Geometric Programming Approximation to Dynamic Programming

Harrison O. Amuji (harrison.amuji@futo.edu.ng)
Federal University of Technology Owerri
Chinemerem Igboanusi
Federal University of Technology Owerri
Obioma G. Onukwube
Federal University of Technology Owerri

Method Article

Keywords: Dynamic programming, Curse of dimensionality, Optimal allocation policy, Geometric programming, Optimal objective function

Posted Date: October 31st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2217286/v1

Additional Declarations: No competing interests reported.

Geometric Programming Approximation to Dynamic Programming

Harrison O. Amuji[1*], Chinemerem Igboanusi [2] and Obioma G.Onukwube[3]

[1,3]Department of Statistics, Federal University of Technology Owerri, Imo State Nigeria

[2]Department of Transport Management Technology, Federal University of Technology, Owerri

[1]harrison.amuji@futo.edu.ng, [2]c.igboanusi@yahoo.com [3]obioma.onukwube @futo.edu.ng

Abstract:

In this paper, we have developed a geometric programming approximation to dynamic programming. The method has additional advantages of providing the cost involved in decision making and also eliminates the course of dimensionality which restricted the application of dynamic programming to small problems. We also obtained the optimal dual decision variables. We stated two lemmas, and through them, we proved that the optimal allocation policy is the same as the optimal primal decision variables and that the sum of cost matrix in Dynamic programming is the same as the cost coefficient in Geometric programming. We applied the method on a problem and obtained the optimal cost for decision making to be ₦97.30 and optimal decision policy to be (0.1, 0.2, 1, 3). This policy means that lecturers should combine to teach courses in year two and year three but each lecturer handles one course in year four and three courses in year five.

Keywords: Dynamic programming, Curse of dimensionality, Optimal allocation policy, Geometric programming, Optimal objective function

1. Introduction

Dynamic programming is a class of non-linear programming that makes use of optimality and recursive relationship to arrive at optimal decision. It has diverse applications, especially in those areas where most of other non-linear and linear optimization cannot be applied. It breaks a problem into stages with each stage having independent optimal decisions. Dynamic programming has variant models, depending on the nature of the problem to be solved, but in general it maintains a unique feature. Our interest in this paper is to provide solution to non applicability of dynamic programming in solving large class of problems. The restriction is due to the curse of dimensionality. The curse of dimensionality is the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space. The curse of dimensionality occurs when the complexity increases rapidly which is caused by the increasing number of possible combinations of inputs. This problem can be solved by converting the high dimensional variables into lower dimensional variables without changing the specific information of the variables. For example, in the principal component analysis, the curse of dimensionality is reduced to the Principle component, where a reasonable high percentage of the variation is accounted for by the principal component. The general objectives of principle component analysis are data reduction and data interpretation, Onyeagu (2003). But principle component has a linear function while we are dealing with a non-linear programming.

For problems with equal quantification levels of the state variables, the amount of high-speed memory, the number of calculations and the amount of off-line memory required increase exponentially according to the number of state variables. The number of calculations has been shown also to increase exponentially according to the number of decision variables, Esogbuo and Marks (1974).

2. Literature Review

Researchers have tried to solve this problem of curse of dimensionality in different ways; for example De Farias and Van-Roy (2003) used linear programming to approach the problem of curse of dimensionality. They converted dynamic programming to linear programming. Their major reason was to avoid the curse of dimensionality in dynamic programming. Of course we know that in the history of linear programming via simplex method, there was an observed shortcoming in the storage space. It was argued against linear programming that it occupies a large memory spaces with some variables that are not needed. Similarly, approximating dynamic programming by linear programming will still not entirely solve the problem of occupying a large memory space.

A similar restriction was observed in geometric programming but researchers such as Kochenberger et al (1973) transformed the non-linear programming problem to a linear programming using separable programming. Though the authors applied this approach to geometric programming problem to overcome the computational difficulties arising from the greater than zero degree of difficulty problem, the same approach can be applied to dynamic programming. They noted that the approximating linear programming problems were larger than the original dual problem but can easily be solved due to the efficiency of the Simplex method. However, Separable Programming has received some criticism; for example, Richard (1978) observed that the problem with Separable linear programming was the inability to obtain a global optimal solution. Separable Programming (SP) is a Linear Programming (LP) extension for handling certain types of non-linear functions within the framework of a general linear format. The author, Kilmer (1978) saw Separable Programming as a technique through which non-linear programming problems may be solved using the Simplex method. The point we are try to drive home here is that linear programming does not produce the global optimal solution and it also requires a large memory space. Hence, the dimensionality we are trying to avert is still not entirely eliminated.

Other authors such as Doraszelski and Judd (2012) devised an approach to avoid the curse of dimensionality. They observed that Discrete-time stochastic games with a finite number of states have been widely applied to study the strategic interactions among players in dynamic environments. They noted that the games suffer from a curse of dimensionality when the cost of computing players' expectations over all possible future states increases exponentially in the number of state variables. They explore the alternative of continuous-time stochastic games with a finite number of states and argued that continuous time may have substantial advantages. They observed that under widely used laws of motion, continuous time avoids the curse of dimensionality in computing expectations, thereby speeding up the computations by orders of magnitude in games with more than a few state variables.

On the other hand, Fernandez-Villaverde et al (2020) resorted to deep learning, which they found useful in solving the problem of curse of dimensionality. They were of the opinion that for researchers to answer a wide range of important economic questions, they must solve high dimensional dynamic programming problems. To break the curse of dimensionality associated with these high-dimensional dynamic programming problems, the authors proposed a deep-learning algorithm that efficiently computes a global solution to this class of problems. This method is by no means easier in finding solution to the problem of curse of dimensionality.

On the other hand, Geometric Programming (GP) on its own is a member of non-linear programming problem with a special feature; that is, its objective function and constraint equation(s) are both posynomials, Amuji et al (2020). Posynomial is a polynomial with positive coefficients. The constraint equation(s) are bounded above by unity to ensures that the function attain global optimal solution; see Avriel and Williams (1970) and Ojha and Das (2010). The standard form of geometric programming is that its objective function must be in the minimization form and the constraint equation must be bounded above by unity for constrained geometric programming problems; see Boyd et al (2007) and Mazumder and Jefferson (1983). It was developed for solving nonlinear optimization problems, and was initially developed to model engineering and economic problems, Ben-Israel (1968), but was later extended to model problems in mathematics, statistics, operations research and other numerous disciplines, Amuji et al (2020). The pioneers of geometric programming having carefully observed the nature of polynomial concluded that it could be used to model several physical and engineering problems Rao (2009). But instead of using the generalized polynomials, they restricted the coefficients of the polynomials and its variables to values greater than zero. This restricted polynomial is called monomial if it has only one term and posynomial (that is, positive polynomial) when the numbers of terms are more than one, [9]. It is not always easy to formulate geometric (Posynomial) programming problem, but once formulated, the solution is easy, Mazumder and Jefferson (2007).

It will be most appreciated to approximate a non-linear programming by another non-linear programming. Hence we propose a geometric programming approximation to dynamic programming problem. The obvious advantage of this approach is not only to give dynamic programming a new look but to eliminate completely the curse of dimensionality associated with dynamic programming problem, establish cost for making decisions, establish that the optimal primal decision variables is the same as the optimal decision policy, attains global optimal solution without going through different stages of the problem and to solve large class of dynamic programming problems.

3. Methodology

3. 1. Development of the Technique

Our interest in this work is to find a new technique for dealing with curse of dimensionality. In the course of our research, we observed that dynamic programming can be transformed into geometric programming and the solution approximated via geometric programming. This

proposed method is novel and will help in the extension of dynamic programming to solve large class of problems.

Dynamic programming model is of the form;

$$f_i(S_i, x_i) = R_i(x_i) + f_{i+1}^{*}(S_i - x_i) \;\; ; \quad \text{see Amuji et al (2017)} \qquad (1)$$

From (1), let S_i be the state variables, x_i be the stage variables; $f_i(S_i, x_1)$ be a total investment; $R_i(x_i)$ be the previous investment and $f_{n+1}^{*}(S_n - x_n)$ be the current investment.
We write the model in equation (1) as:

$$f(R_i, x_i) = \max_{x_i}\left[R_i(x_i) + f_{i+1} * (R_i - x_i)\right] \qquad (2)$$

From the model in equation (2), we observed that the problem is separable in the stage variables, x_i. Also referring to the properties of geometric programming, we noted that geometric programming is closed under addition, multiplication and division, see Boyd et al (2007). Hence, we can write equation (2) as

$$f(R_i, x_i) = \max_{x_i}\left[R_i(x_i)\right] + \max_{x_i}\left[f_{i+1} * (R_i - x_i)\right] \qquad (3)$$

At optimality, the minimum of the primal geometric programming problem is equal to the maximum of the dual geometric programming problem, and maximization of a dual problem is the same as minimization of the inverse of the problem (i,e, the prima problem), see Amuji et al (2021), hence we have;

$$Min\ f(R_i, x_i) = \frac{1}{\left[R_i(x_i)\right]} + \frac{1}{\left[f_{i+1} * (R_i - x_i)\right]} \qquad (4)$$

We noticed that x_i has the same dimension as S_i, and resulting into a square matrix of order n, and hence, a geometric programming problem with zero degree of difficulty, see Amuji et al (2020). From equation (4), we observed that each of the terms contains both state and stage variables with associated cost coefficient; hence, we can write equation (4) as

$$Min\ f(z) = \sum_{j=1}^{n} c_j \prod_{i=1}^{m} x_i^{\,a_{ij}} \qquad (5)$$

Where $(R_n, x_n) = z$, c_j is the cost coefficient associated with each term, n is the number of terms, m is the number of variables, x_i is the primal decision variables and a_{ij} is the exponent matrix. Equation (5) is an unconstrained geometric programming problem. Splitting equation (5) into a constrained geometric programming problem, we have

$$Min\ f_0(z) = \sum_{j=1}^{n_0} c_{0j} \prod_{i=1}^{m_0} x_i^{\,a_{0ij}} + \sum_{j=1}^{n_0+1} c_{0j} \prod_{i=1}^{m_0} x_i^{\,a_{0ij}} \qquad (6)$$

Subject to $g_k(x) = \sum_{j=1}^{n} c_{kj} \prod_{i=1}^{m_k} x_i^{a_{kij}} \leq 1$ (7)

Re-writing Equation (6 and 7) in a standard geometric programming form, we have

$$Min\ f_0(x) = \sum_{j=1}^{n_0} c_{0j} \prod_{i=1}^{m_0} x_i^{a_{0ij}} \quad (8)$$

Subject to $g_k(x) = \sum_{j=1}^{n} c_{kj} \prod_{i=1}^{m_k} x_i^{a_{kij}} \leq 1$ (9)

Equations (8) and (9) are the constrained geometric programming models; where n_0 = number of terms in the objective; $f_0(x)$ = objective function; $g_k(x)$ = constraint equation; C_{0j} = cost coefficient; m_0 = number of variables; a_{oij} = exponent matrix, all in the objective function. Then, x_i = primal decision variables; n_k = number of terms in the constraint equation(s); C_k = cost coefficient; m_k = number of variables; a_{kij} = exponent matrix all in the constraint equation(s); see see Amuji et al (2020).

Lemma 1:

The sum of cost matrix in Dynamic programming is the same as the cost coefficients in Geometric programming.

Proof:

Let $x_{1j}, x_{2j}, \ldots, x_{ij}$ be the column entries of the cost matrix in dynamic programming

Hence,

$$\sum_{ij}^{n} x_{ij} = b_{ij} \quad (10)$$

But $$\sum_{ij}^{n} b_{ij} = \bar{x}_{..} \quad (11)$$

Where b_{ij} is the column sum and $\bar{x}_{..}$ is the grand sum of the entire cost columns and

$$C_{kj} = c_{1j} + c_{2j} + \ldots + c_{kj} \quad (12)$$

where C_{kj} is the total sum of the cost coefficients of geometric programming and c_{kj} is the sub cost coefficients; this implies that

$$\bar{x}_{..} = C_{kj} \text{ and } b_{ij} = c_{kj} \quad (13)$$

Therefore:

$$\sum_{ij}^{n} b_{ij} \to \bar{x}_{..} \text{ as } n \to \infty_{..} \tag{14}$$

Let the column terms be $b_{1j}; b_{2j}; b_{3j}, ..., b_{kj}$ and $r_1 < r_2 < r_3 <, ..., < r_k$; where r_i is the common rations; b_{1j} is the first term 'a' and S_m is the sum of finite series.

Let $|r_i| < 1$

$$\sum_{ij}^{n} b_{ij} = \bar{x}_{..} = b_{1j} + b_{2j} + ... + b_{kj}$$

$$S_m = \sum_{ij}^{n} b_{ij} = \frac{a\left(1 - r^n\right)}{1 - r} \to \bar{x}_{..} \tag{15}$$

$$= \frac{a\left[1 - \frac{b_{1j}}{b_{2j}}\right]^n}{\left[1 - \frac{b_{1j}}{b_{2j}}\right]} = \frac{a\left[\frac{b_{2j} - b_{1j}}{b_{2j}}\right]^n}{\left[\frac{b_{2j} - b_{1j}}{b_{2j}}\right]} = \frac{a\left(k_{ij}\right)^n}{k_{ij}} = a\left(k_{ij}\right)^{n-1} = \bar{x}_{..} \tag{16}$$

From equations (16), we observed that equation (13) holds; and hence, the sum of cost matrix in Dynamic programming is the same as the cost coefficient in Geometric programming.

Lemma 2:

Optimal allocation policy is the same as the optimal primal decision variables.

Proof:

Let S_n be the state variables; x_n be the stage variables and x_n^* be the optimal decision policy in Dynamic programming; then, the optimal decision policy is obtained as follows:

$$S_n = n - x_n = x_{1,d} = x_n^*$$

$$S_{n-1} = (n-1) - x_{n-1} = x_{2,d} = x_{n-1}^*$$

. . . = . . . = . . . = . . .

$$S_{n-(n-1)} = (n - (n-1)) - x_{n-(n-1)} = x_{n,d} = x_1^*$$

$$x_d^* = \left(x_n^*, x_{n-1}^*, ..., x_1^*\right) \tag{17}$$

Equation (17) is the optimal decision policy

Again;

Let $x_1, x_2, x_3, ..., x_n$ be the primal decision variables in Geometric programming,

$$\frac{y^* f^*(x)}{C_{kj}} = \prod_{i=1}^{m} x^{a_{ij}} \quad (18)$$

where y* is the optimal weight of the dual decision variables; f*(x) is the optimal objective function; C_{kj} is the cost coefficient and $\prod_{i=1}^{m} x^{a_{ij}}$ is the product of the primal decision variables and a_{ij} is the exponent matrix; for the relationship in (18), see [9].

The Log linear transformation of equation (18) gives

$$a_{1j} \ln x_1 + a_{2j} \ln x_2 + \cdots + a_{mj} \ln x_m = \ln G \quad (19)$$

where $G = \frac{y^* f^*(x)}{C_{kj}}$

Let $w_i = \ln x_i$ (20)

where w_i are the transformed variables; from equation (20), taking the exponential of both sides, we have

$$\exp(w_i) = x_i^* = x^* \quad (21)$$

Equation (21) is the optimal primal decision variables. We observed that equation (17) agrees with equation (21), and hence,

$$x_d^* = x_i^* = x^* \quad (22)$$

Haven established that the optimal allocation policy is the same as the optimal primal decision variables and that the sum of cost matrix in Dynamic programming is the same as the cost coefficient in Geometric programming; see lemma 1 and 2, we can apply our developed method to approximate dynamic programming problem.

4. Application of the method

We shall demonstrate the application of the developed method by the problem presented in Table 1. The Table presents cadres of lecturers and the level of students.

Table 1: Cadre of Lecturers (S_n) and Levels of Students (x_n)

S_n/x_n		1	2	3	4
1		3	3	3	3
2		3	3	3	3
3		1	3	2	3
4		3	3	3	3
5		3	3	3	3
6		1	3	3	3

Source: Amuji et al (2017)

The problem has six states and four stages. The respective costs are obtained as follows;

$$C_1 = \sum_{i=1}^{6} x_{1i} = 14 ;\ C_2 = \sum_{i=1}^{6} x_{2i} = 18 ;\ C_3 = \sum_{i=1}^{6} x_{3i} = 17 ;\ C_4 = \sum_{i=1}^{6} x_{4i} = 18$$

Formulating the problem, we have

$$\text{Minimize } f(x) = 14x_1 + 18x_2 + 17x_3 + 18x_4 \qquad \text{(i)}$$

Equation (i) is an unconstrained geometric programming problem. Reformulating the problem to a constrained geometric programming problem, we have

$$\text{Minimize } f(x) = 14x_1x_2^{-1}x_4 + 18x_1^{-1}x_2^{-1}x_3^{-1} + 17x_2x_3x_4 + 18x_1x_3^{-1}x_4 \qquad \text{(ii)}$$

$$\text{Subject to } g_1(x) = x_2x_4^{-1} + x_1x_3 \le 1 \qquad \text{(iii)}$$

$$\text{Subject to } g_2(x) = x_2^{-1}x_4^{-1} + x_1^{-1}x_3 \le 1 \qquad \text{(iv)}$$

Equations (ii) – (iv) is in the form of equations (8) and (9)

Subject to normality and orthogonality conditions of equations (v & vi)

$$\sum_{j=1}^{n_0} y_{0j} = 1 \qquad \text{(v)}$$

$$\sum_{i=1}^{m_0}\sum_{j=1}^{n} a_{ij}y_{0j} = 0 \qquad \text{(vi)}$$

Where y_{0j} = dual decision variables

Equations (v) and (vi) are put together as a non-homogenous system of linear equations

$$Ay = B \qquad \text{(vii)}$$

Applying equation (vii) on equations (ii) – (iv), we have the following system of linear equations

$y_1 - y_2 + 0y_3 + y_4 + 0y_5 + y_6 + 0y_7 - y_8 = 0$

$-y_1 + y_2 + y_3 + 0y_4 + y_5 + 0y_6 - y_7 + 0y_8 = 0$

$0y_1 - y_2 + y_3 - y_4 + 0y_5 + y_6 + 0y_7 + y_8 = 0$

$y_1 + 0y_2 + y_3 + y_4 - y_5 + 0y_6 - y_7 + 0y_8 = 0$

$y_1 + y_2 + y_3 + y_4 = 1$

$$\begin{bmatrix} 1 & -1 & 0 & 1 & 0 & 1 & 0 & -1 \\ -1 & 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & -1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & -1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Solving for y_i, we have

A= [1, -1, 0, 1, 0, 1, 0, -1; -1, 1, 1, 0,1 ,0, -1, 0; . . . ; 1, 1, 1, 1, 0, 0, 0, 0];

B= [0; 0; 0; 0; 1];

y* = Pinv(A)*B >0

$$y^* = \begin{bmatrix} 0.3289 \\ 0.3553 \\ 0.2237 \\ 0.0921 \\ 0.1974 \\ 0.0789 \\ 0.4474 \\ 0.1447 \end{bmatrix} > 0$$

The above values of y* are the optimal weights of the dual decision variables and they satisfy the orthogonality and normality conditions.

We obtain the optimal objective function as follows;

$$f^*(y) = \left(\left(\frac{C_1}{y_1^*}\right)^{y_1^*}\right) * \left(\left(\frac{C_2}{y_2^*}\right)^{y_2^*}\right) * \left(\left(\frac{C_3}{y_3^*}\right)^{y_3^*}\right) * \ldots * \left(\left(\frac{C_8}{y_8^*}\right)^{y_8^*}\right)$$
$$* \left((y_5^* + y_6^*) \wedge (y_5^* + y_6^*)\right) * \left((y_7^* + y_8^*) \wedge (y_7^* + y_8^*)\right)$$

$$f^*(y) = \left(\left(\frac{14}{0.3289}\right)^{0.3289}\right) * \left(\left(\frac{18}{0.3553}\right)^{0.3553}\right) * \left(\left(\frac{17}{0.2237}\right)^{0.2237}\right) * \left(\left(\frac{18}{0.0921}\right)^{0.0921}\right)$$
$$* \left(\left(\frac{1}{0.19741}\right)^{0.19741}\right) * \left(\left(\frac{1}{0.0789}\right)^{0.0789}\right) * \left(\left(\frac{1}{0.4474}\right)^{0.4474}\right) * \left(\left(\frac{1}{0.1447}\right)^{0.1447}\right)$$
$$* \left((0.2763) \wedge 0.2763\right) * \left((0.5921) \wedge 0.5921\right) = 97.2682$$

The above is the optimal objective function.

We proceed to calculate the primal decision variables as follows using equation (18);

$$(0.3289)(97.2682) = 14 x_1 x_2^{-1} x_4$$

$$(0.3553)(97.2682) = 18 x_1^{-1} x_2 x_3^{-1}$$

$$(0.2237)(97.2682) = 17 x_2 x_3 x_4$$

$$(0.0921)(97.2682) = 18 x_1 x_3^{-1} x_4$$

$$(0.1974)(97.2682) = x_2 x_4^{-1}$$

$$(0.0789)(97.2682) = x_1 x_3$$

$$(0.4474)(97.2682) = x_2^{-1} x_4^{-1}$$

$$(0.1447)(97.2682) = x_1^{-1} x_3$$

Optimal matrix is located at ($N_r = N_0$), that is, the first four rows since it is a constrained geometric programming problem.

Hence, taking the Ln of both sides, we have

$$0.8264 = \ln x_1 - \ln x_2 + \ln x_4$$

$$0.6523 = -\ln x_1 + \ln x_2 - \ln x_3$$

$$0.2468 = \ln x_2 + \ln x_3 + \ln x_4$$

$$-0.6978 = \ln x_1 - \ln x_3 + \ln x_4$$

let $w = \ln x_i$

Hence, forming the matrix, we have

$w_1 - w_2 + 0w_3 + w_4 = 0.8264$

$-w_1 + w_2 - w_3 + 0w_4 = 0.6523$

$0w_1 + w_2 + w_3 + w_4 = 0.2468$

$w_1 + 0w_2 - w_3 + w_4 = -0.6978$

$$\begin{bmatrix} 1 & -1 & 0 & 1 \\ -1 & 1 & -1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0.8264 \\ 0.6823 \\ 0.2468 \\ -0.6978 \end{bmatrix}$$

A = [1, -1, 0, 1; -1, 1, -1, 0; 0, 1, 1, 1; 1, 0, -1, 1];

B = [0.8264; 0.6823; 0.2468; -0.6978];

w* = inv(A)*B

$$w^* = \begin{bmatrix} -2.1765 \\ -1.4268 \\ 0.02339 \\ 1.1110 \end{bmatrix}$$

Applying equation (21), we have;

$$x^*_i = \exp(w^*) = \begin{bmatrix} 0.1132 \\ 0.2401 \\ 1.0237 \\ 3.0374 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 1 \\ 3 \end{bmatrix}$$

These are the optimal weights of the primal decision variables.

5. Conclusion

We approximated dynamic programming by geometric programming, the major reason for this is because of the curse of dimensionality that restricted the application of dynamic programming to

a small class of problems. In the original dynamic programming problem, there was no provision for the cost of making decision but only the optimal allocation policy. In the proposed method, not only that we were able to obtain the optimal allocation policy using our developed method but the optimal cost involved in making such a decision. The sample problem was adapted from Amuji et al (2017) and we were able to obtain the optimal allocation of courses to lecturers to be (0.1, 0.2, 1, 3). This means that there should be pairing of course to different cadres of lecturers in second and third year; then, in the fourth year, lecturers in each cadre should get one course each and in the fifth year they should get 3 courses each). Note that from the original data, first year was not considered. To take this decision, it will cost the policy makers the sum of ₦97.30. The optimal primal decision variable (x_i*) is the optimal allocation policy,. Finally, we stated two lemmas, and through them, we have proved that the optimal allocation policy is the same as the optimal primal decision variables and that the sum of cost matrix in Dynamic programming is the same as the cost coefficient in Geometric programming.

Acknowledgement

We acknowledge the authors and researchers whose works we have used.

Declaration

Not applicable.

Funding

There is no external funding for this paper.

Conflicts of interest/Competing interests

The authors declare no conflict of interest

Availability of data and material

The data is available at Open Journal of Optimization, Vol.6 (2), 176 – 186.

Code availability

Not applicable

Authors' contributions

All authors contributed in the preparation of the paper and Literature Review. Harrison O. Amuji contributed in the development of the method and its application. Harrison O. Amuji, Chinemerem Igboanusi and G. O. Onukwube contributed in the conclusion of the paper and writing the manuscript.

References

Amuji, H. O., Ugwuanyim, G. U., Ogbonna, C. J., Iwu, H. C. and Okechukwu, B. N. (2017). The Usefulness of Dynamic Programming in Course Allocation in the Nigerian Universities. Open Journal of Optimization, Vol.6 (2), 176 – 186.

Amuji, H. O., Ugwuanyim, G. U. and Nwosu, C. O. (2021). A solution to geometric programming problems with negative degrees of difficulty. Advances and Applications in Discrete Mathematics, 26 (2), 221 – 230.

Amuji, H. O; Ugwuowo, F. I; Chukwu, W. I. E and Uche, P. I. (2020). A Modified Generalized Inverse Method for Solving Geometric Programming Problems with Extended Degrees of Difficulties. International Journal of Operational Research, 38(1), 19-30.

Avriel, M. and Williams, A. C. (1970). Complimentary Geometric Programming. SIAM Journal on Applied Mathematics, **19**(1), 125 – 141.

Ben-Israel, A. (1968). Geometric Programming - Theory and Application. Society for Industrial and Applied Mathematics, **10** (2), 235 – 236.

Boyd, S., Kim, S. J, Vandenberghe, L. and Hassibi, A. (2007). A Tutorial on Geometric Programming. Optimization in Engineering, **8**, 67 –127.

De Farias, D. P. and Van-Roy, B. (2003). The Linear Programming Approach to Approximate Dynamic Programming. Operations Research, 51(6), 850-865.

Doraszelski, U. and Judd, K. L. (2012). Avoiding the Curse of Dimensionality in Dynamic Stochastic Games, University of Pennsylvania.

Esogbuo, A. O. and Marks, B. R. (1974). Non-Serial Dynamic Programming - A Survey. Operational Research Quarterly, 25(2), 253-265.

Fernandez-Villaverde, J., Nuno, G., Sorg-Langhans, G. and Vogler, M. (2020). Solving High-Dimensional Dynamic Programming Problems using Deep Learning. University of Pennsylvania.

Kilmer, R. L. (1978). Optimality and Separable Linear Programming: An Additional Reminder. Western Journal of Agricultural Economics, **3**(1), 81 – 84.

Onyeagu, S. I. (2003). A First Course in Multivariate Statistical Analysis: MegaConcept, Nigeria.

Richard, L. K. (1978). Optimality and Separable Linear Programming: An Additional Reminder. Western Journal of Agricultural Economics, **3**(1), 81 – 84.

Kochenberger, G. A., Woolsey, R. E. D. and McCarl, B. A. (1973). On the Solution of Geometric Programs via Separable Programming. Operational Research Quarterly, **24**(2), 285–294.

Mazumder, M. and Jefferson, T. R. (1983). Maximum Likelihood Estimate for Multinomial Probabilities Via Geometric Programming. Biometrika Trust, **70**(1), 257 – 261.

Rao, S. S. (2009). Engineering Optimization: Theory and Practice 4th ed: John Wiley & Sons Inc: Canada.