Approximate Dynamic Programming Methods for an Inventory Allocation Problem under Uncertainty

Huseyin Topaloglu∗†, Sumit Kunnumkal∗

September 7, 2005

Abstract

In this paper, we propose two approximate dynamic programming methods to optimize the distribution operations of a company manufacturing a certain product at multiple production plants and shipping it to different customer locations for sale. We begin by formulating the problem as a dynamic program. Our first approximate dynamic programming method uses a linear approximation of the value function, and computes the parameters of this approximation by using the linear programming representation of the dynamic program. Our second method relaxes the constraints that link the decisions for different production plants. Consequently, the dynamic program decomposes by the production plants. Computational experiments show that the proposed methods are computationally attractive, and in particular, the second method performs significantly better than standard benchmarks.

Supply chain systems with multiple production plants provide protection against demand uncertainty and opportunities for production smoothing by allowing the demand at a particular customer location to be satisfied through different production plants. However, managing these types of supply chains requires careful planning. When planning the distribution of products to the customer locations, one has to consider many factors, such as the current inventory levels, forecasts of future production quantities and forecasts of customer demands. The decisions for different production plants interact and a decision that maximizes the immediate benefit is not necessarily the “best” decision.

In this paper, we consider the distribution operations of a company manufacturing a certain product at multiple production plants and shipping it to different customer locations for sale. A certain amount of production occurs at the production plants at the beginning of each time period. Before observing the realization of the demand at the customer locations, the company decides how much product should be shipped to the customer locations and how much product should be stored at the production plants. Once a certain amount of product is shipped to a particular customer location, revenue is earned on the sales and shortage cost is incurred on the unsatisfied demand. The left over product is disposed at a salvage value.

∗School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853,
†topaloglu@orie.cornell.edu.
Our work is motivated by the distribution operations of a company processing fresh produce that will eventually be sold at local markets. These markets are set up outdoors for short periods of time, prohibiting the storage of the perishable product. However, the processing plants are equipped with storage facilities. Depending on the supply of fresh produce, the production quantities at the processing plants fluctuate over time and are not necessarily deterministic.

In this paper, we formulate the problem as a dynamic program and propose two approximate dynamic programming methods. The first method uses a linear approximation of the value function whose parameters are computed by using the linear programming representation of the dynamic program. The second method uses Lagrangian relaxation to relax the constraints that link the decisions for different production plants. As a result of this relaxation, the dynamic program decomposes by the production plants and we concentrate on one production plant at a time.

Our approach builds on previous research. Hawkins (2003) proposes a Lagrangian relaxation method for a certain class of dynamic programs in which the evolutions of the different components of the state variable are affected by different types of decisions, but these different types of decisions interact through a set of linking constraints. More recently, Adelman & Mersereau (2004) compare the Lagrangian relaxation method of Hawkins (2003) with an approximate dynamic programming method that uses a separable approximation of the value function. The parameters of the separable approximation are computed by using the linear programming representation of the dynamic program. When applied to the inventory allocation problem described above, both of these methods run into computational difficulties. For example, the Lagrangian relaxation method of Hawkins (2003) requires finding a “good” set of Lagrange multipliers by minimizing the so-called dual function. One way of doing this is to solve a linear program, but the number of constraints in this linear program can easily be on the order of $10^{30}$ for our problem class. We use constraint generation to iteratively construct the constraints of this linear program, and show that this can be done efficiently since constructing each constraint requires simple sorting operations. Another way of finding a “good” set of Lagrange multipliers is to represent the dual function by using cutting planes, but the number of cutting planes needed for this purpose can grow large. We show that we can keep the number of cutting planes at a manageable level by using results from the two-stage stochastic programming literature and that constructing each cutting plane also requires simple sorting operations. The approximate dynamic programming method of Adelman & Mersereau (2004) computes the parameters of the separable value function approximation by solving a linear program whose number of constraints can easily be on the order of $10^{300}$ for our problem class. We use constraint generation to iteratively construct the constraints of this linear program, and show that constructing each constraint requires solving a min-cost network flow problem. Finally, we show that the value function approximations
obtained by the two methods are computationally attractive in the sense that applying the greedy policies characterized by them requires solving min-cost network flow problems.

The approximate dynamic programming field has been active within the past two decades. Most of the work in this field attempts to approximate the value function \( V(\cdot) \) by a function of the form
\[
\sum_{k \in K} r_k V_k(\cdot),
\]
where \( \{V_k(\cdot) : k \in K\} \) are fixed basis functions and \( \{r_k : k \in K\} \) are adjustable parameters. The challenge is to find parameter values \( \{\tilde{r}_k : k \in K\} \) such that \( \sum_{k \in K} \tilde{r}_k V_k(\cdot) \) is a “good” approximation of \( V(\cdot) \). Temporal differences and Q-learning use sampled trajectories of the system to find “good” parameter values (Bertsekas & Tsitsiklis (1996)). On the other hand, linear programming-based methods find “good” parameter values by solving a linear program (Schweitzer & Seidmann (1985), de Farias & Van Roy (2003)). Since this linear program contains one constraint for every state-decision pair, it can be very large and is usually solved approximately. In addition to advances in theory, numerous successful applications of approximate dynamic programming appeared in inventory routing (Adelman (2004), Kleywegt, Nori & Savelsbergh (2002)), dynamic fleet management (Godfrey & Powell (2002), Powell & Topaloglu (2003), Topaloglu & Powell (to appear)), marketing (Bertsimas & Mersereau (2005)) and resource allocation under incomplete information (Yost & Washburn (2000)). Of particular interest are the papers by Bertsimas & Mersereau (2005), Yost & Washburn (2000) and Adelman (2004). The first two of these papers are applications of the Lagrangian relaxation method of Hawkins (2003) and the third one is an application of the linear programming-based method of Adelman & Mersereau (2004).


In this paper, we make the following research contributions. We propose two approximate dynamic programming methods for a stochastic nonstationary multiple-plant multiple-customer inventory allocation problem. Our first method uses a linear approximation of the value function and computes the parameters of this approximation by using the linear programming representation of the dynamic program. We show how to solve this linear program efficiently by using constraint generation. This is one of the few nontrivial linear programming-based approximate dynamic programming methods where the underlying linear program can be solved efficiently. Our second method
uses Lagrangian relaxation to relax the constraints that link the decisions for different production plants. We propose two approaches for minimizing the dual function, one of which is based on constraint generation and the other one is based on representing the dual function by using cutting planes. We show that constructing a constraint or a cutting plane requires simple sorting operations. Finally, we show that applying the greedy policies that are characterized by the value function approximations obtained by these two methods requires solving min-cost network flow problems, which can be done efficiently. Computational experiments show that our methods yield high quality solutions. They also provide insights into the conditions that render stochastic models more effective than deterministic ones.

The organization of the paper is as follows. In Section 1, we describe the problem and formulate it as a dynamic program. Section 2 describes our first solution method that uses a linear approximation of the value function, whereas Section 3 describes our second solution method that uses Lagrangian relaxation. In Section 4, we show that applying the greedy policies that are characterized by the value function approximations obtained by these two solution methods requires solving min-cost network flow problems. Section 5 presents our computational experiments.

1. Problem Formulation

There is a set of plants producing a certain product to satisfy the demand occurring at a set of customer locations. At the beginning of each time period, a random amount of production occurs at each plant. (The production decisions are outside the scope of the problem and are simply modeled as random processes.) Before observing the demand at the customer locations, we have to decide how much product to ship from each plant to each customer location and how much product to hold at each plant. After shipping the product to the customer locations, we observe the demand. The unsatisfied demand is lost. The left over product at the customer locations is disposed at a salvage value, but the plants can store the product. Our objective is to maximize the total expected profit over a finite horizon. We define the following.

\[ T = \text{Set of time periods in the planning horizon. We have } T = \{1, \ldots, T\} \text{ for finite } T. \]

\[ P = \text{Set of plants.} \]

\[ C = \text{Set of customer locations.} \]

\[ c_{ijt} = \text{Cost of shipping one unit of product from plant } i \text{ to customer location } j \text{ at time period } t. \]

\[ \rho_{jt} = \text{Revenue per unit of product sold at customer location } j \text{ at time period } t. \]
\( \sigma_{jt} \) = Salvage value per unit of unsold product at customer location \( j \) at time period \( t \).

\( \pi_{jt} \) = Shortage cost of not being able to satisfy a unit of demand at customer location \( j \) at time period \( t \).

\( h_{it} \) = Holding cost per unit of product held at plant \( i \) at time period \( t \).

\( Q_{it} \) = Random variable representing the production at plant \( i \) at time period \( t \).

\( D_{jt} \) = Random variable representing the demand at customer location \( j \) at time period \( t \).

We assume that the production and demand occur in discrete quantities, and we are allowed to ship in discrete quantities. We also assume that the production quantities at different plants and at different time periods are independent. We define the following decision variables.

\( u_{ijt} \) = Amount of product shipped from plant \( i \) to customer location \( j \) at time period \( t \).

\( w_{jt} \) = Total amount of product shipped to customer location \( j \) at time period \( t \).

\( x_{it} \) = Amount of product held at plant \( i \) at time period \( t \).

\( r_{it} \) = Beginning inventory at plant \( i \) at time period \( t \) (after observing the production).

By suppressing one or more of the indices in the variables defined above, we denote a vector composed of the components ranging over the suppressed indices. For example, we have \( Q_{t} = \{Q_{it} : i \in \mathcal{P}\} \), \( r_{t} = \{r_{it} : i \in \mathcal{P}\} \). We do not distinguish between row and column vectors. We denote the cardinality of set \( \mathcal{A} \) by \(|\mathcal{A}|\). In the remainder of this section, we define the one-period expected profit function and formulate the problem as a dynamic program.

### 1.1. One-period expected profit function

If the amount of product shipped to customer location \( j \) at time period \( t \) is \( w_{jt} \) and the demand is \( D_{jt} \), then the profit we obtain is

\[
F_{jt}(w_{jt}, D_{jt}) = \rho_{jt} \min\{w_{jt}, D_{jt}\} + \sigma_{jt} \max\{w_{jt} - D_{jt}, 0\} - \pi_{jt} \max\{D_{jt} - w_{jt}, 0\}.
\]

Letting \( F_{jt}(w_{jt}) = \mathbb{E}\{F_{jt}(w_{jt}, D_{jt})\} \), \( F_{jt}(w_{jt}) \) is the expected profit obtained at time period \( t \) by shipping \( w_{jt} \) units of product to customer location \( j \). Since the random variable \( D_{jt} \) takes integer values, if we have \( \rho_{jt} + \pi_{jt} \geq \sigma_{jt} \), then we can show that \( F_{jt}(\cdot) \) is a piecewise-linear concave function with points nondifferentiability being a subset of positive integers. In this case, \( F_{jt}(\cdot) \) can be characterized by specifying \( F_{jt}(0) = -\pi_{jt} \mathbb{E}\{D_{jt}\} \) and the first differences \( F_{jt}(w_{jt} + 1) - F_{jt}(w_{jt}) \) for all \( w_{jt} = 0, 1, \ldots \). The latter can be computed by noting that

\[
F_{jt}(w_{jt} + 1, D_{jt}) - F_{jt}(w_{jt}, D_{jt}) = \begin{cases} 
\sigma_{jt} & \text{if } D_{jt} \leq w_{jt} \\
\rho_{jt} + \pi_{jt} & \text{otherwise},
\end{cases}
\]

\( L \) denotes a finite set of positive integers.
which implies that
\[
F_{jt}(w_{jt} + 1) - F_{jt}(w_{jt}) = \mathbb{E}\{F_{jt}(w_{jt} + 1, D_{jt}) - F_{jt}(w_{jt}, D_{jt})\}
\]
\[
= \sigma_{jt} \mathbb{P}\{D_{jt} \leq w_{jt}\} + [\rho_{jt} + \pi_{jt}] \mathbb{P}\{D_{jt} \geq w_{jt} + 1\}. \tag{1}
\]

We let \(f_{jst} = F_{jt}(s + 1) - F_{jt}(s)\) for all \(j \in \mathcal{C}, s = 0, 1, \ldots, t \in \mathcal{T}\). By (1), assuming that the random variables \(\{D_{jt} : j \in \mathcal{C}, t \in \mathcal{T}\}\) are bounded by \(S\), we have \(f_{jst} = \sigma_{jt}\) for all \(j \in \mathcal{C}, s = S, S + 1, \ldots, t \in \mathcal{T}\). Therefore, to characterize \(F_{jt}(\cdot)\), we only need to specify \(F_{jt}(0)\) and \(f_{jst}\) for all \(s = 0, \ldots, S\).

1.2. Dynamic programming formulation

The problem can be formulated as a dynamic program by using \(r_t\) as the state variable at time period \(t\). Assuming that we are not allowed to hold more than \(K\) units of product at any plant and that the random variables \(\{Q_{it} : i \in \mathcal{P}, t \in \mathcal{T}\}\) are bounded by \(B\), the inventory at any plant is bounded by \(K + B\). Therefore, letting \(R = K + B\) and \(\mathcal{R} = \{0, \ldots, R\}\), we use \(\mathcal{R}^{||\mathcal{P}|}\) as the state space. The optimal policy can be found by computing the value functions through the optimality equation

\[
V_t(r_t) = \max \left( - \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} c_{ij} y_{ij} + \sum_{j \in \mathcal{C}} F_{jt}(w_{jt}) - \sum_{j \in \mathcal{C}} F_{jt}(0) - \sum_{i \in \mathcal{P}} h_{it} x_{it} + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\} \right) \tag{2}
\]

subject to
\[
x_{it} + \sum_{j \in \mathcal{C}} y_{ij} = r_{it} \quad \forall \, i \in \mathcal{P} \tag{3}
\]
\[
\sum_{i \in \mathcal{P}} y_{ij} \leq 1 \quad \forall \, j \in \mathcal{C}, \, s = 0, \ldots, S - 1 \tag{9}
\]
\[
x_{it} \leq K \quad \forall \, i \in \mathcal{P} \tag{10}
\]
\[
x_{it}, y_{ij} \in \mathbb{Z}_+ \quad \forall \, i \in \mathcal{P}, \, j \in \mathcal{C}, \, s = 0, \ldots, S \tag{11}
\]

(The reason that we subtract the constant \(\sum_{j \in \mathcal{C}} F_{jt}(0)\) from the objective function will be clear in the proof of Lemma 1 below.) Constraints (3) are the product availability constraints at the plants, whereas constraints (4) keep track of the amount of product shipped to each customer location. The next lemma, which we prove in the appendix, gives an alternative representation of problem (2)-(6).

**Lemma 1.** *The optimality equation*

\[
V_t(r_t) = \max \left( \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} \sum_{s=0}^{S} \left[ f_{jst} - c_{ij} \right] y_{ijst} - \sum_{i \in \mathcal{P}} h_{it} x_{it} + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\} \right) \tag{7}
\]

subject to
\[
x_{it} + \sum_{j \in \mathcal{C}} \sum_{s=0}^{S} y_{ijst} = r_{it} \quad \forall \, i \in \mathcal{P} \tag{8}
\]
\[
\sum_{i \in \mathcal{P}} y_{ijst} \leq 1 \quad \forall \, j \in \mathcal{C}, \, s = 0, \ldots, S - 1 \tag{9}
\]
\[
x_{it} \leq K \quad \forall \, i \in \mathcal{P} \tag{10}
\]
\[
x_{it}, y_{ij} \in \mathbb{Z}_+ \quad \forall \, i \in \mathcal{P}, \, j \in \mathcal{C}, \, s = 0, \ldots, S \tag{11}
\]
is equivalent to the optimality equation in (2)-(6). In particular, if \((x_i^*, y_t^*)\) is an optimal solution to problem (7)-(11), and we let \(u_{ij}^* = \sum_{s=0}^{S} y_{ijst}^*\) and \(w_{jt}^* = \sum_{i \in \mathcal{P}} \sum_{s=0}^{S} y_{ijst}^*\) for all \(i \in \mathcal{P}, j \in \mathcal{C}\), then \((u_i^*, w_t^*, x_i^*)\) is an optimal solution to problem (2)-(6). Furthermore, the optimal objective values of problems (2)-(6) and (7)-(11) are equal to each other.

In the rest of the paper, we use the optimality equation in (7)-(11). For notational brevity, we let \(\mathcal{Y}(r_t)\) be the set of feasible solutions to problem (7)-(11) and

\[
p_t(x_t, y_t) = \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} \sum_{s=0}^{S} [f_{jst} - c_{ijt}] y_{ijst} - \sum_{i \in \mathcal{P}} h_{ist} x_{it}.
\]

In this case, problem (7)-(11) can be written as

\[
V_t(r_t) = \max_{(x_t, y_t) \in \mathcal{Y}(r_t)} \left\{ p_t(x_t, y_t) + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\} \right\}.
\] (12)

2. Linear Programming-Based Approximation

Associating positive weights \(\{\alpha(r_1) : r_1 \in \mathcal{R}^{[|\mathcal{P}|]}\}\) with the initial states, the value functions can be computed by solving the linear program

\[
\begin{align*}
\min & \quad \sum_{r_1 \in \mathcal{R}^{[|\mathcal{P}|]}} \alpha(r_1) v_1(r_1) \\
\text{subject to} & \quad v_t(r_t) \geq p_t(x_t, y_t) + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\} \\
& \quad \forall r_t \in \mathcal{R}^{[|\mathcal{P}|]}, (x_t, y_t) \in \mathcal{Y}(r_t), t \in T \setminus \{T\} \\
& \quad v_T(r_T) \geq p_T(x_T, y_T) \\
& \quad \forall r_T \in \mathcal{R}^{[|\mathcal{P}|]}, (x_T, y_T) \in \mathcal{Y}(r_T),
\end{align*}
\] (13)-(15)

where \(\{v_t(r_t) : r_t \in \mathcal{R}^{[|\mathcal{P}|]}, t \in T\}\) are the decision variables. Letting \(\{v_t^*(r_t) : r_t \in \mathcal{R}^{[|\mathcal{P}|]}, t \in T\}\) be an optimal solution to this problem, it is well-known that if \(\alpha(r_1) > 0\), then we have \(v_t^*(r_1) = V_t(r_1)\) (see Puterman (1994)). Problem (13)-(15) has \(T|\mathcal{R}^{[|\mathcal{P}|]}\) decision variables and \(\sum_{t \in T} \sum_{r_t \in \mathcal{R}^{[|\mathcal{P}|]}} |\mathcal{Y}(r_t)|\) constraints, which can both be quite large.

To deal with the large number of decision variables, we approximate \(V_t(r_t)\) with a linear function of the form \(\theta_t + \sum_{i \in \mathcal{P}} \nu_i r_{it}\). To decide what values to pick for \(\{\theta_t : t \in T\}\) and \(\{\nu_i : i \in \mathcal{P}, t \in T\}\), we substitute \(\theta_t + \sum_{i \in \mathcal{P}} \nu_i r_{it}\) for \(v_t(r_t)\) in problem (13)-(15) to obtain the linear program

\[
\begin{align*}
\min & \quad \sum_{r_1 \in \mathcal{R}^{[|\mathcal{P}|]}} \alpha(r_1) \theta_1 + \sum_{r_1 \in \mathcal{R}^{[|\mathcal{P}|]}} \sum_{i \in \mathcal{P}} \alpha(r_1) r_{i1} \nu_{i1} \\
\text{subject to} & \quad \theta_t + \sum_{i \in \mathcal{P}} r_{it} \nu_i \geq p_t(x_t, y_t) + \theta_{t+1} + \sum_{i \in \mathcal{P}} [x_{it} + \mathbb{E}\{Q_{i,t+1}\}] \nu_{i,t+1} \\
& \quad \forall r_t \in \mathcal{R}^{[|\mathcal{P}|]}, (x_t, y_t) \in \mathcal{Y}(r_t), t \in T \setminus \{T\} \\
& \quad \theta_T + \sum_{i \in \mathcal{P}} r_{iT} \nu_T \geq p_T(x_T, y_T) \\
& \quad \forall r_T \in \mathcal{R}^{[|\mathcal{P}|]}, (x_T, y_T) \in \mathcal{Y}(r_T),
\end{align*}
\] (16)-(18)
where \( \{ \theta_t : t \in T \} \), \( \{ \nu_{it} : i \in P, \ t \in T \} \) are the decision variables. The next proposition shows that we obtain upper bounds on the value functions by solving problem (16)-(18). A result similar to Proposition 2 (and Proposition 5 below) is shown in Adelman & Mersereau (2004) for infinite-horizon problems. However, our proof is for finite-horizon problems and tends to be simpler.

**Proposition 2.** If \( \{ \tilde{\theta}_t \ : t \in T \} \), \( \{ \tilde{\nu}_{it} \ : i \in P, \ t \in T \} \) is a feasible solution to problem (16)-(18), then we have \( V_t(r_t) \leq \tilde{\theta}_t + \sum_{i \in P} \tilde{\nu}_{it} r_{it} \) for all \( r_t \in \mathbb{R}^{|P|}, \ t \in T \).

**Proof.** We show the result by induction. It is easy to show the result for the last time period. Assuming that the result holds for time period \( t + 1 \) and using the fact that \( \{ \tilde{\theta}_t : t \in T \} \), \( \{ \tilde{\nu}_{it} : i \in P, \ t \in T \} \) is feasible to problem (16)-(18), we have

\[
\tilde{\theta}_t + \sum_{i \in P} \tilde{\nu}_{it} r_{it} \geq \max_{(x_t,y_t) \in \mathcal{Y}(r_t)} \left\{ p_t(x_t, y_t) + \mathbb{E}\left\{ \tilde{\theta}_{t+1} + \sum_{i \in P} \tilde{\nu}_{i,t+1} \left[x_{it} + Q_{i,t+1}\right]\right\}\right.
\]

\[
\geq \max_{(x_t,y_t) \in \mathcal{Y}(r_t)} \left\{ p_t(x_t, y_t) + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\}\right\} = V_t(r_t). \quad \Box
\]

Problem (16)-(18) has \( T + T|P| \) decision variables (which is manageable), but has as many constraints as problem (13)-(15). To deal with the large number of constraints, we use constraint generation. The idea is to iteratively solve a master problem, which has the same objective function and decision variables as problem (16)-(18), but has only a few of the constraints. After solving the master problem, we check if any of constraints (17)-(18) is violated by the solution. If there is one such constraint, then we add this constraint to the master problem and resolve the master problem.

In particular, letting \( \{ \tilde{\theta}_t : t \in T \}, \{ \tilde{\nu}_{it} : i \in P, \ t \in T \} \) be the solution to the current master problem, we solve the problem

\[
\max_{r_t \in \mathbb{R}^{|P|}} \left\{ \max_{(x_t,y_t) \in \mathcal{Y}(r_t)} \left\{ p_t(x_t, y_t) + \sum_{i \in P} \tilde{\nu}_{i,t+1} x_{it}\right\} - \sum_{i \in P} \tilde{\nu}_{it} r_{it}\right\} \tag{19}
\]

for all \( t \in T \setminus \{ T \} \) to check if any of constraints (17) is violated by this solution. Letting \( (\tilde{\tilde{\theta}}_t, \tilde{\tilde{x}}_t, \tilde{\tilde{y}}_t) \) be the optimal solution to problem (19), if we have \( p_t(\tilde{\tilde{x}}_t, \tilde{\tilde{y}}_t) + \sum_{i \in P} \tilde{\nu}_{i,t+1} \tilde{\tilde{x}}_{it} - \sum_{i \in P} \tilde{\nu}_{it} \tilde{\tilde{r}}_{it} > \tilde{\tilde{\theta}}_t - \tilde{\tilde{\theta}}_{t+1} - \sum_{i \in P} \mathbb{E}\{Q_{i,t+1}\} \tilde{\nu}_{i,t+1} \) then the constraint

\[
\theta_t + \sum_{i \in P} \tilde{\tilde{r}}_{it} \nu_{it} \geq p_t(\tilde{\tilde{x}}_t, \tilde{\tilde{y}}_t) + \theta_{t+1} + \sum_{i \in P} \left[\tilde{\tilde{x}}_{it} + \mathbb{E}\{Q_{i,t+1}\}\right] \nu_{i,t+1}
\]

is violated by the solution \( \{ \tilde{\theta}_t : t \in T \}, \{ \tilde{\nu}_{it} : i \in P, \ t \in T \} \). We add this constraint to the master problem. Similarly, we check if any of constraints (18) is violated by solving the problem

\[
\max_{r_t \in \mathbb{R}^{|P|}} \left\{ \max_{(x_T,y_T) \in \mathcal{Y}(r_T)} \left\{ p_T(x_T, y_T)\right\} - \sum_{i \in P} \tilde{\nu}_{iT} r_{iT}\right\}. \tag{20}
\]
Fortunately, problems (19) and (20) are min-cost network flow problems, and hence, constraint generation can be done efficiently. To see this, we write problem (19) as

\[
\begin{align*}
\max & \quad \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijt} \right] y_{ijst} + \sum_{i \in P} \left[ \nu_{i,t+1} - h_{it} \right] x_{it} - \sum_{i \in P} \tilde{\nu}_{it} r_{it} \\
\text{subject to} & \quad x_{it} + \sum_{j \in C} \sum_{s=0}^{S} y_{ijst} - r_{it} = 0 \quad \forall \ i \in P \\
& \quad r_{it} \leq R \quad \forall \ i \in P \\
& \quad r_{it}, x_{it}, y_{ijst} \in \mathbb{Z}_+ \quad \forall \ i \in P, \ j \in C, \ s = 0, \ldots, S \\
(9), (10).
\end{align*}
\]

Letting \( \{ \eta_{it} : i \in P \} \) be the slack variables for constraints (22), introducing the new decision variables \( \{ \delta_{jst} : j \in C, \ s = 0, \ldots, S - 1 \} \) and substituting \( \delta_{jst} \) for \( \sum_{i \in P} y_{ijst} \), the problem above becomes

\[
\begin{align*}
\max & \quad \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijt} \right] y_{ijst} + \sum_{i \in P} \left[ \nu_{i,t+1} - h_{it} \right] x_{it} - \sum_{i \in P} \tilde{\nu}_{it} r_{it} \\
\text{subject to} & \quad r_{it} + \eta_{it} = R \quad \forall \ i \in P \\
& \quad - \sum_{i \in P} y_{ijst} + \delta_{jst} = 0 \quad \forall \ j \in C, \ s = 0, \ldots, S - 1 \\
& \quad \delta_{jst} \leq 1 \quad \forall \ j \in C, \ s = 0, \ldots, S - 1 \\
& \quad r_{it}, x_{it}, y_{ijst}, \delta_{jst}, \eta_{it} \in \mathbb{Z}_+ \quad \forall \ i \in P, \ j \in C, \ s = 0, \ldots, S, \ s' = 0, \ldots, S - 1 \quad (27)
\end{align*}
\]

This problem is the min-cost network flow problem shown in Figure 1. Constraints (21), (24) and (25) in problem (23)-(28) are respectively the flow balance constraints for the white, gray and black nodes in this figure. We note that problem (20) is also a min-cost network flow problem, since it is a special case of problem (19).

3. Lagrangian Relaxation-Based Approximation

The idea behind this approximation method is to relax constraints (9) in problem (7)-(11). Associating positive Lagrange multipliers \( \{ \lambda_{jst} : j \in C, \ s = 0, \ldots, S - 1, \ t \in T \} \) with these constraints, this suggests solving the optimality equation

\[
\begin{align*}
V_i^\lambda(r_t) = & \quad \max \quad \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijt} \right] y_{ijst} - \sum_{i \in P} h_{it} x_{it} + \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} \left[ 1 - \sum_{i \in P} y_{ijst} \right] \\
& \quad + \mathbb{E} \{ V_{i+1}^\lambda(x_t + Q_{t+1}) \} \quad (29)
\end{align*}
\]

subject to \( y_{ijst} \leq 1 \quad \forall \ i \in P, \ j \in C, \ s = 0, \ldots, S - 1 \quad (30)

(8), (10), (11),
where we use the superscript \( \lambda \) to emphasize that the solution depends on the Lagrange multipliers.

Noting constraints (9), constraints (30) would be redundant in problem (7)-(11), but we add them to problem (29)-(31) to tighten the relaxation. For notational brevity, we let

\[
Y_i(r_{it}) = \left\{ (x_{it}, y_{it}) \in \mathbb{Z}_+ \times \mathbb{Z}_+^{\left|C\right|(1+S)} : \right. \\
\left. x_{it} + \sum_{j \in C} \sum_{s=0}^{S} y_{ijst} = r_{it} \right.
\]

\[
x_{it} \leq K \right. \\
y_{ijst} \leq 1 \quad \forall j \in C, s = 0, \ldots, S - 1 \} 
\]

\[
p_{it}(x_{it}, y_{it}) = \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijt} \right] y_{ijst} - h_{it} x_{it},
\]

where we use \( y_{it} = \{ y_{ijst} : j \in C, s = 0, \ldots, S \} \). In this case, problem (29)-(31) can be written as

\[
V_\lambda^t(r_t) = \max_{i \in P} \left\{ p_{it}(x_{it}, y_{it}) + \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} \left[ 1 - \sum_{i \in P} y_{ijst} \right] + \mathbb{E}\{ V_\lambda^{t+1}(x_{it} + Q_{i,t+1}) \} \right\}
\]

subject to \( (x_{it}, y_{it}) \in Y_i(r_{it}) \quad \forall i \in P. \) (33)

The benefit of this method is that the optimality equation in (32)-(33) decomposes into \( |P| \) optimality equations, each involving a one-dimensional state variable.

**Proposition 3.** If \( \{ V_\lambda^t(r_{it}) : r_{it} \in \mathcal{R}, t \in T \} \) is a solution to the optimality equation

\[
V_\lambda^t(r_{it}) = \max_{i \in P} \left\{ p_{it}(x_{it}, y_{it}) - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{ijst} + \mathbb{E}\{ V_\lambda^{t+1}(x_{it} + Q_{i,t+1}) \} \right\}
\]

for all \( i \in P \), then we have

\[
V_\lambda^t(r_t) = \sum_{t'=t}^{T} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst'} + \sum_{i \in P} V_\lambda^i(r_{it}).
\]

**Proof.** We show the result by induction. It is easy to show the result for the last time period. Assuming that the result holds for time period \( t + 1 \), the objective function in (32) can be written as

\[
\sum_{i \in P} p_{it}(x_{it}, y_{it}) + \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} \left[ 1 - \sum_{i \in P} y_{ijst} \right] + \sum_{t'=t+1}^{T} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst'} + \sum_{i \in P} \mathbb{E}\{ V_\lambda^{i,t+1}(x_{it} + Q_{i,t+1}) \}.
\]

The result follows by noting that both the objective function above and the feasible set of problem (32)-(33) decompose by the elements of \( \mathcal{P} \). \( \square \)

Although the optimality equation in (34) involves a one-dimensional state variable, it requires solving an optimization problem involving \( 1 + \left|C\right|(1 + S) \) decision variables. The next result shows that this optimality equation can be solved efficiently.

**Lemma 4.** Problem (34) can be solved by a sort operation.
Proof. By using backward induction on time periods, one can show that \( V^\lambda_{i,t+1}(r_{i,t+1}) \) is a concave function of \( r_{i,t+1} \) in the sense that \( V^\lambda_{i,t+1}(r_{i,t+1} - 1) + \frac{V^\lambda_{i,t+1}(r_{i,t+1} + 1)}{2} \leq \frac{V^\lambda_{i,t+1}(r_{i,t+1})}{2} \) for all \( r_{i,t+1} = 1, \ldots, R-1 \), which implies that \( \mathbb{E}\{V^\lambda_{i,t+1}(x_{i,t} + Q_{i,t+1})\} \) is a concave function of \( x_{i,t} \). Therefore, letting \( g_{i,k,t+1} = \mathbb{E}\{V^\lambda_{i,t+1}(x_{i,t} + Q_{i,t+1})\} - \mathbb{E}\{V^\lambda_{i,t+1}(k + Q_{i,t+1})\} \) for all \( k = 0, \ldots, K-1 \) and associating the decision variables \( \{z_{i,k,t+1} : k = 0, \ldots, K-1\} \) with these first differences, problem (34) becomes

\[
\begin{align*}
\max & \quad \sum_{j \in C} \sum_{s=0}^{K-1} \left[ f_{jst} - c_{jst} \right] y_{jst} - h_{st} x_{st} - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{jst} + \mathbb{E}\{V^\lambda_{i,t+1}(Q_{i,t+1})\} + \sum_{k=0}^{K-1} g_{i,k,t+1} z_{i,k,t+1} \\
\text{subject to} & \quad x_{st} + \sum_{j \in C} \sum_{s=0}^{S-1} y_{jst} = r_{st} \\
& \quad x_{st} - \sum_{k=0}^{K-1} z_{i,k,t+1} = 0 \\
& \quad y_{jst} \leq 1 \quad \forall j \in C, s = 0, \ldots, S-1 \\
& \quad z_{i,k,t+1} \leq 1 \quad \forall k = 0, \ldots, K-1 \\
& \quad x_{st}, y_{jst}, z_{i,k,t+1} \in \mathbb{Z}_+ \quad \forall j \in C, s = 0, \ldots, S, k = 0, \ldots, K-1.
\end{align*}
\]

(35)

Dropping the constant term in the objective function and using constraint (35) to substitute \( \sum_{k=0}^{K-1} z_{i,k,t+1} \) for \( x_{st} \), the problem above can be written as

\[
\begin{align*}
\max & \quad \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{jst} \right] y_{jst} - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{jst} + \sum_{k=0}^{K-1} g_{i,k,t+1} - h_{st} \sum_{k=0}^{K-1} z_{i,k,t+1} \\
\text{subject to} & \quad \sum_{j \in C} \sum_{s=0}^{S} y_{jst} + \sum_{k=0}^{K-1} z_{i,k,t+1} = r_{st} \\
& \quad y_{jst}, z_{i,k,t+1} \in \mathbb{Z}_+ \quad \forall j \in C, s = 0, \ldots, S, k = 0, \ldots, K-1
\end{align*}
\]

(36), (37),

which is a knapsack problem where each item consumes one unit of space and the result follows. □

The next proposition shows that the optimality equation in (29)-(31) provides upper bounds on the value functions.

Proposition 5. We have \( V_t(r_t) \leq V^\lambda_t(r_t) \) for all \( r_t \in \mathcal{R}^{\left|\mathcal{P}\right|} \), \( t \in T \).

Proof. We show the result by induction. It is easy to show the result for the last time period. We assume that the result holds for time period \( t+1 \), fix \( r_t \in \mathcal{R}^{\left|\mathcal{P}\right|} \) and let \( (\tilde{x}_t, \tilde{y}_t) = \arg\max_{(x_t, y_t) \in \mathcal{Y}(r_t)} \left\{ p_t(x_t, y_t) + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\} \right\} \). We have

\[
V_t(r_t) = \sum_{i \in \mathcal{P}} p_t(\tilde{x}_it, \tilde{y}_it) + \mathbb{E}\{V_{t+1}(\tilde{x}_t + Q_{t+1})\} \leq \sum_{i \in \mathcal{P}} p_t(\tilde{x}_it, \tilde{y}_it) + \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \lambda_{jst} \left[ 1 - \sum_{i \in \mathcal{P}} \tilde{y}_{jst} \right] + \mathbb{E}\{V^\lambda_{t+1}(\tilde{x}_t + Q_{t+1})\} \leq V^\lambda_t(r_t),
\]
where the first inequality follows from the induction assumption and the fact that \( \lambda_{jst} \geq 0 \) for all \( j \in C, s = 0, \ldots, S - 1 \) and \( (\tilde{x}_t, \tilde{y}_t) \in \mathcal{Y}(r_t) \). Noting the objective function of problem (32)-(33), the second inequality follows from the fact that \( (\tilde{x}_it, \tilde{y}_it) \in \mathcal{Y}_i(r_{it}) \) for all \( i \in P \).

By Proposition 5, we have \( V_1(r_1) \leq V_1^\lambda(r_1) \) for any set of positive Lagrange multipliers. If the initial state is known, then we can solve \( \min_{\lambda \geq 0} V_1^\lambda(r_1) \) for a particular initial state \( r_1 \) to obtain a tight bound on the value function. If, however, the initial state is not known and we need to obtain a “good” approximation to the value function for all possible initial states, then we can associate positive weights \( \{ \alpha(r_1) : r_1 \in \mathcal{R}^{\|P\|} \} \) with the initial states. Assuming that \( \sum_{r_1 \in \mathcal{R}^{\|P\|}} \alpha(r_1) = 1 \) without loss of generality, we can obtain a tight bound on the value function by solving the problem

\[
\min_{\lambda \geq 0} \left\{ \sum_{r_1 \in \mathcal{R}^{\|P\|}} \alpha(r_1) V_1^\lambda(r_1) \right\}
\]

\[
= \min_{\lambda \geq 0} \left\{ \sum_{r_1 \in \mathcal{R}^{\|P\|}} \sum_{t \in T} \sum_{j \in C} \sum_{s=0}^{S-1} \alpha(r_1) \lambda_{jst} + \sum_{r_1 \in \mathcal{R}^{\|P\|}} \sum_{i \in P} \alpha(r_1) V_i^\lambda(r_{i1}) \right\}
\]

\[
= \min_{\lambda \geq 0} \left\{ \sum_{t \in T} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} + \sum_{r_1 \in \mathcal{R}^{\|P\|}} \sum_{i \in P} \beta_i(r_{i1}^t) V_i^\lambda(r_{i1}^t) \right\},
\]

(38)

where the first equality follows from Proposition 3, and we use

\[
1_{a=b} = \begin{cases} 
1 & \text{if } a = b \\
0 & \text{otherwise}
\end{cases}
\]

and \( \beta_i(r_{i1}^t) = \sum_{r_1 \in \mathcal{R}^{\|P\|}} 1_{\{r_1=r_{i1}^t\}} \alpha(r_1) \). The objective function of problem (38) is called the dual function. In the next two sections, we propose two methods to minimize the dual function.

### 3.1. Constraint generation

Since \( \{V_{it}^\lambda(r_{it}) : r_{it} \in \mathcal{R}, t \in T \} \) is a solution to the optimality equation in (34), for any set of Lagrange multipliers \( \lambda \), \( \sum_{i \in P} \sum_{r_{i1} \in \mathcal{R}} \beta_i(r_{i1}) V_i^\lambda(r_{i1}) \) can be computed by solving the linear program

\[
\min \sum_{i \in P} \sum_{r_{i1} \in \mathcal{R}} \beta_i(r_{i1}) v_i(r_{i1})
\]

subject to

\[
v_{it}(r_{it}) \geq p_{it}(x_{it}, y_{it}) - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{ijst} + \mathbb{E}\{v_{i,t+1}(x_{it} + Q_{ist})\} \quad \forall r_{it} \in \mathcal{R}, \ (x_{it}, y_{it}) \in \mathcal{Y}_i(r_{it}), \ i \in P, \ t \in T \setminus \{T\} \quad (39)
\]

\[
v_{iT}(r_{iT}) \geq p_{iT}(x_{iT}, y_{iT}) - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{ijst} \quad \forall r_{iT} \in \mathcal{R}, \ (x_{iT}, y_{iT}) \in \mathcal{Y}_i(r_{iT}), \ i \in P, \ (40)
\]
where \( \{v_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \} \) are the decision variables. Therefore, we can find an optimal solution to problem (38) by solving the linear program

\[
\begin{align*}
\min & \quad \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \lambda_{jst} + \sum_{i \in \mathcal{P}} \sum_{r_{i1} \in \mathcal{R}} \beta_i(r_{i1}) v_{i1}(r_{i1}) \\
\text{subject to} & \quad \lambda_{jst} \geq 0 \quad \forall \ j \in \mathcal{C}, \ s = 0, \ldots, S-1, \ t \in \mathcal{T} \quad (42) \\
& \quad \lambda_{jst}, (39), (40), (41), (43), \quad (39), (40), \quad (39), (40), \quad (39), (40), \quad (39), (40),
\end{align*}
\]

where \( \{v_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \}, \ \{\lambda_{jst} : j \in \mathcal{C}, \ s = 0, \ldots, S-1, \ t \in \mathcal{T} \} \) are the decision variables. This problem has \(|\mathcal{P}| |\mathcal{R}| + |\mathcal{C}| |\mathcal{S}| \) decision variables (which is manageable), but \( \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{P}} \sum_{r_{i1} \in \mathcal{R}} |\mathcal{Y}_i(r_{it})| \) constraints.

To deal with the large number of constraints, we use constraint generation, where we iteratively solve a master problem that has the same objective function and decision variables as problem (41)-(43), but has only a few of the constraints.

Letting \( \{\tilde{v}_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \}, \ \{\tilde{\lambda}_{jst} : j \in \mathcal{C}, \ s = 0, \ldots, S-1, \ t \in \mathcal{T} \} \) be the solution to the current master problem, we solve the problem

\[
\max_{(x_{it}, y_{it}) \in \mathcal{S}_{\tilde{v}_{it}(r_{it})}} \left\{ \tilde{p}_{it}(x_{it}, y_{it}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \tilde{\lambda}_{jst} \tilde{y}_{ijst} + \mathbb{E}\{\tilde{v}_{i,t+1}(x_{it} + Q_{i,t+1})\} \right\} \quad (44)
\]

for all \( r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \setminus \{T\} \) to check if any of constraints (39) in problem (41)-(43) is violated by this solution. Letting \( (\tilde{x}_{it}, \tilde{y}_{it}) \) be the optimal solution to problem (44), if we have

\[
\tilde{v}_{it}(r_{it}) \geq \tilde{p}_{it}(\tilde{x}_{it}, \tilde{y}_{it}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \tilde{\lambda}_{jst} \tilde{y}_{ijst} + \mathbb{E}\{\tilde{v}_{i,t+1}(\tilde{x}_{it} + Q_{i,t+1})\}
\]

is violated by the solution \( \{\tilde{v}_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \}, \ \{\tilde{\lambda}_{jst} : j \in \mathcal{C}, \ s = 0, \ldots, S-1, \ t \in \mathcal{T} \}. \)

We add this constraint to the master problem and resolve the master problem. Similarly, we can solve the problem

\[
\max_{(x_{iT}, y_{iT}) \in \mathcal{S}_{\tilde{v}_{iT}(r_{iT})}} \left\{ \tilde{p}_{iT}(x_{iT}, y_{iT}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \tilde{\lambda}_{jst} \tilde{y}_{ijst} \right\}
\]

for all \( r_{iT} \in \mathcal{R}, \ i \in \mathcal{P} \) to check if any of constraints (40) in problem (41)-(43) is violated.

If \( \tilde{v}_{i,t+1}(r_{i,t+1}) \) is a concave function of \( r_{i,t+1} \) in the sense that

\[
\tilde{v}_{i,t+1}(r_{i,t+1} - 1) + \tilde{v}_{i,t+1}(r_{i,t+1} + 1) \leq 2 \tilde{v}_{i,t+1}(r_{i,t+1}) \quad (45)
\]

for all \( r_{i,t+1} = 1, \ldots, R - 1 \), then problem (44) has the same form as the problem considered in Lemma 4 and can be solved by a sort operation. This makes constraint generation very efficient.
In general, it is not guaranteed that the solution to the master problem will satisfy (45). To ensure that (45) is satisfied, we add the constraints
\[ v_{it}(r_{it} - 1) + v_{it}(r_{it} + 1) \leq 2 v_{it}(r_{it}) \quad \forall r_{it} = 1, \ldots, R - 1, \ i \in \mathcal{P}, \ t \in \mathcal{T} \] (46)
to the master problem before we begin constraint generation. By the next lemma, adding constraints (46) to problem (41)-(43) does not change its optimal objective value. This implies that we can add constraints (46) to the master problem without disturbing the validity of constraint generation.

**Lemma 6.** Adding constraints (46) to problem (41)-(43) does not change its optimal objective value.

**Proof.** We let \( \{v^*_i(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T}\}, \{\lambda^*_jst : j \in \mathcal{C}, \ s = 0, \ldots, S - 1, \ t \in \mathcal{T} \} \) be an optimal solution to problem (41)-(43) and \( \{V^\lambda_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ t \in \mathcal{T}\} \) be obtained by solving the optimality equation in (34) with the set of Lagrange multipliers \( \lambda^* \). As mentioned in the proof of Lemma 4, we have \( V^\lambda_{it}(r_{it} - 1) + V^\lambda_{it}(r_{it} + 1) \leq 2 V^\lambda_{it}(r_{it}) \) for all \( r_{it}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \). We now show that \( \{V^\lambda_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T}\}, \{\lambda^*_jst : j \in \mathcal{C}, \ s = 0, \ldots, S - 1, \ t \in \mathcal{T} \} \) is also an optimal solution to problem (41)-(43) and this establishes the result.

Since \( \{V^\lambda_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ t \in \mathcal{T}\} \) solves the optimality equation in (34), we have
\[
V^\lambda_{it}(r_{it}) \geq p_{it}(x_{it}, y_{it}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \lambda^*_jst y_{ijst} + \mathbb{E}\{V^\lambda_{i,t+1}(x_{it} + Q_{i,t+1})\}
\]
for all \( r_{it} \in \mathcal{R}, (x_{it}, y_{it}) \in \mathcal{Y}_{i}(r_{it}), \ i \in \mathcal{P}, \ t \in \mathcal{T} \). Therefore, \( \{V^\lambda_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T}\}, \{\lambda^*_jst : j \in \mathcal{C}, \ s = 0, \ldots, S - 1, \ t \in \mathcal{T} \} \) is feasible to problem (41)-(43). We now show by induction that \( v^*_i(r_{it}) \geq V^\lambda_{it}(r_{it}) \) for all \( r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T} \), which implies that \( \{V^\lambda_{it}(r_{it}) : r_{it} \in \mathcal{R}, \ i \in \mathcal{P}, \ t \in \mathcal{T}\}, \{\lambda^*_jst : j \in \mathcal{C}, \ s = 0, \ldots, S - 1, \ t \in \mathcal{T} \} \) is an optimal solution to problem (41)-(43). It is easy to show the result for the last time period. Assuming that the result holds for time period \( t + 1 \) and noting constraints (39) in problem (41)-(43), we have
\[
v^*_i(r_{it}) \geq \max_{(x_{it}, y_{it}) \in \mathcal{Y}_{i}(r_{it})} \left\{ p_{it}(x_{it}, y_{it}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \lambda^*_jst y_{ijst} + \mathbb{E}\{v^*_i(x_{it} + Q_{i,t+1})\} \right\}
\geq \max_{(x_{it}, y_{it}) \in \mathcal{Y}_{i}(r_{it})} \left\{ p_{it}(x_{it}, y_{it}) - \sum_{j \in \mathcal{C}} \sum_{s=0}^{S-1} \lambda^*_jst y_{ijst} + \mathbb{E}\{V^\lambda_{i,t+1}(x_{it} + Q_{i,t+1})\} \right\} = V^\lambda_{it}(r_{it}). \]

3.2. Cutting planes

Proposition 7 below shows that \( \sum_{i \in \mathcal{P}} \sum_{r_{1i} \in \mathcal{R}} \beta_i(r_{1i}) V^\lambda_{i}(r_{1i}) \) is a convex function of \( \lambda \). This allows us to represent the dual function by using cutting planes.

We need some new notation in this section. We let \( x^\lambda_{it}(r_{it}), \{y^\lambda_{ijst}(r_{it}) : j \in \mathcal{C}, \ s = 0, \ldots, S \} \) be an optimal solution to problem (34). We use the superscript \( \lambda \) and argument \( r_{it} \) to emphasize that
the solution depends on the Lagrange multipliers and the state. We let \( V_{it}^\lambda \) and \( \lambda_{jt} \) respectively be the vectors \( \{V_{it}^\lambda(r_{it}) : r_{it} \in R \} \) and \( \{\lambda_{jst} : s = 0, \ldots, S-1 \} \). We let \( Y_{ijt}^\lambda = \{y_{ijst}^\lambda(r_{it}) : r_{it} \in R, \ s = 0, \ldots, S-1 \} \) be the \(|R| \times S\)-dimensional matrix whose \((r_{it}, s)\)-th component is \( y_{ijst}^\lambda(r_{it}) \). Finally, we let \( P_{it}^\lambda \) be the \(|R| \times |R|\)-dimensional matrix whose \((r_{it}, r_{i,t+1})\)-th component is \( \mathbb{P}\{x_{it}^\lambda(r_{it}) + Q_{i,t+1} = r_{i,t+1} \} \).

We have the following result.

**Proposition 7.** For any two sets of Lagrange multipliers \( \lambda \) and \( \tilde{\lambda} \), we have

\[
V_{it}^{\tilde{\lambda}} \geq V_{it}^\lambda - \sum_{j \in C} Y_{ijt}^\lambda [\tilde{\lambda}_{jt} - \lambda_{jt}] - P_{it}^\lambda \sum_{j \in C} Y_{ij,t+1}^\lambda [\tilde{\lambda}_{j,t+1} - \lambda_{j,t+1}] \\
- P_{it}^\lambda P_{i,t+1} \sum_{j \in C} Y_{ij,t+2}^\lambda [\tilde{\lambda}_{j,t+2} - \lambda_{j,t+2}] - \cdots - P_{it}^\lambda P_{i,T-1} \sum_{j \in C} Y_{ijT}^\lambda [\tilde{\lambda}_{jT} - \lambda_{jT}]
\]

for all \( i \in P, t \in T \).

**Proof.** We show the result by induction. It is easy to show the result for the last time period. We assume that the result holds for time period \( t+1 \). Letting \( y_{it}^\lambda(r_{it}) = \{y_{ijst}^\lambda(r_{it}) : j \in C, \ s = 0, \ldots, S \} \), since \((x_{it}^\lambda(r_{it}), y_{it}^\lambda(r_{it}))\) is an optimal solution to problem (34), we have

\[
V_{it}^\lambda(r_{it}) = p_{it}(x_{it}^\lambda(r_{it}), y_{it}^\lambda(r_{it})) - \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst} y_{ijst}^\lambda(r_{it}) + \mathbb{E}\{V_{i,t+1}^\lambda(x_{i,t+1}(r_{it}) + Q_{i,t+1})\}
\]

\[
V_{it}^{\tilde{\lambda}}(r_{it}) = p_{it}(x_{it}^{\tilde{\lambda}}(r_{it}), y_{it}^{\tilde{\lambda}}(r_{it})) - \sum_{j \in C} \sum_{s=0}^{S-1} \tilde{\lambda}_{jst} y_{ijst}^{\tilde{\lambda}}(r_{it}) + \mathbb{E}\{V_{i,t+1}^{\tilde{\lambda}}(x_{i,t+1}(r_{it}) + Q_{i,t+1})\}.
\]

Subtracting the first expression from the second one, we obtain

\[
V_{it}^{\tilde{\lambda}}(r_{it}) - V_{it}^\lambda(r_{it}) \geq - \sum_{j \in C} \sum_{s=0}^{S-1} y_{ijst}^\lambda(r_{it}) [\tilde{\lambda}_{jst} - \lambda_{jst}] \\
+ \mathbb{E}\{V_{i,t+1}^{\tilde{\lambda}}(x_{i,t+1}(r_{it}) + Q_{i,t+1}) - V_{i,t+1}^\lambda(x_{i,t+1}(r_{it}) + Q_{i,t+1})\}. \quad (47)
\]

The expectation on the right side can be written as

\[
\sum_{r_{i,t+1} \in R} \mathbb{P}\{x_{it}^\lambda(r_{it}) + Q_{i,t+1} = r_{i,t+1} \} \left[ V_{i,t+1}^{\tilde{\lambda}}(r_{i,t+1}) - V_{i,t+1}^\lambda(r_{i,t+1}) \right].
\]

Therefore, (47) can be written in matrix notation as

\[
V_{it}^{\tilde{\lambda}} - V_{it}^\lambda \geq - \sum_{j \in C} Y_{ijt}^\lambda [\tilde{\lambda}_{jt} - \lambda_{jt}] + P_{it}^\lambda [V_{i,t+1}^{\tilde{\lambda}} - V_{i,t+1}^\lambda].
\]

The result follows by using the induction assumption that

\[
V_{i,t+1}^{\tilde{\lambda}} \geq V_{i,t+1}^\lambda - \sum_{j \in C} Y_{ij,t+1}^\lambda [\tilde{\lambda}_{j,t+1} - \lambda_{j,t+1}] - P_{i,t+1}^\lambda \sum_{j \in C} Y_{ij,t+2}^\lambda [\tilde{\lambda}_{j,t+2} - \lambda_{j,t+2}] \\
- \cdots - P_{i,t+1}^\lambda P_{i,t+2} \cdots P_{i,T-1}^\lambda \sum_{j \in C} Y_{ijT}^\lambda [\tilde{\lambda}_{jT} - \lambda_{jT}]. \quad \square
\]
Letting $\Pi_{ij}^s = P_{s1}^i P_{s2}^i \ldots P_{si}^{i-1}$ with $\Pi_{ij}^1 = Y_{ij1}^1$, we have
\[
V_{il}^\lambda \geq V_{il}^\lambda - \sum_{j \in C} \Pi_{ij1}^\lambda [\tilde{\lambda}_j1 - \lambda_j1] - \sum_{j \in C} \Pi_{ij2}^\lambda [\tilde{\lambda}_j2 - \lambda_j2] - \ldots - \sum_{j \in C} \Pi_{ijT}^\lambda [\tilde{\lambda}_jT - \lambda_jT]
\]
by Proposition 7. In this case, letting $\beta_i$ be the vector $\{\beta_i(r_{i1}) : r_{i1} \in R\}$, we obtain
\[
\sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1}) = \sum_{i \in P} \beta_i V_{il}^\lambda \geq \sum_{i \in P} \Pi_{ij}^\lambda \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1})
\]
Therefore, $\sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1})$ has a subgradient, and hence, is a convex function of $\lambda$.

The cutting plane method that we propose to solve problem (38) has the same flavor as the cutting plane methods for two-stage stochastic programs (Ruszczynski (2003)). We iteratively solve a master problem that has the form
\[
\begin{align*}
\min & \quad \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i V_{il}^\lambda - \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i \Pi_{ij}^\lambda \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1}) \\
\text{subject to} & \quad v \geq \sum_{i \in P} \beta_i V_{il}^\lambda - \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1}) \\
& \quad \lambda_{jst} \geq 0 \quad \forall j \in C, s = 0, \ldots, S - 1, t \in T
\end{align*}
\]
at iteration $k$, where $v$, $\{\lambda_{jst} : j \in C, s = 0, \ldots, S - 1, t \in T\}$ are the decision variables. Constraints (49) are the cutting planes that represent the function $\sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1})$, and they are constructed iteratively by using the solution to the master problem and the subgradient provided by (48). (We have $k = 1$ at the first iteration. Therefore, we do not have any of these cutting planes at the first iteration and the optimal objective value of the master problem is $-\infty$.)

We let $(v^k, \lambda^k)$ be the solution to the master problem at iteration $k$. After solving the master problem, we compute $V_{il}^{\lambda^k}(r_{i1})$ for all $r_{i1} \in R, i \in P$. By Lemma 4, this can be done very efficiently. Noting constraints (49), we always have
\[
v^k = \max_{\ell \in \{1, \ldots, k - 1\}} \left\{ \sum_{i \in P} \beta_i V_{il}^{\lambda^\ell} - \sum_{i \in P} \sum_{j \in C} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1}) \right\} \leq \sum_{i \in P} \beta_i V_{il}^{\lambda^k} = \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1}),
\]
where the inequality follows from (48). If we have $v^k = \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1})$, then for any set of positive Lagrange multipliers $\lambda$, we have
\[
\sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1}) = \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + v^k
\]
\[
= \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + \max_{\ell \in \{1, \ldots, k - 1\}} \left\{ \sum_{i \in P} \beta_i V_{il}^{\lambda^\ell} - \sum_{i \in P} \sum_{j \in C} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1}) \right\}
\]
\[
\leq \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + \max_{\ell \in \{1, \ldots, k - 1\}} \left\{ \sum_{i \in P} \beta_i V_{il}^{\lambda^\ell} - \sum_{i \in P} \sum_{j \in C} \sum_{r_{i1} \in R} \beta_i(r_{i1}) V_{il}^{\lambda^k}(r_{i1}) \right\}
\]
\[
\leq \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + \sum_{i \in P} \beta_i V_{il}^\lambda = \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s = 0}^{S - 1} \lambda_{jst} + \sum_{i \in P} \beta_i(r_{i1}) V_{il}^\lambda(r_{i1}),
\]

where the first inequality follows from the fact that \((v^k, \lambda^k)\) is an optimal solution to the master problem and the second inequality follows from (48). Therefore, if we have \(v^k = \sum_{i \in P} \sum_{r_{i1} \in \mathcal{R}} \beta_i(r_{i1}) V_{i1}^{\lambda^k}(r_{i1})\), then \(\lambda^k\) is an optimal solution to problem (38) and we stop. On the other hand, if we have \(v^k < \sum_{i \in P} \sum_{r_{i1} \in \mathcal{R}} \beta_i(r_{i1}) V_{i1}^{\lambda^k}(r_{i1})\), then we construct the constraint

\[
v \geq \sum_{i \in P} \beta_i V_{i1}^{\lambda^k} - \sum_{i \in T} \sum_{n \in P} \sum_{j \in C} \beta_i \Pi_{ij}^{\lambda^k} [\lambda_{jt} - \lambda_{jt}^k],
\]

add it to the master problem, increase \(k\) by 1 and resolve the master problem.

By using an argument similar to the one used to show the convergence of the cutting plane methods for two-stage stochastic programs, we can show that the cutting plane method described above terminates after a finite number of iterations with an optimal solution to problem (38).

**Proposition 8.** The cutting plane method for solving problem (38) terminates after a finite number of iterations with an optimal solution.

**Proof.** There are a finite number of solutions in \(\mathcal{Y}_i(r_{it})\). Therefore, there is a finite set of \(|\mathcal{R}| \times S\)-dimensional matrices such that \(Y_{ij}^{\lambda} \) takes a value in this set for any \(\lambda\). Similarly, there exists a finite set of \(|\mathcal{R}| \times |\mathcal{R}|\)-dimensional matrices such that \(P_{it}^{\lambda} \) takes a value in this set for any \(\lambda\). Consequently, there exists a finite set of \(S\)-dimensional vectors such that \(\beta_i \Pi_{ij}^{\lambda} \) takes a value in this set for any \(\lambda\).

Letting \(p_{it}^{\lambda} \) be the vector \(\{p_{it}(x_{it}^{\lambda}(r_{it}), y_{it}^{\lambda}(r_{it})): r_{it} \in \mathcal{R}\}\), we write (34) in matrix notation as

\[V_{it}^{\lambda} = p_{it}^{\lambda} - \sum_{j \in C} V_{ij}^{\lambda} \lambda_{jt} + p_{it}^{\lambda} V_{i,T+1}.\]

Using this expression and backward induction on time periods, it is easy to show that

\[V_{i1}^{\lambda} = p_{i1}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} p_{i3}^{\lambda} + \ldots + p_{i1}^{\lambda} p_{i2}^{\lambda} p_{i3}^{\lambda} \ldots p_{i,T-1}^{\lambda} p_{iT}^{\lambda} - \sum_{i \in T} \sum_{j \in C} \Pi_{ij}^{\lambda} \lambda_{jt}.\]

Since there are a finite number of solutions in \(\mathcal{Y}_i(r_{it})\), there exists a finite set of \(|\mathcal{R}|\)-dimensional vectors such that \(p_{it}^{\lambda} \) takes a value in this set for any \(\lambda\). This implies that there exists a finite set of scalars such that \(\beta_i [p_{i1}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} p_{i3}^{\lambda} + \ldots + p_{i1}^{\lambda} p_{i2}^{\lambda} \ldots p_{i,T-1}^{\lambda} p_{iT}^{\lambda}]\) takes a value in this set for any \(\lambda\). Using (50) and (51), we write the constraint added to the master problem at iteration \(k\) as

\[v \geq \sum_{i \in P} \beta_i [p_{i1}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} p_{i3}^{\lambda} + \ldots + p_{i1}^{\lambda} p_{i2}^{\lambda} \ldots p_{i,T-1}^{\lambda} p_{iT}^{\lambda}] - \sum_{i \in T} \sum_{n \in P} \sum_{j \in C} \beta_i \Pi_{ij}^{\lambda} \lambda_{jt}.\]

There exist a finite number of possible values for \(\beta_i \Pi_{ij}^{\lambda} \) and \(\beta_i [p_{i1}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} + p_{i1}^{\lambda} p_{i2}^{\lambda} p_{i3}^{\lambda} + \ldots + p_{i1}^{\lambda} p_{i2}^{\lambda} \ldots p_{i,T-1}^{\lambda} p_{iT}^{\lambda}]\). Therefore, the constraint added to the master problem at iteration \(k\) is one of the finitely many possible constraints. By using the finiteness of the number of possible cutting planes, we can show the result by following the same argument in Ruszczyński (2003) that is used to show the finite convergence of the cutting plane methods for two-stage stochastic programs. \(\square\)
One problem with the proposed method is that the number of cutting planes in the master problem can grow large. We note that since a new cutting plane is added at each iteration, the objective value of the master problem does not decrease from one iteration to the next. If we drop the loose cutting planes (that are satisfied as strict inequalities) from the master problem, then the objective value still does not decrease from one iteration to the next. It turns out that we can drop the loose cutting planes from the master problem and still ensure that the proposed method terminates after a finite number of iterations with an optimal solution.

**Lemma 9.** Assume that we drop the loose cutting planes from the master problem at iteration \( k \) whenever the objective value of the master problem at iteration \( k \) is strictly larger than the objective value at iteration \( k - 1 \). In this case, the cutting plane method for solving problem (38) terminates after a finite number of iterations with an optimal solution.

**Proof.** The proof follows from an argument similar to the one in Ruszczynski (2003) and uses the finiteness of the number of possible cutting planes.

In general, dropping all loose cutting planes is not a good idea, since the dropped cutting planes may have to be reconstructed at the later iterations. In our computational experiments, we drop the cutting planes that remain loose for 50 consecutive iterations.

### 4. Applying the Greedy Policy

This section shows that applying the greedy policies characterized by the value function approximations requires solving min-cost network flow problems.

Letting \( \{ \theta^*_t : t \in T \} \), \( \{ \nu^*_i : i \in P, t \in T \} \) be an optimal solution to problem (16)-(18), the value function approximations obtained by the method given in Section 2 are of the form \( \theta^*_t + \sum_{i \in P} \nu^*_i r_{it} \). On the other hand, letting \( \{ \lambda^*_jst : j \in C, s = 0, \ldots, S - 1, t \in T \} \) be an optimal solution to problem (38) and noting Proposition 3, the value function approximations obtained by the method given in Section 3 are of the form

\[
\sum_{t' = t}^T \sum_{j \in C} \sum_{s = 0}^{S-1} \lambda^*_jst' + \sum_{i \in P} V^\lambda^*_it(r_{it}).
\]

As mentioned in the proof of Lemma 4, \( V^\lambda^*_it(r_{it}) \) is a concave function of \( r_{it} \). Therefore, the value function approximations obtained by the methods given in Sections 2 and 3 are of the form \( \tilde{\theta}_t + \sum_{i \in P} \tilde{V}_it(r_{it}) \), where \( \tilde{\theta}_t \) is a constant and \( \tilde{V}_it(r_{it}) \) is a concave function of \( r_{it} \).

At time period \( t \), the greedy policy characterized by the value function approximations \( \{ \tilde{\theta}_t + \)}
\[ \sum_{i \in P} \hat{V}_{it}(r_{it}) : t \in T \] makes the decisions by solving the problem

\[
\max_{(x_t, y_t) \in Y(r_t)} \left\{ p_t(x_t, y_t) + \hat{\theta}_t + \sum_{i \in P} \mathbb{E}\{\hat{V}_{i,t+1}(x_{it} + Q_{i,t+1})\} \right\}.
\]

Since \( \mathbb{E}\{\hat{V}_{i,t+1}(x_{it} + Q_{i,t+1})\} \) is a concave function of \( x_{it} \), letting \( g_{ik,t+1} = \mathbb{E}\{\hat{V}_{i,t+1}(k + 1 + Q_{i,t+1})\} - \mathbb{E}\{\hat{V}_{i,t+1}(k + Q_{i,t+1})\} \) for all \( k = 0, \ldots, K - 1 \) and associating the decision variables \( \{z_{ik,t+1} : k = 0, \ldots, K - 1\} \) with these first differences, the problem above can be written as

\[
\max \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijst} \right] y_{ijst} - \sum_{i \in P} h_{it} x_{it} + \hat{\theta}_t + \sum_{i \in P} \mathbb{E}\{\hat{V}_{i,t+1}(Q_{i,t+1})\} + \sum_{i \in P} \sum_{k=0}^{K-1} g_{ik,t+1} z_{ik,t+1}
\]

subject to
\[
\begin{align*}
- x_{it} + \sum_{j \in C} \sum_{s=0}^{S} y_{ijst} &= r_{it} & \forall i \in P \quad (52) \\
- x_{it} + \sum_{k=0}^{K-1} z_{ik,t+1} &= 0 & \forall i \in P \quad (53) \\
\sum_{i \in P} y_{ijst} &\leq 1 & \forall j \in C, s = 0, \ldots, S - 1 \quad (54) \\
z_{ik,t+1} &\leq 1 & \forall i \in P, k = 0, \ldots, K - 1 \quad (55) \\
x_{it}, y_{ijst}, z_{ik,t+1} &\in \mathbb{Z}_+ & \forall i \in P, j \in C, s = 0, \ldots, S, k = 0, \ldots, K - 1.
\end{align*}
\]

Dropping the constant terms in the objective function, introducing the new decision variables \( \{\delta_{jst} : j \in C, s = 0, \ldots, S - 1\} \) and substituting \( \delta_{jst} \) for \( \sum_{i \in P} y_{ijst} \), this problem becomes

\[
\max \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S} \left[ f_{jst} - c_{ijst} \right] y_{ijst} - \sum_{i \in P} h_{it} x_{it} + \sum_{i \in P} \sum_{k=0}^{K-1} g_{ik,t+1} z_{ik,t+1}
\]

subject to
\[
\begin{align*}
- \sum_{i \in P} y_{ijst} + \delta_{jst} &= 0 & \forall j \in C, s = 0, \ldots, S - 1 \quad (56) \\
\delta_{jst} &\leq 1 & \forall j \in C, s = 0, \ldots, S - 1 \quad (57) \\
x_{it}, y_{ijst}, z_{ik,t+1}, \delta_{jst} &\in \mathbb{Z}_+ & \forall i \in P, j \in C, s = 0, \ldots, S, k = 0, \ldots, K - 1, s' = 0, \ldots, S - 1 \quad (58)
\end{align*}
\]

This problem is the min-cost network flow problem shown in Figure 2. Constraints (52), (53) and (56) in problem (55)-(59) are respectively the flow balance constraints for the white, gray and black nodes in this figure.

5. Computational Experiments

In this section, we numerically test the performances of the approximate dynamic programming methods given in Sections 2 and 3.
5.1. Experimental setup and benchmark strategy

All of our test problems involve 41 customer locations spread over a $1000 \times 1000$ region. We let $c_{ij} = \bar{c}d_{ij}$, where $\bar{c}$ is the shipping cost applied on a “per-mile” basis and $d_{ij}$ is the Euclidean distance from plant $i$ to customer location $j$. Noting (1), the expected profit function at customer location $j$ at time period $t$ depends on $\sigma_{jt}$ and $\rho_{jt} + \pi_{jt}$, and hence, we let $\pi_{jt} = 0$ without loss of generality. For all $i \in \mathcal{P}$, $j \in \mathcal{C}$, $t \in \mathcal{T}$, we sample $\sigma_{jt}$, $\rho_{jt}$ and $h_{it}$ from the uniform distributions over $[0.5 \bar{\sigma}, 1.5 \bar{\sigma}]$, $[0.5 \bar{\rho}, 1.5 \bar{\rho}]$ and $[0.5 \bar{h}, 1.5 \bar{h}]$ respectively. In all of our test problems, we let $\bar{c} = 1.6$, $\bar{\rho} = 1000$ and $\bar{h} = 20$. We vary the other parameters to obtain test problems with different characteristics.

We model the production random variables through mixtures of uniformly distributed random variables. In particular, we let $Q_{it} = \sum_{n=1}^{N} \mathbf{1}_{\{X_{it}=n\}} U_{it}^{n}$, where $X_{it}$ is uniformly distributed over $\{1, \ldots, N\}$ and $U_{it}^{n}$ is uniformly distributed over $\{a_{it}^{n}, \ldots, b_{it}^{n}\}$ for all $n = 1, \ldots, N$. This allows us to change the variance of $Q_{it}$ in any way we like by changing $N$ and $\{(a_{it}^{n}, b_{it}^{n}) : n = 1, \ldots, N\}$. When presenting the results, we give the coefficients of variation of the production random variables.

The benchmark strategy we use is the so-called rolling horizon method. For a given rolling horizon length $L$, the rolling horizon method solves an optimization problem that spans $L$ time periods and that uses the point forecasts of the future production quantities. In particular, if the state vector at time period $t$ is $r_{t}$, then the rolling horizon method makes the decisions by solving the problem

$$\begin{align*}
\max & \quad -\sum_{t'=t}^{t+L-1} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} c_{ijt'} u_{ijt'} + \sum_{t'=t}^{t+L-1} \sum_{j \in \mathcal{C}} F_{jt'}(w_{jt'}) - \sum_{t'=t}^{t+L-1} \sum_{i \in \mathcal{P}} h_{it'} x_{it'} \\
\text{subject to} & \quad \sum_{j \in \mathcal{C}} u_{ijt} + x_{it} = r_{it} \quad \forall \ i \in \mathcal{P} \\
& \quad \sum_{j \in \mathcal{C}} u_{ijt'} + x_{it'} - x_{i,t'-1} = \mathbb{E}\{Q_{it'}\} \quad \forall \ i \in \mathcal{P}, \ t' = t + 1, \ldots, t + L - 1 \\
& \quad \sum_{i \in \mathcal{P}} u_{ijt'} - w_{jt'} = 0 \quad \forall \ j \in \mathcal{C}, \ t' = t, \ldots, t + L - 1 \\
& \quad x_{it'} \leq K \quad \forall \ i \in \mathcal{P}, \ j \in \mathcal{C}, \ t' = t, \ldots, t + L - 1 \\
& \quad u_{ijt'}, w_{jt'}, x_{it'} \in \mathbb{R}_+ \quad \forall \ i \in \mathcal{P}, \ j \in \mathcal{C}, \ t' = t, \ldots, t + L - 1.
\end{align*}$$

(If we have $t + L - 1 > T$, then we substitute $T$ for $t + L - 1$ in the problem above.) Although this problem includes decision variables for time periods $t, \ldots, t + L - 1$, we only implement the decisions for time period $t$ and solve a similar problem to make the decisions for time period $t + 1$. The rolling horizon method is expected to give better solutions as $L$ increases. For our test problems, increasing $L$ beyond 8 time periods provides marginal improvements in the objective value and we let $L = 8$.

We let $\alpha(r_{1}) = 1/|\mathcal{R}||\mathcal{P}|$ for all $r_{1} \in \mathcal{R}||\mathcal{P}|$. Changing these weights does not noticeably affect the performances of the methods given in Sections 2 and 3.
5.2. Computational results

In Section 3, we give two methods to minimize the dual function. Our setup runs showed that the cutting plane method in Section 3.2 is faster than the constraint generation method in Section 3.1 for our test problems, and hence, we use the cutting plane method to minimize the dual function.

However, the cutting plane method has slow tail performance in the sense that the improvement in the objective value of the master problem slows down as the iterations progress. To deal with this problem, we solve problem (38) only approximately. In particular, letting \(\lambda^*\) be an optimal solution to problem (38) and \((v^k, \lambda^k)\) be an optimal solution to the master problem at iteration \(k\), we have

\[
\sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst}^k + v^k = \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst}^* + \max_{\ell \in \{1, \ldots, k-1\}} \left\{ \sum_{i \in P} \beta_i V_{i1}^\lambda - \sum_{i \in P} \sum_{j \in C} \beta_i \Pi_{ij}^\lambda [\lambda_{jst}^k - \lambda_{jst}^\ell] \right\} \\
\leq \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s=0}^{S-1} \lambda_{jst}^* + \max_{\ell \in \{1, \ldots, k-1\}} \left\{ \sum_{i \in P} \beta_i V_{i1}^\lambda - \sum_{i \in P} \sum_{j \in C} \beta_i \Pi_{ij}^\lambda [\lambda_{jst}^* - \lambda_{jst}^\ell] \right\} \\
\leq \sum_{i \in P} \sum_{r_{i1} \in R} \sum_{s=0}^{S-1} \lambda_{jst}^* + \sum_{i \in P} \beta_i (r_{i1}) V_{i1}^\lambda (r_{i1}) = \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst}^* + \sum_{i \in P} \sum_{r_{i1} \in R} \beta_i (r_{i1}) V_{i1}^\lambda (r_{i1}),
\]

where the first inequality follows from the fact that \((v^k, \lambda^k)\) is an optimal solution to the master problem, the second inequality follows from (48) and the third inequality follows from the fact that \(\lambda^*\) is an optimal solution to problem (38). Therefore, the first and the last terms in the chain of inequalities above give lower and upper bounds on the optimal objective value of problem (38). In Figure 3, we plot the percent gap between the lower and upper bounds as a function of the iteration number \(k\) for a particular test problem, along with the total expected profit that is obtained by the greedy policy characterized by the value function approximations \(\{\sum_{t' = 1}^{t} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst'}^k + \sum_{i \in P} V_{it}^{\lambda^k} (r_{it}) : t \in T\}\). This figure shows that the cutting plane method has slow tail performance, but the quality of the greedy policy does not improve after the first few iterations. Consequently, we stop the cutting plane method when the percent gap between the lower and upper bounds is less than 10\%. This does not noticeably affect the quality of the greedy policy.

We summarize our results in Tables 1-4. In these tables, the first set of columns give the characteristics of the test problems, where \(T\) is the length of the planning horizon, \(|P|\) is the number of plants, \(P\) is the number of plants that serves a particular customer location (we assume that each customer location is served by the closest \(P\) plants), \(\overline{\sigma}\) is the average salvage value, \(\bar{Q}\) is the total expected production quantity (that is, \(\bar{Q} = \mathbb{E}\{\sum_{t \in T} \sum_{i \in P} Q_{it}\}\)) and \(V\) is the average coefficient of variation of the production random variables (that is, \(V\) is the average of \(\sqrt{\text{Var}(Q_{it})}/\mathbb{E}(Q_{it}) : i \in P, t \in T\) ). The
second set of columns give the performance of the linear programming-based approximation method (LP). Letting \( \{\theta_t^* + \sum_{i \in \mathcal{P}} \nu_{it}^* r_{it} : t \in T\} \) be the value function approximations obtained by LP, the first one of these columns gives the ratio of the total expected profit that is obtained by the greedy policy characterized by these value function approximations to the total expected profit obtained by the 8-period rolling horizon method (RH). To estimate the total expected profit that is obtained by the greedy policy characterized by a set of value function approximations, we simulate the behavior of the greedy policy for 500 different samples of \( \{Q_{it} : i \in \mathcal{P}, t \in T\} \). The second column gives the number of constraints added to the master problem. The third column gives the CPU seconds needed to solve problem (16)-(18). The fourth and the fifth columns give what percents of the CPU seconds are spent on solving the master problem and constructing the constraints. The third set of columns give the performance of the Lagrangian relaxation-based approximation method (LG). Letting \( \{\sum_{t' = t}^{T} \sum_{j \in \mathcal{C}} \sum_{s = 0}^{S-1} \lambda_{jst'}^* + \sum_{i \in \mathcal{P}} V_{it}^\lambda^*(r_{it}) : t \in T\} \) be the value function approximations obtained by LG, the first one of these columns gives the ratio of the total expected profit that is obtained by the greedy policy characterized by these value function approximations to the total expected profit obtained by RH. The second and the third columns give the number of cutting planes and the CPU seconds needed to solve problem (38) with 10% optimality gap. The fourth and the fifth columns give what percents of the CPU seconds are spent on solving the master problem and constructing the cutting planes. Letting \( \lambda^0 \) be a trivial feasible solution to problem (38) consisting of all zeros, the sixth column gives the ratio of the total expected profit that is obtained by the greedy policy characterized by the value function approximations \( \{\sum_{t' = t}^{T} \sum_{j \in \mathcal{C}} \sum_{s = 0}^{S-1} \lambda_{jst'}^0 + \sum_{i \in \mathcal{P}} V_{it}^\lambda^0(r_{it}) : t \in T\} \) to the total expected profit obtained by RH. Consequently, the gap between the figures in the columns labeled “Prf” and “In Prf” shows the significance of finding a near-optimal solution to problem (38).

There are several observations that we can make from Tables 1-4. On a majority of the test problems, LP performs worse than RH, whereas LG performs better than RH. (Almost all of the differences are statistically significant at 5% level.) The CPU seconds and the number of constraints for LP show less variation among different test problems than the CPU seconds and the number of cutting planes for LG. Finding a near-optimal solution to problem (38) significantly improves the quality of the greedy policy obtained by LG.

Table 1 shows the results for problems with different values of \( P \). For each value of \( P \), we use low, moderate and high values for the coefficients variation of the production random variables. As the number of plants that can serve a particular customer location increases, the performance gap between LG and RH diminishes. This is due to the fact that if a customer location can be served by a large number of plants, then it is possible to make up an inventory shortage in one plant by using the inventory in another plant. In this case, it is not crucial to make the “correct” inventory allocation.
decisions and RH performs almost as well as LG. It is also interesting to note that the performance of the greedy policy characterized by the value function approximations \( \{ \sum_{t'=t}^{T} \sum_{j \in C} \sum_{s=0}^{S-1} \lambda_{jst'}^{0} + \sum_{i \in P} V_{it}^{\lambda_{0}}(r_{it}) : t \in T \} \) gets better as \( P \) decreases. This shows that if \( P \) is small, then simply ignoring the constraints that link the decisions for different plants can provide good policies. Finally, the performance gap between LG and RH gets larger as the coefficients of variation of the production random variables get large.

Table 2 shows the results for problems with different values of \( \bar{\sigma} \). As the salvage value increases, a large portion of the inventory at the plants is shipped to the customer locations to exploit the high salvage value and the incentive to store inventory decreases. This diminishes the value of a dynamic programming model that carefully balances the inventory holding decisions with the shipment decisions, and the performance gap between LG and RH diminishes.

Table 3 shows the results for problems with different values of \( \bar{Q} \). As the total expected production quantity increases, the performance gap between LG and RH diminishes, which is due to the fact that as the product becomes more abundant, it is not crucial to make the “correct” inventory allocation decisions.

Finally, Table 4 shows the results for problems with different dimensions. The CPU seconds and the number of constraints for LP increase as \( T \) or \( |P| \) increases. However, the CPU seconds and the number of cutting planes for LG do not change in a systematic fashion. (This has been the case for many other test problems we worked on.) Nevertheless, as shown in Figure 3, the quality of the greedy policy obtained by LG is quite good even after a few iterations, and problem (38) does not have to be solved to optimality. This observation is consistent with that of Cheung & Powell (1996), where the authors carry out only a few iterations of a subgradient search algorithm to obtain a good lower bound on the recourse function arising from a multi-period stochastic program.

6. Conclusion

We presented two approximate dynamic programming methods, LP and LG, for an inventory allocation problem. Computational experiments showed that LG performs better than LP and RH.

Adelman & Mersereau (2004) show that if LP uses a nonlinear approximation of the value function, then it provides a tighter upper bound on the value function than does LG. Their computational experiments also show that the greedy policies obtained by LP perform better than the ones obtained by LG. However, for our problem class, if LP uses a nonlinear approximation of the value function, then constraint generation requires solving large integer programs, which can be computationally pro-
hibitive. Consequently, although Adelman & Mersereau (2004) show that LP is superior to LG when it uses a nonlinear “approximation architecture,” computational considerations can be prohibitive and LP along with a linear “approximation architecture” can be inferior to LG.

7. Appendix

Proof of Lemma 1. Since $F_{jt}(\cdot)$ is a piecewise-linear concave function with points of nondifferentiability being a subset of positive integers, noting that $f_{jst} = \sigma_{jt}$ for all $s = S, S + 1, \ldots$ and associating the decision variables $\{z_{jst} : s = 0, \ldots, S\}$ with the first differences of $F_{jt}(\cdot)$, problem (2)-(6) can be written as

\[
V_t(r_t) = \max - \sum_{i \in P} \sum_{j \in C} c_{ijt} u_{ijt} + \sum_{j \in C} \sum_{s=0}^S f_{jst} z_{jst} - \sum_{i \in P} h_{it} x_{it} + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\}
\]

subject to

\[
\sum_{i \in P} u_{ijt} - \sum_{s=0}^S z_{jst} = 0 \quad \forall j \in C
\]
\[
z_{jst} \leq 1 \quad \forall j \in C, s = 0, \ldots, S - 1
\]
\[
u_{ijt}, x_{it}, z_{jst} \in \mathbb{Z}_+ \quad \forall i \in P, j \in C, s = 0, \ldots, S
\]

(3), (5).

(See Nemhauser & Wolsey (1988) for more on embedding piecewise-linear concave functions in optimization problems.) Introducing the new decision variables $\{y_{ijst} : i \in P, j \in C, s = 0, \ldots, S\}$ and substituting $\sum_{s=0}^S y_{ijst}$ for $u_{ijt}$, this problem becomes

\[
V_t(r_t) = \max - \sum_{i \in P} \sum_{j \in C} \sum_{s=0}^S c_{ijt} y_{ijst} + \sum_{j \in C} \sum_{s=0}^S f_{jst} z_{jst} - \sum_{i \in P} h_{it} x_{it} + \mathbb{E}\{V_{t+1}(x_t + Q_{t+1})\}
\]

subject to

\[
x_{it} + \sum_{j \in C} \sum_{s=0}^S y_{ijst} = r_{it} \quad \forall i \in P
\]
\[
\sum_{i \in P} \sum_{s=0}^S y_{ijst} - \sum_{s=0}^S z_{jst} = 0 \quad \forall j \in C
\]
\[
z_{jst} \leq 1 \quad \forall j \in C, s = 0, \ldots, S - 1
\]
\[
x_{it}, y_{ijst}, z_{jst} \in \mathbb{Z}_+ \quad \forall i \in P, j \in C, s = 0, \ldots, S
\]

(5).

By Lemma 10 below, we can substitute $\sum_{i \in P} y_{ijst}$ for $z_{jst}$ in the problem above, in which case constraints (62) become redundant and the result follows.

Lemma 10. There exists an optimal solution $(x^*_t, y^*_t, z^*_t)$ to problem (60)-(65) that satisfies $\sum_{i \in P} y^*_{ijst} = z^*_{jst}$ for all $j \in C, s = 0, \ldots, S$. 24
Proof of Lemma 10. We let \((x^*_t, y^*_t, z^*_t)\) be an optimal solution to problem (60)-(65), \(\mathcal{I}^+ = \{(j, s) : \sum_{i \in P} y^*_{ijst} > z^*_{jst}\}\) and \(\tilde{\mathcal{I}}^- = \{(j, s) : \sum_{i \in P} y^*_{ijst} < z^*_{jst}\}\). If we have \(|\mathcal{I}^+| + |\mathcal{I}^-| = 0\), then we are done. Assume that we have \(|\mathcal{I}^+| + |\mathcal{I}^-| > 0\). We now construct another optimal solution \((\tilde{x}_t, \tilde{y}_t, \tilde{z}_t)\) with \(|\tilde{\mathcal{I}}^+| + |\tilde{\mathcal{I}}^-| < |\mathcal{I}^+| + |\mathcal{I}^-|\), where we use \(\tilde{\mathcal{I}}^+ = \{(j, s) : \sum_{i \in P} \tilde{y}_{ijst} > \tilde{z}_{jst}\}\) and \(\tilde{\mathcal{I}}^- = \{(j, s) : \sum_{i \in P} \tilde{y}_{ijst} < \tilde{z}_{jst}\}\). This establishes the result.

Assume that \((j', s') \in \mathcal{I}^+\). Since \((x^*_t, y^*_t, z^*_t)\) satisfies constraints (62), there exists \(s''\) such that \((j', s'') \in \mathcal{I}^-\). (If we assume that \((j', s'') \in \mathcal{I}^+\), then there exists \(s'\) such that \((j', s') \in \mathcal{I}^+\) and the proof remains valid.) We let \(\delta = \sum_{i \in P} y^*_{ij's't} - z^*_{j's't} > 0\) and assume that \(\delta \leq z^*_{j's't} - \sum_{i \in P} y^*_{ij's't}\). We pick \(i_1, \ldots, i_n \in \mathcal{P}\) such that \(y^*_{i_1j's't} + \ldots + y^*_{i_nj's't} \geq \delta\) and \(y^*_{i_1i'_js't} + \ldots + y^*_{i_ni'_js't} < \delta\).

We let \(\tilde{x}_t = x^*_{it}\), \(\tilde{z}_j = z^*_{jst}\) for all \(i \in \mathcal{P}\), \(j \in \mathcal{C}\), \(s = 0, \ldots, S\) and
\[
\tilde{y}_{ijst} = \begin{cases} 
0 & \text{if } i \in \{i_1, \ldots, i_{n-1}\}, j = j', s = s' \\
y^*_{i_1j's't} + \ldots + y^*_{i_nj's't} - \delta & \text{if } i = i_n, j = j', s = s' \\
y^*_{ij's't} + y^*_{ij's't} & \text{if } i \in \{i_1, \ldots, i_{n-1}\}, j = j', s = s'' \\
y^*_{ij's't} - y^*_{i_1j's't} - \ldots - y^*_{i_nj's't} + \delta & \text{if } i = i_n, j = j', s = s'' \\
y^*_{ijst} & \text{otherwise.} 
\end{cases}
\]

Since we have \(\sum_{s=0}^S \tilde{y}_{ijst} = \sum_{s=0}^S y^*_{ijst}\) for all \(i \in \mathcal{P}\), \(j \in \mathcal{C}\), \((\tilde{x}_t, \tilde{y}_t, \tilde{z}_t)\) is feasible to problem (60)-(65) and yields the same objective value as \((x^*_t, y^*_t, z^*_t)\). Therefore, \((\tilde{x}_t, \tilde{y}_t, \tilde{z}_t)\) is an optimal solution. Furthermore, (66) implies that
\[
\sum_{i \in \mathcal{P}} \tilde{y}_{ijst} = \begin{cases} 
z^*_{j's't} & \text{if } j = j', s = s' \\
\sum_{i \in \mathcal{P}} y^*_{ij's't} + \delta & \text{if } j = j', s = s'' \\
\sum_{i \in \mathcal{P}} y^*_{ijst} & \text{otherwise.} 
\end{cases}
\]

Since we have \(\tilde{z}_{jst} = z^*_{jst}\) for all \(j \in \mathcal{C}\), \(s = 0, \ldots, S\) and \(\sum_{i \in \mathcal{P}} \tilde{y}_{ijst} = \sum_{i \in \mathcal{P}} y^*_{ijst}\) whenever \((j, s) \notin (j', s'), (j', s'')\), the elements of \(\tilde{\mathcal{I}}^+\) and \(\tilde{\mathcal{I}}^-\) are respectively the same as the elements of \(\mathcal{I}^+\) and \(\mathcal{I}^-\), except possibly for \((j', s')\) and \((j', s'')\). Since we have \(\sum_{i \in \mathcal{P}} \tilde{y}_{ij's't} = z^*_{j's't} = \tilde{z}_{j's't}\), we have \((j', s') \notin \tilde{\mathcal{I}}^+\) and \((j', s') \notin \tilde{\mathcal{I}}^-\). Finally, since we have \(\sum_{i \in \mathcal{P}} \tilde{y}_{ij's't} = \sum_{i \in \mathcal{P}} y^*_{ij's't} + \delta \leq \sum_{i \in \mathcal{P}} y^*_{ij's't} + z^*_{j's't} - \sum_{i \in \mathcal{P}} y^*_{ij's't} = z^*_{j's't}\), we have \((j', s'') \notin \tilde{\mathcal{I}}^+\). Therefore, we have \(|\tilde{\mathcal{I}}^+| = |\mathcal{I}^+| - 1\) and \(|\tilde{\mathcal{I}}^-| \leq |\mathcal{I}^-|\). The proof for the case \(\delta > z^*_{j's't} - \sum_{i \in \mathcal{P}} y^*_{ij's't}\) follows from a similar argument. \qed

References


Figure 1: Problem (23)-(28) is a min-cost network flow problem. The flow balance constraint for the root node is redundant and problem (23)-(28) does not have this constraint. The legend indicates which variables in problem (23)-(28) correspond to which arcs in the network above.

Figure 2: Problem (55)-(59) is a min-cost network flow problem. The flow balance constraint for the root node is redundant and problem (55)-(59) does not have this constraint. The legend indicates which variables in problem (55)-(59) correspond to which arcs in the network above.
Figure 3: The percent gap between the lower and upper bounds on the optimal objective value of problem (38), and the total expected profit that is obtained by the greedy policy characterized by the value function approximations $\{\sum_{t' = t}^T \sum_{j \in C} \sum_{s = 0}^{S-1} \lambda^k_{jst'} + \sum_{i \in P} V^k_{it}(r_{it}) : t \in T\}$.

Table 1: Computational results for problems with different numbers of plants that serve a customer location.
<table>
<thead>
<tr>
<th>Problem parameters</th>
<th>LP</th>
<th>LG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2: Computational results for problems with different salvage values.

<table>
<thead>
<tr>
<th>Problem parameters</th>
<th>LP</th>
<th>LG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3: Computational results for problems with different total expected production quantities.

<table>
<thead>
<tr>
<th>Problem parameters</th>
<th>LP</th>
<th>LG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4: Computational results for problems with different dimensions.