta_i|+\sum_{i=1}^n\beta_i\xi_i-\frac{c_{\min}}{2}\sum_{i=1}^n|\xi_i-\eta_i|+C(-\boldsymbol{z})\\
&\le \sum_{i=1}^n\beta_i\eta_i+\frac{\rho c_{\max}-c_{\min}}{2}\sum_{i=1}^n|\xi_i-\eta_i|+C(-\boldsymbol{z})\\
&\le \sum_{i=1}^n(\beta_i+\rho c_{\max}-c_{\min})\eta_i+C(-\boldsymbol{z}),
\end{aligned}$$

where the second inequality follows by Lemma EC.4 and the third inequality follows by Lemma EC.3.

Now we show equation (EC.12). Suppose $(\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta})$ is a feasible direction. Again, let $\boldsymbol{y}^*$ be an optimal solution to equation (16) at $(\boldsymbol{x},\boldsymbol{\gamma})$. Then $\boldsymbol{y}^*+t\boldsymbol{\eta}$ is clearly a feasible solution to equation (16) at $(\boldsymbol{x}+t\boldsymbol{z}+t\boldsymbol{\eta},\boldsymbol{\gamma}-t\boldsymbol{\eta})$ and thus we have

$$\begin{aligned}
v'(\boldsymbol{x},\gamma;\boldsymbol{z}+\boldsymbol{\eta},-\boldsymbol{\eta}) &= \lim_{t\to0}\frac{v(\boldsymbol{x}+t\boldsymbol{z}+t\boldsymbol{\eta},\boldsymbol{\gamma}-t\boldsymbol{\eta})-v(\boldsymbol{x},\boldsymbol{\gamma})}{t}\\
&\le \lim_{t\to0}\frac{u(\boldsymbol{y}^*+t\boldsymbol{\eta},\boldsymbol{\gamma}-t\boldsymbol{\eta})+C(\boldsymbol{y}^*-\boldsymbol{x}-t\boldsymbol{z})-u(\boldsymbol{y}^*,\boldsymbol{\gamma})-C(\boldsymbol{y}^*-\boldsymbol{x})}{t}\\
&\le u'(\boldsymbol{y}^*,\gamma;\boldsymbol{\eta},-\boldsymbol{\eta})+C(-\boldsymbol{z})\\
&\le \sum_{i=1}^n\beta_i\eta_i+C(-\boldsymbol{z}),
\end{aligned}$$

where the second inequality follows from the subadditivity and the postive homogeneity of $C(\cdot)$ and the last inequality follows from the assumption.

To show the result in (EC.11), suppose $(\boldsymbol{0},\boldsymbol{z})$ is a feasible direction at $(\boldsymbol{x},\boldsymbol{\gamma})$. Let $\boldsymbol{y}^*$ be an optimal solution to equation (16) at $(\boldsymbol{x},\boldsymbol{\gamma})$. Then $\boldsymbol{y}^*$ is a feasible solution for $(\boldsymbol{x},\boldsymbol{\gamma}+t\boldsymbol{z})$ and thus,

$$v'(\boldsymbol{x},\gamma;\boldsymbol{0},\boldsymbol{z}) = \lim_{t\to0}\frac{v(\boldsymbol{x},\boldsymbol{\gamma}+t\boldsymbol{z})-v(\boldsymbol{x},\boldsymbol{\gamma})}{t}$$

$$\le \lim_{t\to 0} \frac{u(\boldsymbol{y}^*, \boldsymbol{\gamma}+t\boldsymbol{z}) + C(\boldsymbol{x}-\boldsymbol{y}^*) - u(\boldsymbol{y}^*, \boldsymbol{\gamma}) - C(\boldsymbol{x}-\boldsymbol{y}^*)}{t}$$
$$= \lim_{t\to 0} \frac{u(\boldsymbol{y}^*, \boldsymbol{\gamma}+t\boldsymbol{z}) - u(\boldsymbol{y}^*, \boldsymbol{\gamma})}{t} = u'(\boldsymbol{y}^*, \boldsymbol{\gamma}; \boldsymbol{0}, \boldsymbol{z}) \le \frac{\rho c_{\max}}{2} \sum_{i=1}^n |z_i|,$$

which completes the proof.

The last piece of Theorem 1 is the Lipschitz continuity of $u_t(\cdot)$. In the next proposition, we show that item (b) and (c) imply item (d).

LEMMA EC.7. *Consider a bounded function $f : \Delta \to \mathbb{R}$ that is convex, continuous, and satisfies*

1. $|f'(\boldsymbol{y}, \boldsymbol{\gamma}; \mp\boldsymbol{\eta}, \pm\boldsymbol{\eta})| \le \sum_{i=1}^n \beta_i \eta_i$ *for all $(\boldsymbol{x}, \boldsymbol{\gamma}) \in \Delta$ and any feasible direction $(\mp\boldsymbol{\eta}, \pm\boldsymbol{\eta})$ with $\boldsymbol{\eta} \ge \boldsymbol{0}$;*
2. $f'(\boldsymbol{y}, \boldsymbol{\gamma}; \boldsymbol{0}, \boldsymbol{v}) \le (\rho c_{\max}/2) \sum_{i=1}^n |v_i|$ *for all $(\boldsymbol{x}, \boldsymbol{\gamma}) \in \Delta$ and any feasible direction $(\boldsymbol{0}, \boldsymbol{v})$ with $\boldsymbol{e}^T \boldsymbol{v} = 0$.*

Then, the function $f(\cdot)$ is Lipschitz continuous on Δ° with Lipschitz constant $(3/2)\sqrt{2n}\,\beta_i$.

Proof. Let $(\boldsymbol{y}, \boldsymbol{\gamma}) \in \Delta^\circ$. Let $\boldsymbol{\xi}^+ = \max(\boldsymbol{\xi}, 0)$ and $\boldsymbol{\xi}^- = \min(\boldsymbol{\xi}, 0)$, then $(\boldsymbol{\xi}, \boldsymbol{\eta}) = (\boldsymbol{\xi}^+, -\boldsymbol{\xi}^+) + (\boldsymbol{\xi}^-, -\boldsymbol{\xi}^-) + (0, \boldsymbol{\xi}^+ + \boldsymbol{\xi}^- + \boldsymbol{\eta})$ and it follows from Theorem 2 and Assumption 2 that

$$\begin{aligned}
|f'(\boldsymbol{y}, \boldsymbol{\gamma}; \boldsymbol{\xi}, \boldsymbol{\eta})| &\le |f'(\boldsymbol{y}, \boldsymbol{\gamma}; \boldsymbol{\xi}^+, -\boldsymbol{\xi}^+)| + |f'(\boldsymbol{y}, \boldsymbol{\gamma}; \boldsymbol{\xi}^-, -\boldsymbol{\xi}^-)| + |f'(\boldsymbol{y}, \boldsymbol{\gamma}; 0, \boldsymbol{\xi}^+ + \boldsymbol{\xi}^- + \boldsymbol{\eta})| \\
&\le \sum_{i=1}^n \beta_i |\boldsymbol{\xi}_i^+| + \sum_{i=1}^n \beta_i |\boldsymbol{\xi}_i^-| + (\rho c_{\max}/2)\, \|\boldsymbol{\xi}^+ + \boldsymbol{\xi}^- + \boldsymbol{\eta}\|_1 \\
&\le \beta_{\max} \|\boldsymbol{\xi}\|_1 + (\rho c_{\max}/2)\, \|\boldsymbol{\xi}^+ + \boldsymbol{\xi}^- + \boldsymbol{\eta}\|_1 \\
&\le \beta_{\max} \|\boldsymbol{\xi}\|_1 + (\beta_{\max}/2) \left(\|\boldsymbol{\xi}^+\|_1 + \|\boldsymbol{\xi}^-\|_1 + \|\boldsymbol{\eta}\|_1\right) \\
&\le (3/2)\, \beta_{\max} \|(\boldsymbol{\xi}, \boldsymbol{\eta})\|_1 \le (3/2)\sqrt{2n}\, \beta_{\max} \|(\boldsymbol{\xi}, \boldsymbol{\eta})\|_2.
\end{aligned}$$

With all the pieces ready, the proofs for Theorem 1 for $(u_t(\cdot))$ and $(v_t(\cdot))$ thus follow from Proposition EC.1, Proposition EC.2, Proposition EC.3, and the induction.

EC.1.4. Other Proofs

Proof of Theorem 3: Fix $\boldsymbol{\gamma} \in S$. Let $\boldsymbol{y}^*(\boldsymbol{x}, \boldsymbol{\gamma}) = \{\boldsymbol{y} \in \Delta_{n-1}(I) : u(\boldsymbol{x}, \boldsymbol{\gamma}) = C(\boldsymbol{y} - \boldsymbol{x}) + u(\boldsymbol{y}, \boldsymbol{\gamma})\}$ be the set of optimal solutions corresponding to the system state $\boldsymbol{x} \in S$. It is easy to verify that

$$\Omega_u(\boldsymbol{\gamma}) = \cup_{\boldsymbol{x} \in \Delta_{n-1}(I)} \boldsymbol{y}^*(\boldsymbol{x}, \boldsymbol{\gamma}). \tag{EC.14}$$

As $C(\cdot)$ and $u(\cdot)$ are continuous and $\Delta_{n-1}(I)$ is compact, by Berge's Maximum Theorem, $\boldsymbol{y}^*(\cdot)$ is a nonempty-valued and compact-valued upper hemicontinuous[2] correspondence. As $C(\cdot)$ and $u(\cdot)$

[2] Upper hemicontinuity can be defined as follows. Suppose X and Y are topological spaces. A correspondence $f : X \to \mathcal{P}(Y)$ (power set of Y) is upper hemicontinuous if for any open set V in Y, $f^{-1}(V) = \{x \in X | f(x) \subset V\}$ is open in X.

are also convex, $\boldsymbol{y}^*(\cdot)$ is also convex-valued. So, it is clear from (EC.14) that $\Omega_u(\boldsymbol{\gamma})$ is nonempty. To show $\Omega_u(\boldsymbol{\gamma})$ is compact, suppose $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ is a sequence in $\Omega_u(\boldsymbol{\gamma})$ such that $\boldsymbol{y}_n \in \boldsymbol{y}^*(\boldsymbol{x}_n, \boldsymbol{\gamma})$ for $n \in \mathbb{N}$ and $\boldsymbol{y}_n \to \boldsymbol{y}$. We need to show that $\boldsymbol{y} \in \Omega_u(\boldsymbol{\gamma})$. By passing through a subsequence, we may assume that $\boldsymbol{y}_{n_k} \in \boldsymbol{y}^*(\boldsymbol{x}_{n_k}, \boldsymbol{\gamma})$, $\boldsymbol{x}_{n_k} \to \boldsymbol{x}$ and $\boldsymbol{y}_{n_k} \to \boldsymbol{y}$. As $\boldsymbol{y}^*(\cdot)$ is compact-valued, by the Closed Graph Theorem, $\boldsymbol{y}^*(\cdot)$ has a closed graph. This implies that $\boldsymbol{y} \in \boldsymbol{y}^*(\boldsymbol{x}, \boldsymbol{\gamma}) \subset \Omega_u(\boldsymbol{\gamma})$, and therefore $\Omega_u(\boldsymbol{\gamma})$ is compact.

To show that $\Omega_u(\boldsymbol{\gamma})$ is connected, suppose the reverse is true. Then, there exist open sets V_1, V_2 in $\Delta_{n-1}(I)$ such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 \supset \Omega_u(\boldsymbol{\gamma})$, and $V_1 \cap \Omega_u(\boldsymbol{\gamma})$ and $V_2 \cap \Omega_u(\boldsymbol{\gamma})$ are nonempty. As $\boldsymbol{y}^*(\cdot)$ is convex-valued, this implies that, for any $\boldsymbol{x} \in \Delta_{n-1}(I)$, $\boldsymbol{y}^*(\boldsymbol{x}, \boldsymbol{\gamma})$ is either in V_1 or in V_2, but not both. Let $(U_1, \boldsymbol{\gamma}) = \boldsymbol{y}^{*-1}(V_1)$ and $(U_2, \boldsymbol{\gamma}) = \boldsymbol{y}^{*-1}(V_2)$. Then U_1, U_2 are open, $U_1 \cap U_2 = \emptyset$, $U_1 \cup U_2 \supset \Delta_{n-1}(I)$, and $U_1 \cap \Delta_{n-1}(I)$ and $U_2 \cap \Delta_{n-1}(I)$ are nonempty. This implies that the $(n-1)$-dimensional simplex $\Delta_{n-1}(I)$ is not connected. We have reached a contradiction. Therefore, $\Omega_u(\boldsymbol{\gamma})$ is also connected.

Next, to show that π^* is optimal, note that $\pi^*(\boldsymbol{x}, \boldsymbol{\gamma}) = \boldsymbol{x}$ for $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$ is clear from (17). If $\boldsymbol{x} \notin \Omega_u(\boldsymbol{\gamma})$, then, by (EC.14), $\pi^*(\boldsymbol{x}, \boldsymbol{\gamma}) \in \Omega_u(\boldsymbol{\gamma})$. Now, suppose there exists $\pi^*(\boldsymbol{x}, \boldsymbol{\gamma}) = \boldsymbol{y} \in \Omega_u(\boldsymbol{\gamma})^\circ$, then $\boldsymbol{y} + t(\boldsymbol{x} - \boldsymbol{y}) \in \Omega_u(\boldsymbol{\gamma})$ for small enough $t > 0$. Set $\boldsymbol{z} = \boldsymbol{y} + t(\boldsymbol{x} - \boldsymbol{y})$. Then $u(\boldsymbol{z}, \boldsymbol{\gamma}) + C(\boldsymbol{z} - \boldsymbol{x}) \le u(\boldsymbol{y}, \boldsymbol{\gamma}) + C(\boldsymbol{y} - \boldsymbol{z}) + C(\boldsymbol{z} - \boldsymbol{x}) = u(\boldsymbol{y}, \boldsymbol{\gamma}) + tC(\boldsymbol{y} - \boldsymbol{x}) + (1-t)C(\boldsymbol{y} - \boldsymbol{x}) = u(\boldsymbol{y}, \boldsymbol{\gamma}) + C(\boldsymbol{y} - \boldsymbol{x})$. So, $\boldsymbol{z}$ is as good a solution as $\boldsymbol{y}$. Therefore, there exists an optimal solution $\pi^*(\boldsymbol{x}, \boldsymbol{\gamma}) \in \mathcal{B}(\Omega_u(\boldsymbol{\gamma}))$ if $\boldsymbol{x} \notin \Omega_u(\boldsymbol{\gamma})$. □

Proof of Proposition 1: Suppose $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$. Take any feasible direction $(\boldsymbol{z}, \boldsymbol{0})$ at $(\boldsymbol{x}, \boldsymbol{\gamma})$. Then, by (17),

$$\frac{u(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma}) - u(\boldsymbol{x}, \boldsymbol{\gamma})}{t} \ge -C(\boldsymbol{z})$$

for $t > 0$. Taking the limit as $t \downarrow 0$, we have $u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0}) \ge -C(\boldsymbol{z})$. Conversely, suppose $u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0}) \ge -C(\boldsymbol{z})$ for any feasible direction $\boldsymbol{z}$ at $\boldsymbol{x}$ in $\mathcal{H}$. Let $\phi(t) = u(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma})$. Then, $\phi(\cdot)$ is convex, $\phi(0) = u(\boldsymbol{x})$, and $\phi'(0+) = u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0}) \ge -C(\boldsymbol{z})$. By the subgradient inequality, $t\phi'(0+) + \phi(0) \le \phi(t)$. This implies that $-tC(\boldsymbol{z}) + u(\boldsymbol{x}, \boldsymbol{\gamma}) \le u(\boldsymbol{x} + t\boldsymbol{z}, \boldsymbol{\gamma})$ is true for any feasible direction $(\boldsymbol{z}, \boldsymbol{0})$. Therefore, we have $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$. □

Proof of Proposition 2: For the "if" part, suppose $\boldsymbol{x} \notin \Omega_u(\boldsymbol{\gamma})$. Then, there exists $\boldsymbol{y} \in \Delta_{n-1}(I)$ such that $u(\boldsymbol{x}, \boldsymbol{\gamma}) > C(\boldsymbol{y} - \boldsymbol{x}) + u(\boldsymbol{y}, \boldsymbol{\gamma})$. Take any $\boldsymbol{g} \in \partial_x u(\boldsymbol{x}, \boldsymbol{\gamma})$. By the subgradient inequality, $u(\boldsymbol{x}, \boldsymbol{\gamma}) + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{x}) \le u(\boldsymbol{y}, \boldsymbol{\gamma})$. It follows that

$$C(\boldsymbol{y} - \boldsymbol{x}) < -\boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{x}).$$

Suppose $\boldsymbol{w} = (w_{ij})$ is an optimal solution to problem (1). Then $C(\boldsymbol{y}-\boldsymbol{x}) = \sum_i \sum_j c_{ij} w_{ij}$, and by Lemma EC.2, $-\boldsymbol{g}^T(\boldsymbol{y}-\boldsymbol{x}) = \sum_i g_i(y_i - x_i)^- - \sum_j g_j(y_j - x_j)^+ = \sum_i \sum_j (g_i - g_j) w_{ij}$. So, we have

$$\sum_i \sum_j c_{ij} z_{ij} < \sum_i \sum_j (g_i - g_j) w_{ij}.$$

Hence, there exists i and j such that $g_i - g_j > c_{ij}$. This implies $\boldsymbol{g} \notin \mathcal{G}$.

For the "only if" part, suppose $\boldsymbol{x} > \boldsymbol{0}$ and $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$. Assume $\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) \cap \mathcal{G} = \emptyset$. We will show that this leads to a contradiction. Let P be the orthogonal projection from $\mathbb{R}^n$ to the subspace $\mathcal{H} = \{\boldsymbol{x} \in \mathbb{R}^n : \sum_i x_i = 0\}$. Then

$$P(\boldsymbol{x}) = \boldsymbol{x} - \frac{\sum_i x_i}{n} \boldsymbol{e},$$

where $\boldsymbol{e} = (1, \ldots, 1)$ in $\mathbb{R}^n$. Noting that $\mathcal{G} + \alpha \boldsymbol{e} \subset \mathcal{G}$ for any $\alpha \in \mathbb{R}$, it is easy to verify that

$$\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) \cap \mathcal{G} = \emptyset \quad \text{if and only if} \quad \partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) \cap P(\mathcal{G}) = \emptyset,$$

since $\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) \subseteq \mathcal{H}$. As $\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma})$ is closed and $P(\mathcal{G})$ is compact, by Hahn-Banach Theorem, there exists $\boldsymbol{z} \in \mathcal{H}$, $a \in \mathbb{R}$ and $b \in \mathbb{R}$ such that

$$\langle \boldsymbol{g}, \boldsymbol{z} \rangle < a < b < \langle \boldsymbol{\lambda}, \boldsymbol{z} \rangle$$

for every $\boldsymbol{g} \in P(\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}))$ and for every $\boldsymbol{\lambda} \in P(\mathcal{G})$, or equivalently, as $\langle \boldsymbol{g}, \boldsymbol{z} \rangle = \langle P(\boldsymbol{g}), \boldsymbol{z} \rangle$ and $\langle \boldsymbol{\lambda}, \boldsymbol{z} \rangle = \langle P(\boldsymbol{\lambda}), \boldsymbol{z} \rangle$, for every $\boldsymbol{g} \in \partial_x u(\boldsymbol{x}, \boldsymbol{\gamma})$ and for every $\boldsymbol{\lambda} \in \mathcal{G}$. As $\boldsymbol{z}$ is a feasible direction in $\mathcal{H}$ at $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$, by Proposition 1, we have $u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0}) \geq -C(\boldsymbol{z})$. It follows that

$$\sup\{\langle \boldsymbol{g}, \boldsymbol{z} \rangle : \boldsymbol{g} \in \partial_x u(\boldsymbol{x}, \boldsymbol{\gamma})\} = u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0}) \geq -C(\boldsymbol{z}).$$

So, we have

$$-C(\boldsymbol{z}) \leq a < b < \langle \boldsymbol{\lambda}, \boldsymbol{z} \rangle$$

for every $\boldsymbol{\lambda} \in \mathcal{G}$. However, by the dual formulation (EC.1), there exists $\boldsymbol{\lambda} \in \{(y_1, y_2 \ldots, y_n) \,|\, y_j - y_i \leq c_{ij} \;\forall\, i, j\}$ such that $\langle \boldsymbol{\lambda}, \boldsymbol{z} \rangle = C(\boldsymbol{z})$, or equivalently, $\langle -\boldsymbol{\lambda}, \boldsymbol{z} \rangle = -C(\boldsymbol{z})$. Recognizing that $-\boldsymbol{\lambda} \in \mathcal{G}$ leads to the contradiction. Therefore, it follows that $\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) \cap \mathcal{G} \neq \emptyset$. □

Proof of Corollary 2: If $u(\cdot, \boldsymbol{\gamma})$ is differentiable at $\boldsymbol{x}$, then

$$\partial_x u(\boldsymbol{x}, \boldsymbol{\gamma}) = \left\{ \left(\frac{\partial u(\boldsymbol{x}, \boldsymbol{\gamma})}{\partial x_1}, \frac{\partial u(\boldsymbol{x}, \boldsymbol{\gamma})}{\partial x_2}, \ldots, \frac{\partial u(\boldsymbol{x}, \boldsymbol{\gamma})}{\partial x_n} \right) \right\}.$$

In this case, it is easy to see that (20) is simplified to (21). To show that $\boldsymbol{x} \in \Omega_u(\boldsymbol{\gamma})$ implies (21) for $\boldsymbol{x} \in \mathcal{B}(\Delta_{n-1}(I))$. Note that the equality $\sup\{\boldsymbol{g}^T \boldsymbol{z} : \boldsymbol{g} \in \partial_x u(\boldsymbol{x}, \boldsymbol{\gamma})\} = u'(\boldsymbol{x}, \boldsymbol{\gamma}; \boldsymbol{z}, \boldsymbol{0})$ now holds for $\boldsymbol{x} \in \mathcal{B}(\Delta_{n-1}(I))$. The rest of the proof is the same as Proposition 2. □

Proof of Theorem 2: To show that value function retains its structure in the infinite horizon setting, we invoke the general approach outlined in Porteus (1975) and Porteus (1982) which "iterates" the structural properties of the one-stage problem. Let $\mathcal{V}^*$ be the space of convex, continuous and bounded functions over Δ. Note that a one-step *structure preservation* property holds by Lemma EC.1, Lemma EC.2, and Lemma EC.3: combined, they say that if the next period value function is in $\mathcal{V}^*$, then the optimal value of the current period is also in $\mathcal{V}^*$. Furthermore, the set $\mathcal{V}^*$ with the sup-norm $\|\cdot\|_\infty$ is a complete metric space. These two observations allow us to apply Corollary 1 of Porteus (1975) and conclude that $v \in \mathcal{V}^*$ (the remaining assumptions needed to apply the result can be easily checked). The rest of the proof follows from Lemma EC.1, Lemma EC.2, Lemma EC.3, Theorem 3, Proposition 1, Proposition 2, and Corollary 2. $\square$

Proof of Lemma 1: If $\boldsymbol{x} \in D^k$, $\boldsymbol{a}_k \in \partial_x u_J(\boldsymbol{x},\boldsymbol{\gamma})$. Since $a_{ki} - a_{kj} \le c_{ij}$ for all i,j, according to Proposition 2, we have $\boldsymbol{x} \in \Omega_{u_J}(\boldsymbol{\zeta})$. For the second part, we first write down the primal formulation for problem (22):

$$\begin{aligned}\bar{v}_J(\boldsymbol{x},\boldsymbol{\zeta}) = \quad \min \quad & \boldsymbol{c}\cdot\boldsymbol{w} + \xi \\ \text{subject to} \quad & \sum_{i=1}^n w_{i,j} - \sum_{k=1}^n w_{j,k} = z_j - x_j \quad \forall \quad j = 1,2,\ldots,n \\ & \boldsymbol{w} \ge 0 \\ & \boldsymbol{e}^T\boldsymbol{z} = \boldsymbol{e}^T\boldsymbol{x} \\ & \xi \ge (\boldsymbol{z} - \boldsymbol{y}_k)^T\boldsymbol{a}_k + (\boldsymbol{\zeta} - \boldsymbol{\gamma}_k)^T\boldsymbol{b}_k + \boldsymbol{\iota}_k \quad \forall k = 1,2,\ldots,J \\ & \boldsymbol{z} \ge 0.\end{aligned}$$

Since $\boldsymbol{x} \in \Omega_{u_J}(\boldsymbol{\zeta})$, one optimal solution to the primal formulation is $\boldsymbol{w} = 0, \boldsymbol{z} = \boldsymbol{x}, \xi = (\boldsymbol{x} - \boldsymbol{y}_k)^T\boldsymbol{a}_k + (\boldsymbol{\gamma} - \boldsymbol{\gamma}_k)^T\boldsymbol{b}_k + \boldsymbol{\iota}_k$. The dual solution $\boldsymbol{\lambda} = \boldsymbol{a}_k, \mu_k = 1, \mu_l = 0, \forall l \ne k$ is clearly feasible. It also satisfies the complementary slackness condition. Therefore, the solution is optimal. $\square$

Proof of Theorem 4: Let us first introduce some notation. For any bounded function $f : \Delta \to \mathbb{R}$, we define the mapping $\mathcal{L}$ such that $\mathcal{L}f : \Delta \to \mathbb{R}$ is another bounded function such that

$$(\mathcal{L}f)(\boldsymbol{y},\boldsymbol{\gamma}) = l(\boldsymbol{y}) + \rho \int \min_{\boldsymbol{y}' \in \Delta_{n-1}(\boldsymbol{e}^T\boldsymbol{x}')} C(\boldsymbol{y}' - \boldsymbol{x}') + f(\boldsymbol{y}',\boldsymbol{\gamma}')\, d\mu \quad \forall\, (\boldsymbol{y},\boldsymbol{\gamma}) \in \Delta, \tag{EC.15}$$

where $\boldsymbol{x}' = \tau_x(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d},\boldsymbol{P})$ and $\boldsymbol{\gamma}' = \tau_\gamma(\boldsymbol{y},\boldsymbol{\gamma},\boldsymbol{d},\boldsymbol{P})$. Note that $\mathcal{L}$ is closely related to the standard Bellman operator associated with the MDP defined in (9); see, for example, Bertsekas and Tsitsiklis (1996). The difference from the standard definition is that $\mathcal{L}$ comes from the Bellman recursion for $u(\boldsymbol{y},\boldsymbol{\gamma})$ instead of $v(\boldsymbol{x},\boldsymbol{\gamma})$. With this in mind, we henceforth simply refer to $\mathcal{L}$ as the "Bellman operator" and note a few standard properties.

LEMMA EC.8. *The Bellman operator $\mathcal{L}$ has the following properties:*

1. *(Monotonicity) Given bounded $f_1, f_2 : \Delta \to \mathbb{R}$ with $f_1 \le f_2$, then $\mathcal{L}f_1 \le \mathcal{L}f_2$.*

2. *(Contraction) For any bounded $f_1, f_2 : \Delta \to \mathbb{R}$, it holds that $\|\mathcal{L}f_1 - \mathcal{L}f_2\|_\infty \le \rho \|f_1 - f_2\|_\infty$.*

3. *(Fixed Point) The optimal value function u is the unique fixed point of $\mathcal{L}$, i.e., $\mathcal{L}u = u$.*

4. *(Constant Shift) Let $\mathbf{1}$ be the constant one function, i.e., $\mathbf{1}(\cdot) = 1$, and let α be a scalar. For any bounded $f : \Delta \to \mathbb{R}$, it holds that $\mathcal{L}(f + \alpha\mathbf{1}) = \mathcal{L}f + \rho\alpha\mathbf{1}$.*

Proof. These basic properties for our Bellman operator $\mathcal{L}$ are well-known for the standard Bellman operator and can be proved in an analogous manner; see, for example, Puterman (1994) or Bertsekas and Tsitsiklis (1996). □

We now move on to the main proof. We want to show that for each $\epsilon > 0$, there exists an almost surely finite iteration index $J(\epsilon)$ such that for all $J \ge J(\epsilon)$, it holds that $\|u_J - u\|_\infty \le \epsilon$. Let $B_r(\boldsymbol{y}, \boldsymbol{\gamma})$ be a $(2n-1)$-dimensional ball centered at $(\boldsymbol{y}, \boldsymbol{\gamma}) \in \Delta^\circ$ with radius r. Consider some $\epsilon' > 0$ (to be specified later) and let $\mathcal{C}(\epsilon')$ be an ϵ'-covering of Δ, meaning that $\mathcal{C}(\epsilon')$ is a finite collection of points in Δ° (representing the centers of a finite collection of balls with radius ϵ') and $\Delta \subseteq \bigcup_{(\boldsymbol{y},\boldsymbol{\gamma}) \in \mathcal{C}(\epsilon')} B_{\epsilon'}(\boldsymbol{y}, \boldsymbol{\gamma})$. Let $(\tilde{\boldsymbol{y}}_1, \tilde{\boldsymbol{\gamma}}_1), (\tilde{\boldsymbol{y}}_2, \tilde{\boldsymbol{\gamma}}_2), \ldots$ denote the sequence of sample points visited by the algorithm (one per iteration). Thus, by Assumption 3, we have $\sum_J \mathbf{P}\{(\tilde{\boldsymbol{y}}_J, \tilde{\boldsymbol{\gamma}}_J) \in B_{\epsilon'}(\boldsymbol{y}, \boldsymbol{\gamma})\} = \infty$, and an application of the Borel-Cantelli lemma tells us that each ball $B_{\epsilon'}(\boldsymbol{y}, \boldsymbol{\gamma})$ associated with the covering is visited infinitely often with probability one. To reduce notation, we will often suppress $(\boldsymbol{y}, \boldsymbol{\gamma})$ and use $B_{\epsilon'}$ to denote a generic ball in the covering. Our proof follows three main ideas:

1. For any infinite trajectory of sampled states, we can split it into an infinite number of "phases" such that in each phase, every ball associated with the ϵ'-covering is visited at least once.

2. We can then construct an auxiliary "batch" algorithm whose iteration counter aligns with the sequence of phases from the previous step. This new algorithm is defined as another instance of Algorithm 1, where on any given iteration, we group all states visited in the corresponding phase of the main algorithm into a single batch and perform all updates at once. For clarity, we will refer to the main algorithm as the "asynchronous" version of the batch algorithm.

3. The auxiliary batch algorithm can be viewed as an approximate version of value iteration. Using the properties of $\mathcal{L}$, we can show that it converges to an approximation of u (with error depending on ϵ'). Finally, we conclude by arguing that the main algorithm does not deviate too far from the auxiliary version.

Let $J_0 = 0$ and for $K = 1, 2, \ldots$, define the random variable

$$J_{K+1} = \min\{J > J_K : \forall\,(\boldsymbol{y}, \boldsymbol{\gamma}) \in \mathcal{C}(\epsilon'),\ \exists\, J' \text{ s.t. } J_K < J' \le J,\ (\tilde{\boldsymbol{y}}_{J'}, \tilde{\boldsymbol{\gamma}}_{J'}) \in B_{\epsilon'}(\boldsymbol{y}, \boldsymbol{\gamma})\}$$

to be the first time after J_K such that every ball in the ϵ'-covering is visited at least once. Notably, J_1 is the first time that the entire covering is visited at least once. We denote the set of iterations

$$\mathcal{J}_K = \{J_{K-1} + 1, J_{K-1} + 2, \ldots, J_K\}$$

to be the "Kth phase" of the algorithm and let $\hat{\mathcal{S}}_K = \{(\tilde{\boldsymbol{y}}_J, \tilde{\boldsymbol{\gamma}}_J)\}_{J \in \mathcal{J}_K}$ be the set of states visited throughout the course of phase K.

We now describe "path-dependent" instances of Algorithm 1 to assist with the remaining analysis. To be precise with the definitions, let us consider a sample path ω. The auxiliary batch algorithm associated with ω is a new instance of Algorithm 1 that uses iteration counter K and generates hyperplanes at the set of states $\mathcal{S}_K = \hat{\mathcal{S}}_K(\omega)$ for all $K \geq 1$. The initial approximation is $\hat{u}_0 = u_0$ and the estimate after K batch updates is denoted $\hat{u}_K(\boldsymbol{y}, \boldsymbol{\gamma})(\omega) = \max_{i=1,\ldots,N_K} \hat{g}_i(\boldsymbol{y}, \boldsymbol{\gamma})(\omega)$. We are now interested in studying the stochastic process $\{\hat{u}_K(\boldsymbol{y}, \boldsymbol{\gamma})\}$.

Next, we observe that the hyperplanes generated at iteration $K+1$ of the batch algorithm are tangent to $\mathcal{L}\hat{u}_K$ at the points in $\mathcal{S}_{K+1}$. Let $\kappa = (3/2)\sqrt{2n}\,\beta$. Note that by repeatedly applying Lemma EC.7, Proposition EC.3, Proposition EC.1, and Proposition EC.2, and using $u_0 = 0$, we can argue that all (tangent) hyperplanes generated throughout the algorithm have directional derivatives bounded by κ. It follows that if $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{\gamma}})$ is a sample point in $\mathcal{S}_{K+1}$ that lies in a ball $B_{\epsilon'}$ and it generates a hyperplane $\hat{g}$, then the underestimation error within the ball is upper-bounded by $\max_{(\boldsymbol{y},\boldsymbol{\gamma}) \in B_{\epsilon'}} \big[(\mathcal{L}\hat{u}_K)(\boldsymbol{y}, \boldsymbol{\gamma}) - \hat{g}(\boldsymbol{y}, \boldsymbol{\gamma})\big] \leq 2\kappa\epsilon'$ (using the fact that there is zero estimation error at $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{\gamma}})$, the tangent point). Applying this across the ϵ'-covering, we have:

$$\mathcal{L}\hat{u}_K - (2\kappa\epsilon')\,\mathbf{1} \leq \max_{i=N_K+1,\ldots,N_{K+1}} \hat{g}_i \leq \max_{i=1,\ldots,N_{K+1}} \hat{g}_i = \hat{u}_{K+1}. \tag{EC.16}$$

Therefore, we have a form of approximate value iteration and can analyze it accordingly (see Bertsekas and Tsitsiklis (1996)). Utilizing the monotonicity and shift properties of Lemma EC.8, we apply $\mathcal{L}$ to both sides of (EC.16) for $K = 0$ to obtain

$$\mathcal{L}(\mathcal{L}\hat{u}_0 - 2\kappa\epsilon'\,\mathbf{1}) = \mathcal{L}^2\,\hat{u}_0 - \rho\,(2\kappa\epsilon')\,\mathbf{1} \leq \mathcal{L}\hat{u}_1.$$

Subtracting $(2\kappa\epsilon')\,\mathbf{1}$ from both sides and then applying (EC.16) for $K = 1$, we have

$$\mathcal{L}^2\hat{u}_0 - \rho\,(2\kappa\epsilon')\,\mathbf{1} - (2\kappa\epsilon')\,\mathbf{1} \leq \mathcal{L}\hat{u}_1 - (2\kappa\epsilon')\,\mathbf{1} \leq \hat{u}_2.$$

Iterating these steps, we see that $\mathcal{L}^K\hat{u}_0 - (2\kappa\epsilon')(1 + \rho + \cdots + \rho^{K-1})\,\mathbf{1} \leq \hat{u}_K$. Taking limits, using the convergence of the value iteration algorithm (see Puterman (1994)), and noting that $\hat{u}_K \leq u$ for all K, we arrive at

$$u(\boldsymbol{y}, \boldsymbol{\gamma}) - \frac{2\kappa\epsilon'}{1-\rho} \leq \lim_{K\to\infty} \hat{u}_K(\boldsymbol{y}, \boldsymbol{\gamma}) \leq u(\boldsymbol{y}, \boldsymbol{\gamma}), \quad \forall\,(\boldsymbol{y}, \boldsymbol{\gamma}) \in \Delta. \tag{EC.17}$$

Hence, we have shown that the auxiliary batch algorithm generates value function approximations that closely approximate u in the limit.

The final step is to relate the main asynchronous algorithm to the auxiliary batch version. We claim that the value function approximation $\hat{u}_K$ generated by the Kth phase, for $K \geq 1$, of the batch algorithm is within a certain error bound of the approximation from the asynchronous algorithm at J_K:

$$\hat{u}_K - (4\kappa\epsilon')(1+\rho+\cdots+\rho^{K-1})\,\mathbf{1} \le u_{J_K}. \tag{EC.18}$$

To prove (EC.18), let us consider the first phase, $K=1$. Recall that the two algorithms are initialized with identical approximations, so $\hat{u}_0 = u_0$. Since $\{u_J\}$ is a nondecreasing sequence of functions, we have $\hat{u}_0 \le u_J$ and $\mathcal{L}\hat{u}_0 \le \mathcal{L}u_J$ for any $J \in \mathcal{J}_1$ by the monotonicity property of Lemma EC.8. Also note that the auxiliary batch algorithm builds a uniform underestimate of $\mathcal{L}\hat{u}_0$ with points of tangency belonging to $\hat{\mathcal{S}}_1$, so we have $\hat{u}_1 \le \mathcal{L}\hat{u}_0 \le \mathcal{L}u_J$. The hyperplane g_{J+1} added in iteration $J+1 \in \mathcal{J}_1$ of the asynchronous algorithm is tangent to $\mathcal{L}u_J$ at $(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1})$, so it follows that

$$\hat{u}_1(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}) \le (\mathcal{L}u_J)(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}) = g_{J+1}(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}).$$

Suppose $(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1})$ is in a ball $B_{\epsilon'}$. Then g_{J+1} can fall below $\hat{u}_1$ by at most $\max_{(\boldsymbol{y},\boldsymbol{\gamma})\in B^{\epsilon'}} \big[\hat{u}_1(\boldsymbol{y},\boldsymbol{\gamma}) - g_{J+1}(\boldsymbol{y},\boldsymbol{\gamma})\big] \le 4\kappa\epsilon'$ within the ball. Since this holds for every hyperplane (and corresponding ball) added throughout the phase $K=1$ and noting that every point in Δ can be associated with some well-approximating hyperplane (due to the property that each phase contains at least one visit to every ball), we have

$$\hat{u}_1 - (4\kappa\epsilon')\,\mathbf{1} \le \max_{J=1,\ldots,J_1} g_J = u_{J_1}, \tag{EC.19}$$

which proves (EC.18) for $K=1$. Applying $\mathcal{L}$ to both sides of (EC.19), utilizing Lemma EC.8, noting that $\hat{u}_2$ underestimates $\mathcal{L}\hat{u}_1$, and applying the nondecreasing property of the sequence $\{u_J\}$, we have

$$\hat{u}_2 - \rho\,(4\kappa\epsilon')\,\mathbf{1} \le \mathcal{L}\hat{u}_1 - \rho\,(4\kappa\epsilon')\,\mathbf{1} \le \mathcal{L}u_{J_1} \le \mathcal{L}u_J, \quad \forall\, J \in \mathcal{J}_2.$$

By the same reasoning as above, for $J+1 \in \mathcal{J}_2$, it must hold that

$$\hat{u}_2(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}) - \rho\,(4\kappa\epsilon') \le (\mathcal{L}u_J)(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}) = g_{J+1}(\tilde{\boldsymbol{y}}_{J+1}, \tilde{\boldsymbol{\gamma}}_{J+1}),$$

and we obtain $\hat{u}_2 - \rho\,(4\kappa\epsilon')\,\mathbf{1} - (4\kappa\epsilon')\,\mathbf{1} \le u_{J_2}$, proving (EC.18) for $K=2$. We can iterate these steps to argue (EC.18) for any K. Taking limits (all subsequent limits exist due to the boundedness and monotonicity of the sequences), we get

$$\lim_{K\to\infty} \hat{u}_K(\boldsymbol{y},\boldsymbol{\gamma}) - \frac{4\kappa\epsilon'}{1-\rho} \le \lim_{K\to\infty} u_{J_K}(\boldsymbol{y},\boldsymbol{\gamma}) = \lim_{J\to\infty} u_J(\boldsymbol{y},\boldsymbol{\gamma}) \le u(\boldsymbol{y},\boldsymbol{\gamma}), \quad \forall\,(\boldsymbol{y},\boldsymbol{\gamma}) \in \Delta, \tag{EC.20}$$

where the equality follows by the fact $u_{J_K}(\boldsymbol{y},\boldsymbol{\gamma})$ is a subsequence of $u_J(\boldsymbol{y},\boldsymbol{\gamma})$. Combining (EC.17) and (EC.20), we obtain

$$u(\boldsymbol{y},\boldsymbol{\gamma}) - \frac{6\kappa\epsilon'}{1-\rho} \le \lim_{K\to\infty} \hat{u}_K(\boldsymbol{y},\boldsymbol{\gamma}) - \frac{4\kappa\epsilon'}{1-\rho} \le \lim_{J\to\infty} u_J(\boldsymbol{y},\boldsymbol{\gamma}) \le u(\boldsymbol{y},\boldsymbol{\gamma}), \quad \forall\,(\boldsymbol{y},\boldsymbol{\gamma}) \in \Delta,$$

showing that the asynchronous algorithm generates value function approximations that are arbitrarily close to u: if we set $\epsilon' = (1-\rho)\epsilon/(6\kappa)$, this implies the existence of $J(\epsilon)$ such that for $J \ge J(\epsilon)$, $\|u_J - u\|_\infty \le \epsilon$. $\square$

EC.1.5. Subroutine for Adding Cuts

In Table EC.2, we provide details regarding the subroutine for computing a new cut, which used in the R-ADP algorithm.

```
Parameters: $l, c_{ij}, \rho > 0$
Data: $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_M, \boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_M$
Input: initial approximation $u_J(\boldsymbol{y}, \boldsymbol{\gamma}) = \max_{k=1,\ldots,N_J} g_k(\boldsymbol{y}, \boldsymbol{\gamma})$, new sample point $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{\gamma}})$
Initialization: $\mathcal{K} = \emptyset$
for $k = 1, 2, \ldots, N_J$,
    if $a_{ki} - a_{kj} \le c_{ij}\ \forall i, j$
        Add $k$ into the set $\mathcal{K}$
    end if
end for
for $s = 1, 2, \ldots, M$,
    $\bar{\boldsymbol{x}}_s = (\tilde{\boldsymbol{y}} - \boldsymbol{d}_s)^+ + \boldsymbol{P}_s^T(\tilde{\boldsymbol{\gamma}} + \min(\tilde{\boldsymbol{y}}, \boldsymbol{d}_s))$,
    $\bar{\boldsymbol{\gamma}}_s = (\tilde{\boldsymbol{\gamma}} + \min(\tilde{\boldsymbol{y}}, \boldsymbol{d}_s)) \circ (\boldsymbol{e} - \boldsymbol{P}_s \boldsymbol{e})$,
    Let $\mathcal{D} = \arg\max_{k=1,\ldots,N_J}\ g_k(\bar{\boldsymbol{x}}_s, \bar{\boldsymbol{\gamma}}_s)$
    if $\mathcal{D} \cap \mathcal{K} = \emptyset$
        Solve $\bar{v}_J(\bar{\boldsymbol{x}}_s, \bar{\boldsymbol{\gamma}}_s)$ in Equation (23)
        Note the optimal solution as $\lambda_0^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ and $v^*$ as the objective value
    else
        Pick any $k \in \mathcal{D} \cap \mathcal{K}$
        $\lambda_0^* = 0, \boldsymbol{\lambda}^* = \boldsymbol{a}_k, \boldsymbol{\mu}^* = \boldsymbol{e}_k, v^* = (\bar{\boldsymbol{x}}_s - \boldsymbol{y}_k)^T \boldsymbol{a}_k + (\bar{\boldsymbol{\gamma}}_s - \boldsymbol{\gamma}_k)^T \boldsymbol{b}_k + \boldsymbol{\iota}_k$
    end if
    Set $\bar{c}_s = \beta_i \sum_{i=1}^n (d_{s,i} - \tilde{y}_i)^+ + \rho v^*, \bar{\boldsymbol{a}}_s = \lambda_0^* \boldsymbol{e} + \boldsymbol{\lambda}^*, \bar{\boldsymbol{b}}_s = \sum_{j=1}^{N_J} \mu_j^* \boldsymbol{b}_j$
    Set $\nabla^s_{\bar{x},y} = \mathrm{Diag}(1_{\tilde{\boldsymbol{y}} > \boldsymbol{d}_s}) + \boldsymbol{P}_s \circ (1_{\tilde{\boldsymbol{y}} \le \boldsymbol{d}_s} \boldsymbol{e}^T), \nabla^s_{\bar{x},\gamma} = \boldsymbol{P}_s$
    Set $\nabla^s_{\bar{\gamma},\gamma} = \mathrm{Diag}(\boldsymbol{e} - \boldsymbol{P}_s \boldsymbol{e}), \nabla^s_{\bar{\gamma},y} = \mathrm{Diag}\left((\boldsymbol{e} - \boldsymbol{P}_s \boldsymbol{e}) \circ 1_{\tilde{\boldsymbol{y}} \le \boldsymbol{d}_s}\right)$
end for
Set $\tilde{\boldsymbol{a}} = \frac{1}{M} \sum_{s=1}^M \left(-\beta_i \sum_{i=1}^n 1_{\tilde{y}_i \le d_{s,i}} \boldsymbol{e}_i + \rho \nabla^s_{\bar{x},y} \bar{\boldsymbol{a}}_s + \rho \nabla^s_{\bar{\gamma},y} \bar{\boldsymbol{b}}_s\right)$
Set $\tilde{\boldsymbol{b}} = \frac{1}{M} \sum_{s=1}^M \left(\rho \nabla^s_{\bar{x},\gamma} \bar{\boldsymbol{a}}_s + \rho \nabla^s_{\bar{\gamma},\gamma} \bar{\boldsymbol{b}}_s\right), \tilde{c} = \frac{1}{M} \sum_{s=1}^M \bar{c}_s$
Output: $\tilde{g}(\boldsymbol{y}, \boldsymbol{\gamma}) = \tilde{\boldsymbol{a}}^T (\boldsymbol{y} - \tilde{\boldsymbol{y}}) + \tilde{\boldsymbol{b}}^T (\gamma - \tilde{\gamma}) + \tilde{c}$
```

Table EC.2 Subroutine for Adding Cuts

EC.1.6. Some Practical Considerations in implementing R-ADP

There are two primary practical challenges that arise when implementing Algorithm 1: (1) the value function approximations are represented by an unbounded number of cuts and (2) the design of the state-sampling strategy, which becomes crucial in a high-dimensional state space.

EC.1.6.1. Removing Redundant Cuts If we keep adding cuts to the existing approximation, some cuts become dominated by others, i.e., there exists some $k \in \{1, 2, \ldots, J\}$ such that $g_k(\boldsymbol{y}, \boldsymbol{\gamma}) < \max_{j=1,\ldots,J} g_j(\boldsymbol{y}, \boldsymbol{\gamma})$ for all $(\boldsymbol{y}, \boldsymbol{\gamma}) \in \Delta$. It is important to remove these redundant cuts since they can lower the efficiency of solving optimization problem (23). Fortunately, the simple structure of the simplex enables us to check whether a piece is redundant efficiently and effectively.

We first show how to determine whether a cut is completely dominated by another cut over the simplex.

PROPOSITION EC.4. *$\boldsymbol{a}_1^T(\boldsymbol{y}-\boldsymbol{y}_1)+\boldsymbol{b}_1^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_1)+c_1 \geq \boldsymbol{a}_2^T(\boldsymbol{y}-\boldsymbol{y}_2)+\boldsymbol{b}_2^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_2)+c_2$ for all $(\boldsymbol{y},\boldsymbol{\gamma}) \in \Delta$ if and only if*

$$c_1-\boldsymbol{a}_1^T\boldsymbol{y}_1-\boldsymbol{b}_1^T\boldsymbol{\gamma}_1-c_2+\boldsymbol{a}_2^T\boldsymbol{y}_2+\boldsymbol{b}_2^T\boldsymbol{\gamma}_2+\min\{\min_i\{a_{1i}-a_{2i}\},\min_i\{b_{1i}-b_{2i}\}\} \geq 0.$$

Therefore, to check whether a cut is completely dominated by another cut, one just needs to perform a series of elementary operations and check one inequality, the computational effort of which is negligible compared to solving a linear program. Therefore, we can always check whether the current cut either dominates or is dominated by other cuts by checking at most $2N_J$ inequalities. Though Proposition EC.4 is simple to implement, it does not cover the situation where a cut is dominated by the maximum of several other cuts, which occurs more frequently. The next proposition addresses this situation.

PROPOSITION EC.5. *$D^k \neq \emptyset$ if and only if the objective function value of the following linear program*

$$\begin{array}{rc} \min & t \\ \textit{subject to} & \boldsymbol{e}^T\boldsymbol{y}+\boldsymbol{e}^T\boldsymbol{\gamma}=N, \\ & t \geq \boldsymbol{a}_l^T(\boldsymbol{y}-\boldsymbol{y}_l)+\boldsymbol{b}_l^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_l)+c_l-\boldsymbol{a}_k^T(\boldsymbol{y}-\boldsymbol{y}_k)-\boldsymbol{b}_k^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_k)-\boldsymbol{\iota}_k,\ \forall\ l \neq k \\ & \boldsymbol{y},\boldsymbol{\gamma} \geq 0, \end{array} \tag{EC.21}$$

is negative.

Proof. Cut k having a dominating region in Δ means that

$$\min_{(\boldsymbol{y},\boldsymbol{\gamma})\in\Delta} \max_{l\neq k}\ \boldsymbol{a}_l^T(\boldsymbol{y}-\boldsymbol{y}_l)+\boldsymbol{b}_l^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_l)+c_l-\boldsymbol{a}_k^T(\boldsymbol{y}-\boldsymbol{y}_k)-\boldsymbol{b}_k^T(\boldsymbol{\gamma}-\boldsymbol{\gamma}_k)-\boldsymbol{\iota}_k < 0.$$

Introducing the dummy variable t to denote the maximum, we obtain the formulation (EC.21). □

By solving the linear program (EC.21) at most N_J times, one can remove all of the redundant pieces. One can perform this redundant check periodically. Our numerical implementation also employs the following strategy. We track the iterations at which a cut dominates when identifying $\mathcal{D}$. If a cut does not become dominating for a number of iterations greater than some threshold, we consider it potentially redundant and check for redundancy by solving (EC.21) once. Despite these attempts to reduce the number of cuts, problem instances with more locations naturally require more cuts to accurately represent the value function $u(\cdot)$. To control the computation required for solving linear programs, we also make the practical recommendation to set an absolute upper bound on the number of cuts used in the approximation, with older cuts dropped as needed.

EC.1.6.2. Sampling Distribution We also propose a more effective method of sampling states for the ADP algorithm beyond the naive choice of a uniform distribution over Δ. Our tests indicate that, especially when the number of locations is large, uniform sampling is unable to prioritize the sampling in important regions of the state space (for example, states with large $\boldsymbol{\gamma}$ are unlikely to occur in problem instances where the return probability is high). A reasonable strategy is to periodically simulate the ADP policy (i.e., the one implied by the current value function approximation) and collect the set of states visited under this policy — termed a *replay buffer*. On future iterations, we can sample a portion of states at which to compute cuts directly from the replay buffer. This idea is based on the notion of *experience replay* within the reinforcement learning literature (see Lin (1992) and also Mnih et al. (2015) for a recent application).

EC.1.7. Comparative Statics

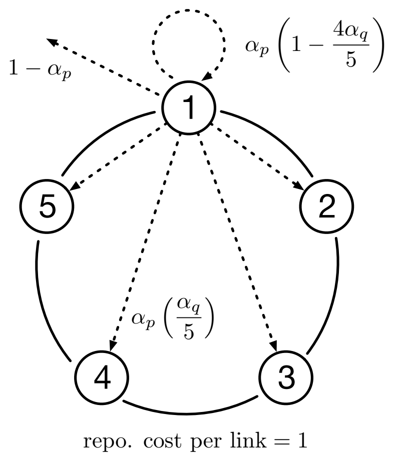

$1-\alpha_p$
$\alpha_p\left(1-\frac{4\alpha_q}{5}\right)$
1
2
3
4
5
$\alpha_p\left(\frac{\alpha_q}{5}\right)$
repo. cost per link = 1

Figure EC.1 Network used in Section EC.1.7

Parameter Name	Value	Range
n, number of locations	5	–
N, total inventory	1.0	–
ρ, discount factor	0.95	–
β_i, lost sales cost	4.5	–
c_{ij}, repositioning cost	$\{1,2\}$	–
α_ν, demand mean	0.3	$[0.1, 0.5]$
α_σ, demand volatility	1.0	$[0.5, 1.5]$
α_p, return fraction	0.75	$[0.4, 1.0]$
α_q, return uniformity	0.5	$[0.0, 1.0]$

Table EC.3 Parameter Values used in Section EC.1.7

In this section, we aim to compare the R-ADP, myopic, and no-repositioning policies across a range of parameter settings, with the goal of studying the impacts of (1) total demand, (2) demand volatility, (3) return fraction (i.e., fraction of vehicles returned per period), and (4) uniformity of return locations. We again use $N=1$ and set $\rho=0.95$. Due to the large number of MDP instances that we need to solve, we only consider $n=5$ locations, creating a set of 9-dimensional MDPs. Let $\tilde{\nu}_i=0.2$ for $i=1,2,\ldots,5$ and we set the mean demand at each location to be $\nu_i=\alpha_\nu\,\tilde{\nu}_i$ for some scaling parameter α_ν, so that $\sum_i \nu_i=\alpha_\nu$. Similar to before, we set $\sigma_i=\alpha_\sigma\,\nu_i$ for another scaling parameter α_σ. The repositioning costs c_{ij} are illustrated in Figure EC.1: the cost between adjacent locations is 1 and the cost between non-adjacent locations (e.g., 1 and 3 or 5 and 2) is 2. The return behavior is controlled by two parameters α_p, the fraction of rented vehicles returned (thus, $1-\alpha_p$ is the fraction of rented vehicles that remain rented), and α_q, which we interpret as the "return uniformity." For $\alpha_q = 1$, returns are split evenly between the 5 locations and for $\alpha_q = 0$, vehicles are returned to their rental origins. These two parameters are also illustrated in Figure EC.1 for location 1. We vary each of the scaling parameters α_ν, α_σ, α_p, and α_q individually; their nominal values and test ranges are summarized in Table EC.3. In all solved instances, we use a maximum of 1,000 cuts in the approximation while the ADP algorithm is run for 10,000 iterations[3]; all other algorithmic settings are as described in Section 6.3. The results are given in Figure EC.2. In each plot, lines illustrate absolute cost values of each policy for each parameter setting (x-axis), while the bars illustrate the percentage decrease in cost achieved by the ADP policy compared to the myopic and no-repositioning policies.

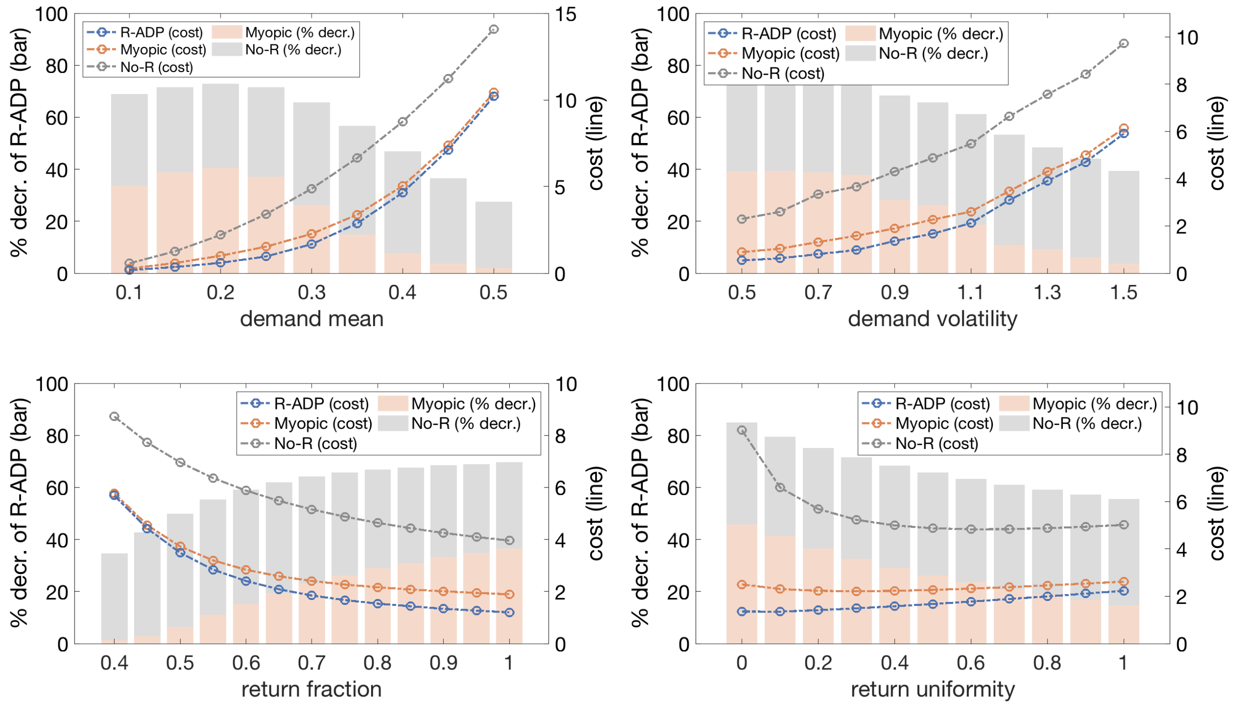


Figure EC.2 **Impact of Parameters (Left-Axis & Bar: % Improvement of ADP; Right-Axis & Line: Raw Cost)**

There are a few key takeaways from these experiments, which we now summarize. We see that when the mean demand α_ν in the system is high (greater than 45% of the total inventory), the performance of the myopic policy essentially matches that of the ADP policy. This can be explained by the observation that lost sales in high demand systems are somewhat inevitable, so the impact of considering return behavior and future cost is diminished. On the other hand, when demand is between 10% and 40% of the total inventory, a substantial improvement of between 7%–40% beyond the myopic performance is observed. The largest improvement, 40%, is seen for $\alpha_\nu = 0.2$.

[3] As shown in the previous section, 10,000 iterations is enough to produce a near-optimal policy for $n = 5$.

Demand volatility α_σ has a strong influence on cost for all three policies, with the cost of the ADP policy ranging from 0.54 for $\alpha_\sigma = 0.5$ and growing tenfold to 5.90 for $\alpha_\sigma = 1.5$. We also observe that the gap between the ADP and myopic policies shrinks for higher volatility systems.

The latter two plots are related to the return behavior parameters α_p and α_q. Although the cost decreases when the return fraction α_p increases (as there is more inventory with which to satisfy demand), the improvement upon the myopic policy increases. Intuitively, given more available inventory due to fewer ongoing rentals, the ADP policy has more "opportunities" to reposition and plan for future periods. We also see that return uniformity α_q tends to increase the cost under the ADP and myopic policies, but interestingly, if no-repositioning is used, the cost is actually reduced as α_q increases in the range of $[0, 0.6]$. This can perhaps be explained by the "natural repositioning" induced by the return behaviors, an effect that disappears when active repositioning (i.e., ADP or myopic) is allowed. Similar to the case of demand volatility, the gap between the ADP and myopic policy decreases as uniformity increases.

EC.1.8. Details of the Clustering Approach used in the CR-ADP Algorithm

Details of the clustering approach used in Section 7.1 are provided below.

- For simplicity, suppose that n is divisible by $\tilde{n}$. We take a geographical clustering approach[4] and group adjacent locations by setting $\mathcal{C}_k = \{(k-1)\,n/\tilde{n} + 1, \ldots, kn/\tilde{n}\}$.
- We transform the demand by summing within clusters. For cluster k, we set $\tilde{d}_{t,k} = \sum_{i \in \mathcal{C}_k} d_{t,i}$. Moreover, we define "demand weights" χ_{ik} that conceptually represent the weight of a particular location in a cluster of locations. For each $i \in \mathcal{C}_k$, let $\chi_{ik} = \mathbf{E}\, d_{t,i} / \sum_{j \in \mathcal{C}_k} \mathbf{E}\, d_{t,j}$.
- We keep the lost sales cost unchanged: $\tilde{\beta}_i = \beta_i$. For the repositioning cost from cluster k to cluster q, we take an averaging approach:

$$\tilde{c}_{kq} = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \frac{1}{|\mathcal{C}_q|} \sum_{j \in \mathcal{C}_q} c_{ij}$$

- The transformed return fraction from cluster k to cluster q is given by:

$$\tilde{p}_{t,kq} = \sum_{i \in \mathcal{C}_k} \chi_{ik} \sum_{j \in \mathcal{C}_q} p_{t,ij},$$

where we first sum over $\mathcal{C}_q$ and then take a weighted average over $\mathcal{C}_k$ using the weights χ_{ik} defined above.

- Finally, for a location i in cluster k, we define our splitting heuristic to be: $y_i = \chi_{ik}\, \tilde{y}_k$.

[4] Another conceivable approach is to group locations by demand (i.e., group many small locations together).

EC.1.9. An Illustration using Car2Go Data

In this section, we illustrate an application of the CR-ADP approach on a real world system, based on the same Car2Go dataset used in Appendix EC.1.1. We use the same technique described in Appendix EC.1.1 to deconstruct the GPS location time series data into individual trips that start and end within the bounds of [-122.7, -122.6] longitude and [45.5, 45.58] latitude. Since Car2Go is a free-floating service, we need to define discrete regions in order to apply our repositioning model and policy: to do so, we used k-means on a dataset of 64,884 GPS locations that represent either trip start or end locations to obtain $n = 200$ regions. Figure EC.3 shows the regions (in orange) along with the underlying GPS data (in blue).

To estimate $\boldsymbol{P}_t$, we first compute the fraction of trips started from i that end at j for all (i, j). Next, we estimate the return fraction p_t by computing the fraction of trips that end within an hour from each region, and then average over regions to arrive at $p_t \approx 0.712$. Finally, we apply a Gaussian filter with $\sigma = 5.0$ on the resulting matrix to construct a smoother estimate. Figure EC.4 illustrates the estimated $\boldsymbol{P}_t$ matrix using a heatmap. Examining the heatmap, we can see a prevalence of two-way trips (the bright diagonal) and one-way trips to downtown Portland (the bright vertical strip around region 50 as the destination). The mean demand at each location is simply taken to be proportional to the number of trips that started in that region; see Figure EC.5.

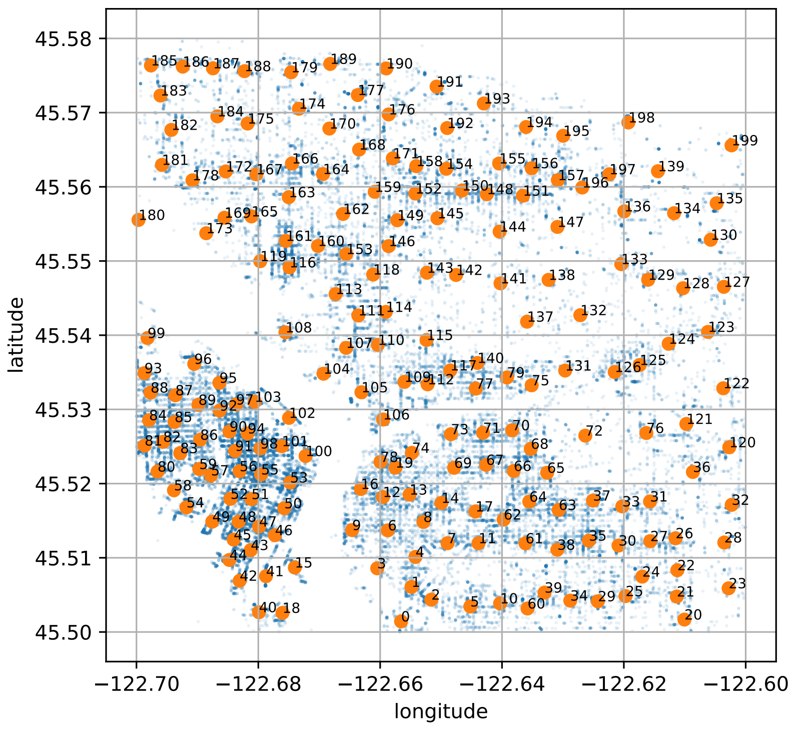


Figure EC.3 200 Regions (k-Means) Obtained from Car2Go GPS Data

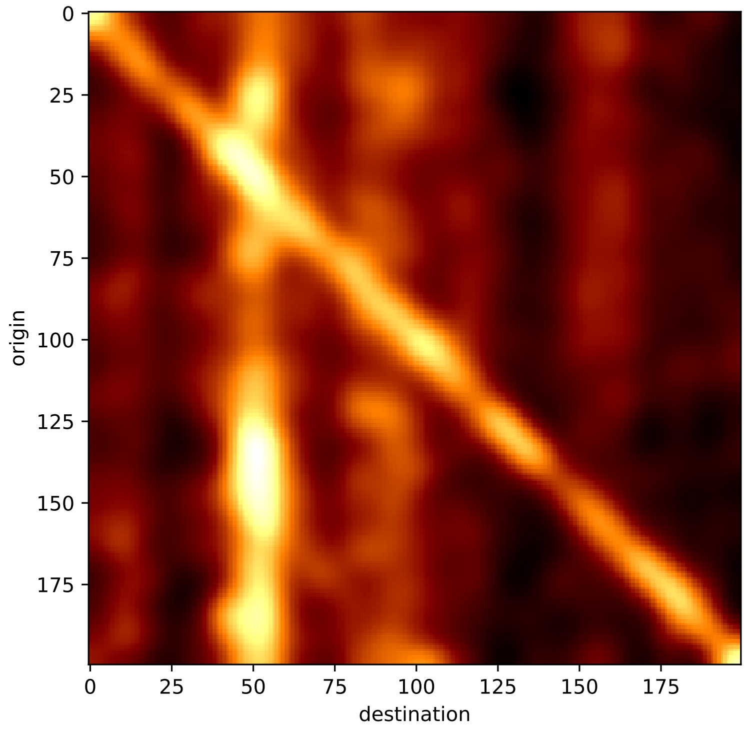


Figure EC.4 Heatmap of Estimated Transition P_t

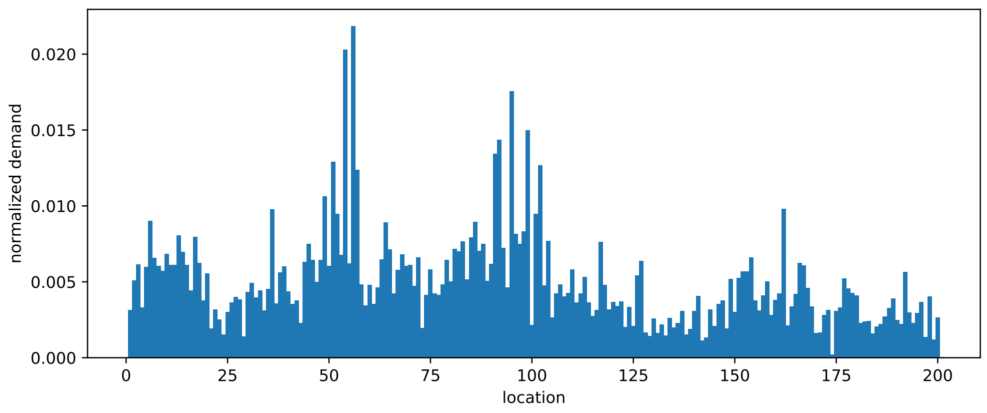


Figure EC.5 Normalized Demand Means

$\beta_i \mid \alpha_\nu$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1	1.2	1.3
2	21.9%	36.2%	34.8%	54.0%	59.2%	67.0%	74.2%	79.3%	83.2%	86.1%	88.4%
4	18.4%	26.1%	32.7%	46.2%	62.9%	66.7%	73.7%	79.1%	83.0%	86.0%	88.2%
6	32.8%	28.7%	31.7%	47.2%	60.5%	67.4%	73.5%	79.0%	82.9%	85.9%	88.2%
8	17.9%	21.8%	33.0%	46.2%	66.2%	71.7%	73.7%	78.8%	82.8%	85.9%	88.2%

Table EC.4 Performance of CR-ADP as Percentage of the No-Repositioning Policy's Cost (Lower is Better)

We let the lost sales cost $\beta_i \in \{2,4,6,8\}$ and the sum of mean demand $\alpha_\nu \in \{0.3, 0.4, \ldots, 1.3\}$ (total inventory is normalized to be 1.0). The repositioning costs are taken to be proportional to the Euclidean distance between regions, ranging between 0 and 1. The discount factor is assumed to be $\rho = 0.99$. Our CR-ADP policy reduces the problem of 200 locations to a system with $\tilde{n} = 10$ clusters of equal size, with cluster k containing regions $20(k-1)+1$ to $20k$, as labeled in Figure EC.3 (these are roughly grouped by geographical location). R-ADP is run on the auxiliary MDP for 20,000 iterations and a policy for the $n = 200$ location case is then defined via the approach described in Section EC.1.8. The results, relative to the cost of the no-repositioning policy[5], are shown in Table EC.4. We can see that CR-ADP produces significantly lower costs in all instances. The largest relative improvements compared to no-repositioning are when the total demand is low, consistent with what is shown in Figure EC.2.

References

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-Dynamic Programming* (Belmont, MA: Athena Scientific).

Boyd S, Boyd SP, Vandenberghe L (2004) *Convex optimization* (Cambridge university press).

Lin LJ (1992) Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8(3-4):293–321.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Porteus E (1982) Conditions for characterizing the structure of optimal strategies in infinite-horizon dynamic programs. *Journal of Optimization Theory and Applications* 36(3):419–432.

Porteus EL (1975) On the optimality of structured policies in countable stage decision processes. *Management Science* 22(2):148–157.

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (New York, NY, USA: John Wiley & Sons, Inc.), 1st edition.

Rockafellar RT (1970) *Convex Analysis* (Princeton, NJ, USA: Princeton University Press), 1st edition.

[5] The other policies are omitted due to the high computational cost of running them for $n = 200$.