Science Journal of Applied Mathematics and Statistics
2026, Vol. 14, No. 1, pp. 1–5
https://doi.org/10.11648/j.sjams.20261401.11

Research Article

A Method for Problem Solving Dynamic Programming Using Quadratic Inequalities and Convex Monotonicity Theory

Kim Kwon Jun, So Ung Bom, Kim Hyon Chol*

Faculty of Information Science and Technology, Kim Chaek University of Technology, Pyongyang, DPR Korea

Abstract

Dynamic programming is an important discipline in the fields of applied mathematics, operations, and computer science, and standard solution methods are used in various fields of engineering, economics, commerce, management, etc. Dynamic programming is that, whatever the initial state of optimality and the initial control, a sequence of subsequent controls with respect to the resulting state must be the optimal strategy. Thus, it is an optimization method that solves problems in a time-dependent process using the principle of optimality for economic problems that cannot be solved by linear programming. The continued emergence of publications describing new settings, reformulations and general theory in the development of dynamic programming demonstrates the continuing interest in the fundamental problems of dynamic programming. In this paper, we introduce an acceleration method to improve the execution time, which is the most challenging problem in solving many real-life dynamic programming problems. And we prove that the acceleration method using decision monotonicity is effective in improving the execution time when the input data is large compared to the existing method. We also consider the execution time when the mathematical model of the plant is referred to as dynamic programming and the state transition equation satisfies convex monotonicity. We have used the quadrilateral inequality and convex monotonicity theory to solve the traditional computational complexity and time-increasing problem and find reasonable solutions quickly and accurately. In this paper, we introduce an accelerated method to improve the execution time, which is the most challenging problem in solving many real-life dynamic programming problems. We define the quadrilateral inequality and convex monotonicity theory and consider the acceleration of the dynamic programming solution of the mathematical model satisfying it.

Keywords

Dynamic Programming, Acceleration, Decision Monotonicity, Algorithm, Complexity

1. Introduction

Dynamic programming (DP) generally investigates how to find controls that optimize the objective for controllable objects that can be implemented differently in practical activities.

Thus, DP is a mathematical approach to optimize the automation and management of the production technological process.

DP is a method that reduces the problem of extrema of several variable functions based on the principle of optimality.

DP has been widely applied in various fields, including military and economic sectors, as well as in academic fields.

*Corresponding author: HC.Kim@star-co.net.kp (Kim Hyon Chol)

Received: 16 October 2025; **Accepted:** 31 October 2025; **Published:** 7 January 2026

Applications of DP include the management of freight cars, where decisions about when, where and how many to move have to be made in the presence of numerous sources of uncertainty, including customer demands, transit times and equipment problems.

Many researchers have been working on the fault diagnosis of jet engines using DP in the military field, and various methods and techniques have been proposed for it.

The idea is to approximate DP solutions by using a function approximation structure such as neural networks to approximate the cost function [4, 5].

DP is a method that in general solves optimization problems that involve making a sequence of decisions by determining, for each decision, subproblems that can be solved in like fashion, such that an optimal solution of the original problem can be found from optimal solutions of subproblems [6-12].

Let us assume that the problem is given as follows:

Let us consider a problem where the entire interval is divided into several intervals and the sum of cost of each interval is minimized.

When the cost of the interval from i to j is called $w(i,j)$, this problem reduces to a DP problem which yields [1, 2, 13-15].

$$R = \sum_{i=1}^{n} f(x_i)$$

$$f(x) = min\{f(i) + w(i,x) | 1 \le i < x\} \quad (1)$$

Let the entire interval be $[1,n]$ and the solution of the problem becomes $f(n)$.

Given this, the previous algorithm performed the corresponding computations by iterating i for each n.

Since the state quantity is two, the execution complexity of this algorithm is $O(n^2)$ [3].

Since the overall algorithm complexity is $O(n^2)$, we cannot implement this algorithm when n is very large.

For example, when $n = 10^5$, the total number of operations is 10^{10}, so the operation time will take about 100s.

Due to the too long execution time of the algorithm, we will discuss an accelerated method to improve the complexity of the algorithm to $O(nlogn)$.

2. Acceleration Method in the DP

2.1. Quadrilateral Inequality and Convex Monotonicity

Before discussing the acceleration method in the DP, let us discuss the quadrilateral inequality and convex monotonicity.

Definition 1: For any binary function $w(x,y): N \times N \to R (\forall x \forall y, x \ge y => w(x,y) = 0)$, we define a one-membered function $f(x) = w(x,i+1) - w(x,i): N \to R, i \in N$.

If $f(x)$ is a non-increasing function, i.e., $w(x,i+1) - w(x,i) \ge w(x+1,i+1) - w(x+1,i)$, then $w(x,y)$ is called convex monotonicity.

If $w(x,y)$ satisfies convex monotonicity, when $\forall k > 0$, then $f(x) = w(x,i+k) - w(x,i): N \to R, i \in N$ becomes a non-increasing function.

If $f(x) = w(x,i+k) - w(x,i): N \to R, i \in N$ is a non-increasing function, then $g(x) = w(i+k,x) - w(i,x): N \to R, i \in N$ is also a non-increasing function for $\forall k > 0$.

Hence, if $w(x,y)$ satisfies convex monotonicity, then satisfies $w(i,j) + w(\acute{\imath},\acute{\jmath}) \le w(i,\acute{\jmath}) + w(\acute{\imath},j)$ for $\forall i,\acute{\imath},j,\acute{\jmath} \in N, i \le \acute{\imath} \le j \le \acute{\jmath}$.

Definition 2: For $\forall i,\acute{\imath},j,\acute{\jmath} \in N, i \le \acute{\imath} \le j \le \acute{\jmath}$, if $w(i,j) + w(\acute{\imath},\acute{\jmath}) \le w(i,\acute{\jmath}) + w(\acute{\imath},j))$, then $w(x,y)$ is called to satisfy the quadrilateral inequality.

Similarly, we can derive the convex monotonicity of $w(x,y)$ from the quadrilateral inequality.

Theorem 1: If $w(x,y)$ satisfies convex monotonicity, then it is necessary and sufficient to satisfy the quadrilateral inequality.

2.2. Decision Monotonicity and Acceleration Method

Recall the above-mentioned isomorphic state transition.

$$f(x) = min\{f(i) + w(i,x) | 1 \le i < x\} \quad (2)$$

Definition 3: Assume $t(i,x) = f(i) + w(i,x) (1 \le i < x)$.

For $\exists x$, $f(x)$ is called to satisfy decision monotonicity *if* $t(i,x) \ge t(j,x) (i < j)$, when for $\forall y > x$, $t(i,y) \ge t(j,y)$.

In other words, for $i,j (i < j)$, if at any time the decision i is not superior to the decision j, then the decision i cannot be superior to the decision j.

Let us discuss an accelerated method for finding $f(n)$ in $O(nlogn)$ time if the decision monotonicity is satisfied.

For each decision i from 1 to n, we find $B(i) (i < B(i) \le n+1)$ in turn.

Here $B(i) = min\{x | \forall j (1 \le j < n), j < i => t(j,x) \ge t(i,x)\}$.

That is, $B(i)$ is the minimum value of x such that decision i is superior to any decision $j (j < i)$.

For any $j (j < i)$, if $B(j) \ge B(i)$, then $f(x)$ satisfies decision monotonicity, so decision j is not needed thereafter.

Thus, the sequence of useful decisions is $B(i_1) < B(i_2) < ... < B(i_t)$ at any time, when $i_1, i_2, ..., i_t (i_1 < i_2 < \cdots < i_t)$.

When $f(x)$ is found, the decision i_k is the optimal decision if $k = max\{k | B(i_k) \le x, k \le l\}$ is found in the decision sequence.

That is,

$$f(x) = t(i_k, x) \quad (3)$$

Because it is necessary and sufficient that $B(i_k) \leq x$ so that $i_k(1 \leq i_k < x)$ is superior to any random decision $j(j \leq i_k)$.

From the characteristics of the decision sequence, we can then find $\acute{k}$ for $\acute{x} > x(\acute{x} = x + 1)$ in the decision sequence, after the kth decision in the decision sequence, in the following.

Thus, the complexity of finding the optimal decision i_k n times becomes $O(n)$.

After finding $f(x)$, x is a useful decision, so we find $B(x)$ and sort out the decision sequence.

When $y = min\{y|t(x,y) < t(i_l,y)\}$, let's see the time to find y as T.

If $y \leq B(i_l)$, then $B(x) \leq B(i_l)$ must be true.

Thus, i_l is a useless decision and must be removed from the decision sequence (simply set $1 = 1 - 1$).

This process continues until $y = min\{y|t(x,y) < t(i_l,y)\} > B(i_l)$.

Then $B(x) = y$, and x is added to the end of the decision sequence. We can see that using the stack facilitates this sequence maintenance. Each decision is made to enter the stack at most once or to exit the stack.

Since each decision takes T time when it comes out of the stack, the time complexity of maintaining the sequence is $O(nT)$.

On the other hand, the time complexity of finding the optimal decision from decision monotonicity is $O(n)$.

Thus, the total time complexity is $O(nT)$.

In any case, from the expression of the w function, we can easily find $y = min\{y|t(x,y) < t(i_l,y)\}$, and then $T = O(1)$.

In a more general case, if the w function satisfies convex monotonicity, then $t(x,y) - t(i_k,y)$ is a monotone non-increasing function with respect to y, so we can find y by a two-minute search in $O(logn)$ time.

Thus, if the w function satisfies convex monotonicity, this type of DP problems can be solved in the worst case $O(nlogn)$ time and in the best case $O(n)$ time.

As can be seen, this method is very efficient.

2.3. Quadrilateral Inequality and Decision Monotonicity

To improve the mathematical model of the type of DP that satisfies the decision monotonicity, the state transition of the corresponding mathematical model must satisfy the quadrilateral inequality.

The relationship between the introduced quadrilateral inequality and the decision monotonicity theory will be solved by developing a mathematical model for the following problem and improving the mathematical model.

There is a row of $n(n \leq 1000)$ piles of stones.

Now we want to merge these piles into a single pile of stones, each time we can merge two adjacent piles into a single pile of stones, and the number of newly built stones is the cost of this merger.

When n piles are made of one pile, a mathematical model to find the minimum cost can be established as follows:

[Algorithm Analysis] This problem is a typical application of DP.

Let n stone piles' numbers be $1, 2, \ldots, n$ in turn, and let the number of stones of each pile be $d[1], d[2], \ldots, d[n]$. Then, $m[i,j], 1 \leq i \leq j \leq n$ are the minimum cost of merging $d[i], \ldots, d[j]$, and the state-transfer equation and edge conditions of the dynamic scheme are as follows:

$$\begin{cases} m[i,j] = 0, i = j \\ m[i,j] = min\{m[i,k-1] + m[k,j] + \sum_{l=i}^{j} d[l]\}, i < j \end{cases} \quad (4)$$

In addition, by convention $s[i,j] = k$, we can find the minimum value by plotting the partition position before the merging of $d[i], \ldots, d[j]$ and then find the merging method.

In the above expression, the value of $\sum_{l=i}^{j} d[l]$ can be calculated in $O(1)$ time as $t[i] = \sum_{j=1}^{i} d[j], i = 1 \ldots n, t[0] = 0$, first calculated during the preprocessing process $\sum_{l=i}^{j} d[l] = t[j] - t[i-1]$.

In the above algorithm, the total number of states is $O(n^2)$, each state is derived from $O(n)$ states, and since the state transition time is $O(1)$, the total time complexity is $O(n^3)$.

[Improvement of time complexity]

In this problem, $w[i,j]$ satisfies the quadrilateral inequality when $w[i,j] = \sum_{l=i}^{j} d[l]$.

Also, since $d[i] \geq 0, t[i] \geq 0$, we see that $w[i,j]$ satisfies monotonicity with respect to interval membership.

From the definition of $w[i,j]$, the above state transfer can be written as

$$\begin{cases} m[i,j] = 0, i = j \\ m[i,j] = min\{m[i,k-1] + m[k,j] + w[i,j]\}, i < j \end{cases} \quad (5)$$

It can be derived that $m[i,j]$ also satisfies the quadrilateral inequality when $w[i,j]$ satisfies the above two conditions.

That is, for $\forall i, \acute{\imath}, j, \acute{\jmath} \in N, i \leq \acute{\imath} \leq j \leq \acute{\jmath}, m[i,j] + m[\acute{\imath},\acute{\jmath}] \leq m[i,\acute{\jmath}] + m[\acute{\imath},j]$.

And this property can be proved by mathematical induction.

In the quadrilateral inequality we have a recursion for length $l = j - i = \acute{\jmath} - \acute{\imath}$.

For $i = \acute{\imath}$ or $j = \acute{\jmath}$, the inequality must be true.

Hence, when $l \leq 1$, the function $m[i,j]$ satisfies the quadrilateral inequality.

We proceed by splitting the proof into two cases.

In case $i < \acute{\imath} = j < \acute{\jmath}$

In this case, the quadrilateral inequality is simplified to the semi-triangular inequality as follows:

$$m[i,j] + m[j,\acute{\jmath}] \leq m[i,\acute{\jmath}] \quad (6)$$

Putting $k = max\{p|m[i,j] = m[i,p-1] + m[p,j] + w[i,j]$, k is either $k \le j$ or $k > j$.

In the following, we consider only if $k \le j$.

The proof is similar for $k > j$.

When $k \le j$

$$m[i,j] + m[j,j] \le w[i,j] + m[i,k-1] + m[k,j] + m[j,j]$$

$$\le w[i,j] + m[i,k-1] + m[k,j] + m[j,j]$$

$$\le w[i,j] + m[i,k-1] + m[k,j] = m[i,j] \quad (7)$$

In case $i < i < j < j$

$$y = max\{p|m[i,j] = m[i,p-1] + m[p,j] + w[i,j]\}$$

$$z = max\{p|m[i,j] = m[i,p-1] + m[p,j] + w[i,j] \quad (8)$$

Here, we also divide into two cases: the case $z \le y$ or $z > y$.

In the following, we consider only the case of $z \le y$.

Since $i < z \le y \le j$

$$m[i,j] + m[i,j] \le w[i,j] + m[i,z-1] + m[z,j] + w[i,j] + m[i,j] + m[i,y-1] + m[y,j] \le w[i,j] + w[i,j] + m[i,y-1] + m[i,z-1] + m[z,j] + m[y,j] \le w[i,j] + w[i,j] + m[i,y-1] + m[i,z-1] + m[y,j] + m[z,j] \le m[i,j] + m[i,j] \quad (9)$$

Coordinating 2 cased above, we see that $m[i,j]$ satisfies the quadrilateral inequality.

Putting $s[i,j] = max\{k|m[i,j] = m[i,k-1] + m[k,j] + w[i,j]\}$, we can derive the monotonicity of the function $s[i,j]$ from the fact that the function $m[i,j]$ satisfies the quadrilateral inequality.

To prove $s[i,j] \le s[I,j+1]$, we first set $m_k[i,j] = m[i,k-1] + m[k,j] + w[i,j]$, and then we prove that $\forall i < k < \acute{k} < j, m_k[i,j] \le m_{\acute{k}}[i,j] => m_{\acute{k}}[i,j+1] \le m_k[i,j+1]$ from the definition of $s[i,j]$.

In fact, we can prove the inequality $m_k[i,j] - m_{\acute{k}}[i,j] \le m_k[i,j+1] - m_{\acute{k}}[i,j+1]$ which embeds this expression.

Thus, by expanding the expression from the definition $m_k[i,j] + m_{\acute{k}}[i,j+1] \le m_k[i,j+1] + m_{\acute{k}}[i,j]$, it follows that $[k,j] + m[\acute{k},j+1] \le m[\acute{k},j] + m[k,j+1]$, and it is a quadrilateral inequality for the case $k < \acute{k} \le j < j+1$.

The proof of $s[i,j+1] \le s[i+1,j+1]$ can also be proved in a similar way.

Summarizing the above description, we see that $s[i,j]$ is monotonic when $w[i,j]$ satisfies the quadrilateral inequality.

The improved state transfer equation obtained from this is

$$\begin{cases} m[i,j] = 0, i = j \\ m[i,j] = min\{m[i,k-1] + m[k,j] + w[i,j]\}, i < j, s[i,j-1] < k \le s[i+1,j] \end{cases} \quad (10)$$

The time complexity of the algorithm can be improved in a similar way in the problem of finding the maximum cost of merging stone piles.

Let us analyze the time complexity of the algorithm.

For certain states of length $l = j - i$, $m[1,l+1]$ is $s[1,l]$ to $s[2,l+1]$, $m[2,l+2]$ is $s[2,l+1]$ to $s[3,l+2]$.

…

Thus, if the transition times of these states are summed together, then $s[n-l+1,n] - s[1,l] + n - l + 1 = O(n)$.

Since $0 \le l \le n-1$, the total time complexity is $O(n^2)$.

3. Results and Analysis

To see the superiority of the previous method of acceleration in solving the above dynamics, let us compare the running time of the algorithm for different n values (Table 1).

Table 1. *Simulation analysis result.*

n	Existing Method(s)	Acceleration method (s)
100	<0.01	<0.01
1000	10	<0.01
10000	>1min	1

As shown in the table above, the execution times are similar for small n values, but the differences in execution times are significant with increasing n values.

4. Conclusions

In this paper, we have introduced an acceleration method to improve the execution time, which is the most challenging problem for solving many real-world DP problems.

The test results show that the acceleration method using decision monotonicity makes a significant improvement in execution time when the input data is large compared to the existing method.

We found that the mathematical modeling of the plant resulted in a DP approach and significantly improved the execution time than the existing approach when the state transition equation satisfied decision monotonicity.

Abbreviations

DP Dynamic Programming

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Moshe Sniedovich. Dynamic Programming Foundations and Principles 2nd Edition, 2012.

[2] F. Frances. Yao. Speed-Up in Dynamic Programming, 2016.

[3] Warren B. Powel, Approximate Dynamic Programming, 2007.

[4] Huaguang Zhang, Derong Liu, Yanhong Luo, Ding Wang, Adaptive Dynamic Programming for Control, 2012.

[5] Art Lew, Holger Mauch, Dynamic Programming, 2006.

[6] Samir M. Koriem, T. E. Dabbous, W. S. El-Kilani, A new Petri net modeling technique for the performance analysis of discrete event dynamic systems, 2004, The journal of systems and software 72, https://doi.org/10.1016/S0164-1212(03)00211-5

[7] Sokol Bush Kaliaj, A functional equation arising in dynamic programming, Springer International Publishing, 2017, https://doi.org/10.1007/s00010-017-0495-6

[8] Hamed Khaledi, Mohammad Reisi-Nafchi, Dynamic production planning model: a dynamic programming approach, 2012, https://doi.org/10.1007/s00170-012-4600-7

[9] Fawaz Alsolami, Talha Amin, Igor Chikalov, Mikhail Moshkov, Dynamic Programming Approach for Construction of Association Rule Systems, 2016, https://doi.org/10.3233/FI-2016-1402

[10] L. P. Myshlyaev, S. N. Starovatskaya, Dynamic Programming in the Presence of Indeterminacy, 2011, https://doi.org/10.3103/S096709121106009X

[11] Jiangyan Pu, Qi Zhang, Dynamic Programming Principle and Associated Hamilton-Jacobi—Bellman Equation for Stochastic Recursive Control Problem with Non-Lipschitz Aggregator, 2018, https://doi.org/10.1051/cocv/2017016

[12] Yu. V. Bugaev, S. V. Chikunov, Generalization of the Dynamic Programming Scheme, 2007, https://doi.org/10.1134/S0005117909020076

[13] Warren B. Powell, Perspectives of approximate dynamic programming, 2012, https://doi.org/10.1007/s10479-012-1077-6

[14] E. A. Galperin, Reflections on Optimality and Dynamic Programming, 2006, https://doi.org/10.1016/j.camwa.2006.08.015

[15] S. Kindermann, A. Leitão, Regularization by dynamic programming, 2007, https://doi.org/10.1515/JIIP.2007.016