

The Usefulness of Dynamic Programming in Course Allocation in the Nigerian Universities

Harrison O. Amuji, Geoffrey U. Ugwuanyim, Chukwudi J. Ogbonna, Hycinth C. Iwu, Bridget N. Okechukwu

Department of Statistics, Federal University of Technology, Owerri, Nigeria
Email: amujioji@yahoo.com, gugwuanym@yahoo.com, d_ogbonna59@yahoo.com, iwuchuk@yahoo.com, nwanyi_buife@yahoo.com

How to cite this paper: 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*, 6, 176-186.

<https://doi.org/10.4236/ojop.2017.64012>

Received: August 9, 2017

Accepted: December 26, 2017

Published: December 29, 2017

Copyright © 2017 by authors and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

Having lectured in some universities and polytechnics in Nigeria, the researchers observed problems in course allocations. There are no lay-down techniques on how courses should be allocated with respect to the minimum and maximum credit a lecturer should carry in a semester. Many lecturers were overloaded while others were under-loaded. For this reason, dynamic programming model was developed for allocating courses among lecturers in the Nigerian universities using the Department of Statistics, Federal University of Technology Owerri, as a case study. From our analysis, we observed that among all the optimal allocations discovered in the study, the best optimal allocation policy was achieved at the point (1, 2, 1, 2). Allocation of courses in this order will yield an optimal credit hour of 12 per lecturer per semester.

Keywords

Multi-Stage Decision Problem, Dynamic Programming, Serial Decision Problem

1. Introduction

Dynamic programming is a mathematical technique well suited for the optimization of multistage decision problem. The dynamic programming technique decomposes a multistage decision problem as a sequence of single-stage decision problems. A multistage decision process is one in which a number of single-stage processes are connected in series so that the output of one stage is the input of the succeeding stage. This is a serial dynamic programming since the individual stages are connected head to tail with no recycle. A decision process can be characterized by certain input parameters, S (or data), certain decision

variables (X), and certain output parameters (T) representing the outcome obtained as a result of making the decision. The input parameters are called input state variables, and the output parameters are called output state variables. The objective function R , measures the effectiveness of the decisions made and the output that results from these decisions.

A dynamic programming problem can be stated as follows: find x_1, x_2, \dots, x_n , which optimizes

$$f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n R_i = \sum_{i=1}^n r_i(x_i, s_i + 1) \quad (1)$$

and satisfies the design equations

$$s_i = t_i(s_i + 1, x_i), i = 1, 2, \dots, n. \quad (2)$$

The dynamic programming makes use of the concept of sub-optimization and the principle of optimality in solving a problem. An optimal policy (or a set of decisions) has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [1]. It can be grouped into the following four categories depending on the underlying structures of the systems under study, they are: a) serial processes; b) non-serial processes; c) Markov processes; and d) fuzzy processes [2]. These researchers concentrated mainly on the non-serial dynamic programming. They applied dynamic programming to model problems in chemical engineering, natural gas pipeline systems and water resource systems.

A simplified work was done on dynamic programming formulation [3]. The authors demonstrated the application of dynamic programming using musical instruments and their player and the total man-hour required for playing music. Many researchers on dynamic programming lament the lack of practical applications of the technique. The increasingly powerful computing facilities now available mean that the solution of many hitherto intractable problems is becoming a reality. However, there remains a problem in encouraging students and practitioners to adopt a dynamic programming approach to solution [4]. The author used dynamic programming to model a system whereby revenue support grants are distributed to local authorities, with apparently strong incentives for authorities to reduce expenditure levels. Future spending targets remained heavily dependent on past spending levels.

Dynamic programming is conceptually a powerful computational technique that can solve nonlinear stochastic control problems involving constraints in the state and control variables [5]. His work presents a new decomposition procedure that reduces both the high-speed memory requirement and the computational time for interpolations in the dynamic programming algorithm. He developed dynamic programming model that can calculating the optimal values of cost and control for states and store only the optimal values of cost for states which is what is needed to be stored in the high-speed memory. Dynamic pro-

gramming is a mathematical technique for solving certain types of sequential decision problems. A sequential decision problem is a problem in which a sequence of decisions must be made with each decision affecting future decisions [6]. Dynamic programming is quite different in form and concept from linear programming. Dynamic programming is conceptually more powerful and computationally less powerful than linear programming.

Dynamic programming was used to determine the optimum mix of widths of steel used to “pack” a transformer coil. The approach enables the designer to specify and vary the number of widths to be incorporated into the design. It was found that, beyond a relatively small number, the “coverage” obtained is largely independent of the coil diameter. Transformer coil comprises a series of “plates” of transformer steel packed together. The closer the “packing” is to being circular, the better the design. Thus the designer starts off aiming to fill a circular area with as much transformer steel as possible, this material typically being made available in thicknesses of around 1 mm [7].

Dynamic programming was used for cluster sampling [8]. His work considers the problem of partitioning N entities into M disjoint and nonempty subsets (clusters). Except when both N and $N-M$ are very small, a search for the optimal solution by total enumeration for clustering alternatives is quite impractical. The author presents a dynamic programming approach that reduces the amount of redundant transitional calculations implicit in a total enumeration approach. Unlike most clustering approaches used in practice, the dynamic programming algorithm will always converge on the best clustering solution. The efficiency of the dynamic programming approach depends upon the rapid-access computer memory available. Dynamic programming was used for capital allocation of resources. Dynamic programming algorithms were developed for optimal capital allocation subject to budget constraints. By including multi-level projects, reinvesting returns, borrowing and lending, capital deferrals, and project interactions, the authors were able to handle dynamic programming models with several state variables because the optimal returns are monotone non-decreasing step functions [9].

Due to problems arising from course allocation in various departments in the Nigerian Universities, where some lecturers will carry excess work load and others carry less, the researchers saw it as a matter of priority to develop a model that will equally distribute courses to different cadres of lecturers in different departments using the department of Statistics, Federal University of Technology Owerri as a case study. A Dynamic programming model, otherwise known as optimal allocation policy, was developed for the study. Dynamic programming is a multi-stage decision process where the solution of the final stage proceeds from the first stage with a recursive relationship. The first stage forms a base for the computation of the second stage etc. From this study, it was observed that among all the optimal allocations, the best optimal allocation policy was achieved at the point (1, 2, 1, 2). This means that each lecturer should be as-

signed at most one statistics course in both the harmattan and rain semester second year and one course in the harmattan semesters fourth year respectively. On the other hand, lecturers should be allocated two courses each in the third and fifth year rain and harmattan semesters. Allocation of courses in this order will yield an optimal credit hour of 12 per lecturer per semester. In the second year, Statistics courses are being introduced to the students and they have few of those courses to offer while in the fourth year, the students offer fewer courses only in the harmattan semesters due to six-month industrial attachment they attend in the rain semester.

2. Methodology

Dynamic programming begins by sub-optimizing the last component, numbered 1. This involves the determination of

$$f_1^*(s_2) = \text{opt}_{x_1} [R_1(x_1, s_2)] \quad (3)$$

The best value of the decision variable x_1 is that which makes the return (or objective) function R_1 assume its optimum value, denoted by f_1 . Both x_1 and f_1 depend on the condition of the input that the component 1 receives from s_2 . Since the particular value s_2 will assume after the upstream components are optimized is not known at this time, this last stage sub-optimization problem is solved for a range of possible values of s_2 and the results are entered into a table. This table contains a complete summary of the results of sub-optimization of Stage 1. Next we move up the serial system to include the last two components. In this two-stage sub-optimization, we have to determine

$$f_2^*(s_3) = \text{opt}_{x_2, x_1} [R_2(x_2, s_3) + R_1(x_1, s_2)] \quad (4)$$

Since all the information about component 1 has already been encoded in the table corresponding to f_1 , this information can then be substituted for R_1 and so on till the final stage

$$f_i^*(s_n, x_n) = \text{opt}_{x_i} [R_n(x_i) + f_{n+1}^*(s_n - x_n)] \quad (5)$$

The final thing needed is to retrace the steps through the tables generated, to gather the complete set of x_i^* ($i = 1, 2, \dots, n$) for the system. This can be done as follows. The n th sub-optimization gives the values of x_n^* and f_n^* for the specified value of s_{n+1} . The known design equation

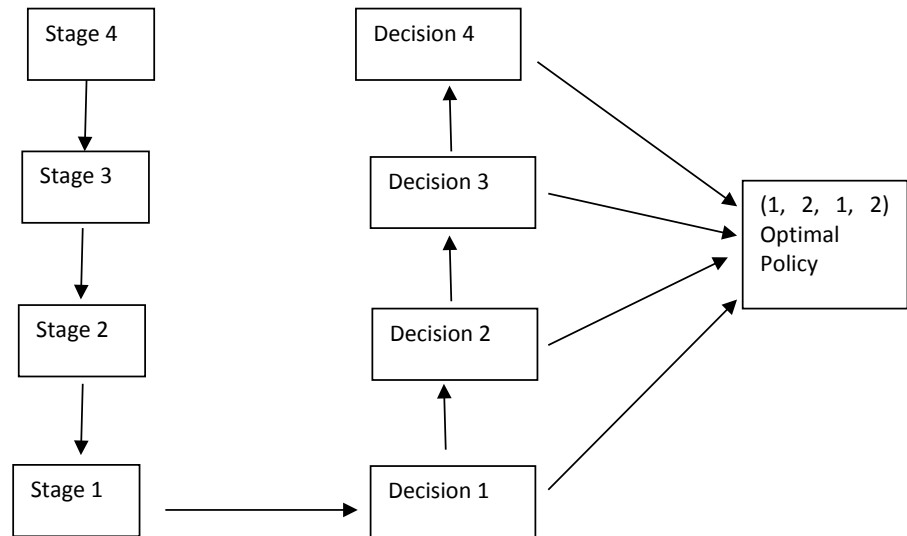
$$s_n = f_n(s_{n+1}, x_n^*) \quad (6)$$

can be used to find the input, s_n^* , to the $(n - 1)$ th stage.

The data for this paper was collected from the department of Statistics, Federal University of Technology, Owerri. The courses are designated by titles, codes and credit hours attached to each of them. Each of these courses is to be allocated to different cadres of academic staff (lecturers) in the department. The data comprise different Statistics courses offered by the department of Statistics from second year to final year (fifth year) when Statistics courses are actually of-

ferred by the students of the department. Few courses are offered in both second and fourth year because in the second year, students are introduced to statistics courses and in the fourth year, second semester (rain), students' go for industrial training. Out of the entire courses offered by the department, simple random sampling was used to draw six courses each from third and fifth year respectively where the bulk of the courses are offered and the entire courses are drawn in both second and fourth year where fewer courses are offered.

A Flow Diagram of the Method



3. Presentation and Analysis of Data

3.1. Presentation of Data

The data to be presented here are:

- 1) Cadres of lecturers and level of statistics courses, see **Table 1**.
- 2) Cadres of lecturers and credit hours per statistics courses, see **Table 2**.

3.2. Analysis of Data

The standard form of dynamic programming when the final state is fixed and initial state is free is given by

$$f_n(s_n, x_n) = \text{opt}_{x_i} [R_n(x_i) + f_{n+1}^*(s_n - x_n)] \tag{7}$$

where x_i are the stages and S_i are the states, $i = 1, 2, \dots, n$.

In this section, we analyze the data presented in **Tables 3-7**.

1) We represent different cadres of lecturers as “State” and level of courses as “Stages”. Here, we have six states and four stages. Zeroes in the second row represent the origin where no allocation has been made. This is represented in **Table 3**.

2) **Table 4** is derived from the state and last column of Stage 4 from **Table 3**, since the last stage is fixed.

Table 1. Cadres of lecturers and level of STA courses.

Cadre of lecturers		200 Level	300 Level	400 Level	500 Level
Professors	1	STA 211	STA 301	STA 411	STA 501
Readers	2	STA 221	STA 321	STA 421	STA 511
Senior lecturers	3	STA 223	STA 331	STA 431	STA 513
Lecturer 1	4	STA 212	STA 312	STA 433	STA 502
Lecturer 11	5	STA 222	STA 336	STA 435	STA 512
Ass. lecturer	6	STA 224	STA 342	STA 451	STA 514

Source: Department of Statistics, FUTO (2017).

Table 2. Cadres of lecturers and credit hours per STA courses.

Cadre of lecturers		200 Level 1	300 Level 2	400 Level 3	500 Level 4
Professors	1	3	3	3	3
Readers	2	3	3	3	3
Senior lecturers	3	1	3	2	3
Lecturer 1	4	3	3	3	3
Lecturer 11	5	3	3	3	3
Ass. lecturer	6	1	3	3	3

Table 3. State and stages.

State/Stages	1	2	3	4
0	0	0	0	0
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

Table 4. Stage 1 of the problem.

S_4	f_4^*	X_4^*
0	0	0
1	3	1
2	3	2
3	3	3
4	3	4
5	3	5
6	3	6

Table 5. Stage 2 of the problem.

S_3/X_3	0	1	2	3	4	5	6	f_3^*	X_3^*
0	0							0	0
1	3	3						3	0 or 1
2	3	6	3					6	1
3	3	6	6	2				6	1 or 2
4	3	6	6	5	3			6	1 or 2
5	3	6	6	5	6	3		6	1, 2 or 4
6	3	6	6	5	6	6	3	6	1, 2, 4 or 5

Table 6. Stage 3 of the problem.

S_2/X_2	0	1	2	3	4	5	6	f_2^*	X_2^*
0	0							0	0
1	3	3						3	0 or 1
2	6	6	3					6	0 or 1
3	6	9	6	3				9	1
4	6	9	9	6	3			9	1 or 2
5	6	9	9	9	6	3		9	1, 2 or 3
6	6	9	9	9	9	6	3	9	1, 2, 3 or 4

Table 7. Stage 4 (final stage) of the problem.

S_1/X_1	0	1	2	3	4	5	6	f_1^*	X_1^*
0	0							0	0
1	3	3						3	0 or 1
2	6	6	3					6	0 or 1
3	9	9	6	1				9	0 or 1
4	9	9	9	4	3			9	0, 1 or 2
5	9	12	12	7	6	3		12	1, or 2
6	9	12	12	10	9	6	1	12	1 or 2

3) **Tables 5-7** are obtained recursively from Equation (7) using **Table 4** as the first stage.

Since the final stage was fixed, we obtain **Table 4** from the first and last column of **Table 3** above. The optimal value X_4^* correspond to the values in the column of S_4 .

Applying the formula in Equation (7) and using **Table 4** as a starting point, we obtain **Tables 5-7** below. See **Appendices 1-3** for the computation of these Tables respectively.

In **Table 8**, we calculate the optimal course allocations starting from **Table 7**. This is calculated by taking the various combinations from the last value of state (S_1) as 6 and $X_1^* = 1$. And we do the needed computations as shown in **Table 8** to obtain the optimal course allocation.

Table 8. Optimal allocations.

$S_1 = 6; X_1^* = 1$	2) $S_1 = 6; X_1^* = 2$
$S_2 = 6 - 1 = 5; X_2^* = 1$	$S_2 = 6 - 2 = 4; X_2^* = 1$
$S_3 = 5 - 1 = 4; X_3^* = 1$	$S_3 = 4 - 1 = 3; X_3^* = 1$
$S_4 = 4 - 1 = 3; X_4^* = 3$	$I_4 = 3 - 1 = 2; X_4^* = 2$
(1, 1, 1, 3)	(2, 1, 1, 2)
3) $S_1 = 6; X_1^* = 1$	5) $S_1 = 6; X_1^* = 2$
$S_2 = 6 - 1 = 5; X_2^* = 2$	$S_2 = 6 - 2 = 4; X_2^* = 2$
$S_3 = 5 - 2 = 3; X_3^* = 1$	$I_3 = 4 - 2 = 2; X_3^* = 1$
$S_4 = 3 - 1 = 2; X_4^* = 2$	$S_4 = 2 - 1 = 1; X_4^* = 1$
(1, 2, 1, 2)	(2, 2, 1, 1)
4) $S_1 = 6; X_1^* = 1$	
$S_2 = 6 - 1 = 5; X_2^* = 3$	
$S_3 = 5 - 3 = 2; X_3^* = 1$	
$S_4 = 2 - 1 = 1; X_4^* = 1$	
(1, 3, 1, 1)	

Thus, the allocation of courses to lectures according to **Table 8** above will yield estimated optimum credit hours of 12 for each lecturer per semester. See the last value in f_1^* of **Table 7**.

4. Interpretation of Results

Calculations in Section 3.2 are presented in **Appendices 1-3**. Hence, the optimal allocation policy of (1, 1, 1, 3) in **Table 8** means the allocation of one Statistics course to each lecturer in each cadre in second, third, fourth year and three courses to each cadre of lecturer in the fifth (final) year will yield optimal credit hour of 12 for each lecturer per semester.

(2, 1, 1, 2) in **Table 8** means the allocation of two Statistics courses to each lecturer in each cadre in second year, one course each in third and fourth year and two courses to each cadre of lecturer in the fifth (final) year will yield optimal credit hour of 12 for each lecturer per semester.

(1, 2, 1, 2) in **Table 8** means the allocation of one Statistics course to each lecturer in each cadre in second year, two courses each to each lecturer in third year, one course to each lecturer in fourth year and two courses to each cadre of lecturer in the fifth (final) year will yield optimal credit hour of 12 for each lecturer per semester.

(1, 3, 1, 1) in **Table 8** means the allocation of one statistics course to each lecturer in each cadre in second year, three courses each to each lecturer in third year, one course to each lecturer in fourth year and one course to each lecturer in each cadre in the fifth (final) year will yield optimal credit hour of 12 for each lecturer per semester.

(2, 2, 1, 1) in **Table 8** means the allocation of two Statistics courses to each lecturer in each cadre in second and third year and allocation of one fourth and

fifth year courses to each lecturer in each cadre will yield optimal credit hour of 12 for each lecturer per semester.

5. Conclusion

In this paper, dynamic programming was applied in allocating courses to lecturers in the Nigerian universities, using the Department of Statistics, Federal University of Technology, Owerri, Imo State as a case study. An optimal 12 credit hours was obtained for each lecturer in each cadre for the semester. Among all the optimal allocations policies presented in section 4, the best optimal allocation is achieved at the point **(1, 2, 1, 2)**. This means that each lecturer should be assigned at most one course in both harmattan and rain semester of second year and one course in harmattan semester of fourth year when fewer number of Statistics courses are offered due to introduction of Statistics courses to second year students and also due to six-month industrial training by the fourth year students in the rain semester. On the other hand, lecturers should be allocated two courses each in the third and fifth year rain and harmattan semesters, when all the students are back for their normal academic session. Allocation of courses in this order will yield an optimal credit hour of 12 per lecturer per semester. This is highly reasonable as opposed to the use of rule of thumb which overload some lecturers with more courses and under load others with fewer courses.

References

- [1] Rao, S.S. (2009) *Engineering Optimization: Theory and Practice*. 4th Edition, John Wiley & Sons, Inc., Canada.
- [2] Esogbue, A.O. and Marks, B.R. (1974) Non-Serial Dynamic Programming: A Survey. *Journal of the Operational Research Society*, **25**, 253-265.
- [3] Adelson, R.M., Norman, J.M. and Laporte, G. (1976) A Dynamic Programming Formulation with Diverse Applications. *Journal of the Operational Research Society*, **27**, 119-121.
- [4] Smith, P. (1989) Dynamic Programming in Action. *Journal of the Operational Research Society*, **40**, 779-787.
- [5] Wong, P.J. (1970) A New Decomposition Procedure for Dynamic Programming. *Operations Research*, **18**, 119-131. <https://doi.org/10.1287/opre.18.1.119>
- [6] Howard, R.A. (1966) Dynamic Programming. *Operations Research*, **12**, 317-348. <https://doi.org/10.1287/mnsc.12.5.317>
- [7] Moores, B. (1986) Dynamic Programming in Transformer Design. *Journal of the Operational Research Society*, **37**, 967-969.
- [8] Jensen, R.E. (1969) A Dynamic Programming Algorithm for Cluster Analysis. *Operations Research*, **17**, 1034-1057. <https://doi.org/10.1287/opre.17.6.1034>
- [9] Nemhauser, G.L. and Ullmann, Z. (1969) Discrete Dynamic Programming and Capital Allocation. *Operations Research*, **15**, 494-505. <https://doi.org/10.1287/mnsc.15.9.494>

Appendices

Computations of Tables 5-7

In these appendices, we show how **Tables 5-7** were computed from Equation (7). The values of current state and stages were substituted while the previous stage forms the starting point in the current stage. This recursive relationship continues until the final stage is obtained.

Appendix 1. Computation of Stage 2 presented in Table 5.

For $n = 3$, we have	$n(3,0) = R_3(0) + f_4^*(3) = 0 + 3 = 3$
$f_3(S_3, X_3) = R_3(X_3) + f_4^*(S_3 - X_3)$	$n(3,1) = R_3(1) + f_4^*(2) = 3 + 3 = 6$
$n(0,0): f_3(0,0) = R_3(0) + f_4^*(0 - 0)$	$n(3,2) = R_3(2) + f_4^*(1) = 3 + 3 = 6$
$= R_3(0) + f_4^*(0) = 0$	$n(3,3) = R_3(3) + f_4^*(0) = 2 + 0 = 2$
$n(1,0): f_3(1,0) = R_3(0) + f_4^*(1 - 0)$	$n(4,0) = R_3(0) + f_4^*(4) = 0 + 3 = 3$
$= R_3(0) + f_4^*(1) = 0 + 3 = 3$	$n(4,1) = R_3(1) + f_4^*(3) = 3 + 3 = 6$
$n(1,1): f_3(1,1) = R_3(1) + f_4^*(0) = 3 + 0 = 3$	$n(4,2) = R_3(2) + f_4^*(2) = 3 + 3 = 6$
$n(1,1): f_3(1,1) = R_3(1) + f_4^*(0) = 3 + 0 = 3$	$n(4,3) = R_3(3) + f_4^*(1) = 2 + 3 = 5$
$n(2,0) = R_3(0) + f_4^*(2) = 0 + 3 = 3$	$n(4,4) = R_3(4) + f_4^*(0) = 3 + 0 = 3$
$n(2,1) = R_3(1) + f_4^*(1) = 3 + 3 = 6$	$n(5,1) = R_3(1) + f_4^*(4) = 3 + 3 = 6$
$n(2,2) = R_3(2) + f_4^*(0) = 3 + 0 = 3$	$n(5,2) = R_3(2) + f_4^*(3) = 3 + 3 = 6$
$n(5,0) = R_3(0) + f_4^*(5) = 0 + 3 = 3$	$n(5,3) = R_3(3) + f_4^*(2) = 2 + 3 = 5$
$n(6,0) = R_3(0) + f_4^*(6) = 3$	$n(5,4) = R_3(4) + f_4^*(1) = 3 + 3 = 6$
$n(6,1) = R_3(1) + f_4^*(5) = 6$	$n(5,5) = R_3(5) + f_4^*(0) = 3 + 0 = 3$
$n(6,2) = R_3(2) + f_4^*(4) = 6$	
$n(6,3) = R_3(3) + f_4^*(3) = 5$	
$n(6,4) = R_3(4) + f_4^*(2) = 6$	
$n(6,5) = R_3(5) + f_4^*(1) = 6$	$n(6,6) = R_3(6) + f_4^*(0) = 3$

Appendix 2. Computation of Stage 3 presented in Table 6.

For $n = 2$, we have	$n(4,3) = R_2(3) + f_3^*(1) = 3 + 3 = 6$
$f_2(S_2, X_2) = R_2(X_2) + f_3^*(S_2 - X_2)$	$n(4,4) = R_2(4) + f_3^*(0) = 3 + 0 = 3$
$f_2(0,0) = R_2(0) + f_3^*(0) = 0$	$n(5,0) = R_2(0) + f_3^*(5) = 0 + 6 = 6$
$n(1,0) = R_2(0) + f_3^*(1) = 3$	$n(5,1) = R_2(1) + f_3^*(4) = 3 + 6 = 9$
$n(1,1) = R_2(1) + f_3^*(0) = 3$	$n(5,2) = R_2(2) + f_3^*(3) = 3 + 6 = 9$
$n(2,0) = R_2(0) + f_3^*(2) = 6$	$n(5,3) = R_2(3) + f_3^*(2) = 3 + 6 = 9$
$n(2,1) = R_2(1) + f_3^*(1) = 6$	$n(5,4) = R_2(4) + f_3^*(1) = 3 + 3 = 6$
$n(2,2) = R_2(2) + f_3^*(0) = 3$	$n(5,5) = R_2(5) + f_3^*(0) = 3 + 0 = 3$
$n(3,0) = R_2(0) + f_3^*(3) = 6$	$n(6,0) = R_2(0) + f_3^*(6) = 0 + 6 = 6$

Continued

$n(3,1) = R_2(1) + f_3^*(2) = 3 + 6 = 9$	$n(6,1) = R_2(1) + f_3^*(5) = 3 + 6 = 9$
$n(3,2) = R_2(2) + f_3^*(1) = 3 + 3 = 6$	$n(6,2) = R_2(2) + f_3^*(4) = 3 + 6 = 9$
$n(3,3) = R_2(3) + f_3^*(0) = 3 + 0 = 3$	$n(6,3) = R_2(3) + f_3^*(3) = 3 + 6 = 9$
$n(4,0) = R_2(0) + f_3^*(4) = 0 + 3 = 3$	$n(6,4) = R_2(4) + f_3^*(2) = 3 + 6 = 9$
$n(4,2) = R_2(2) + f_3^*(2) = 3 + 6 = 9$	$n(6,5) = R_2(5) + f_3^*(1) = 3 + 3 = 6$
	$n(6,6) = R_2(6) + f_3^*(0) = 3 + 0 = 3$

Appendix 3. Computation of Stage 4 presented in Table 7.

For $n = 1$, we have	$n(4,3) = R_1(3) + f_2^*(1) = 1 + 3 = 4$
$f_1(S_1, X_1) = R_1(X_1) + f_2^*(S_1 - X_1)$	$n(4,4) = R_1(4) + f_2^*(0) = 3 + 0 = 3$
$f_1(0,0) = R_1(0) + f_2^*(0) = 0$	$n(5,0) = R_1(0) + f_2^*(5) = 9$
$n(1,0) = R_1(0) + f_2^*(1) = 3$	$n(5,1) = R_1(1) + f_2^*(4) = 3 + 9 = 12$
$n(1,1) = R_1(1) + f_2^*(0) = 3 + 0 = 3$	$n(5,2) = R_1(2) + f_2^*(3) = 3 + 9 = 12$
$n(2,0) = R_1(1) + f_2^*(2) = 6$	$n(5,3) = R_1(3) + f_2^*(2) = 1 + 6 = 7$
$n(2,1) = R_1(1) + f_2^*(1) = 3 + 3 = 6$	$n(5,4) = R_1(4) + f_2^*(1) = 3 + 3 = 6$
$n(2,2) = R_1(2) + f_2^*(0) = 3 + 0 = 3$	$n(5,5) = R_1(5) + f_2^*(0) = 3 + 0 = 3$
$n(3,0) = R_1(0) + f_2^*(3) = 9$	$n(6,0) = R_1(0) + f_2^*(6) = 9$
$n(3,1) = R_1(1) + f_2^*(2) = 3 + 6 = 9$	$n(6,1) = R_1(1) + f_2^*(5) = 3 + 9 = 12$
$n(3,2) = R_1(2) + f_2^*(1) = 3 + 3 = 9$	$n(6,2) = R_1(2) + f_2^*(4) = 3 + 9 = 12$
$n(3,3) = R_1(3) + f_2^*(0) = 1$	$n(6,3) = R_1(3) + f_2^*(3) = 1 + 9 = 10$
$n(4,0) = R_1(0) + f_2^*(4) = 9$	$n(6,4) = R_1(4) + f_2^*(2) = 3 + 9 = 9$
$n(4,1) = R_1(1) + f_2^*(3) = 3 + 6 = 9$	$n(6,5) = R_1(5) + f_2^*(1) = 3 + 3 = 6$
$n(4,2) = R_1(2) + f_2^*(2) = 3 + 6 = 9$	$n(6,6) = R_1(6) + f_2^*(0) = 1 + 0 = 1$