1978, Vol. 3, pp. 183–192

BUILDING DESIGN USING FEEDFORWARD NON-SERIAL DYNAMIC PROGRAMMING

JOHN. S. GERO

Associate Professor, Department of Architectural Science, University of Sydney, Sydney, Australia; formerly, Harkness Research Fellow, Department of Civil Engineering, M.I.T., Cambridge, Mass.

PATRICK J. SHEEHAN

Formerly Graduate Student, Department of Civil Engineering, M.I.T., Cambridge, Mass.

and

JAMES M. BECKER

Assistant Professor, Department of Civil Engineering, M.I.T., Cambridge, Mass.

(*Received July 27, 1977*)

Dynamic programming is proving to be an important optimization technique used in building design. There are many design problems which prima facie do not fit the rigorous serial structure of dynamic programming. This paper describes a procedure for the solution of that class of nonserial dynamic programs which contains feedforward loops which do not intersect. The procedure condenses the various independent paths between stages by absorbing them to produce an equivalent serial structure. The method is demonstrated by solving a problem which can readily be solved by serial dynamic programming upon suitable reformulation. It is then used to design the floor-ceiling sandwich of a multi-storey building, a problem formulated with nested feedforward loops.

NOTATION

D_n	The decision variable at stage n which can be manipulated to optimize the objective
$f_n(X_n)$	The optimal return to stage n
$N_n = \{n_n\}$	The return for state X in stage n
$P_{n-1} = [p^j_{n-1}]$	The return for the path going from state X in stage n to state j in stage $n-1$
$P_{skl} = [p^j_{skl}]$	The return for the path in the feedforward loop going from state j in stage k to state X in stage l
$P_{pkl} = [p^j_{pkl}]$	The optimal return for the primary path going from state j in stage k to state X in stage l
$P_{ekl} = [p^j_{ekl}]$	The optimal return for the equivalent path going from state j in stage k to state X in stage l
r_n	The return associated with stage n
X_n	Particular state variable at stage n
$\oplus$	The composition operator which stipulates separability

1 INTRODUCTION

Optimization techniques are being used in the design of building and architectural subsystems (Gero).[1] However, many of these design problems present themselves as integer linear or nonlinear programming problems which are often difficult to solve. One optimization technique which appears to be particularly applicable is dynamic programming because it handles the entire range of discrete formulations found in building design (Gero).[2] Dynamic programming is a more powerful design tool than most integer nonlinear programming algorithms not only because it guarantees that the global optimum is reached but also because of the facility with which sensitivity and stability studies may be carried out (Gero).[3] In addition the concept of invariant imbedding allows for the solution of a wide range of design problems in one pass. This increases its efficacy for the designer.

Dynamic programming is a sequential decision making optimization technique based on Bellman's optimality principle (Bellman):[4]

"An optimal policy 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."

The requirements that a problem must satisfy in order to be soluble using dynamic programming all relate to the user's ability to decompose the problem into sequential stages. These requirements manifest themselves as separability and monoticity conditions. A burgeoning number of building design problems are being formulated to satisfy these requirements and are soluble directly by dynamic programming.

There are building design problems, however, which prima facie fail to satisfy the separability requirements. Thus, there is the need for nonserial dynamic programming formulations to handle problems such as those in which the output states of one stage are not only connected to the adjacent stage but also to other stages. This occurs in such problems as the design of building subsystems in which, say, the roof is logically related directly to both the walls (which support it) and perhaps the foundations which have to be white ant proofed if the roof is made of timber.

Additionally, nonserial formulations can be used to transmit information between stages which would not be transmitted by the normal sequential paths. Such secondary paths can be used to transmit information which is dimensionally inhomogeneous with the information transmitted along the primary paths.

Nonserial systems have been explored by Nemhauser[5] who has elucidated four separate structures; diverging branches, converging branches, feedforward loops and feedback loops. He has developed suitable formulations for the elementary representations of each of these structures. This paper explores feedforward nonserial systems, develops a dynamic programming formulation for relatively complex systems and demonstrates its applicability to building design.

2 FEEDFORWARD NONSERIAL DYNAMIC PROGRAMMING

The general recursion equations for dynamic programming take the form

$$f_1(X_1) = D_1^{\text{opt}}[r_1(X_1, D_1)] = r_1(X_1) \tag{1}$$

and

$$f_n(X_n) = D_n^{\text{opt}}[r_n(X_n, D_n) \oplus f_{n-1}(X_{n-1})] \tag{2}$$

where X_n = a particular state variable at stage n
r_n = the return associated with stage n
D_n = a decision variable which can be manipulated to optimize the objective
$f_n(X_n)$ = the optimal return to stage n
$\oplus$ = composition operator which stipulates separability

A feedforward nonserial system consists of a staged serial system with a diverging branch at one stage which converges with the main system at a later stage. The transformations and returns are:

i) the usual serial ones for all stages other than the stages at which divergence or convergence takes place;

ii) the diverging stage transformations and returns at the diverging stage; and

iii) the converging stage transformations and returns at the converging stage.

Nemhauser[5] has developed the pertinent relations for the general case. This paper develops the relations for the situation where the returns are composed of two parts. Those associated with the path in going from a state in one stage to a state in another stage and those associated with the state itself. Furthermore, the composition operator will be strictly addition.

Let $P_{n-1} = [p^j_{n-1}]$ be the return for the path going from the state X in the stage n to state j in stage $n-1$

and $N_n = \{n_n\}$ be the return for the state X in stage n

Hence, Eqs. (1) and (2) can be rewritten as

$$f_1(X_1) = \{n_1\} \tag{3}$$

and

$$f_n(X_n) = {}^{\text{opt}}_{j}[(\{n_n\} + [p^j_{n-1}]) + f_{n-1}(X_{n-1})] \tag{4}$$

The optimal decisions can be determined using a trace back procedure. This serial formulation can be readily interpreted graphically.

Consider the system shown in Figure 1 consisting of three stages with three states in each stage plus a feedforward loop between stages 1 and 3.

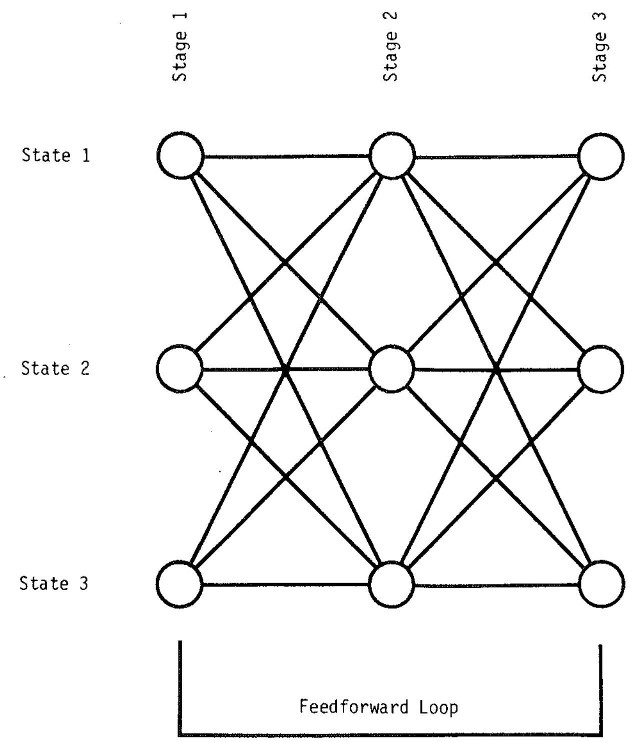

FIGURE 1 A three stage sequential system with one feedforward loop.

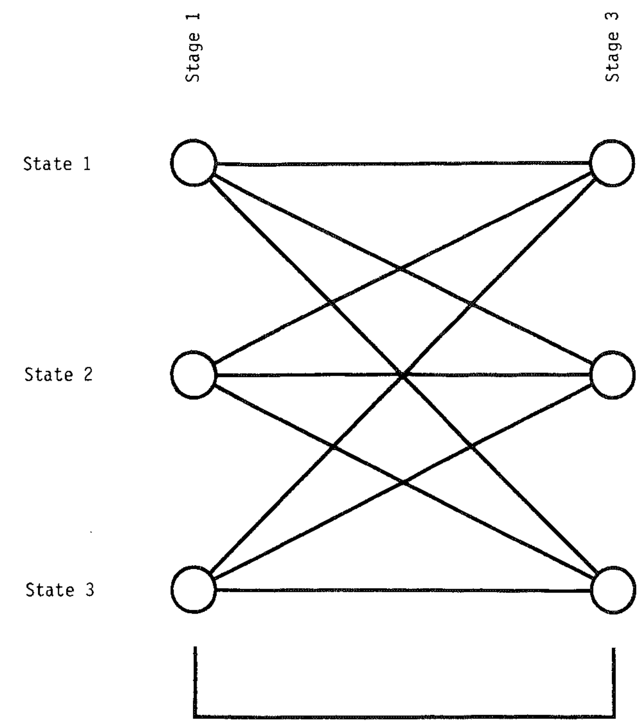

FIGURE 2 System of Figure 1 after condensation.

This could be represented as

$$\{n_1\}\ [p_1^{\,j}]\ \{n_2\}\ [p_2^{\,j}]\ \{n_3\}$$
$$\underbrace{\qquad\qquad\qquad}_{[p_{s13}]}$$

where

$P_{s13} = [p^j_{s13}]$ the return for the path in the feedforward loop going from the state j in stage 1 to state X in stage 3.

In order to solve this problem by dynamic programming there can only be one independent path from stage to stage. To achieve this it is necessary to condense the independent paths into one so that the serial nature of the formulation is preserved. In the example in Figure 1 the primary paths may be condensed between stages 1 and 3 to produce an equivalent path return from stages 1 to 3 shown in Figure 2.

Let

$P_{p13} = [p^j_{p13}]$ be the optimal return for the primary path going from state j in stage 1 to state X in stage 3

and

$P_{e13} = [p^j_{e13}]$ be the optimal return for the equivalent path going from state j in stage 1 to state X in stage 3,

i.e.

$$P_{e13} = P_{s13} + P_{p13}$$

The result after condensation can now be represented as

$$\{n_1\}\ [p^j_{e13}]\ \{n_3\}$$

which can be solved by the serial formulation of Eqs. (3) and (4).

2.1 General Formulation

The general formulation for a dynamic program with nested feedforward loops can now be outlined using the following steps:

1) Formulate problem with primary and secondary paths (Figure 3).
2) Condense all primary paths completely encompassed by feedforward (secondary) loops and replace with the equivalent path (Figure 4).
3) Repeat step 2 until all feedforward loops are absorbed (Figure 5).
4) Solve resultant problem (Figure 5) as a serial dynamic program to obtain the optimal decisions and return.

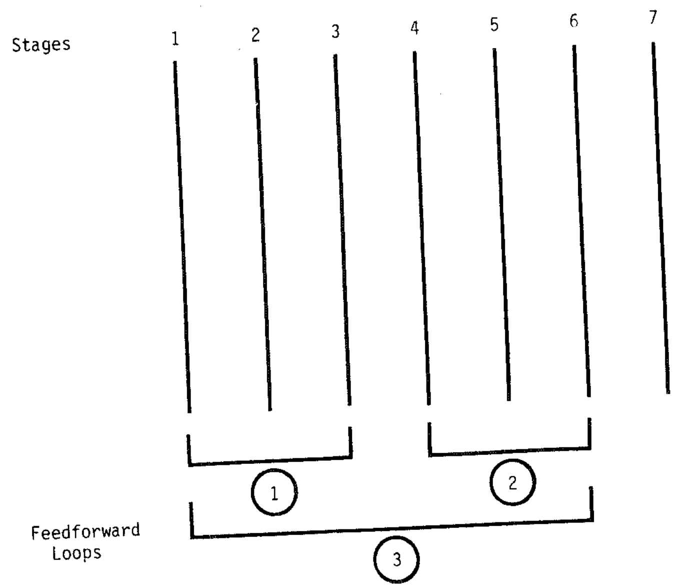

FIGURE 3 A seven stage sequential system with secondary paths (nested feedforward loops).

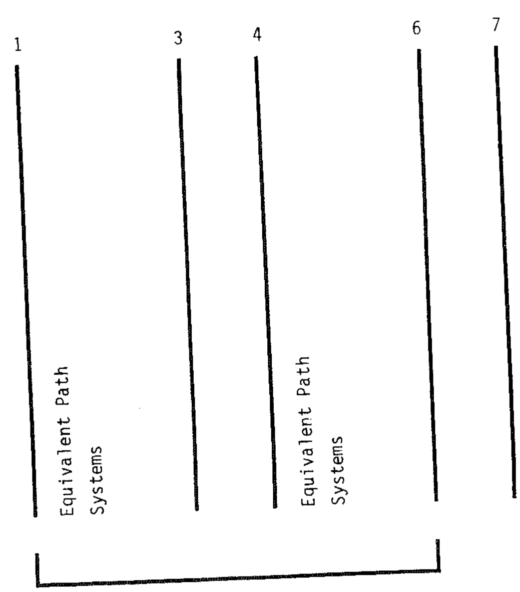

FIGURE 4 System of Figure 3 after two condensations.

Because of the procedure adopted here feedforward loops may be independent or nested but may not cross each other. If they did this would require an additional procedure to handle the additional dimensions produced. This occurs in problems which have extensive feedforward and feedback loops (Gero and Radford).[6]

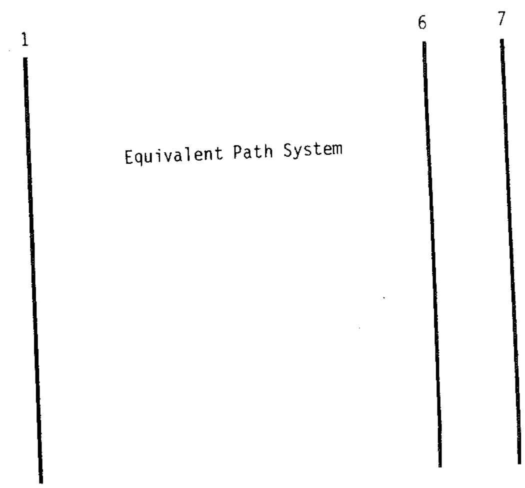

FIGURE 5 System of Figure 3 after all feedforward loops have been absorbed.

3 EXAMPLES

Two examples will be presented to demonstrate the applicability of feedforward nonserial dynamic programming to the design of building systems. The first deals with a problem soluble by serial dynamic programming which may be readily solved using feedforward because of the nature of the problem. The second is a problem which could not be formulated as a serial dynamic program.

3.1 Design of an Elementary Structural System

Aguilar[7] has presented and solved the following problem: Design the most economical raised platform to support a uniform load of 200 pounds per square foot of horizontal projection. The structural system consists of a rectangular surface 30 feet long and 20 feet wide; four columns each 40 feet in length, braced as required; and an appropriate foundation to safely transfer all forces to the ground.

The problem is shown diagrammatically in Figure 6. There are 7 platform types, 3 column types and 3 foundation types, see Table I. For the original formulation the reader is referred to Aguilar.[7] This system may be represented graphically in Figure 7. In this formulation costs rather than weight is the state variable. The node costs for the columns and foundations are based on the cost of one of the elements designed to carry a load of 130,000 pounds. The path costs are estimates of whatever additional costs would be incurred to use a particular column with a particular foundation in the design. The size of the foundation varies, inter alia,

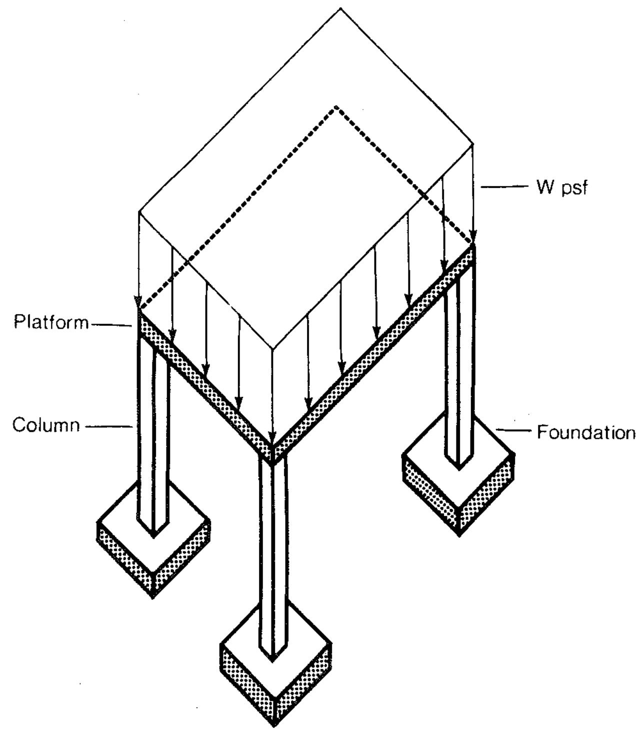

FIGURE 6 Diagrammatic representation of the elementary structural system design problem.

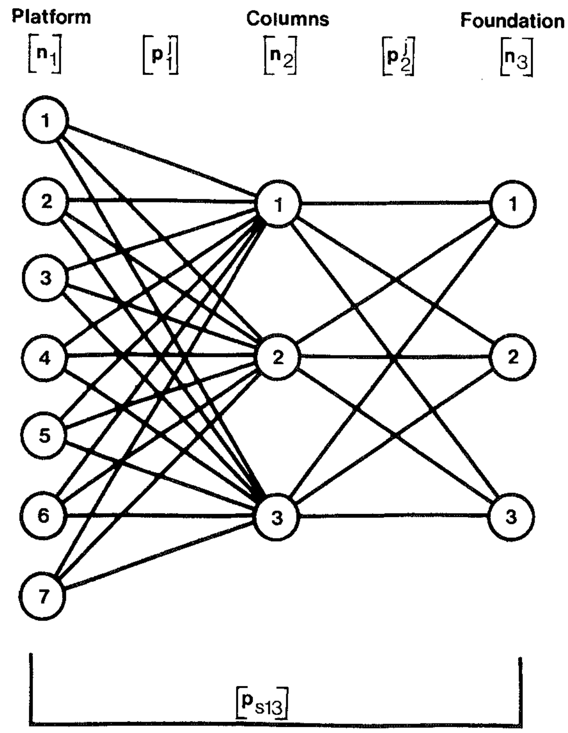

FIGURE 7 Graphical representation of the elementary structural system design problem in feedforward nonserial form.

TABLE I

Elementary structural system design—elements

States	Stages		
	Platforms	Columns	Foundations
1.	solid one-way concrete slab	reinforced concrete tied columns	spread footings
2.	one-way concrete pan joist	reinforced concrete spiral columns	drilled concrete piles
3.	solid two-way concrete slab	structural steel	driven steel piles
4.	concrete waffle slab		
5.	steel beams and steel deck		
6.	steel bar joists		
7.	steel beams and composite concrete deck		

with the type of platform used. This information can be conveyed as a cost in a feedforward or secondary loop from stage 1 to stage 3.

The various matrices are

$$\{n_1\} = \begin{bmatrix} 1100 \\ 1650 \\ 1300 \\ 1800 \\ 3000 \\ 2400 \\ 2000 \end{bmatrix} \quad [p_1^j] = \begin{bmatrix} 1765 & 2350 & 1865 \\ 710 & 1080 & 745 \\ 1070 & 1560 & 1165 \\ 450 & 720 & 450 \\ 50 & 80 & 50 \\ 100 & 160 & 100 \\ 500 & 800 & 500 \end{bmatrix} \quad \{n_2\} = \begin{bmatrix} 1400 \\ 1400 \\ 4000 \end{bmatrix} \quad [p_2^j] = \begin{bmatrix} 350 & 375 & 290 \\ 190 & 300 & 250 \\ 90 & 150 & 180 \end{bmatrix} \quad \{n_3\} = \begin{bmatrix} 400 \\ 600 \\ 600 \end{bmatrix}$$

$$[p^j_{s13}] \quad \begin{bmatrix} 1050 & 675 & 485 \\ 450 & 225 & 150 \\ 650 & 375 & 280 \\ 280 & 125 & 60 \\ 0 & 0 & 0 \\ 0 & 50 & 310 \\ 300 & 150 & 90 \end{bmatrix}$$

Thus,

$$[p^j_{p13}] = \begin{bmatrix} 3515 & 3540 & 3455 \\ 2460 & 2485 & 2400 \\ 2820 & 2845 & 2760 \\ 2200 & 2225 & 2140 \\ 1770 & 1780 & 1730 \\ 1850 & 1860 & 1790 \\ 2250 & 2275 & 2190 \end{bmatrix}$$

and

$$[p^j_{e13}] = \begin{bmatrix} 4565 & 4215 & 3940 \\ 2910 & 2710 & 2550 \\ 3470 & 3220 & 3010 \\ 2450 & 2350 & 2200 \\ 1770 & 1780 & 1730 \\ 1850 & 1910 & 2100 \\ 2550 & 2425 & 2280 \end{bmatrix}$$

Thus,

$$f_n(X_n) = \begin{bmatrix} 5640 \\ 4800 \\ 4910 \\ 4600 \\ 5170 \\ 4650 \\ 4880 \end{bmatrix}$$

The optimal return is \$4600 and, by tracing back, the optimal decisions are

platform — concrete waffle slab
columns — reinforced concrete tied
foundations — driven steel piles.

This result is identical with that obtained by Aguilar.

3.2. *Design of the Floor-Ceiling Sandwich in a Multi-Storey Building*

In a multi-storey building the floor-ceiling sandwich includes all the systems contained within the space defined by the ceiling below and the floor cover of the floor above it. This could include the following systems: structural; heat, ventilating and air conditioning; electrical and fire. Because of the complexity of interactions between these various subsystems designers have treated each separately on the assumption that the optimum design is simply the sum of each system optimized separately. This is not necessarily the case, feed-forward nonserial dynamic programming offers the opportunity to integrate these systems to produce the optimal design for the combined system.

3.2.1 The model Consider the multi-storey office building in which the outer core is structural and carries all the lateral as well as the vertical loads that are not carried by the service core. The service core provides the vertical circulation paths for all the other systems. Thus, the floor-ceiling sandwich is concerned with horizontal distribution.

In order to put numerical values onto the variables, let the building be 30 storeys high with a plan as shown in Figure 8. Assume that as part of the design the column spacing around the periphery is 20 ft. Due to symmetry only a typical two bay area needs to be considered.

Table II shows the results of the initial selection process of the designers of the various systems. For example, the structural system is composed of three subsystems: the framing, the reinforcing and the slab type. There are 6, 5 and 5 possible choices in each of these subsystems respectively. Similarly for the remaining systems. The decisions which produce this are outside the scope of this paper, they would form part of the preliminary design process which would define the limits on the resultant solution.

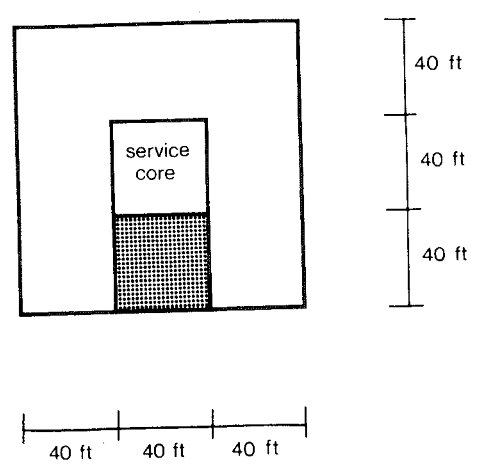

FIGURE 8 Plan of typical floor of building for which the floor-ceiling sandwich is to be designed.

TABLE II

Floor–ceiling sandwich design—available elements after initial selection

Structural systems

Frame	Reinforcing	Slab type
1. concrete girders and beams	one-way slab	cast in place reinforced concrete
2. concrete girders and joists	rib slab	precast reinforced concrete
3. steel girders and beams	two-way slab	precast prestressed
4. steel girders and joists	waffle slab	post tensioned
5. steel girders	cellular steel deck	concrete topping only
6. concrete girders		

Hvac system

HVAC System	Ducting	Terminal units	Return air
1. independent	conventional low velocity	mixing box	ducted
2 central	single, high velocity	mixing box with reheat	ceiling plenum
3.	double, high velocity	mixing box with variable volume	
4.		reheat box	
5.		reheat, variable volume	

Electrical system
1. through floor slab
2. below floor slab

Fire protection system
1. sprinklers
2. no sprinklers

In order to develop a suitable dynamic programming model from the systems in Table II the subsystems can be treated as the stages whilst the choices within the subsystems become the states. This generates a 9 stage dynamic program. Now the interactions between the states in one stage and the states in another stage need to be developed, these produce the feedforward loops. The solution methodology constrains these feedforward loops from intersecting. It is feasible to produce a good model of this problem within this restriction. One way is to rate the interactions between the subsystems and reduce the weak interactions to zero whilst increasing the strong interactions to unity to produce a binary interaction matrix to satisfy the above constraint. The resultant matrix may well be different for different designers and will certainly vary between geographical regions.

Since cost will be defined as the criterion for this problem the interactions will relate the cost interactions. A typical resultant interaction matrix for this problem is shown below. The number of the forward interactions between one subsystem and another is the row sum from the matrix, whilst the number of backward interactions is the column sum from the matrix.

	Frame	Reinforcing	Slab Type	HVAC System	Ducting	Term Units	Return Air	Electrical	Fire
Frame	–	1	1	0	0	0	0	0	0
Reinforcing	0	–	0	0	0	0	0	0	0
Slab Type	0	1	–	0	0	0	0	1	0
HVAC System	0	0	0	–	0	0	0	0	0
Ducting	1	1	0	1	–	0	0	0	0
Term Units	0	0	0	0	1	–	0	0	0
Return Air	0	1	0	0	0	0	–	0	0
Electrical	0	1	0	0	0	0	0	–	0
Fire	0	0	0	0	0	0	0	0	–

The ordinal function of the graph represented by this matrix can be used to sequence these subsystems into stages. Figure 9a shows the sequencing as a graph with six stages. Since it intended to treat each subsystem as a stage the graph can be "straightened out" and the Return Air and Fire subsystems arbitrarily placed as the first and last stage without altering the structure. Figure 9b shows this, the dotted connectivity lines are used to indicate that the path cost is zero.

The detailed model is shown in Figure 10 and now only requires numerical values for node and path costs. The objective will be to minimize the cost of the floor-ceiling sandwich.

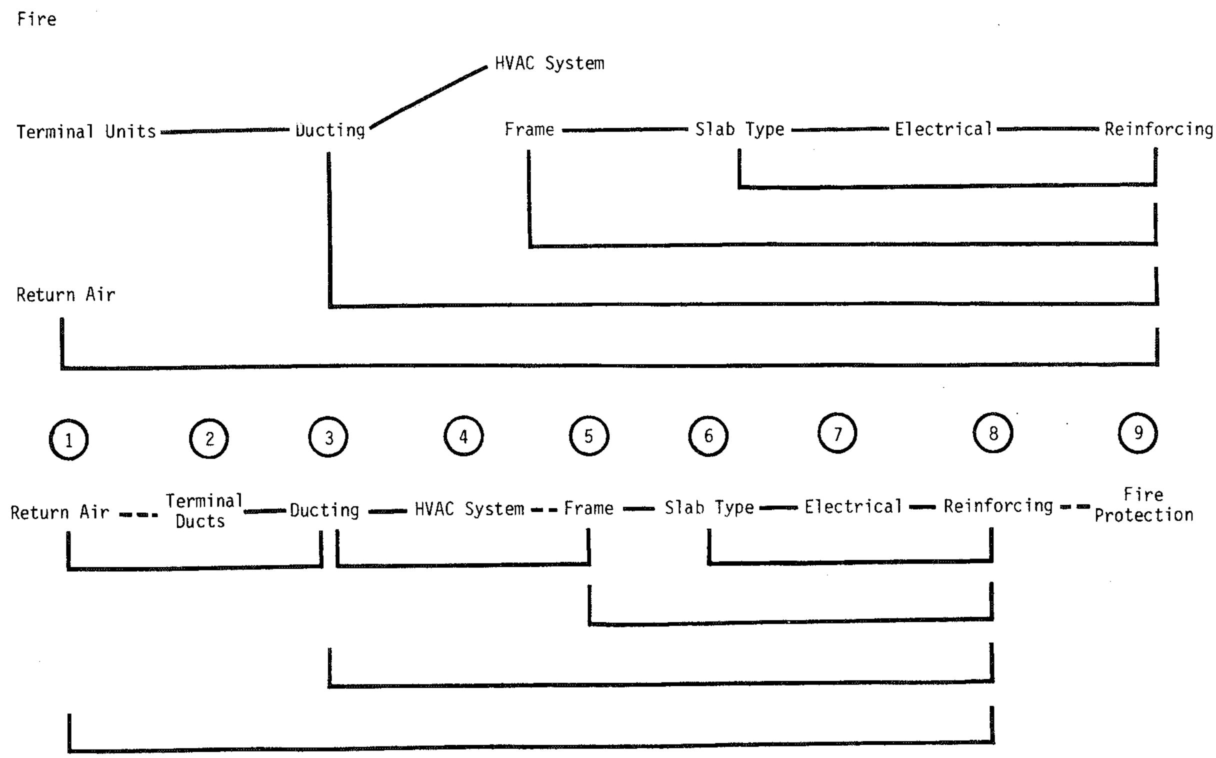

FIGURE 9 a) Graph of the sequencing of the dependent stages of the floor-ceiling sandwich problem. b) Graph of "straightened" sequence of stages. The heavy lines indicate non-zero path costs. The dotted lines indicate zero path costs (i.e., independence).

3.2.2 Node and path costs Careful study is required to produce the node and path costs since they are in a form which is not customary. The node cost is the relative cost of an item (state) in a stage. The path cost relates only to the cost of combining states. They reflect relative costs of different combinations. Thus, a path cost of zero would indicate that there is no supernumerary cost associated with that combination. The path cost is used to prevent two states being combined (because it is infeasible) by applying an infinite cost to that path. Shown below are the node and path cost matrices associated with the problem as explicated in Figure 10.

3.2.3 Node Costs

$$N_1 = \begin{bmatrix} 235 \\ 50 \end{bmatrix} \quad N_2 = \begin{bmatrix} 515 \\ 757 \\ 812 \\ 535 \\ 627 \end{bmatrix} \quad N_3 = \begin{bmatrix} 235 \\ 400 \\ 800 \end{bmatrix} \quad N_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad N_5 = \begin{bmatrix} 2178 \\ 2352 \\ 1790 \\ 1700 \\ 1449 \\ 1784 \end{bmatrix} \quad N_6 = \begin{bmatrix} 190 \\ 303 \\ 170 \\ 0 \\ 0 \end{bmatrix} \quad N_7 = \begin{bmatrix} 40 \\ 30 \end{bmatrix} \quad N_8 = \begin{bmatrix} 1650 \\ 1600 \\ 1810 \\ 1700 \\ 1668 \end{bmatrix} \quad N_9 = \begin{bmatrix} 690 \\ 370 \end{bmatrix}$$

3.2.4 Primary path costs

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad P_2 = \begin{bmatrix} \infty & \infty & 10 \\ \infty & \infty & 20 \\ \infty & \infty & 25 \\ 10 & 10 & \infty \\ \infty & 15 & \infty \end{bmatrix} \quad P_3 = \begin{bmatrix} 0 & 770 \\ 0 & 1027 \\ 0 & 1612 \end{bmatrix} \quad P_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 20 & 30 & 35 & 40 & 100 \\ 10 & 15 & 20 & 20 & 125 \\ 50 & 60 & 70 & 100 & 10 \\ 40 & 70 & 80 & 80 & 10 \\ 200 & 180 & 160 & 150 & 160 \\ 120 & 120 & 100 & 50 & \infty \end{bmatrix}$$

$$
\overset{P_6}{\begin{bmatrix} 30 & 40 \\ 40 & 30 \\ \infty & 20 \\ \infty & 20 \\ 0 & 60 \end{bmatrix}}
\overset{P_7}{\begin{bmatrix} 50 & 50 & 50 & 50 & 0 \\ 0 & 0 & 0 & 0 & 50 \end{bmatrix}}
\overset{P_8}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}
$$

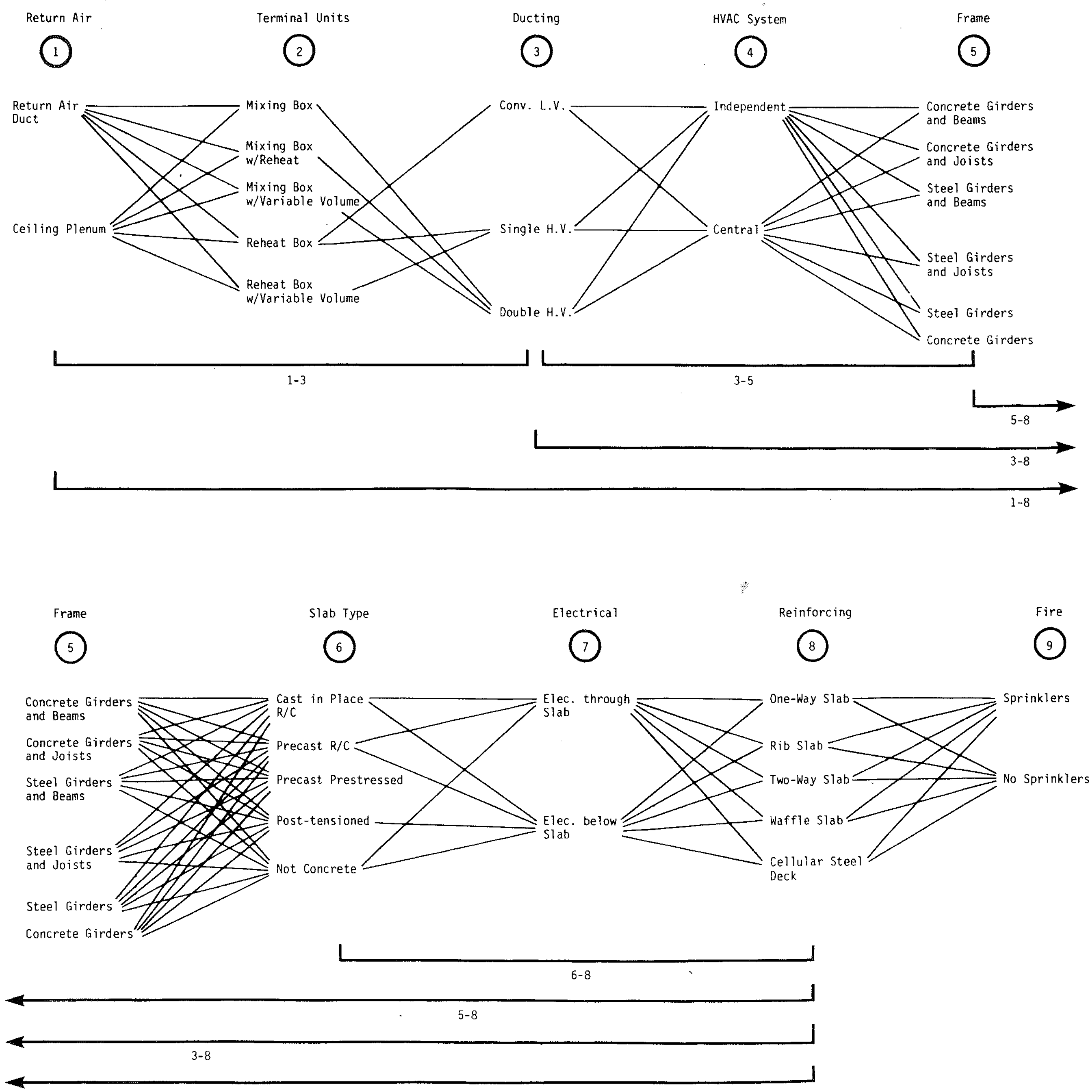

FIGURE 10 The nodes and detailed paths for the floor-ceiling sandwich design problem presented as a feedforward nonserial dynamic program with 9 stages.

3.2.5 Secondary path costs

$$P_{s13} = \begin{bmatrix} 15 & 15 & 15 \\ 60 & 60 & 60 \end{bmatrix} \quad P_{s35} = \begin{bmatrix} 125 & 120 & 115 & 110 & 0 & 0 \\ 125 & 120 & 115 & 110 & 0 & 0 \\ 140 & 130 & 125 & 120 & 0 & 0 \end{bmatrix} \quad P_{s68} = \begin{bmatrix} 913 & 668 & 890 & 892 & \infty \\ 38 & 22 & 32 & 20 & \infty \\ 759 & 648 & 937 & 782 & \infty \\ 1313 & 1241 & 1394 & 1343 & \infty \\ \infty & \infty & \infty & & 0 \end{bmatrix}$$

$$P_{s58} = \begin{bmatrix} 200 & 40 & 180 & 30 & 600 \\ 300 & 100 & 275 & 95 & 650 \\ 50 & 125 & 35 & 140 & 25 \\ 60 & 140 & 40 & 165 & 25 \\ 70 & 160 & 50 & 180 & 35 \\ 240 & 60 & 230 & 50 & \infty \end{bmatrix} \quad P_{s38} = \begin{bmatrix} 175 & 175 & 175 & 175 & \infty \\ 190 & 190 & 190 & 190 & 290 \\ 255 & 255 & 255 & 255 & 310 \end{bmatrix} \quad P_{s18} = \begin{bmatrix} 175 & 175 & 175 & 175 & 0 \\ 0 & 0 & 0 & 0 & 175 \end{bmatrix}$$

3.2.5 The solution For the data given the optimal solution derived using feedforward nonserial dynamic programming is

return air:	ceiling plenum
terminal units:	reheat box
ducting:	conventional low velocity
HVAC system:	independent
frame:	steel girders
slab type:	precast reinforced concrete
electrical:	below slab
reinforcing:	one way
fire:	no sprinklers

This can be summarized for the four original systems as:

structural system:	steel girders with one way precast reinforced concrete slab
HVAC system:	conventional low velocity ducting with reheat box, an independent system with ceiling plenum for return air
electrical system:	carried below slab
fire system:	no sprinklers

4 CONCLUSION

It has been shown that complex feedforward nonserial representations can be formulated as dynamic programs. This extends the applicability of dynamic programming as a design tool. Whilst the objective function in the examples used addition as the composition operator, considerably more complex functions can be used without changing the degree of difficulty of the solution procedure. Stability and sensitivity analyses can be carried out simply in the same manner as for serial dynamic programming. However, the concept of invariant imbedding needs to be re-examined in the light of the feed-forward loops to determine its validity.

ACKNOWLEDGEMENTS

This work has been variously supported by the US NSF (Project ENG75-02566) and the ARGC (Project F74/15278). The material has been drawn from a thesis by P. Sheehan.[8]

REFERENCES

1. J. S. Gero, "Architectural optimization—a review," *Eng. Opt.*, **1**, 3, pp. 189–199 (1975).
2. J. S. Gero, "Dynamic programming in the CAD of buildings," *CAD 76*, (G. Jones and D. Smith (eds)) (IPC Press, Guildford, pp. 31–37, 1976).
3. J. S. Gero, "Note on 'Synthesis and optimization of small rectangular floor plans'," *Env. and Plan. B.* Vol. 4, pp. 81–88 (1977).
4. R. E. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, N.J., 1957).
5. G. L. Nemhauser, *Introduction to Dynamic Programming* (John Wiley, New York, 1966).
6. J. S. Gero and A. D. Radford, "A dynamic programming approach to the optimum lighting problem", *Eng. Opt.*, **3**, 2, pp. 71–82 (1977).
7. R. J. Aguilar, *Systems Analysis and Design in Engineering, Architecture, Construction and Planning* (Prentice-Hall, Englewood Cliffs, N.J., pp. 296–301, 1973).
8. P. J. Sheehan, *"Application of Forward Feed Control Dynamic Programming to Building Design,"* M. S. Thesis (unpublished), Department of Civil Engineering, M.I.T., 1975.