# Dual-based methods for solving infinite-horizon nonstationary deterministic dynamic programs

Christopher Thomas Ryan[*] Robert L Smith[†]

June 25, 2018

**Abstract**

We develop novel dual-ascent and primal-dual methods to solve infinite-horizon nonstationary deterministic dynamic programs. These methods are finitely implementable and converge in value to optimality. Moreover, the dual-ascent method produces a sequence of improving dual solutions that pointwise converge to an optimal dual solution, while the primal-dual algorithm provides a sequence of primal basic feasible solutions with value error bounds from optimality that converge to zero. Our dual-based methods work on a more general class of infinite network flow problems that include the shortest-path formulation of dynamic programs as a special case. To our knowledge, these are the first dual-based methods proposed in the literature to solve infinite-horizon nonstationary deterministic dynamic programs.

## 1 Introduction

Industry is often accused of being short-sighted in its planning, obsessed with next quarter's projected profit margin. Many planning tools themselves exacerbate this focus by only considering decisions over a fixed finite horizon. Tools that incorporate an infinite horizon typically make stationarity assumptions that restrict the future to be exactly the world we confront today. Only relatively few attempts have been made at tackling infinite-horizon and nonstationary environments. Examples of this nonstationary setting in practice include the sizing and timing of capacity expansions, planning for production scheduling, the replacement and acquisition of new equipment, optimal search, dynamic traffic assignment, among others. Many of these problems can be modeled as infinite-horizon nonstationary dynamic programs (or simply DPs), the context of this paper.

How do we deal with a problem that is nonstationary and non-ending? Tackling this challenge invites many unsavory characters into our midst: an infinite amount of data, pathologies of infinite-dimensional spaces, challenges in expressing solutions finitely, etc. Traditionally, this problem has

[*]Booth School of Business, University of Chicago, E-mail: chris.ryan@chicagobooth.edu
[†]Industrial and Operations Engineering, University of Michigan, E-mail: rlsmith@umich.edu

been tackled with one of two approaches. The first approach, typically called a planning horizon approach, is to (if possible) find a finite horizon $T$ sufficiently distant to make the associated errors from the infinite setting negligible (Bean and Smith, 1984; Bès and Sethi, 1988; Charnes et al., 1966; Modigliani and Hohn, 1955). A key insight here is that although decisions made near $T$ can be distorted by end-of-horizon effects, the near term (in particular, the first decision) is less affected. This decoupling of the present and future decisions can be achieved by discounting (Bean and Smith, 1984), uncertainty Alden and Smith (1992); Hopp et al. (1987), or through tie-breaking selection (Ryan et al., 1992; Schochetman and Smith, 1991). Luckily, the current decision is the only one the decision maker must commit to at the time of implementation. If calculations are efficient, a rolling horizon approach is feasible.

The second approach, typically called the strategy horizon approach, searches for optimal solutions within the space of infinite horizon decision sequences, or strategies. Most success in this direction is for stationary problems, where the model is simplified to allow for analytic, and hence finite, solutions. An example is solving infinite-horizon homogeneous Markov decision processes that can restrict search within stationary strategies (Denardo, 2003; Puterman, 2014; Ross, 1996). The difficulty here is that the necessary model simplifications can exclude many realistic problems.

Since nonstationary data is sometimes unavoidable, other methods have been undertaken in the strategy-horizon context. Most notable are solving DPs as infinite-dimensional linear programs. Simplex methods and duality theory are then leveraged to establish properties of optimal strategies without resort to finite truncations of the horizon. Ghate and Smith (2013) and Lee et al. (2017) develop simplex methods to solve non-homogeneous versions of Markov decision processes (MDPs) based on their formulation as countably-infinite linear programs (CILPs). We mention here in passing settings with "continuous" state spaces, that are less related to the approach adopted here because of their stationarity assumptions. Several authors have adapted Anderson and Nash's Anderson and Nash (1987) general framework for infinite-dimensional linear programs for Markov decision processes to solve uncountable state-space stationary Markov and semi-Markov decision problems using linear programming formulations of Bellman equations. See for instance, (Feinberg and Shwartz, 2002; Hernández-Lerma and Lasserre, 2012; Klabjan and Adelman, 2006, 2007).

These simplex methods for MDPs rely on the condition that all extreme points are *nondegenerate*, a crucial property for showing asymptotic convergence. By contrast, a deterministic DP corresponds to a CILP that is highly *degenerate*. To the authors' knowledge, only three papers study deterministic DP via a strategy horizon approach using CILP formulations. Ghate, Sharma and Smith Ghate et al. (2010) develop a simplex-like method they call the *shadow simplex method* for a more general class of CILPs that have deterministic DP as a special case (see their Section 4). Sharkey and Romeijn (2008) propose a simplex method for capacitated infinite network flow problems that can model deterministic dynamic programming as a shortest path problem (following the construction we set out in Section 2 below). Ryan et al. (2018) propose a model for uncapacitated network flow problems and model deterministic DP as an all-to-infinity shortest-path problem in their Section 5.2.

However, the approaches derived from each of these three papers have their own limitations. All three can be shown to converge in optimal value, meaning iterates of their respective simplex methods converge in value to optimality. However, all three fail to guarantee solution convergence, where successive policy iterates converge to an optimal solution. Such convergence is only guaranteed when there is a unique optimal solution to the problem. Moreover, the methods of (Sharkey and Romeijn, 2008) and (Ryan et al., 2018) may not even be finitely-implementable, meaning each iteration of the algorithm may run in infinite time in the worst case.

In this paper, we also build on the "infinite network" view of dynamic programming, but instead of constructing a primal simplex method, as in (Sharkey and Romeijn, 2008) and (Ryan et al., 2018), we generalize dual-ascent and primal-dual methods for finite network flow problems (see, for instance, Chapter 9 of (Bertsimas and Tsitsiklis, 1997)) to an infinite setting. This setting captures deterministic DP as a special case. Unlike the primal methods discussed in the previous paragraph, we show that the dual-ascent method is finitely-implementable and has guaranteed pointwise solution convergence. The primal-dual method adapts the dual-ascent method to concurrently generate primal feasible solutions in the form of spanning trees. An optimality error bound on the gap between objective value of these primal feasible solutions and the optimal value is provided, based on the current dual solution. This optimality guarantee is refined as the primal-dual method proceeds and the gap disappears in the limit. Both the dual-ascent method and primal-dual method leverage the special structure of the infinite graphs we study and cannot easily be seen as a straightforward extension of the corresponding methods in the finite-network case.

To our knowledge, this paper is the first to develop dual-based algorithms for infinite network flow problems. As mentioned above, previous work has focused on primal simplex methods (see also earlier work in (Anderson and Philpott, 1989) on a primal simplex method for continuous time network flow problems). There has been a recent surge of work on infinite network flow problems and their generalizations (see (Nourollahi and Ghate, 2017) and (Ghate, 2015), in addition to (Ryan et al., 2018)) but these works either focus on weak and strong duality properties or primal simplex methods. It is well known that duality theory for infinite-dimensional optimization problems is problematic. Weak duality and complementary slackness may not hold Romeijn et al. (1992), strong duality may fail even when weak duality holds (Anderson and Nash, 1987), the primal may not have any extreme points even when each variable is bounded (Anderson and Nash, 1987), extreme points may not be characterizable as basic feasible solutions (Ghate and Smith, 2009), and a sequence of basic feasible solutions with strictly improving objective value may not pointwise converge to an optimal solution (Ghate et al., 2010). Starting in Ryan et al. (2018) and continuing here, all these issues are resolved for the class of *pure supply* infinite-network flow problems, which includes infinite-horizon nonstationary deterministic dynamic programming as a special case. Moreover, we establish strong duality using dual-based arguments, a departure from previous approaches.

This paper is organized as follows. In Section 2, we introduce the dynamic program we study and formulate it as an infinite network flow problem. In Section 3 the more general class of pure-supply network flow problems is introduced. Section 4 describes our dual ascent method. Section 5 describes our primal-dual method which builds on the primal method of Ryan et al. (2018) and the dual ascent method of the previous section. Section 6 concludes.

## 2 Infinite-horizon nonstationary dynamic programming

We consider a nonstationary infinite-horizon deterministic dynamic programming problem (or simply a DP) with finite per period state and action spaces, defined as follows. A system evolves over time periods $t = 0, 1, 2, \ldots$ in one of finitely many states $s_t \in S_t$ in each period. The starting state $s_0$ is arbitrarily chosen from the set $S_0$. An action $a_t$ is chosen from a finite and nonempty set $A_{s,t}$ that depends on each state $s \in S_t$ and time $t$. The system transitions to a new state according to the law $s' = \tau_t(s_t, a_t) \in S_{t+1}$, yielding a nonnegative immediate cost $c_t(s_t, a_t)$. If there exists a state $s' \in S_{t+1}$ that is not reachable from any state in $S_t$ (that is, if there does not exist an $s \in S_t$ and $a \in A_{s,t}$ with $s' = \tau_t(s,t)$) then we remove state $s'$ from $S_{t+1}$ since it is redundant.

There is a time-discounting factor $\delta \in (0,1)$ associated with costs. The discounted cost at time $t$ is $\delta^t c_t(s,a)$. A policy $d = \{d(\cdot,t) : t = 0,1,2,\ldots\}$ prescribes an action $d(s,t) \in A_{s,t}$ for every $s \in S_t$ and every $t$. Let $D$ denote the set of all policies from starting state $s_0$. The strategy $\pi(d)$ corresponding to policy $d$ is the sequence of actions $\pi_t(d)$ out of initial state $s_0$ provided by policy $d$ where $\pi_t(d) = d(s_t(d), t)$ and $s_{t+1}(d) = \tau_t(s_t(d), d(s_t(d), t)$ for $t = 0, 1, 2, \ldots$. Then $V(\pi(d), s_0) := \sum_{t=0}^{\infty} \delta^t c_t(s_t(d), \pi_t(d))$ is the cost of policy $d$ (and its corresponding strategy $\pi(d)$) with starting state $s_0$.

We assume some additional structure as follows:

(S1) immediate costs are uniformly bounded by $\bar{c} < \infty$; that is, $c_t(s,a) \le \bar{c}$ for all $s, a$.

(S2) there exists a sub-exponential function $g(t)$ such that the cardinality of the set of states $S_t$ at time $t$ is bounded by $g(t)$; that is, $\#(S_t) \le g(t)$ for all $t = 0, 1, \ldots$.

The focus of this paper is how to determine a policy that minimizes total discounted cost. That is, we solve

$$V^* := \min_{d \in D} V(\pi(d), s_0), \tag{1}$$

where $s_0$ is a fixed starting state. We consider a non-stationary version of the problem, where state sets $S_t$, available actions $A_{s,t}$, immediate costs $c_t$, and transition laws $\tau_t$ can all depend on $t$. Our approach to solving this problem is to formulate it as an minimum cost network flow problem on a network with countably-many nodes (see Section 3 below).

We define the network $N = (\mathcal{N}, \mathcal{A}, b, c)$ as follows. The set $\mathcal{N}$ consists of nodes for each state-time pair $(s,t)$ where $t = 0, 1, 2 \ldots$ and $s \in S_t$. The set $\mathcal{A}$ consists of arcs $((s,t),(s',t+1))$ between pairs of nodes for which there exists $a \in A_{s,t}$ such that $\tau_t(s,a) = s'$. Node $(s_0, 0)$ is endowed with a unit supply of 1 and all other nodes have a supply of 0, which defines the vector of supplies $b$. Each arc has an associated cost $c_{((s,t),(s',t+1))} = c_t(s,a)$, which defines arc costs $c$. Arcs are uncapacitated.

Using this network we may define a min-cost flow problem as follows. For brevity, for every node $(s,t) \in \mathcal{A}$, let $O(s,t)$ denote the set $\{(s', t+1) : \tau_t(s,a) = s' \text{ for some } a \in A_{s,t}\}$ of nodes adjacent to outgoing arcs of $(s,t)$ and $I(s,t)$ denote the set $\{(s', t-1) : \tau_{t-1}(s', a) = s \text{ for some } a \in A_{s',t-1}\}$ of nodes adjacent to incoming arcs into $(s,t)$. Note that $I(s_0, 0) = \emptyset$ since the system starts at time 0 in state $s_0$.

The network formulation of (1), given the starting state $s_0$, is:

$$Z^* := \min_x \sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} x_{(s,t)(s',t+1)} \tag{2a}$$

$$\text{subject to} \sum_{(s',t+1) \in O(s_0,0)} x_{(s_0,0)(s',t+1)} = 1 \tag{2b}$$

$$\sum_{(s',t+1) \in O(s,t)} x_{(s,t)(s',t+1)} - \sum_{(s',t-1) \in I(s,t)} x_{(s',t-1)(s,t)} = 0 \text{ for } (s,t) \in \mathcal{N} \tag{2c}$$

$$x_{(s,t)(s',t+1)} \geq 0 \text{ for } ((s,t),(s',t+1)) \in \mathcal{A}, \tag{2d}$$

where $x_{((s,t)(s',t+1))}$ is interpreted as the flow along arc $((s,t),(s',t+1))$. Let

$$Z(x) := \sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} x_{(s,t)(s',t+1)}.$$

The next result relates problems (1) and (2). Details of the proof are contained in the appendix. The argument there follows familiar reasoning from the finite-dimensional case, but must take special care of some topological issues that arise in the infinite-dimensional setting.

**Proposition 1.** Problem (1) and (2) are equivalent in the following sense:

(i) every optimal policy $d^*$ can be used to construct an optimal flow $x^*$ where $V(\pi(d^*), s_0) = Z(x^*)$.

(ii) there exists an optimal flow $x^*$ that can be used to construct an optimal policy $d^*$ where $V(\pi(d^*), s_0) = Z(x^*)$.

It is straightforward to note that (2) is a CILP. As noted in the introduction, CILPs are hopelessly difficult to solve in general. Luckily, (2) comes from a particularly well-structured class that we describe in the next section.

## 3 Pure-supply infinite network flow problems

We now discuss a generalization of the network flow problem (2), which was introduced in Ryan et al. (2018). We recall the necessary notions from that paper and describe how (2) is a special case of that setting.

Let $G = (\mathcal{N}, \mathcal{A})$ be a directed graph with countably many nodes $\mathcal{N} = \{1, 2, \dots\}$ and arcs $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. Let $I(i)$ denote the set of nodes that are tails of arcs entering node $i$: $I(i) := \{j \in \mathcal{N} : (j, i) \in \mathcal{A}\}$. Similarly, the set of nodes that are heads of arcs leaving $i$ is $O(i) := \{j \in \mathcal{N} : (i, j) \in \mathcal{A}\}$. Each arc $(i, j)$ has *cost* $c_{ij}$. Each node has supply $b_i$. The countably-infinite network flow problem (CINF) problem on network $(\mathcal{N}, \mathcal{A}, b, c)$ is to a find a real nonnegative flow vector $x$ that minimizes the cost and maintains flow balance at every node:

$$\text{(CINF)} \qquad \begin{aligned} Z^* := \inf_x & \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} && \text{(3a)} \\ \text{s.t.} & \sum_{j\in O(i)} x_{ij} - \sum_{j\in I(i)} x_{ji} = b_i \text{ for } i \in \mathcal{N} && \text{(3b)} \\ & x_{ij} \geq 0 \text{ for } (i, j) \in \mathcal{A}. && \text{(3c)} \end{aligned}$$

Let $Z(x) := \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}$. A feasible solution $x$ to (CINF) is called a feasible flow.

For an arc $(i, j) \in \mathcal{A}$, node $i$ is called the *tail node* of arc $(i, j)$ and node $j$ its *head node*. The *in-degree* and *out-degree* of node $i$ in $G$ are the cardinalities of $I(i)$ and $O(i)$, respectively. A graph is *locally finite* if every node has finite in- and out-degree. A *finite (undirected) path* in $G$ is a finite sequence of distinct nodes $i_1$, $i_2, \dots, i_n$, where $(i_k, i_{k+1}) \in \mathcal{A}$ or $(i_{k+1}, i_k) \in \mathcal{A}$ for $k = 1, \dots, n-1$. A *path to infinity* is a sequence of distinct nodes $i_1, i_2, \dots$ where $(i_k, i_{k+1}) \in \mathcal{A}$ or $(i_{k+1}, i_k) \in \mathcal{A}$ for $k = 1, 2, \dots$. We typically use $P_{ij}$ to denote a finite path from node $i$ to node $j$, and $P_{i\infty}$ to denote a path from node $i$ to infinity. Two nodes $i$ and $j$ are *finitely connected* in $G$ if there exists a finite path $P_{ij}$. Two nodes $i$ and $j$ are *connected at infinity* if $G$ contains two paths to infinity, $P_{i\infty}$ and $P_{j\infty}$, that share no common nodes. Nodes $i$ and $j$ are *connected* if they are either finitely connected or connected at infinity. The graph $G$ is *finitely connected* if all nodes $i$ and $j$ in $G$ are finitely connected.

A *finite cycle* in $G$ is a finite sequence of nodes $i_1$, $i_2, \dots, i_n$, $i_1$ where $i_1$, $i_2, \dots, i_n$ is a path and either $(i_1, i_n) \in \mathcal{A}$ or $(i_n, i_1) \in \mathcal{A}$. An *infinite cycle*, also called a *cycle at infinity*, consists of two paths to infinity from some node $i$, $(i, i_1, i_2 \dots)$ and $(i, j_1, j_2, \dots)$, where all intermediate nodes $i_k$ and $j_\ell$ are distinct.

We also need directed versions of these definitions. A *finite directed path* from $i_1$ to $i_n$, denoted $P^{\rightarrow}_{i_1 i_n}$, is a finite path $i_1, \dots, i_n$ where $(i_k, i_{k+1}) \in A$ for all $k = 1, 2, \dots, n-1$. We call $P^{\rightarrow}_{i_1 i_n}$ an *out-path from* $i_1$ and an *in-path into node* $i_n$. A *directed path from node $i$ to infinity*, denoted $P^{\rightarrow}_{i\infty}$, is a path to infinity $P_{i\infty}$ where each arc in the path is directed *away* from node $i$. A *directed* *path from infinity to node $i$*, denoted $P^{\leftarrow}_{i\infty}$, is a path to infinity $P_{i\infty}$ where each arc in the path is directed *towards* node $i$. A *directed finite cycle* is a finite cycle that consists of a finite directed path $i_1, \dots, i_n$ and the arc $(i_n, i_1) \in \mathcal{A}$. A *directed cycle at infinity* is a cycle at infinity where both paths to infinity from a given node $i$ are directed, one from infinity to $i$, and the other from $i$ to infinity.

A graph is *acyclic* if it contains no finite or infinite directed cycles. A subgraph of $G$ is a *forest* if it contains no (undirected) cycles. A connected forest is called a *tree*. A tree that contains all the nodes of $G$ is called a *spanning tree*. We are a bit sloppy when it comes to subgraphs of $G$, and will think of them alternatively as sets of arcs, sets of nodes, or entire subgraphs, as the context requires.

An *in-tree rooted at node $r$* is a tree where from every node $i$ in the tree there is a directed path to node $r$ where all arcs are directed to node $r$. An *out-tree rooted at node $r$* has directed paths from node $r$ to every other node in the tree, with now each arc directed away from node $r$. When $r$ is designated as the node at infinity, we call the tree an *in-tree rooted at infinity* and has the property that every node $i$ in the tree has a unique infinite directed path $P^{\rightarrow}_{i\infty}$.

Following Ryan et al. (2018), we will assume that the network $(\mathcal{N}, \mathcal{A}, b, c)$

(A1) is locally finite,

(A2) is finitely connected,

(A3) contains no finite or infinite directed cycles,

(A4) has finitely many nodes with in-degree 0

(A5) has integer-valued supplies; that is, $b_i$ is integer for all $i \in \mathcal{N}$

(A6) has nonnegative supplies; that is, $b_i \geq 0$ for all $i \in \mathcal{N}$,

(A7) has uniformly-bounded supplies; that is, $b \in \ell_\infty(\mathcal{N})$ and so there exists a $\bar{b} = ||b||_\infty$ is the uniform upper bound on all node supplies, and

(A8) the arc costs are nonnegative; that is, $c_{ij} \geq 0$ for all $(i, j) \in \mathcal{A}$.

Assumption (A8) is not assumed in Ryan et al. (2018) but is needed here for reasons that will be remarked on below.

A vector $x$ satisfying constraints (3b) is a *basic flow* if the arcs $\{(i, j) \in \mathcal{A} : x_{ij} \neq 0\}$ form a forest in $G$. Theorem 3.13 of Romeijn et al. (2006) shows that every forest can be extended to a spanning tree of $G$. This ensures that every basic flow can be associated with (at least one) spanning tree. A basic flow is a *basic feasible flow (bff)* if the flow is also nonnegative.

Until now we have not set any assumptions on the costs of arcs. To do so, we introduce another concept: *stages* of nodes. As carefully detailed in Ryan et al. (2018), this requires some preprocessing. Identify all transshipment nodes with out-degree zero. Since no feasible flow will send positive flow along arcs into such nodes, they can be removed along with all of their adjacent arcs without loss of generality. Apply this rule recursively until no such nodes remain. Moreover, without loss of *feasibility*, each supply node has out-degree at least one — otherwise flow balance is violated. This establishes the following.

**Proposition 2** (Proposition 2.1 in Ryan et al. (2018))**.** In every feasible instance of (CINF) whose underlying instance satisfies (A1)–(A7), each node has out-degree at least one without loss of generality.

We can now define stages of nodes. Stage 0 is the set of all nodes with in-degree 0. From assumption (A4), this set is finite. Stage 1 consists of all nodes with in-degree 0 in the modified graph that results from removing all stage 0 nodes and their adjacent arcs. Observe that all Stage 1 nodes are adjacent to Stage 0 nodes in the graph. Repeat this procedure to construct the remaining stages. Since the graph is acyclic, each node is contained in exactly one stage. Let $\mathcal{S}_t$ denote the set of nodes in Stage $t$ and $s(i)$ denote the stage of node $i$. By construction, each stage is a finite collection of nodes. A key property for our dual-based methods is the following.

**Lemma 1.** For every arc $(i, j) \in \mathcal{A}$, $s(i) < s(j)$.

*Proof.* Only after removing node $i$ will it be possible that $j$ has an in-degree of 0 since the arc $(i, j)$ itself gives node $j$ an in-degree of at least one. □

Since there are countably-many nodes, they can be numbered $1, 2, \ldots$. There are many possible numberings, but we require that the numberings of nodes obeys the staging. That is, for every $i \in \mathcal{S}_m$ and $j \in \mathcal{S}_n$ with $m < n$, node $i$ is numbered before node $j$. Succinctly we write this as $i < j$.

We make the following assumptions on stages and structure of the costs of the graph.

(A9) there exist $\beta \in (0, 1)$ and $\gamma \in (0, +\infty)$ such that for every $(i, j) \in \mathcal{A}$, $|c_{ij}| \leq \gamma\beta^{s(i)}$, where $\beta$ can be interpreted as a discount factor (bound on arc costs),

(A10) there exists a sub-exponential function $g(t)$ such that the cardinality of stage $t$ is bounded above by $g(t)$: $\#(\mathcal{S}_t) \leq g(t)$ for all $t$ (bound on the cardinalities of stages), and

**Definition 1.** Following Ryan et al. (2018), we call a network that satisfies assumptions (A1)–(A10) a *pure supply network*. A (CINF) problem with an underlying pure supply network is called a *pure supply problem.*

The name "pure supply" is inspired by assumption (A6), but also includes all the additional assumptions stated. As noted in Ryan et al. (2018), we can also handle the complementary setting where assumption (A6) is replaced by $b_i \leq 0$ for all $i \in \mathcal{N}$, by simply reversing arc directions and changing the signs of the demands.

## 3.1 Dynamic programming is a pure supply problem

We next show that deterministic dynamic programming falls within our focus class of network flow problems.

**Proposition 3.** The network flow problem (2) associated with dynamic program (1) is a pure-supply problem.

*Proof.* Let $N$ denote the network underlying problem (2) and $G$ its associated graph. Hence, $G$ is locally finite (condition (A1)) under the stated assumption that $A_{s,t}$ is finite for all $s$ and $t$. As for local connectedness (condition (A2)), this holds given our structure of action sets and transitions. Indeed, we have assumed that $A_{s,t} \neq \emptyset$ for all $s, t$ and for every $s' \in S_{t+1}$ there exists an $s \in S_t$ and $a \in A_{s,t}$ such that $s' = \tau_t(s,a)$. This ensures finite connectedness since all nodes can trace a path back to $(s_0, 0)$ and so are connected (in the undirected sense that is required). Turning to the nonexistence of directed cycles (condition (A3)), there exists no arc from a node $(s,t)$ to a node $(s',t')$ with $t' \leq t$. This makes both finite and infinite directed cycles impossible. Regarding 0 in-degree (condition (A4)), the only node with in-degree 0 is the starting node $(s_0, 0)$ and so there are clearly finitely-many such nodes. Conditions (A5)–(A7) hold since the right-hand of (2) only takes on values 0 and 1. Turning to the stage structure, observe that stage $\mathcal{S}_t = \{(s,t) : s \in S_t\}$. Conditions (A9) and (A10) then follow immediately by boundedness of immediate costs (property (S1)), cost discounting by $\delta$, and assumptions on the growth of states (property (S2)) stated in Section 2. Condition (A8) holds since $c_t(s,a)$ are nonnegative for all $s$,$a$, and $t$. $\square$

**Remark 1.** The previous result shows that the dynamic programming network flow problem can be studied using the methodology of Ryan et al. (2018). As we argued in Lemma 14, the primal *shadow simplex method* of Ghate et al. (2010) also applies. One may also check that the primal simplex method of Sharkey and Romeijn (2008) is valid for this setting. The network structure of (2), if we explicitly add the implied constraints $x_{(s,t)(s',t+1)} \leq 1$ to the formulation, does satisfy the basic assumptions of Sharkey and Romeijn (2008) as well as the critical condition that the flow between stages is uniformly bounded (Proposition 2.5). The latter property is needed to ensure their simplex method converges in optimal value.

On the other hand, primal simplex methods described in (Ghate and Smith, 2013; Lee et al., 2017) do not directly apply. Their methods crucially rely on nondegeneracy properties of the underlying hypernetwork that corresponds to a stochastic dynamic program. By contrast, (2) is highly degenerate since all but one node has a supply of zero. $\triangleleft$

## 3.2 Constructing trees and basic feasible flows in pure supply networks

The following properties of pure supply networks are used in our analysis of the dual-ascent method. Consider the following procedure:

**Procedure 1** (Constructing a spanning tree)**.** Given a pure supply network,

(i) for every node $i$ select a single outgoing arc $a_i$ (such an arc is guaranteed to exist by Proposition 2), and

(ii) construct the subgraph $T$ with arc set $\{a_i : i \in \mathcal{N}\}$.

**Lemma 2** (Lemma 4.2 in Ryan et al. (2018))**.** A subgraph $T$ of a pure supply network is a spanning in-tree rooted at infinity if and only if it can be constructed by Procedure 1.

Consider also the following related procedure.

**Procedure 2** (Constructing a basic flow from a tree)**.** Given a spanning tree $T$ of a pure supply network, construct a flow $x^T$ as follows:

(i) initially set $x_i^S = 0$ for all $i \in \mathcal{N}$

(ii) for each $i \in \mathcal{N}$, identify the unique path $P_{i\infty}$ from $i$ to infinity in $T$, with forward arcs $P_{i\infty}^F$ and backward arcs $P_{i\infty}^B$, and add a flow of $b_i$ to all arcs in $P_{i\infty}^F$ and remove a flow of $b_i$ from all arcs in $P_{i\infty}^B$.

For a general spanning tree $T$, $x^T$ need not be a basic *feasible* flow. However, when $T$ is an in-tree rooted at infinity, this is indeed the case.

**Lemma 3** (Lemma 4.4 in Ryan et al. (2018))**.** If $T$ is a spanning in-tree rooted at infinity in a pure supply network (e.g., if $T$ is constructed by Procedure 1) then $x^T$ is a basic feasible flow.

## 4 A dual-ascent method for pure-supply problems

We construct a simple dual-ascent method to solve (CINF), inspired by the success of dual-ascent methods for finite network flow problems. From Ryan et al. (2018), the dual problem to (3) is

$$\text{(CINFD)} \qquad \begin{aligned} D^* := \max_{\pi} \ & \sum_{i \in \mathcal{N}} b_i \pi_i && && \text{(4a)} \\ \text{s.t. } & \pi_i - \pi_j \le c_{ij} && \text{for } (i,j) \in \mathcal{A} && \text{(4b)} \\ & \pi \in c_0, && && \text{(4c)} \end{aligned}$$

where $c_0$ is the vector space of all sequences that converge to 0. Let $D(\pi) := \sum_{i \in \mathcal{N}} b_i \pi_i$.

Weak duality for (3) and (4) was proved in Ryan et al. (2018), which is critical to our Dual-Ascent Method. We restate the result here for completeness.

**Theorem 1** (Weak duality, Theorem 7.3 of Ryan et al. (2018))**.** In an instance of (3) where assumption (A1)–(A10) hold, every primal feasible $x$ and dual feasible $\pi$ satisfy $Z(x) \ge D(\pi)$. In particular, if there exist a primal feasible $x^*$ and dual feasible $\pi^*$ such that $Z(x^*) = D(\pi^*)$ then $x^*$ is an optimal solution to (3) and $\pi^*$ is an optimal solution to (4).

The idea of a dual-ascent method is to iteratively generate dual feasible solutions $\pi^n$ that strictly improves $D(\pi^n)$ (unless a termination condition is reached). Each iteration of our algorithm adjusts $\pi$ at a single node $i$, incrementing it by the slack value $s_{ij} = c_{ij} + \pi_j - \pi_i$ of constraint (4b) for some $j$ where $(i,j) \in \mathcal{A}$. This construction implies that $\pi_i$ maintains the cost of a finite path from

Dual-Ascent Method

1. (Initialization) Input a pure supply network (that is, an infinite network satisfying assumptions (A1)–(A10)) and set $\pi_i^0 = 0$ for all $i \in \mathcal{N}$ and $n = 1$.
2. (Construct slacks) Set

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \qquad \text{for all } (i,j) \in \mathcal{A}, \tag{5}$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \qquad \text{for all } i \in \mathcal{N} \tag{6}$$

$$s^n \leftarrow \max_{i \in \mathcal{N}} s_i^n. \tag{7}$$

3. (Check for termination) If $s^n = 0$ then *terminate*. Else go to 4.
4. (Update $\pi^n$) Set $i^n \in \arg\max_{i \in \mathcal{N}} s_i^n$ and $j^n \in \arg\min_{j \in O(i)} s_{i^n j}^n$ and update

$$\pi_i^n \leftarrow \begin{cases} \pi_{j^n}^{n-1} + c_{i^n j^n} & \text{if } i = i^n \\ \pi_i^{n-1} & \text{otherwise.} \end{cases} \tag{8}$$

5. (Update $n$) Increment $n$ to $n+1$ and go to step 2.

node $i$ to some node $j$ in the graph. Our algorithm chooses the node $i$ that admits the largest possible increase in the value of $\pi$. That is, node $i$ is chosen so that $s_{ij}$ is as large as possible and yet still maintains dual feasibility. Since the $s_{ij}$ are nonnegative for all $(i,j) \in \mathcal{A}$, this implies that $\pi$ is strictly increased at each iteration unless all $s_{ij}$ are zero. Careful details follow.

The following provides an alternate interpretation of the updating rule (8).

**Lemma 4.** For all $i$ and $n$, the iterate $\pi_i^n$ of the Dual-Ascent Method satisfies

$$\pi_i^n = \begin{cases} \pi_{i^n}^{n-1} + s_{i^n j^n}^n & \text{if } i = i^n \\ \pi_i^n & \text{otherwise.} \end{cases} \tag{9}$$

*Proof.* Observe that $c_{i^n j^n} + \pi_{j^n}^{n-1} = \pi_i^{n-1} + s_{i^n j^n}^n$ by (5) and so we can alternately express the updating (8) of $\pi_i^n$ as in (9). □

**Remark 2.** It is important to point out the possibility that the same node $i^n$ may be visited more than once by the algorithm. Observe that when an arc $a = (i^{n_1}, j^{n_1})$ is selected at time $n_1$, then (8) ensures that $s_{i^{n_1} j^{n_1}}^{n_1+1} = 0$ when the algorithm next visits (5). That is,

$$\pi_{i^{n_1}}^{n_1} - \pi_{j^{n_1}}^{n_1} = c_{i^{n_1} j^{n_1}} \tag{10}$$

This implies $s_{i^{n_1} j^{n_1}}^{n_1+1} = 0$ and so node $i^{n_1}$ will not be immediately selected as $i^{n_1+1}$ at iteration $n_1 + 1$. However, we now show it is possible for $s_{i^{n_1} j^{n_1}}^{n_2} > 0$ at a later iteration for some iteration

$n_2$, allowing the possibility that $i^{n_2} = i^{n_1}$. Suppose $i^{n_2-1} = j^{n_1}$ (that is, the head node of arc $a$ is selected immediately before iteration $n_2$) and neither node incident to arc $a$ was selected since iteration $n_1$. This implies that (10) still holds at iteration $n_2 - 1$; that is,

$$\pi_{i^{n_1}}^{n_2-1} - \pi_{j^{n_1}}^{n_2-1} = c_{i^{n_1} j^{n_1}}. \tag{11}$$

Now, suppose arc $(j^{n_1}, j^{n_2-1})$ is the arc selected in iteration $n_2 - 1$. This implies

$$\pi_{j^{n_1}}^{n_2} = \pi_{j^{n_1}}^{n_2-1} + c_{j^{n_1} j^{n_2-1}} \tag{12}$$

and

$$\pi_{i^{n_1}}^{n_2} = \pi_{i^{n_1}}^{n_2-1}. \tag{13}$$

Putting equations (11)–(13) together implies

$$\begin{aligned} \pi_{i^{n_1}}^{n_2} - \pi_{j^{n_1}}^{n_2} &= \pi_{i^{n_1}}^{n_2-1} - \pi_{j^{n_1}}^{n_2-1} - c_{i^{n_1} j^{n_1}} \\ &= c_{i^{n_1} j^{n_1}} - c_{i^{n_1} j^{n_1}} \\ &< c_{i^{n_1} j^{n_1}}, \end{aligned}$$

under the assumption of nonnegative costs and assuming for sake of argument that $c_{j^{n-1} j^{n_2-1}} > 0$. This implies that $s_{i^{n_1} j^{n_1}}^{n_2} > 0$, allowing for the possibility that $i^{n_2} = i^{n_1}$. This return to a previously processed node may cause the reader to worry the algorithm cycles. The issue of cycling is addressed below in Remark 4. $\triangleleft$

The following observations give us greater insights into the Dual-Ascent Method. The first key observation is that iterates are dual feasible solutions.

**Proposition 4** (Dual feasibility of iterates)**.** For all $n$, the iterates $\pi^n$ of the Dual-Ascent Method are feasible to (CINFD).

*Proof.* We argue inductively. For the base ($n = 0$) case, plugging $\pi^0$ into (4b) reveals $0 \le c_{ij}$ for all $(i, j) \in \mathcal{A}$ and clearly $\pi^0 \in c_0$. Thus, $\pi^0$ satisfies (4b) and (4c) and so is dual feasible.

For the inductive step, suppose $\pi^{n-1}$ is dual feasible and argue the same is true for $\pi^n$. It suffices to check that $s_{ij}^{n+1} \ge 0$ for all arcs $(i, j) \in \mathcal{A}$. If arc $(i, j)$ is such that neither $i$ nor $j$ are equal to $i^n$ then $s_{ij}^{n+1} \geq 0$ follows by induction. Next suppose $j = i^n$ in arc $(i,j) \in \mathcal{A}$ and hence

$$\begin{aligned} s_{ii^n}^{n+1} &= c_{ii^n} + \pi_{i^n}^n - \pi_i^n \\ &= c_{ii^n} + \pi_{i^n}^n - \pi_i^{n-1} \\ &\geq c_{ii^n} + \pi_{i^n}^{n-1} - \pi_i^{n-1} \\ &= s_{ii^n}^n \\ &\geq 0, \end{aligned}$$

where the second equality holds since $\pi^n = \pi_i^{n-1}$, the first inequality uses the fact that $\pi_{i^n}^n \geq \pi_{i^n}^{n-1}$ from (9) and the inductive hypothesis that $s_{i^n j^n}^n \geq 0$, and the third equality holds by the definition of $s_{ii^n}^n$ and the final inequality holds by the inductive hypothesis.

Next, suppose $(i,j)$ is such that $i = i^n$. Then we have for all $j \in O(i)$,

$$\begin{aligned} s_{i^n j}^{n+1} &= c_{i^n j} + \pi_j^n - \pi_{i^n}^n \\ &= c_{i^n j} + \pi_j^n - \pi_{i^n}^{n-1} - s_{i^n j^n}^n \\ &= s_{i^n j}^n - s_{i^n j^n}^n \\ &\geq 0, \end{aligned}$$

where the second equality uses (9), the third equality uses the definition of $s_{i^n j}^n$ and the inequality uses the fact that $s_{i^n j^n} = \min_{j \in O(i^n)} s_{i^n j}^n$.

It only remains to argue that $\pi^n \in c_0$. Note that the updating process of $\pi^{n-1}$ in Step 3 of the Dual-Ascent Method changes at most one entry from $\pi^{n-1}$ to $\pi^n$. Hence, since $\pi_i^0 = 0$ for all $i \in \mathcal{N}$, $\pi^n$ has at most $n$ nonzero entries, implying $\pi^n \in c_0$. □

Proposition 4 implies, in particular, that $s_{ij}^n \geq 0$ for all $(i,j) \in \mathcal{A}$ and for all $n$. The next result discusses the termination condition, which is related to the values of the slacks.

**Proposition 5** (Optimality upon termination)**.** If the termination condition $s^n = 0$ in the Dual-Ascent Method is reached at iteration $n$, $\pi^{n-1}$ is an optimal dual solution. Moreover, a primal optimal flow can be constructed at termination.

*Proof.* By Proposition 4, $\pi^{n-1}$ is a dual feasible solution. If $s^n = 0$ then $s_i^n = 0$ for all $i$ and thus for every node $i$ there exists an arc $(i,j)$ with $\pi_i^{n-1} - \pi_j^{n-1} = c_{ij}$. Construct a tree $T$ using Procedure 1 where $(i,j)$ is chosen for each node $i$. By Lemma 2, the resulting tree $T$ is a spanning in-tree rooted at infinity and so the flow $x^T$ constructed using Procedure 2 is a basic feasible flow by Lemma 3.

Next, we argue that $\pi^{n-1}$ and $x^T$ have the same objective values; that is, $D(\pi^{n-1}) = Z(x^T)$.

Indeed,

$$\begin{aligned}
Z(x^T) &= \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}^T \\
&= \sum_{(i,j)\in T} c_{ij}x_{ij}^T \\
&= \sum_{i\in\mathcal{N}} \sum_{(i,j)\in P_{i\infty}} b_i c_{ij} \\
&= \sum_{i\in\mathcal{N}} b_i \left[ \sum_{(i,j)\in P_{i\infty}} (\pi_i^{n-1} - \pi_j^{n-1}) \right] \\
&= \sum_{i\in\mathcal{N}} b_i \pi_i^{n-1} \\
&= D(\pi^{n-1}),
\end{aligned}$$

where the first equality is the definition of $Z(x^T)$, the second equality uses the fact that $x_{ij}^T = 0$ for $(i,j) \notin T$, the third equality is the method of constructing basic feasible solutions in Procedure 2, the fourth equality uses the fact that $s_{ij} = 0$ for all arcs $(i,j)$ in $T$ and so $c_{ij} = \pi_i^{n-1} - \pi_j^{n-1}$, the fifth equality comes from telescoping along the path $P_{i\infty}$, and the final equality is the definition of $D(\pi^{n-1})$.

Since $Z(x^T) = D(\pi^{n-1})$ and $x^T$ is primal feasible and $\pi^{n-1}$ is dual feasible, Theorem 1 implies $\pi^{n-1}$ is an optimal dual solution and $x^T$ is a primal optimal flow. □

The next results discuss interpretations of the dual iterates $\pi^n$.

**Lemma 5.** For all $i$ and $n$, the iterate $\pi_i^n$ of the DUAL-ASCENT METHOD is the cost of a finite out-tree from node $i$ (and possibly an empty tree with no arcs and cost 0).

*Proof.* This follows by induction. For the base case, note that $\pi_i^0 = 0$ is the cost of an empty directed tree. The inductive hypothesis is that for all $i$, $\pi_i^{n-1}$ is the cost of a finite out-tree from node $i$. By the updating formula (8), when $i \neq i^n$, $\pi_i^n = \pi_i^{n-1}$ and clearly $\pi_i^{n-1}$ is the cost of a finite out-tree from node $i$ by the inductive hypothesis. When $i = i^n$, we know that there is a finite out-tree $T$ from node $j^n$ with cost $\pi_{j^n}^{n-1}$. Consider adding arc $(i^n, j^n)$ to $T$. The only worry is that adding this arc creates a cycle, which only happens where $j^n$ is in $T$ already. However, if $j^n$ is in $T$ already, adding $(i^n, j^n)$ creates a finite directed cycle. Under assumption (A3) this cannot happen. Hence, adding $(i^n, j^n)$ to $T$ creates a finite out-tree from node $i$ and thus $\pi_i^n$ is the cost of that newly created tree. This completes the argument. □

**Remark 3.** Note that it need not be the case that the $T$ defined in the proof of the previous lemma is a directed path. Indeed, the possibility that a node $i$ is visited more than once by the algorithm (as discussed in Remark 2) gives rise to this possibility. ◁

The next results concern the "computational finiteness" of the algorithm. In particular, there is an important step (step (7)) that naïvely could involve infinite work if the maximum that defines $s^n$ is attained at all. The next two results resolve this issue. Observe that the minimum defining $s_i^n$ in (6) is attained since $O(i)$ is a finite set by assumption (A1).

**Lemma 6.** The slack variable $s^n$ defined in (7) is well-defined; that is, despite the fact $\mathcal{N}$ is infinite, the maximum defining $s^n$ is finite and attained.

*Proof.* First, we show each $s_{ij}^n$ is finite. All $\pi_i$ can be bounded above by $||c||_1$ since $\pi_i$ is the length of a finite out-tree in the graph (by Lemma 5) which is bounded by the sum of all costs $||c||_1$ of all arcs in the graph (this uses assumption (A8)). This implies that $s_{ij}^n = c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \leq 4||c||_1 < \infty$. This gives a uniform upper bound on the $s_{ij}^n$, and so $s_i^n$ and $s^n$ are all finite.

Next, we claim that $s_i^n$ converges to 0 as $i \to \infty$. This will imply there exists an $M$ such that $\max_{i \in \mathcal{N}} s_i^n \leq \max_{i \in \{1,2,\dots,M\}} s_i^n$. By Lemma 1, the stage $s(j)$ of node $j$ for any $(i,j) \in \mathcal{A}$ is in a stage greater than $s(i)$. Hence, the out-trees that defines $\pi_i^n$ and $\pi_j^n$ consist of arcs with tail nodes in stage at least $s(i)$. As $i$ goes to infinity, $s(i) \to \infty$ since nodes are numbered according to stages and each stage contains finitely many nodes. Thus, $\pi_i^n \to 0$, $\pi_j^n \to 0$, and $c_{ij} \to 0$ as $i \to \infty$. This implies $s_{ij}^n \to 0$ as $i \to \infty$ for all $j \in O(i)$ and thus $s_i^n \to 0$ as $i \to \infty$. This completes the argument. □

**Proposition 6** (Finite implementability)**.** If $s^n \neq 0$ then steps 2 to 5 of the DUAL-ASCENT METHOD can be executed in finite time at iteration $n$ of the algorithm.

*Proof.* Since $s^n \neq 0$ then there exists an $i$ such that $s_i^n > 0$. Hence, there exists an $M$ such that $s_j^n \leq s_i^n$ for all $j \geq M$, again using similar reasoning as in the proof of the previous theorem. In fact, $M$ can be computed using the cost structure in assumption (A9) once $i$ is specified. Thus, there is no need to consider the graph beyond its first $M$ nodes. Accordingly, steps 2 to 5 can be executed in finite time. □

An important observation is that the termination condition $s^n = 0$ of the DUAL-ASCENT METHOD cannot, in general, be verified in finite time. Interestingly, if the algorithm reaches a stage where the condition is true, then the current solution is optimal (via Proposition 5) but there is no way (in finite time) to know that this is the case. However, Proposition 6 says that if we ignore Step 3 and we happen to be in the condition that $s^n \neq 0$, then all other steps of the algorithm can be processed in finite time. It is in this sense that we say that the DUAL-ASCENT METHOD is finitely-implementable. A similar phenomenon was observed in Ghate and Smith (2013).

We now turn to the asymptotic performance of the DUAL-ASCENT METHOD. That is, we consider the limiting behavior of the iterates as $n$ tends towards infinity.

**Lemma 7** (Convergence to dual feasibility)**.** The sequence of iterates $\pi^n$ of the Dual-Ascent Method converges pointwise to a dual feasible vector $\pi^*$ as $i \to \infty$. That is, $\pi_i^n \to \pi_i^*$ as $n \to \infty$ for all $i \in \mathcal{N}$.

*Proof.* We first show that the $\pi^n$ converge pointwise. As argued in the proof of Lemma 6, the $\pi_i^n$ are nonnegative and uniformly bounded above by $||c||_1$. Moreover, since $c_{ij} \geq 0$, $\pi_i^n$ is monotone nondecreasing as a sequence in $n$. Hence, for each $i$, the sequence $\pi_i^n$ converges to some limit, say $\pi_i^*$. Let $\pi^* = (\pi_i^* : i = 1, 2, \dots)$.

Next, we argue that $\pi^*$ is dual feasible. From Proposition 4 we know $c_{ij} + \pi_j^n - \pi_i^n \geq 0$ for all $n$. Hence $\lim_{n\to\infty}(c_{ij} + \pi_j^n - \pi_i^n) \geq 0$ or $c_{ij} + \pi_j^* - \pi_i^* \geq 0$ and $\pi^*$ satisfies (4b). It remains to argue that $\pi^*$ is in $c_0$. Suppose otherwise. There exists a subsequence $\{i_k\}_{k=1}^{\infty}$ of nodes such that

$$\pi_{i_k}^* > \epsilon \text{ for some } \epsilon > 0 \tag{14}$$

As argued in the previous paragraph, $\pi_{i_k}^n \to \pi_{i_k}^*$ as $n \to \infty$. Via Lemma 5, we know $\pi_{i_k}^n$ is the cost of a finite out-tree $T^n$ from node $i_k$. By Lemma 1, all arcs in $T^n$ have tails in stage $s(i_k)$ or higher. Thus by assumption (A9), the cost of every arc in $T^n$ is bounded by $\gamma\beta^{s(i)}$. Moreover, we can upper bound the number of arcs in $T^n$ by the subexponential function $\sum_{t=s(i_k)}^{s_n} g(t)$, where $s_n$ is the stage of the largest head node of an arc of $T^n$ (such an arc exists since $T^n$ is finite). Thus, $\pi_{i_k}^n \leq \sum_{t=s(i_k)}^{s_n} g(t)\gamma\beta^{s(i_k)}$, which converges to 0 as $k$ goes to infinity. This implies that there exist $k_0$ and $n_0$ such that $\pi_{i_k}^n \leq \epsilon$ for $n \geq n_0$ and $k \geq k_0$. Since $\pi_{i_k}^n \to \pi_{i_k}^*$ as $n \to \infty$, this implies that $\pi_{i_k}^* \leq \epsilon$ for $k$ sufficiently large. This contradicts (14), completing the proof. □

**Lemma 8** (Slacks tend to zero)**.** The $s_i^n$ defined in (6) of the Dual-Ascent Method satisfy $\lim_{n\to\infty} s_i^n \to 0$ for all $i \in \mathcal{N}$.

*Proof.* We proceed by contradiction. Suppose not so that there exists an $i \in \mathcal{N}$, an $\epsilon > 0$, and a subsequence $s_o^{n_k}$ of the $s_i^n$ such that $s_i^{n_k} > \epsilon > 0$ for all $k$. This implies that $s^n > \epsilon$ for infinitely many $n$. Hence, from (9) in Lemma 4, an entry of $\pi^n$ is increased by at least $\epsilon > 0$ infinitely often. But this is impossible since $\pi_i^n$ is the cost of a finite out-tree from node $i$, which is bounded above by $||c|||_1$. □

**Theorem 2** (Convergence to dual optimality)**.** The sequence of iterates $\pi^n$ generated by the Dual-Ascent Method converge pointwise to an optimal dual solution $\pi^*$.

*Proof.* By Lemma 7, the limit sequence $\pi^*$ exists and is dual feasible. Let $s_{ij}^* := c_{ij} + \pi_j^* - \pi_i^*$, which denotes the slacks of the dual constraints (4b) for the dual solution $\pi^*$. Since $\pi_j^n \to \pi_j^*$ and $\pi_i^n \to \pi_i^*$ as $n \to \infty$,

$$s_{ij}^n \to c_{ij} + \pi_j^* - \pi_i^* = s_{ij}^* \tag{15}$$

as $n \to \infty$. From Lemma 8, for all $i$, $s_i^n \to 0$. By the pigeon-hole principle, for every $i$ there exists a $j \in O(i)$ such that $s_i^n = s_{ij}^n$ for infinitely many $n$. Restricting to the subsequence $n_k$ of the $n$

such that $s_i^{n_k} = s_{ij}^{n_k}$, $s_{ij}^* = \lim_k s_i^{n_k} = 0$, using (15) and Lemma 8, respectively (and noting all subsequences of convergent sequences have the same limit). This implies that for each node $i$, we can identify an arc $(i, j)$ such that $s_{ij}^* = 0$.

By Lemma 2, the subgraph $T^*$ of $G$ consisting of the identified arcs where $s_{ij}^* = 0$ form a spanning in-tree rooted at infinity. Thus, by Lemma 3, the associated basic feasible flow $x^{T^*}$, constructed by Procedure 2, is a basic feasible flow. We note that $Z(x^{T^*}) = D(\pi^*)$. This is precisely the same argument captured in the string of equalities in the proof of Proposition 5, replacing $T$ with $T^*$ and $\pi^{n-1}$ with $\pi^*$. Finally, by Theorem 1, $Z(x^{T^*}) = D(\pi^*)$ implies $\pi^*$ is an optimal dual solution (and $x^{T^*}$ is a primal optimal flow). □

**Remark 4.** It is useful to note that it is impossible for the sequence of iterates to cycle; that is, revisit an earlier value for $\pi$. The reason is that the updating of $\pi$ (as reformulated in (9)) always occurs when $s_{i^n j^n}^n$ is strictly bigger than zero (otherwise the method terminates in step 3). ◁

**Remark 5.** The proof of Theorem 2 establishes the existence of a primal optimal solution $x^{T^*}$ and a dual optimal solution $\pi^*$ with the same objective value, that is, $Z(x^{T^*}) = D(\pi^*)$. In other words, the Dual-Ascent Method can be used to establish the strong duality of (CINF) and (CINFD) under assumptions (A1)–(A10). This was established in Ryan et al. (2018) using a primal simplex method, and in previous papers (such as (Nourollahi and Ghate, 2017; Sharkey and Romeijn, 2008)) using topological arguments. ◁

As the above results show, the Dual-Ascent Method produces a sequence of improving dual feasible solutions that pointwise converge to an optimal dual solution. In other words, the Dual-Ascent Method is an approach to solve the *dual* of the pure supply problem (3). Another usage is to develop optimality error bounds for the primal problem in finite time. In Ryan et al. (2018), the authors propose a primal simplex method that generates an improving sequence of primal feasible iterates that converge in value to optimality. Running the primal simplex method and dual ascent method in parallel produces a primal feasible solution $x^n$ and dual feasible solution $\pi^n$ after $n$ iterations of each algorithm. The optimality error $Z(x^n) - Z^*$ of the primal is thus bounded by $D(\pi^n) - Z(x^n)$ after finitely many iterations of both algorithms. In the next section, we show how to adjust the Dual-Ascent Method to produce a sequence of primal and dual feasible iterates simultaneously, and whose structures are related, by devising a primal-simplex method.

## 5 A primal-dual method for pure supply problems

We now develop a primal-dual method. It is standard practice in the finite-dimensional setting to develop first a dual-ascent method and then modify it slightly (mostly by just tracking more of what is already generated in the dual ascent case) to construct a primal-dual method that maintains a sequence of both primal and dual feasible iterates. The connection between the primal and dual feasible solution is through the following notion.

Primal-Dual Method

1. (Initialization) Input a pure supply network (that is, an infinite network satisfying assumptions (A1)–(A10)) and set $\pi_i^0 = 0$ for all $i \in \mathcal{N}$, $B^0 = \{i \in \mathcal{N} : c_{ij} = 0 \text{ for some } j \in O(i)\}$, and $n = 1$. For every $i \in \mathcal{N}$ select a single arc $a_i = (i, j)$ such that $c_{ij} = \min\{c_{ij} : j \in O(i)\}$ and $T^0 = \{a_i : i \in \mathcal{N}\}$
2. (Construct initial slacks) Set

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \qquad \text{for all } (i,j) \in \mathcal{A}, \tag{16}$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \qquad \text{for all } i \in \mathcal{N} \tag{17}$$

3. (Termination check) Set $s^n \leftarrow \max_{i \in \mathcal{N}} s_i^n$. If $s^n = 0$ then *terminate*. Else go to Step 4.
4. (Balancing step) Set $i^n \in \arg\max_{i \in \mathcal{N}} s_i^n$, $j^n \in \arg\min_{j \in O(i)} s_{i^n j}^n$, and

$$\pi_i^n \leftarrow \begin{cases} \pi_{j^n}^{n-1} + c_{i^n j^n} & \text{if } i = i^n \\ \pi_i^{n-1} & \text{otherwise.} \end{cases} \tag{18}$$

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^n - \pi_i^n \text{ for all } (i,j) \in \mathcal{A} \text{ with } i = i' \text{ or } j = i' \tag{19}$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \text{ for all } i \in \mathcal{N} \tag{20}$$

$$B^n \leftarrow B^{n-1} \cup \{i^n\} \tag{21}$$

$$T^n \leftarrow T^{n-1} \cup \{(i^n, j^n)\} \setminus \left\{(i,j) \in T^{n-1} : i = i^n\right\} \tag{22}$$

5. (Rebalancing step)
   **while** $\{i \in B^n | s_i^n > 0\} \neq \emptyset$

$$i' \leftarrow \max\{i \in B^n | s_i^n > 0\} \tag{23}$$

$$j' \leftarrow \text{ a node } j' \in O(i') \text{ such that } s_{i'j'}^n = s_i^n \tag{24}$$

$$\pi_{i'}^n \leftarrow \pi_{j'}^n + c_{i'j'} \tag{25}$$

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^n - \pi_i^n \text{ for all } (i,j) \in \mathcal{A} \text{ with } i = i' \text{ or } j = i' \tag{26}$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \text{ for all } i \in \mathcal{N} \tag{27}$$

$$T^n \leftarrow T^n \cup \left\{(i', j')\right\} \setminus \left\{(i,j) \in T^n : i = i'\right\} \tag{28}$$

   **endwhile**
6. (Update $x^n$) Set

$$x^n \leftarrow x^{T^n} \tag{29}$$

   where $x^{T^n}$ is as constructed in Procedure 2.
7. (Update $n$) Set $n \leftarrow n + 1$ and go to Step 3.

**Definition 2.** An arc $(i, j)$ is balanced with respect to $\pi \in \mathbb{R}^{\mathcal{N}}$ if $\pi_i - \pi_j = c_{ij}$.

In the finite-dimensional setting, the dual feasible iterates $\pi^n$ are built so that they correspond to costs of a forest of balanced arcs (with respect to $\pi^n$). This forest can then be extended to a spanning tree, and thus a primal feasible flow (typically using some max flow algorithm). As the algorithm proceeds, arcs are added to this forest and eventually there is a dual feasible iterate found that corresponds (in cost) to a spanning tree of balanced arcs. Then an argument very similar to the proof of Proposition 5 produces an optimal dual solution and optimal primal feasible solution.

The issue in our setting is that the dual iterates in the Dual-Ascent Method do not correspond to forests of balanced arcs. This was noted in Remark 2, that arcs can become "unbalanced" as the method proceeds. This highlights a difference between our Dual-Ascent Method and the typical methods in the finite-dimensional setting. We carefully leverage the pure supply structure of the underlying network to avoid running max-flow calculations in each iteration (as, for instance, is done in Chapter 7 of Bertsimas and Tsitsiklis (1997)). We adapt the method to maintain as much of its underlying simplicity as possible.

Accordingly, we adapt the method not to maintain a forest of balanced arcs (as in the finite-dimensional case), but instead of growing set of balanced *nodes*, defined as follows.

**Definition 3.** A node $i$ is balanced with respect to $\pi \in \mathbb{R}^{\mathcal{N}}$ if there exists a $j \in O(i)$ such that $(i, j)$ is balanced with respect to $\pi$. That is, a node is balanced if it has an outgoing arc that is balanced.

The updating step in the Dual-Ascent method has the potential to unbalance arcs (and thus possibly unbalance nodes). So in our revised algorithm (called the Primal-Dual Method) we add a "rebalancing step" that balances (potentially different) arcs to assure that every previously balanced node remains balanced. Accordingly, the set of balanced nodes grows monotonically in size with every iterations of the algorithm, while the set of balanced arcs typically loses arcs and gains others as the algorithm proceeds.

In the limit, we will show that every node becomes balanced, and so a similar conclusion to the finite case can be drawn. Namely, there is a limiting dual solution that corresponds to a limiting primal feasible solution where every arc in the underlying spanning tree is balanced.

We now look more carefully at the two major differences between the Dual-Ascent Method and the Primal-Dual Method: Step 5 (the rebalancing step) and Step 6 (the step that will generate primal-feasible iterates). The rebalancing step is essential as it allows us to grow the set of balanced nodes. Before establishing this, we first point out that the algorithm does not get caught in the while loop in Step 5.

**Lemma 9.** In every iteration of the Primal-Dual Method, the while loop in Step 5 finitely terminates.

*Proof.* It suffices to prove that the set $\{i \in B^n : s_i^n > 0\}$ eventually becomes empty. Suppose $i'$ and $j'$ are chosen as in (23) and (24) for one iteration of the while. Setting $s_{i'j'}^n$ to 0 in step (26) removed node $i'$ from the set $\{i \in B^n : s_i^n > 0\}$ at the end the end of the iteration of the while. Next, we argue that this node $i'$ never returns to the set $\{i \in B^n : s_i^n > 0\}$.

The only way that $i'$ re-enters the set $\{i \in B^n : s_i^n > 0\}$ is if $\pi_{j'}^m$ is changed from its value $\pi_{j'}^n$ at some later iteration $m$. However, by definition of $i'$, we know $s_j^n = 0$ for all $j \in B^n$ with $j > i$ (by the definition of $i'$ as the maximum element of the set $\{i \in B^n : s_i^n > 0\}$). This implies that $\pi_{j'}$ will not be adjusted in the course of the rebalancing step, since only tail nodes of arcs have their $\pi$ value adjusted, and all arcs $(j', k)$ with $j'$ as their tail node have $s_{j'k}^n = 0$ and this slack will not be adjusted since the same property holds for the nodes $k$.

Hence, a node is removed from $\{i \in B^n : s_i^n > 0\}$ at every iteration of the while, and since $B^n$ is initially a finite set and no nodes are added to $B^n$ in the rebalancing step, this implies that $\{i \in B^n : s_i^n > 0\}$ is eventually empty, breaking the while loop. □

**Remark 6.** Implicit in the argument above is that the only nodes that be rebalanced are predecessor nodes of the node $i^n$ that is balanced in Step 4 of the Primal-Dual Method. How such nodes can become unbalanced was explored in concrete terms in Remark 2. ◁

As the algorithm proceeds, $n$ is successively iterated, and according to (3), the set $B^n$ grows with every iteration. We now show that the set $B^n$ contains only balanced nodes. This explains the naming of Steps 4 and 5 as balancing and rebalancing steps, respectively.

**Proposition 7.** Every node in $B^n$ after the rebalancing step (Step 4 of the Primal-Dual Method) is balanced with respect to $\pi^n$.

*Proof.* The proof is by induction. This property is true for $B^0$ by construction since every node $i$ in $B^0$ has an arc $j \in O(i)$ such that $c_{ij} = 0$ and so

$$\pi_i^0 - \pi_j^0 = c_{ij}$$

since both sides are equal to 0. For the inductive step, note that in (21), the node $i^n$ that is added to $B^n$ is balanced by the nature of the updating in (18). Indeed, arc $(i^n, j^n)$ is balanced in step (18) (and, in particular $s_{i^n j^n}^n$ is set to 0 in (19)) and so $i^n$ is balanced when added to $B^n$ in (21).

It only remains to argue that the nodes in $B^n$ added at an earlier stage remain balanced with respect to the new value $\pi^n$. For a node $i \in B^n$, if $s_i^n$ is unchanged in (20), then it remains at $s_i^n = 0$ and so node $i$ remains balanced. If, however, $s_i^n > 0$ after step (20), this is addressed in the rebalancing step. Once this node is processed as $i'$ in (23) then after step (27) we have $s_i^n = 0$ and so node $i$ is again balanced. As argued in Lemma 9, the while loop terminates with $s_i^n = 0$ for all $i \in B^n$. That is, all nodes in $B^n$ remain balanced with respect to $\pi^n$ at end of the rebalancing step. □

Next, we show that Step 6 generates a sequence of primal feasible flows.

**Proposition 8.** For all $n$, the iterates $x^n$ of the Primal-Dual Method are feasible to (CINF).

*Proof.* We first show that $T^n$ contains exactly one outgoing arc for every node $i \in \mathcal{N}$. We establish this by induction. The base case for $T^0$ is true by construction. For the inductive hypothesis, note that $T^n$ is updated in (22) and revised in (28). By the inductive hypothesis, $T^{n-1}$ has a single outgoing arc for every node $i \in \mathcal{N}$ and so in (22) the set $\left\{(i,j) \in T^{n-1} : i = i^n\right\}$ is a singleton. Thus, the update takes out one outgoing arc $(i^n, j)$ of $i^n$ from $T^{n-1}$ and replaces it with another outgoing arc $(i^n, j^n)$ of $i^n$. Hence, $T^n$ maintains a single outgoing arc for every node after (22). By the same logic, this property is maintained every time (28) is invoked in the rebalancing step. Thus, in (29), $T^n$ always contains a single outgoing arc for every node, and hence $x^n$ is a basic feasible flow by Lemmas 2 and 3. $\square$

The proof of dual feasibility mimics that of the Dual-Ascent Method, applying the logic of the balancing step to the rebalancing step. Details are omitted.

**Proposition 9** (cf. Proposition 4)**.** For all $n$, the iterates $\pi^n$ of the Primal-Dual Method are feasible to (CINFD).

The previous results explain some of the logic of the Primal-Dual Method. It works to build balanced nodes and generates a sequence of both primal and dual feasible solutions. The concepts come together in establishing a bound on the gap in value between the primal and dual iterates.

**Theorem 3.** For every iteration $n$, $Z(x^n) - D(\pi^n) \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$, where $T_B^n$ denotes the arcs in $T^n$ that are balanced with respect to $\pi^n$.

*Proof.* This proof uses the following notation. As $x^n$ is constructed by Procedure 2 we may define $P_{i\infty}$ as the unique directed path to infinity from node $i$ in the spanning tree $T^n$. Using an analog of Lemma 5, which applies equally to the $\pi^n$ generated by the Primal-Dual Method, let $T_i^n$ denote the finite out-tree of node $i$ whose cost is represented by $\pi_i^n$. Note that the set of arcs in $T_i^n$ must contain the arcs $P_i^n$ in $P_{i\infty}$ with tail nodes in $B^n$. This is because every arc added to $T^n$ in (22) and (28) that is on a directed path from node $i$ is captured in $T_i^n$.

Using this notation we observe that (with careful justification below):

$$\begin{aligned}
Z(x^n) - D(\pi^n) &= \sum_{i\in\mathcal{N}} c(P_{i\infty}) - \sum_{i\in\mathcal{N}} b_i \pi_i^n && (30)\\
&= \sum_{i\in\mathcal{N}} b_i(c(P_{i\infty}) - \pi_i^n) && (31)\\
&= \sum_{i\in\mathcal{N}} b_i(c(P_{i\infty}) - c(T_i^n)) && (32)\\
&\leq \sum_{i\in\mathcal{N}} b_i(c(P_{i\infty}) - c(P_i^n)) && (33)\\
&= \sum_{i\in\mathcal{N}} b_i c(P_{i\infty} \setminus P_i^n) && (34)\\
&= \sum_{(i,j)\in T^n \setminus T_B^n} c_{ij} x_{ij}^n. && (35)
\end{aligned}$$

Step (30) uses Procedure 2. Step uses the definition of $T_i^n$. Since the set of arcs in $T_i^n$ contain the arcs $P_i^n$ in $P_{i\infty}$ with tail nodes in $B^n$, this justifies dropping the costs of arcs in $T_i^n$ not in $P_i^n$. The direction of the inequality follows since the costs of all arcs are nonnegative. Step (35) uses the fact that $P_i^n$ is a subset of $P_{i\infty}$ for all $i$. Finally, step (35) uses the understanding from Procedure 2 about how $x^n$ is constructed and the fact that since there are no demand nodes, *all* flow $b_i$ originating from node $i$ must traverse the arcs in $P_{i\infty} \setminus P_i^n$. $\square$

An immediate consequence of Propositions 8 and 9 and Theorem 3 and weak duality (Theorem 1) is the following non-asymptotic error bounds on the quality of the primal and dual feasible solutions generated by the Primal-Dual Method.

**Corollary 1.** For every iteration $n$, $Z(x^n) - Z^* \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ and $D^* - D(\pi^n) \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$.

This is a non-asymptotic bound on the performance of both the primal method and dual ascent method. Moreover, Theorem 3 yields asymptotic convergence results for the Primal-Dual Method. Observe that the optimality error bound value $\sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ depends on how many nodes are balanced, and thus how many arcs are in $T^n \setminus T_B^n$. The next two results show that all nodes will eventually be balanced.

**Lemma 10** (Slacks tend to zero, cf. Lemma 8)**.** The $s_i^n$ updated in (20) and (27) of the Primal-Dual Method satisfy $\lim_{n\to\infty} s_i^n \to 0$ for all $i \in \mathcal{N}$.

The proof of this lemma is analogous to that of Lemma 8, adapted to the primal-dual setting, and thus omitted.

**Proposition 10.** All nodes are eventually balanced. That is, for all $i \in \mathcal{N}$ there exists an $n_i$ such that node $i$ is balanced with respect to $\pi^n$ for all $n \geq n_i$.

*Proof.* Suppose, by way of contradiction, there exists an $i \notin \mathcal{B}$ and thus $i \notin B_n$ for all $n \geq 0$. This implies, in particular since $i \notin B_0$, that

$$\min_{j \in O(i)} c_{ij} > 0. \tag{36}$$

Also, for all $n$ and $k \in O(i)$ we have

$$s_{ik}^n = c_{jk} + \pi_j^n - \pi_i^n \geq c_{ij} \tag{37}$$

since $i$ is not in $B_n$ and $\pi_j^n \geq 0$ for all $j \in O(i)$. Thus, from (36) and (37) we have

$$\min_{k \in O(i)} s_{ij}^n \geq \min_{j \in O(i)} c_{ij} > 0,$$

and so $\lim_{n\to\infty} s_i^n > 0$. This contradicts Lemma 10, thus completing the proof. □

**Lemma 11.** The optimality error bound $\sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ from Theorem 3 (and Corollary 1) converges to 0 as $n \to \infty$.

*Proof.* Since, by Proposition 10, all nodes are eventually balanced, there exists an $n_s$ such that all nodes in the first $s-1$ stages are in $B_{n_s}$. This implies that $T^n \setminus T_B^n$ can only contain arcs with tail nodes in stage $s$ or later for $n \geq n_s$. Thus, the cost of every arc in $T^n \setminus T_B^n$ can be bounded by $\gamma\beta^s$. That is,

$$\begin{aligned}\sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n &\leq \sum_{k=s}^{\infty} \sum_{i \in S_k} \sum_{j \in O(i)} c_{ij} x_{ij} \\ &\leq ||b||_\infty \gamma \sum_{k=s}^{\infty} \beta^k \sum_{j=0}^{k} g(j) \end{aligned} \tag{38}$$

for $n \geq n_s$, where the second inequality uses assumptions (A7)–(A10). Taking $n$ sufficiently large, we can send $s \to \infty$ which sends the right-hand side (38) to zero under assumption (A10). This gives the result. □

**Theorem 4.** The iterates $\pi^n$ and $x^n$ converge in value to $D^*$ and $Z^*$. That is, $D(\pi^n) \to D^*$ and $Z(x^n) \to Z^*$.

*Proof.* This is immediate from Corollary 1 and Lemma 11. □

In Theorem 2, the stronger result of the convergence of $\pi^n$ to an optimal solution $\pi^*$, and not just convergence in value, was established. On the primal side, we cannot guarantee solution convergence. The reason is that there is a choice of arcs in the spanning tree sets $T^n$, which may not converge to a unique spanning tree in the limit.

However, we can also guarantee finite implementability of the PRIMAL-DUAL METHOD, as was argued in Proposition 6 for the DUAL-ASCENT METHOD. The same result holds for the PRIMAL-DUAL METHOD. Here both the primal and dual solutions need only to be constructed on a finite subgraph corresponding to nodes in $B^n$, since as Lemma 11 showed, costs of arcs outside of $T_B^n$ can be made negligible. It should be noted that we cannot guarantee *finite convergence* of the overall algorithm, the termination condition in step 3 will, in general, be unreachable in finite time.

Returning to the deterministic DP problem, as established in Proposition 3, it corresponds to a pure supply network flow problem and so the DUAL-ASCENT METHOD and PRIMAL-DUAL METHOD both apply. To make this concrete, we may write the dual of (2) as

$$\max_{\pi} \pi_{(s_0,0)} \tag{39a}$$

$$\text{s.t. } \pi_{(s,t)} - \pi_{(s',t+1)} \leq \delta^t c_{(s,t)(s',t+1)} \qquad \text{for all } t \geq 0 \text{ and } ((s,t),(s',t+1)) \in \mathcal{A} \tag{39b}$$

$$\pi \in c_0. \tag{39c}$$

To our knowledge, the DUAL-ASCENT METHOD and PRIMAL-DUAL METHOD are the first dual-based algorithms proposed to solve (2) and (39). These dual based methods directly provide computable bounds on the value error from optimal for the strategy provided by a separate primal algorithm or within the primal-dual algorithm for the $n$th iterate. These bounds would, in general, be superior to the crude bound heretofore provided by the maximum discounted cost-to-go as, for example, in traditional finite horizon approximations.

## 6 Conclusion

In this paper, we establish a dual-ascent and primal-dual method for deterministic DP. In fact, we develop the methodology in the more general class of pure supply network flow problems. We show that these methods have desirable properties, including finite implementability, value and solution convergence (in the case of the dual-ascent method).

We should remark that the dual-ascent method studied here is a simple one, where at each iteration only a single arc is balanced (or relaxed). More general dual-ascent methods (such as those described in Bertsimas and Tsitsiklis (1997) and Bertsekas (1991)) are considered in the finite network case. Iterations of our dual-ascent method most closely resembles a *single-node relaxation* (or coordinate ascent) approach proposed in Section 3.3 of Bertsekas (1991). Single-node relaxations have been shown to be computationally efficient in practice, but for general finite networks (such as those with both supply and demand nodes) such relaxations do not suffice to compute an optimal dual solution, they must be combined with more general relaxation steps. The reason we can confine ourselves to single node relaxations is the simple structure of the pure supply network.

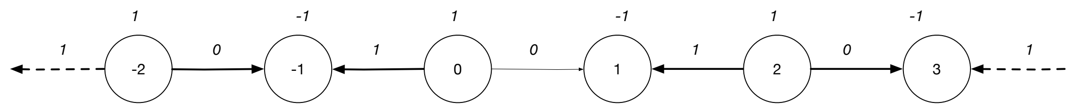

Figure 1: Illustration of a network with supply and demand nodes where a sequence of spanning trees fails to converge to a spanning tree.

For future work, one could explore analogs of more general dual-ascent methods from the finite-network settings and adapt them to the infinite setting. However, this path is fraught with potential difficulties, as demonstrated in the following example.

**Example 1.** Consider the network in Figure 1 with $b_i$ indicated next to each $i$: odd-numbered nodes have supply 1 and even-numbered nodes have supply $-1$, i.e., demand 1. Any spanning tree in this network consists of all arcs except for one. Bold arcs in the figure depict one such tree (arc (0, 1) is excluded), and values next to the arcs form the corresponding basic flow. This is the unique feasible flow in this network; note that it is degenerate and a tree constructed by omitting the arc $(i, i+1)$ for any even $i$ has the same corresponding basic feasible flow.

Consider the sequence of spanning trees that exclude the arcs $(0, 1)$, $(2, 3)$, $(4, 5)$, etc. The limit of this sequence (in the product discrete topology defined in (Ryan et al., 2018)) is the entire network, which is not a spanning tree.

One reason this example is problematic is as follows. In the pure supply setting, the trees $T^n$ always correspond to spanning trees due to Lemma 3. Moreover, the limit tree $T^*$ (studied in the proof of Theorem 2) is also a spanning tree because of Lemma 3. The above example shows that this need not be true for the mixed supply and demand case. The dual-ascent and primal-dual methods in this paper strongly leverage the properties of the pure supply network (and particularly, Lemma 3) and so careful thought must be put into adapting to more general settings.

## Acknowledgements

We thank Marina A. Epelman for several enlightening discussions that helped improve the paper. The first author thanks the Chicago Booth of Business for its generous financial research support. The second author is supported in part by the National Science Foundation Grant CMMI-1333260.

## A Proof of Proposition 1

We begin with some preliminary lemmas. To state the lemmas we need the notion of the *characteristic vector* $\chi^S \in \{0, 1\}^{\mathcal{A}}$ of a subset of arcs $S \subseteq \mathcal{A}$ that is defined by $\chi^S_{(s,t)(s',t+1)} = 1$ if $((s, t), (s', t+1)) \in S$ and 0 if $((s, t), (s', t+1)) \in \mathcal{A} \setminus S$. Note that $Z(\chi^S) = \sum_{((s,t),(s',t+1))\in S} \delta^t c_{(s,t)(s',t+1)}$ is the sum of the discounted costs of the arcs in $S$.

**Lemma 12.** The following hold:

(i) Every integral solution $x$ to (2) is the characteristic vector of a path $P_x$ from $(s_0, 0)$ to infinity and so $Z(x) = Z(\chi^{P_x})$.

(ii) Conversely, every path $P$ from $(s_0, 0)$ to infinity corresponds to an integral feasible solution $x_P$ to (2), where $Z(\chi^P) = Z(x_P)$.

*Proof.* To establish (i) note that (2b), (3b), and integrality imply that the arcs with flow 1 for any integral solution $x$ precisely form a path $P$ from $(s_0, 0)$ to infinity. Thus, $x = \chi_P$. For (ii), observe that the characteristic vector of a path from $(s_0, 0)$ satisfies constraints (2b) and (3b). The result then follows. $\square$

**Lemma 13.** The following hold:

(i) Every policy $d$ to (1) gives rise to a unique path $P_d$ from $(s_0, 0)$ to infinity such that $V(\pi(d)_{s_0}) = Z(\chi^{P_d})$, where $\chi^{P_d}$ is the characteristic vector of a path $P_d$ from $(s_0, 0)$ to infinity.

(ii) Conversely, every path $P$ from $(s_0, 0)$ to infinity consisting of arcs $(s_0, 0), (s_1, 1), (s_2, 2), \ldots$ corresponds to a set of policies $D_P$ where each policy $d$ in that set satisfies:

$$d(s,t) \begin{cases} = a \in A_t \text{ such that } \tau(s_t, a) = s_{t+1} & \text{if } s = s_t \\ \in A_t & \text{if } s \in S_t \text{ and } s \neq s_t \end{cases} \tag{40}$$

and $V(\pi(d), s_0) = Z(\chi^P)$ for every $d \in D_P$.

*Proof.* To establish (i) note that given a policy $d$ and starting with initial state $s_0$ at time 0, the policy determines the unique path $P_d$ consisting of arcs $(s_0, 0), (s_1, 1), (s_2, 2), \ldots$ where $s_{t+1} = \tau(s_t, d(s_t, t))$ for $t = 0, 1, 2, \ldots$. It is straightforward to see that $V(\pi(d), s_0) = Z(\chi^{P_d})$.

For (ii), clearly each $d$ constructed in (40) has $s_{t+1} = \tau(s_t, d(s_t, t))$ for $t = 0, 1, 2, \ldots$. This is all the information needed to specific the cost $V(\pi(d), s_0)$, as the choice of $d(s,t)$ for $s \in S_t$ with $s \neq s_t$ is irrelevant to the outcome of the dynamic program. Hence, $V(\pi(d), s_0) = Z(\chi^P)$ for every $d \in D_P$. $\square$

The following result is standard in finite-horizon versions of the problem. The argument here relies on a couple of useful results in the CILP literature.

**Lemma 14.** Problem (2) possesses an optimal flow that is integer-valued.

*Proof.* Recall Proposition 2.7 of (Ghate et al., 2010): a CILP has an optimal extreme point solution if

(a) the feasible region is non-empty,

(b) every constraint has a finite number of variables,

(c) the feasible region is bounded by some vector $u$; that is, $x_{(s,t)(s',t+1)} \leq u_{(s,t)(s',t+1)}$ for all $((s,t),(s',t+1)) \in \mathcal{A}$ for as feasible $x$, and

(d) $\sum_{((s,t),(s',t+1))\in\mathcal{A}} \delta^t c_{(s,t)(s',t+1)} u_{(s,t)(s',t+1)} < \infty$.

Condition (a) is clearly satisfied since every path from $(s_0, 0)$ to infinity corresponds to a feasible solution by Lemma 12(ii). Condition (b) follows by assumption (A1). As for (c), the vector $u_{(s,t)(s',t+1)} = 1$ for all $((s,t),(s',t+1)) \in \mathcal{A}$ suffices. Indeed, the only source of flow in the network is from node $(s_0, 0)$ with a supply of 1, hence no arc will ever have a flow larger than 1 (note that we do not impose this upper bound of flow explicitly, but it is implied by the data). Thus, (c) holds. As for (d) this follows since

$$\sum_{((s,t),(s',t+1))\in\mathcal{A}} \delta^t c_{(s,t)(s',t+1)} u_{(s,t)(s',t+1)} \leq \sum_{t=0}^{\infty} \delta^t g(t) \cdot \bar{c} \cdot 1 = \bar{c}\tfrac{1}{1-\delta} < \infty,$$

where the inequality follows from (S1) and (S2). Thus, (2) has an optimal extreme point solution. Next, by Theorem 3.14 in (Romeijn et al., 2006), every extreme point solution is integral, since the right-hand sides of the constraints (2b)–(3b) are integral. Taken together, this implies (2) has an integer solution that is optimal. □

*Proof of Proposition 1.* Lemmas 12 and 13 imply that (a) every feasible policy $d$ to (1) gives rise to an integer flow $x_d$ of (2) where $V(\pi(d), s_0) = Z(x_d)$ and (b) every integer flow $x$ of (2) gives rise to policies $d_x$ that obey (40) where $P_x$ is the path associated with integer flow $x$ described in Lemma 12(i) and $Z(x) = V(\pi(d_x), s_0)$ for any such policy.

We now establish (i). By (a), $d^*$ has a corresponding integer flow $x^*$ with $V(\pi(d^*), s_0) = Z(x^*)$. Next, by (b), every integer flow $x$ of (2) gives rise to a policy $d_x$ such that $V(\pi(d_x), s_0) = Z(x)$. Taken together this implies that

$$Z(x^*) = V(\pi(d^*), s_0) \leq V(\pi(d_x), s_0) = Z(x),$$

where the inequality uses optimality of the policy $d^*$ in (1). Thus, $x^*$ has the minimum cost among all integer flows to (2). By Lemma 14, this implies that $x^*$ is an optimal flow for (2). This completes (i).

Returning to (ii), there exists an optimal *integer flow* $x^*$ to (2). Let $d^*$ be any policy that obeys (40) where $P = P_{x^*}$ with $V(\pi(d^*), s_0) = Z(x^*)$. Every feasible policy $d$ corresponds to an integer vector $x_d$ with $V(\pi(d), s_0) = Z(x_d)$ and so

$$V(\pi(d), s_0) = Z(x_d) \geq Z(x^*) = V(\pi(d^*), s_0),$$

where the inequality follows by the optimality of $x^*$ in (2). Thus, $d^*$ is an optimal policy since its value no greater than that of any other feasible policy. □

## References

Jeffrey M. Alden and Robert L. Smith. Rolling horizon procedures in nonhomogeneous Markov decision processes. *Operations Research*, 40(3-supplement-2):S183–S194, 1992.

Edward J. Anderson and Peter Nash. *Linear Programming in Infinite-Dimensional Spaces: Theory and Applications.* Wiley, 1987.

Edward J Anderson and Andrew B Philpott. A continuous-time network simplex algorithm. *Networks*, 19(4):395–425, 1989.

James C. Bean and Robert L. Smith. Conditions for the existence of planning horizons. *Mathematics of Operations Research*, 9(3):391–401, 1984.

Dimitri P Bertsekas. *Linear Network Optimization: Algorithms and Codes.* MIT Press, 1991.

Dimistris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena, 1997.

Christian Bès and Suresh P. Sethi. Concepts of forecast and decision horizons: Applications to dynamic stochastic optimization problems. *Mathematics of Operations Research*, 13(2):295–310, 1988.

Abraham Charnes, Jacques Dreze, and Merton Miller. Decision and horizon rules for stochastic planning problems: A linear example. *Econometrica*, pages 307–330, 1966.

Eric V. Denardo. *Dynamic Programming: Models and Applications.* Dover, 2003.

Eugene A. Feinberg and Adam Shwartz. *Handbook of Markov Decision Processes: Methods and Applications.* Kluwer, 2002.

Archis Ghate. Circumventing the Slater conundrum in countably infinite linear programs. *European Journal of Operations Research*, 246(3):708–720, 2015.

Archis Ghate and Robert L. Smith. Characterizing extreme points as basic feasible solutions in infinite linear programs. *Operations Research Letters*, 37(1):7–10, 2009.

Archis Ghate and Robert L. Smith. A linear programming approach to non stationary infinite-horizon Markov decision processes. *Operations Research*, 61:413–425, 2013.

Archis Ghate, Dushyant Sharma, and Robert L. Smith. A shadow simplex method for infinite linear programs. *Operations Resesearch*, 58(4):865–877, 2010.

Onesimo Hernández-Lerma and Jean B. Lasserre. *Discrete-time Markov Control Processes: Basic Optimality Criteria*, volume 30. Springer, 2012.

Wallace J. Hopp, James C. Bean, and Robert L. Smith. A new optimality criterion for nonhomogeneous Markov decision processes. *Operations Research*, 35(6):875–883, 1987.

Diego Klabjan and Daniel Adelman. Existence of optimal policies for semi-Markov decision processes using duality for infinite linear programming. *SIAM Journal on Control and Optimization*, 2006.

Diego Klabjan and Daniel Adelman. An infinite-dimensional linear programming algorithm for deterministic semi-Markov decision processes on borel spaces. *Mathematics of Operations Research*, 32(3):528–550, 2007.

Ilbin Lee, Marina A. Epelman, H. Edwin Romeijn, and Robert L. Smith. Simplex algorithm for countable-state discounted Markov decision processes. *Operations Research*, 65(4):1029–1042, 2017.

Franco Modigliani and Franz E. Hohn. Production planning over time and the nature of the expectation and planning horizon. *Econometrica*, pages 46–66, 1955.

Sevnaz Nourollahi and Archis Ghate. Duality in convex minimum cost flow problems on infinite networks and hypernetworks. *Networks*, 70(2):98–115, 2017.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, 2014.

H. Edwin Romeijn, Robert L. Smith, and James C. Bean. Duality in infinite dimensional linear programming. *Mathematical Programming*, 53(1-3):79–97, 1992.

H. Edwin Romeijn, Dushyant Sharma, and Robert L. Smith. Extreme point characterizations for infinite network flow problems. *Networks*, 48(4):209–22, 2006.

Sheldon M. Ross. *Stochastic Processes.* Wiley, 1996.

Christopher Thomas Ryan, Robert L. Smith, and Marina A. Epelman. A simplex method for uncapacitated pure-supply infinite network flow problems. *To appear in SIAM Journal on Optimization*, 2018.

Sarah M. Ryan, James C. Bean, and Robert L. Smith. A tie-breaking rule for discrete infinite horizon optimization. *Operations Research*, 40(1-supplement-1):S117–S126, 1992.

Irwin E. Schochetman and Robert L. Smith. Convergence of selections with applications in optimization. *Journal of Mathematical Analysis and Applications*, 155(1):278–292, 1991.

Thomas C. Sharkey and H. Edwin Romeijn. A simplex algorithm for minimum-cost network-flow problems in infinite networks. *Networks*, 52(1):14–31, 2008.