

UC Berkeley

IURD Working Paper Series

Title

A Normalization Procedure in Building Methodology

Permalink

<https://escholarship.org/uc/item/37r066g9>

Author

Distefano, Nestor

Publication Date

1971-11-01

A NORMALIZATION PROCEDURE IN BUILDING METHODOLOGY

Néstor Distéfano

Departments of Architecture and Civil Engineering
University of California at Berkeley

Dennis Nagy

Institut für Statik und Dynamik, University of Stuttgart
(Formerly at the University of California, Berkeley)

November 1971

Working Paper No. 164

(To be presented at the Third EDRA and Eighth AR Conference, UCLA
January 22-24, 1972.)

ABSTRACT

In the design of large electrical, mechanical, structural, etc., systems, the architect frequently faces a normalization problem: given a system made by a large number of components, a procedure, generally a numerical procedure, is available for the determination of the minimum size required by each one of the components. The adoption of the sizes obtained in this fashion would then represent the optimum design solution in the sense of minimum cost if it were not for the well-known fact that repetition of components in a system yields a reduction in fabrication and assemblage costs. If the fractional reduction in cost due to repetition is assumed to be known for each type of component, the problem consists in determining the combination of sizes for which the greatest reduction is achieved. This is a combinatorial problem of vast arithmetic proportions unless a methodological approach is employed. In the present research a method based in the theory of Dynamic Programming has been developed using elements of Graph Theory and Optimization. The model and its solution is presented using as an example of application the problem of normalizing the structural sections of a housing project. Application to other areas of building methodologies is also discussed. A computer program and numerical illustrative results complete the presentation of the model. At present, a stochastic version of the model is being developed.

Introduction

At the "dimensioning" stage of large electrical, mechanical, structural, etc. systems, the architect frequently faces a number of normalization problems. In fact, at this stage the system will be formed by a large number of components whose dimensions have been determined by procedures that in general fail to recognize the true complexity of the system. For example, we possess very powerful methods to optimize the structural elements of a building but these procedures do not in general account for the interaction of the structural topology and the associated cost of fabrication and assemblage, and much less do they account for the interaction of the structure with other systems composing the entire building. The same is true in considering a number of mechanical, electrical, etc. systems. Clearly, the adoption of the optimum sizes obtained with those methods will not in general lead to the optimum design solution of the entire system, because what is optimum with respect to one function will be in general not optimum for two or more functions. This is particularly true when we consider operative and manufacture costs in addition to straightforward consideration of cost due to volume of materials. It is in effect known that repetition of components in a large system yields in general a reduction in fabrication and assemblage costs. A meaningful step in the design process is therefore to normalize the dimensions of the elements comprising a system in such a way as to achieve a reduction in cost. The present paper is devoted to a methodological treatment of this particular design stage. We shall consider here the simplest version of this problem as an example of a methodological approach that admits considerable generalization in the realm of building methodology. Assuming that the fractional reduction

in cost due to repetition of a number of elements is known, the problem now consists in determining the combination of sizes for which the greatest reduction in cost is achieved. This is a combinatorial problem of vast arithmetic proportions unless a methodological approach is employed. The dynamic programming algorithm of the allocation problem has been previously pointed out as an efficient technique for the solution of problems of this kind in the realm of structural design [1]. In this paper we present a methodological approach for the formulation and the solution of problems of the present type based in elements of Graph Theory and Optimization. This approach has proved to be highly efficient and easy to teach. This is demonstrated by the fact that it is the main theme for an undergraduate interdepartmental course in methodological aspects of design that is regularly taught by the authors to students of architecture and engineering at the Berkeley Campus.

The model and its solution is presented using as an example of application the problem of normalizing the structural sections of a housing project. Application to other areas of building methodologies are also discussed. A computer program and numerical illustrative results complete the presentation of the model. At present, a stochastic version of the model is being developed.

The Model

A system containing n components (elements) of the same category is to be designed to perform adequately under various conditions. The definition of adequacy of performance depends on the type of system under consideration. In a building, for example, the relevant components might be all the beams of the same length in the structural frame. The performance of the system would then be its response to various loading

conditions and the adequacy of response might be composed of certain requirements on stress and deflection values, yielding of material, or failure of members due to buckling or fatigue. It is assumed that methods of analysis and design relevant to the conditions of adequacy are available to determine the minimum allowable component sizes, i.e., to select a set of n components, each of which is just sufficient to satisfy all conditions of adequate performance. This set of components represents a design for the system. In many systems of interest, adequacy of performance is an increasing function of the amount of material used, in efficient system components. Thus, if there exists a finite, discrete set of m available component sizes (and thus m different levels of performance) from which to choose, the selected set of minimum allowable performance components for the system can be expressed as

$$n_j \text{ components of size } j, \quad j=1,2,\dots,m \quad (1)$$

with $\sum_{j=1}^m n_j = n$, the total number of components in the system (2)

and this design is the minimum weight design due to the relationship between performance and amount of material.

For some classes of systems (e.g., structures such as airplanes and spacecraft) weight is the predominant factor in determining the ultimate total cost of constructing and using the system, and thus the minimum-weight design is also effectively the minimum-cost design. However, in many other systems the fabrication costs are equal or even dominant factors with the material and weight-penalty costs. Fabrication costs per component can often be reduced through normalization of components, i.e., the manufacture of many identical components, but the material costs increase because some components in the system are then

larger or heavier than necessary for adequate performance. The problem is thus a smoothing problem of selecting the set of components that minimizes the total cost by balancing these two competing effects.

If the m different component sizes are arranged in order of decreasing size or weight, then we define

$$c_j = \text{the cost of one component of size } j, \text{ assuming} \\ \text{only one is fabricated } (j=1,2,\dots,m) \quad (3)$$

and

$$r_j(i) = \text{overall fractional cost reduction factor for} \\ \text{size } j \text{ if } i \text{ components of that size are fabrica-} \\ \text{ted together.} \quad (4)$$

Then the total cost of k_j components of size j is given by

$$a(k_j, j) = k_j c_j [1 - r_j(k_j)] \quad (5)$$

and the total cost of the system is

$$A = \sum_{j=1}^m a(k_j, j). \quad (6)$$

An artificial stratification of the selection problem is possible because of the separable form (6) of the total cost function.

The optimization problem of minimizing cost may then be viewed as an m -stage sequential component selection process, with constraints on the selection (decision) imposed by the requirements of adequate system performance. These constraints are given by the previously determined minimum-performance design (1), (2):

$$\begin{aligned} &\text{at least } n_1 \text{ components of size 1 (the largest)} \\ &\text{at least } n_2 \text{ components of size 2 or larger} \\ &\text{at least } n_3 \text{ components of size 3 or larger} \\ &\quad \cdot \\ &\quad \cdot \\ &\text{at least } n_m \text{ components of size } m \text{ or larger} \end{aligned} \quad (7)$$

If the selection begins with the largest size component, these constraints become

$$\text{at stage (size) } j, \quad N_j = \sum_{i=1}^j k_j \geq \sum_{i=1}^j n_j \quad (8)$$

where k_j is the number of components chosen to be of size j .

The Methodology

In many types of discrete-state, multistage decision processes, a map or graph is useful in illustrating the process and visualizing its solution procedure. If the horizontal axis of the graph represents successive component sizes and the vertical axis represents cumulative number of components (N_j) chosen up to and including size (stage) j , then the graph of possible combinations of selected elements appears as in Figure 1. A possible combination of selected elements (a sequence of decisions or a policy) is represented by a path from point (0,0) to point (n,m). All possible paths are bounded by two limiting system designs:

- 1) the fully-normalized or uniform-component design (upper bound)
- 2) the minimum component performance of minimum-weight design (lower bound).

In terms of coordinates on the graph, for example, the paths representing these two designs can be written as

$$\begin{aligned} \text{UNIFORM COMPONENT: } & (0,0) \rightarrow (n,1) \rightarrow (n,2) \rightarrow \dots \rightarrow (n,m) \\ \text{MINIMUM COMPONENT: } & (0,0) \rightarrow (n_1,1) \rightarrow (n_1+n_2,2) \rightarrow \dots \rightarrow (n,m) \end{aligned} \quad (9)$$

Exhaustive enumeration of all allowable paths in the graph (Figure 1) would be one way of computing the costs of different system designs and finding, by direct search, the lowest resulting cost. However, the number of allowable paths is of the order of $\left(\frac{n}{2}\right)^m$, which makes direct enumeration computationally impossible for large systems.

An approach to multistage analysis called dynamic programming [2] is ideally suited for this class of problems because it exploits the separable nature of the cost function (equation 6) to obtain an efficient recursive solution procedure. The dynamic programming approach, which renders much larger problems computationally solvable than does a direct enumeration approach, will be explained in the following section.

The Dynamic Programming Solution

Following the ideas of discrete dynamic programming [2], we define

$$f_j(N_j) = \text{the minimum cost after selection of } N_j \text{ total components from among } j \text{ successive sizes.} \quad (10)$$

Then the Principle of Optimality of dynamic programming (page 15 of [2]) provides the key to obtaining the recursion relations needed to solve this problem. From equation (6), equation (10) becomes

$$f_j(N_j) = \text{minimum}_{k_1, k_2, \dots, k_j} \left[\sum_{i=1}^j a(k_i, i) \right] \quad (11)$$

Proceeding formally with equation (11), we may separate the minimum operation to obtain

$$f_j(N_j) = \min_{k_j} \left[\text{minimum}_{k_1, k_2, \dots, k_{j-1}} \left\{ \sum_{i=1}^j a(k_i, i) \right\} \right] \quad (12)$$

Then (12) becomes, by separation of the summation terms,

$$f_j(N_j) = \min_{k_j} \left[a(k_j, j) + \text{minimum}_{k_1, k_2, \dots, k_{j-1}} \left\{ \sum_{i=1}^{j-1} a(k_i, i) \right\} \right] \quad (13)$$

From the definition (10) and equation (11), we finally obtain

$$f_j(N_j) = \min_{k_j} \left[a(k_j, j) + f_{j-1}(N_j - k_j) \right] \quad (14)$$

Upon substitution of (5) and consideration of the constraints (7), (14) can finally be written as a set of functional recurrence relations

$$f_j(N_j) = \underset{0 \leq k_j \leq L_j}{\text{minimum}} \left[k_j c_j \{1 - r_j(k_j)\} + f_{j-1}(N_j - k_j) \right] \quad (15)$$

(j=2,3,...,m)

where

$$L_j = N_j - \sum_{i=1}^{j-1} n_i, \quad \left(\sum_{i=1}^j n_i \leq N_j \leq n \right) \quad (16)$$

and

$$f_1(N_1) = N_1 c_1 [1 - r_1(N_1)] \quad (\text{initial condition}) \quad (17)$$

The equations (15), (16), and (17) are then solved successively for each value of j , and at each value of j for each possible value of N_j , up to the end-point $f_m(n)$, which is then the minimum cost of the total system. The selected optimal policy (optimal sequence) of component sizes is then determined by a trace-back through the sets of values $k_j^*(N_j)$ which satisfied the minima in equation (15) at each stage j and each level N_j . In the following section, a FORTRAN subroutine computer program is developed to perform the dynamic programming solution (15)-(17).

The Computer Program

The digital computer solution of the above general component selection problem may be accomplished by a FORTRAN subroutine. Define the following FORTRAN variables:

Data variables -

NBT = number of components in the total system (n)

NT = number of different component sizes available (m)

Data arrays (dimensions) -

COSTF(NT) = cost of one component of each size, if only one is fabricated (c_j)

CRED(NBT) = overall cost reduction factors for multiple-component fabrication ($r[i]$)

NBMIN(NT) = number of components of each type in the minimum weight design (n_j)

Computation and storage arrays (dimensions) -

F(NBT,2) = current stage (col. 2) and previous stage (col. 1) minimum cost values ($f_j[N_j]$)

IPOL(NBT,NT) = selection policy values ($k_j^*[N_j]$)

Result or solution array (dimension) -

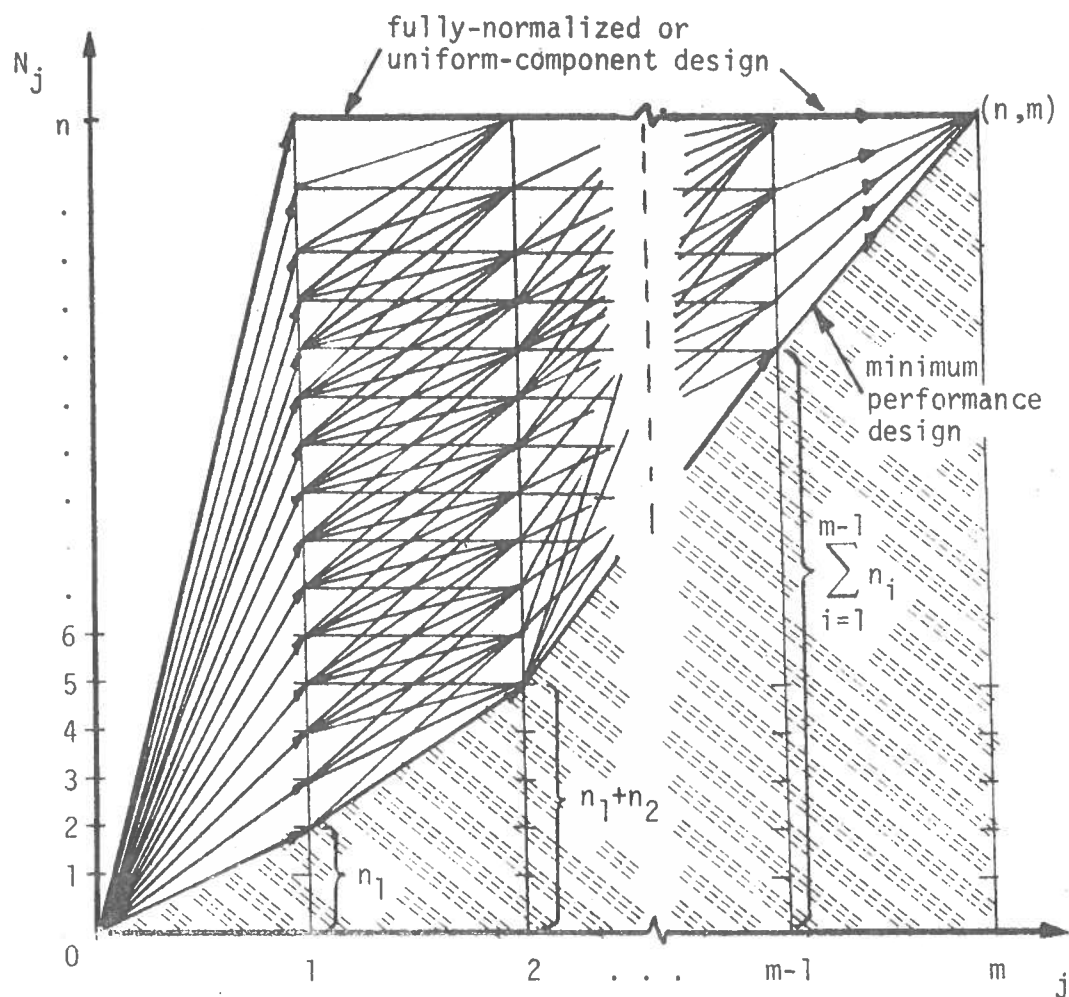
NBOPT(NT) = number of components of each size selected to minimize the total cost of the structure

The minimum cost of the system is located in F(NBT,1) at the end of execution of the subroutine. A complete FORTRAN listing of subroutine MCESDP (for minimum cost element selection by dynamic programming) appears in Figure 2.

A Sample Problem

Consider a three-story, multi-bay building frame containing 15 beams of the same length dimension. The structure is to be constructed from pre-cast, pre-stressed concrete members. Some method of hand calculations or a computer analysis, e.g. SAP (Structural Analysis Program) has been used to determine the minimum-weight design. This design, consisting of five different beam cross-section sizes, is given in Table 1. Costs for each size of each member, assuming only one member of that size is being fabricated, are also given in Table 1. The information supplied by the concrete beam manufacturers concerning the cost reduction factors for multiple-beam fabrication is given in Table 2.

The solution of this problem is straightforward. Subroutine MCESDP, coupled with a main program for Input/Output, finds the minimum-cost design in .014 seconds of CDC-6400 central processor time (Table 3). The output of the main program includes Tables 1, 2, and 3.



STAGES = component sizes (in decreasing order)

(shaded portion of graph violates adequate-performance)

FIGURE 1 DIRECTED GRAPH OF POSSIBLE COMPONENT SELECTION COMBINATIONS IN THE DYNAMIC PROGRAMMING APPROACH

```

SUBROUTINE MCESDP(NT,NBT,COSTF,NEMII,CILD,NBOPT,F,IPCL)
C*****
C
C SUBROUTINE FOR THE MINIMUM-COST SELECTION OF COMPONENTS OF A SYSTEM
C BY DYNAMIC PROGRAMMING
C*****
C DIMENSION COSTF(NT),NBMIN(NT),CRED(NBT),NBOPT(NT),F(NBT,NT),IPCL(NB
  1T,NT)
C*****
C FIRST STAGE (SIZE)
C*****
      II=NBMIN(1)
      DO 1 I=II,NBT
        IPCL(I,1)=1
        1 F(I,1)=I*COSTF(1)*(1.-CRED(1))
C*****
C LOOP FOR REMAINING STAGES (2 THROUGH NT)
C*****
      DO 2 J=2,NT
        IPK=II
        1 I=I+NBMIN(J)
C*****
C LOOP FOR ALL ALLOWABLE N(J) VALUES (LEVELS)
C*****
        DO 3 I=II,NBT
          FMIN=F(I,2)
          IPCL(I,J)=0
          KMAX=I-IPK
C*****
C LOOP FOR PERFORMING MINIMIZATION AT EACH POINT IN THE GRAPH (I,J)
C*****
          DO 4 K=1,KMAX
            TEST=F(I-K,1)+K*COSTF(J)*(1.-CRED(K))
            IF(TEST.GT.FMIN) GO TO 4
            FMIN=TEST
            IPCL(I,J)=K
          4 CONTINUE
          3 F(I,2)=FMIN
C*****
C SHIFT VALUES OF MINIMUM-COST FUNCTION TO PREVIOUS COLUMN
C*****
          DO 6 I=II,NBT
            6 F(I,1)=F(I,2)
          2 CONTINUE
C*****
C TRACE-BACK THROUGH POLICY ARRAY TO OBTAIN NUMBER
C OF COMPONENTS OF EACH TYPE SELECTED
C*****
          DO 5 J=1,NT
            NBOPT(NT+1-J)=IPCL(II,NT+1-J)
          5 II=I+NBMIN(NT+1-J)
          RETURN
          END

```

Figure 2

DYNAMIC PROGRAMMING SELECTION OF 15 STRUCTURAL
ELEMENTS FROM AMONG 5 TYPES, FOR MINIMUM COST.

Table 1. Element Type Data

Element Type	Cost for One Element	Number of Elements in Min. Wt. Design
1	110.50	3
2	90.00	1
3	88.50	8
4	65.00	1
5	50.00	2

Table 2. Cost Reduction Factors

Number of Similar Elements Used	Overall Cost Reduction Factor
1	.
2	.070
3	.110
4	.140
5	.140
6	.140
7	.140
8	.140
9	.140
10	.140
11	.140
12	.140
13	.140
14	.140
15	.140

Table 3. Minimum Cost Selection Table

Element Type	Number of Elements	Cost (In Dollars)
1	3	295.03
2	4	309.60
3	5	380.55
4	1	65.00
5	2	93.00

Total elements = 15 Total cost = 1143.18 dollars
 Selection took .014 seconds of CDC-6400 Central Processor time.

REFERENCES

1. Moses, F., and Goble, G., "Minimum-Cost Structures by Dynamic Programming," AISC Engineering Journal, July 1970, pp. 97-100.
2. Bellman, R., and Dreyfus, S. E., Applied Dynamic Programming, Princeton University Press, 1962.