Every Unit Matters: Dynamic Programming for Medical Supply Chain Decisions

Bassel Shaheen, Kareem Taha, Ward Salkeni, Mohammad Nasser, and Mohammad Ehab
Biomedical Engineering Department, Cairo University, Egypt

Course Instructor: Prof. Eman Ayman and Teaching Assistant: Yara ElShamy

Abstract: Efficient medical inventory management is a critical challenge in healthcare systems, where uncertainty in demand and the high cost of shortages can directly impact patient outcomes. This report presents a dynamic programming based decision support system for optimizing medical inventory ordering decisions over a finite planning horizon. The proposed approach explicitly balances regular ordering costs, inventory holding costs, and emergency shortage costs, while respecting warehouse capacity constraints.

The algorithm is implemented from first principles without relying on built in optimization solvers, allowing full transparency and control over the decision process. A simulation and visualization framework is developed to analyze optimal policies, cost accumulation, and inventory trajectories. The obtained results are discussed and qualitatively compared with greedy ordering strategies and Markov Decision Process based formulations reported in the literature.

I. INTRODUCTION

Modern healthcare delivery relies heavily on efficient medical supply chains to ensure continuous access to essential items such as medications, consumables, and protective equipment. Inventory decisions made at hospitals directly influence patient safety, operational efficiency, and financial sustainability [4].

Unlike commercial inventory systems, healthcare supply chains operate under strict service level requirements and high penalties for failure. A missing medical item is not merely an inconvenience but a direct risk to patient outcomes. Shortages of critical medications or surgical supplies can delay treatment and compromise care quality, while excessive stocking increases waste due to expiration [7].

These characteristics motivate the use of structured optimization methods that explicitly model time and future consequences of current decisions. Dynamic programming provides a natural framework for such problems, as it decomposes sequential decision making into tractable stages [6]. In this work, a finite horizon dynamic programming formulation is adopted to optimize hospital inventory replenishment decisions.

II. BIOMEDICAL PROBLEM DESCRIPTION

Hospitals must continuously decide how much inventory to order and when to place these orders. Ordering insufficient quantities leads to stockouts that may delay or compromise patient care, while over ordering increases holding costs and expiration risk [3].

The biomedical consequences of poor inventory decisions are well documented, particularly in emergency medicine, surgical settings, and blood banks [5]. These challenges are amplified in healthcare systems with limited supplier flexibility or constrained budgets.

This work focuses on a single hospital managing a critical medical item. While simplified, this formulation captures the essential dynamics of healthcare inventory systems and serves as a foundation for more complex extensions.

III. PROBLEM STATEMENT AND SYSTEM REQUIREMENTS

The objective of this project is to design a decision support system for hospital inventory management of a critical medical supply.

The system must:

- Operate over a finite planning horizon of T periods.
- Determine optimal regular order quantities at each period.
- Respect a maximum storage capacity constraint.
- Automatically place emergency orders when demand exceeds available inventory.
- Minimize total cost composed of ordering, holding, and emergency shortage costs.
- Output both the minimum total cost and the corresponding optimal ordering policy.

IV. RELATED WORK

Classical inventory models such as Economic Order Quantity and base stock policies provide closed form solutions under simplifying assumptions such as stationary demand and infinite horizons [1]. While computationally efficient, these approaches are often inadequate for healthcare settings where demand varies over time and shortages carry severe penalties.

Sequential inventory control problems are commonly formulated as Markov Decision Processes and solved using dynamic programming techniques [2]. These formulations explicitly account for the evolution of system state and future costs. Applications in healthcare include blood inventory management, pharmaceutical supply planning, and emergency preparedness systems [4, 5].

Greedy and myopic policies are frequently used in practice due to their simplicity, but they ignore future cost to go and often result in excessive emergency procurement [3]. Approximate and heuristic approaches trade optimality for scalability [8]. In contrast, dynamic programming offers exact optimal solutions for moderate problem sizes with full interpretability [6].

V. MODEL AND OPTIMIZATION FRAMEWORK

The inventory management problem is formulated as a finite horizon sequential decision process. At each time period, the hospital selects an order quantity before observing demand, while accounting for future consequences of the decision.

Dynamic programming is employed due to its ability to explicitly represent time dependent tradeoffs between current cost and future risk [6].

A. System State and Decisions

The system state at time t is defined as the available inventory level I_t. The decision variable is the regular order quantity q_t, constrained by storage capacity.

B. Cost Structure

Three cost components are considered:

- Regular ordering cost with fixed and per unit components.
- Inventory holding cost.
- Emergency shortage cost when demand exceeds available inventory.

This structure reflects healthcare procurement realities, where emergency sourcing is significantly more expensive than planned replenishment [9].

C. Inventory Dynamics and Objective

Inventory evolves according to:

$$I_{t+1} = \max(0, I_t + q_t - D_t)$$

The objective is:

$$\min_{\{q_t\}} \sum_{t=1}^{T} C_t$$

subject to inventory dynamics and capacity constraints.

VI. ALGORITHMIC SOLUTION PROCEDURE

The optimization problem is solved using backward dynamic programming. A value table stores the minimum future cost for each time period and inventory level.

A. Bellman Recursion

Let $V_t(I_t)$ denote the minimum cost to go. The Bellman equation is:

$$V_t(I_t) = \min_{q_t \geq 0} [C_t(I_t, q_t, D_t) + V_{t+1}(I_{t+1})]$$

with terminal condition:

$$V_T(I) = 0$$

B. Policy Extraction

The optimal policy is recovered through backtracking from the initial inventory level. This produces the complete ordering plan, inventory trajectory, and cost breakdown.

VII. CORRECTNESS OF THE ALGORITHM

The proposed dynamic programming algorithm satisfies the principle of optimality, which states that an optimal policy has the property that, regardless of the initial state and decision, the remaining decisions must constitute an optimal policy with respect to the state resulting from the first decision [2].

At each time step, the Bellman recursion evaluates all feasible ordering decisions for every possible inventory state and selects the decision that minimizes the sum of immediate cost and future cost-to-go. Since the value function is computed backward from the terminal condition and all feasible state–action pairs are exhaustively evaluated, the resulting policy is globally optimal for the defined model.

Therefore, the algorithm guarantees minimization of the total accumulated cost over the entire planning horizon, subject to the inventory dynamics and capacity constraints.

VIII. EXAMPLE RUN

To illustrate the proposed dynamic programming framework, we consider a single hospital managing a critical medical supply over a planning horizon of 12 months. The input parameters used in this example are summarized in Table I.

TABLE I: Input Parameters for Example Run

Parameter	Value
T	12 months
I_0	0 units
$I_{\max}$	10 units
$c_{\text{order, fixed}}$	100
$c_{\text{order, unit}}$	10
c_{hold}	2
$c_{\text{emergency, fixed}}$	150
$c_{\text{emergency, unit}}$	60
D_t	[8, 4, 10, 9, ...]

This scenario allows us to observe how the algorithm anticipates demand variations, manages inventory efficiently, and minimizes emergency orders compared to naive or greedy policies.

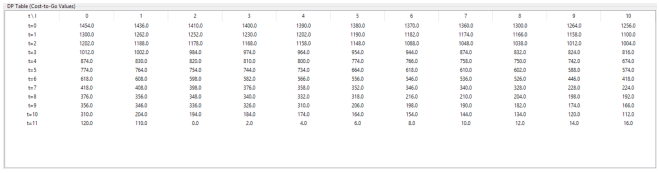

FIG. 1: DP value function snapshot for selected periods. Each cell shows the minimum cost-to-go for the corresponding inventory level.

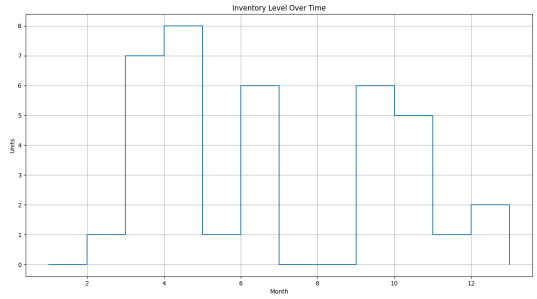

FIG. 2: Inventory trajectory over 12 periods under DP (blue) policy.

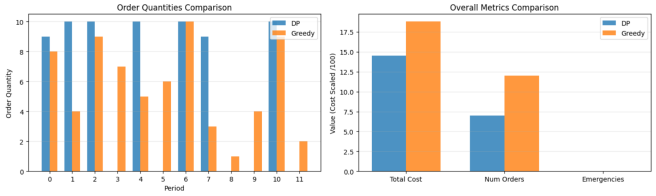

FIG. 3: Optimal order quantities under DP compared to greedy strategy.

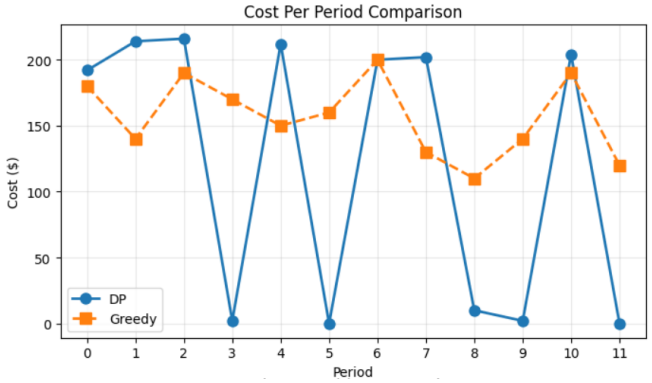

FIG. 4: Period-wise cost components for DP and greedy policies. DP reduces emergency costs significantly.

IX. TIME COMPLEXITY AND COMPUTATIONAL ANALYSIS

The proposed solution employs a finite-horizon, discrete-state dynamic programming formulation that explicitly evaluates the Bellman optimality equation using backward recursion. The computational cost arises from exhaustively enumerating all feasible inventory states and ordering decisions across the planning horizon.

Unlike solver-based approaches, the algorithm is implemented using explicit table-based iteration, enabling transparent validation of each state-action transition and cost component.

A. Bellman Equation and Iterative Evaluation

At each time period t, the algorithm evaluates the Bellman equation:

$$V_t(I_t) = \min_{q_t} \left[C(I_t, q_t, d_t) + V_{t+1}(I_{t+1}) \right]$$

where:

- I_t is the inventory level at the start of period t,
- q_t is the regular order quantity,
- d_t is the realized demand,
- $I_{t+1} = \max(I_t + q_t - d_t, 0)$,

- $C(\cdot)$ represents ordering, holding, and emergency costs.

This equation is implemented directly through nested iteration in the solver, following the backward recursion logic in `validation.py`.

B. State and Action Discretization

Let:

- T denote the number of time periods,
- $I_{\max}$ denote the maximum inventory capacity,
- $N = I_{\max} + 1$ denote the number of discrete inventory states,
- $U_t = I_{\max} - I_t + 1$ denote the number of feasible order quantities at state I_t.

The value function $V_t(I)$ is stored in a two-dimensional table of size $(T + 1) \times N$. A corresponding policy table is maintained to enable optimal policy backtracking.

C. Nested Loop Structure in the Implementation

The backward dynamic programming recursion consists of three nested loops:

1. An outer loop iterating backward over time periods $t = T - 1, \ldots, 0$,
2. A middle loop iterating over all inventory states $I_t \in \{0, \ldots, I_{\max}\}$,
3. An inner loop iterating over all feasible order quantities $q_t \in \{0, \ldots, I_{\max} - I_t\}$.

For each tuple (t, I_t, q_t), the algorithm:

- Computes the immediate cost for regular ordering, holding, and emergency shortages,
- Computes the resulting next inventory state I_{t+1},
- Performs a constant-time lookup of $V_{t+1}(I_{t+1})$,
- Updates the minimum cost and corresponding optimal action.

All operations within the innermost loop execute in constant time.

D. Time Complexity Derivation

In the worst case, the number of evaluated state-action pairs is:

$$\sum_{t=1}^{T} \sum_{I_t=0}^{I_{\max}} (I_{\max} - I_t + 1)$$

This expression is upper bounded by:

$$O(T \cdot N \cdot U)$$

Since $U \leq N$, the overall time complexity simplifies to:

$$O(TN^2)$$

This quadratic dependence on inventory resolution reflects the exhaustive evaluation required to guarantee global optimality.

E. Space Complexity

The value function table and policy table each store $T \times N$ entries, resulting in a space complexity of:

$$O(TN)$$

This memory requirement enables exact policy extraction and interpretability of the optimal solution.

F. Scalability and Practical Implications

While the proposed formulation is computationally tractable for moderate planning horizons and inventory capacities, the quadratic dependence on inventory resolution introduces a natural scalability limit. Increasing inventory granularity or capacity directly increases runtime.

This tradeoff is acceptable in medical inventory management, where emergency shortages incur substantial costs and decision quality outweighs computational speed. Nevertheless, for large-scale healthcare systems, this motivates future exploration of approximate dynamic programming and heuristic methods [8], while exact dynamic programming remains a principled benchmark [6].

X. SYSTEM IMPLEMENTATION AND EVALUATION SETUP

The optimization framework is implemented in Python using explicit table-based dynamic programming without reliance on external solvers. This approach ensures full control over state transitions, cost validation, and policy extraction. The system inputs include:

- Planning horizon T,
- Demand sequence $\{d_t\}_{t=1}^{T}$,
- Initial inventory level,
- Storage capacity,
- Regular ordering, holding, and emergency cost parameters.

The system outputs include:

- Minimum total cost,
- Optimal ordering policy,
- Inventory trajectories,
- Emergency order quantities,
- Period-wise cost breakdowns.

Evaluation is conducted using synthetic demand scenarios designed to capture baseline operation, capacity stress, and emergency-driven conditions. These scenarios are validated through the Bellman recursion implemented in `verification.py`.

Performance metrics include total cost, frequency of emergency orders, and inventory stability over time. Visualization results are presented in the subsequent section following empirical evaluation.

XI. DISCUSSION AND CONCLUSION

The dynamic programming policy consistently anticipates future demand and reduces reliance on emergency procurement compared to greedy strategies, which act myopically and ignore future costs [3, 8].

The deterministic dynamic programming formulation used in this work can be interpreted as a special case of a Markov Decision Process with degenerate transition probabilities [2]. While simplified, this assumption enables transparency and interpretability.

Overall, this work presents an explainable and structured decision support framework for medical inventory management. The formulation provides a solid foundation for future extensions incorporating stochastic demand, multiple items, and real hospital data.

[1] P. Zipkin, *Foundations of Inventory Management*, McGraw-Hill, 2000.
[2] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.
[3] E. Silver, D. Pyke, and R. Peterson, *Inventory Management and Production Planning and Scheduling*, Wiley, 1998.
[4] A. Shah et al., *Healthcare Supply Chain Management*, Springer, 2021.
[5] J. Blake and M. Hardy, "A decision support system for blood bank inventory management," *Annals of Operations Research*, 2014.
[6] D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2017.
[7] S. Nahmias, *Production and Operations Analysis*, 6th Edition, McGraw-Hill Irwin, 2009.
[8] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley, 2011.
[9] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*, McGraw-Hill/Irwin, 2008.