RATIONALIZING TOOL SELECTION IN A FLEXIBLE MANUFACTURING SYSTEM FOR SHEET-METAL PRODUCTS

MARK DASKIN and PHILIP C. JONES

Northwestern University, Evanston, Illinois

TIMOTHY J. LOWE

University of Iowa, Iowa City, Iowa

(Received August 1988; revisions received July, October 1989; accepted November 1989)

Substantial simplification of existing processes and designs may be required before the potential benefits of modern manufacturing technology can be realized. This paper analyzes implementation problems associated with a flexible system that produces flat sheet-metal parts with interior holes. The paper makes three main contributions. First, we formulate the problem of selecting tooling and design standards as an optimization model and demonstrate that the model yields insight by applying it to one manufacturer's problem, thereby reducing substantially the required tooling. Second, we show that the model has a totally balanced constraint matrix, and hence, there are polynomial time algorithms for various versions of the problem. Third, we provide new algorithms with substantially improved performance bounds for two important versions of the problem.

The traditional method for producing flat sheet-metal parts, punch and die technology, requires that a separate die be designed and built for each part. After the die is mounted and adjusted, cycle time is small because a single operation forms perimeters and punches all interior holes. One drawback of punch and die technology is that setups can be time consuming (often more than 1 shift), thereby inducing large production lot sizes with correspondingly high work-in-process inventory costs. Another drawback is that part redesign often requires designing and producing a new die, an expensive and lengthy process.

Laser punch press technology, a new manufacturing method, has the potential to overcome these drawbacks of punch and die technology, thereby significantly increasing the firm's flexibility in production and redesign. The laser cuts the perimeter and some irregularly shaped holes in the interior of a part. A flexible arm with a gripper withdraws tools from a rack to punch the remaining interior holes one at a time. Although cycle times are often larger than their punch and die technology counterparts, setups can virtually be eliminated.

One of the authors worked on a project with a medium-sized Chicago area firm whose management was very concerned about how to introduce an expensive, new laser punch press into their existing operations. Each different size and shape of hole requires a different tool, and a major issue in implementing the new technology is selecting appropriate tooling. To see why tool selection is significant, note that the firm's existing designs had been developed under the previous punch and die technology when a different die was designed and produced for each part. Standardizing hole sizes between parts would have made the dies no less difficult or expensive to manufacture, and hence, the firm previously had no compelling financial incentive to standardize. Indeed, when we examined data for existing designs of the firm's over 3,000 active parts, we found that existing designs required over 900 different specifications for round holes alone, where a specification is a combination of hole diameter, tolerance, and material thickness. (Material thickness is included because different material thicknesses require different tools, even when hole diameters are identical.) Furthermore, ignoring material thickness, there were over 470 different combinations of hole diameter and tolerances with over 380 different nominal diameters alone.

With the new laser punch press, the situation is different. The same tool can (and will) punch holes of the same size in different parts. Also, the rack on the new punch press holds a maximum of 200 tools. Thus, with the new technology, standardization is an important economic issue for two reasons: First, tooling is very expensive, even when using standard tools in stock sizes. If custom tooling is required, it is even more costly. Thus, using fewer tools and eliminating

Subject classifications: Facilities/design: tooling. Inventory/production: applications. Manufacturing: automated systems.

0030-364X/90/3806-1104 $01.25

as much custom tooling as possible can yield significant savings. Second, if the 200 tool capacity of the tool rack is exceeded, setups are required to mount different tools in the rack and to reprogram the system. Since eliminating setups is a primary advantage of the new system, implementing it in a fashion that requires frequent setups is an ineffective strategy.

The most obvious way to purchase tooling for producing existing designs is to order a tool that corresponds to each different combination of material thickness and hole diameter. This obvious approach, however, was ineffective for the firm under consideration. (As mentioned before, this approach would have required the firm to purchase over 900 different tools for round holes alone. Furthermore, most of these tools would have to be custom-made.)

Thus, introducing new technology into an already existing production environment can be considerably more difficult than it would be in a *clean-slate* situation. This paper shows how optimization modeling can be used to gain insight into some of the implementation and design problems associated with introducing laser punch press technology. Tang and Denardo (1988a, b) examine some of the operational issues associated with scheduling tool changes in such flexible manufacturing systems.

We now give an overview of the rest of the paper. Section 1 presents a more detailed statement of the problem, formulates the problem as an optimization model, and discusses how different versions of the model produce insight into different aspects of the problem. Section 2 shows that the constraint matrix of the proposed model is totally balanced. This allows us to apply polynomial-time algorithms of Broin and Lowe (1986) as well as Hoffman, Kolen and Sakarovitch (1985) to different versions of the problem. Sections 3 and 4 present algorithms that are substantial improvements over previous algorithms for solving two important versions of the problem. Specifically, Section 3 derives a linear-time algorithm for selecting the minimal cardinality set of tools that can punch all the required holes. Section 4 develops a dynamic programming algorithm that solves versions of the problem in which the upper bound on the number of tools (imposed by tool-rack capacity) becomes a binding constraint. Section 5 presents results based on data from a manufacturing firm.

1. PROBLEM STATEMENT AND MODEL

Our presentation concentrates on the case of round holes. When a design requires a round hole, the nominal diameter of the hole as well as its required tolerance must be specified. A particular hole may, for example, have a nominal diameter of 0.150 inch with tolerances of +0.007 inch and −0.009 inch.

Each hole specification, therefore, defines an interval on the real line. For the example immediately above, the corresponding interval is [0.141, 0.157]. Similarly, each tool defines an interval on the real line. According to Bolz (1977), tools can be expected to punch holes to tolerances of +0.0015 inch, −0.0000 inch. Thus, the interval corresponding to a tool whose nominal diameter is 0.1250 inch would be [0.1250, 0.1265]. The nominal diameter of the tool need not equal the nominal diameter of the hole for the tool to be able to punch the hole. For example, if a hole with a nominal diameter of 0.1260 inch had a tolerance of ±0.005 inch, it could be punched by the tool described above because the hole actually punched (defined by the tool interval) would be within specified tolerances (defined by the hole interval). Figure 1 provides an illustration.

Since a hole is defined by specifying its nominal diameter and tolerances, the same hole may be punched in many different parts and may also be punched more than once in the same part. Clearly, the same tool can be used to punch every instance of the same hole, however, several important issues immediately arise.

First, how many tools (and which ones) will it take to punch all the holes required by current part designs? Furthermore, if we restrict ourselves to standard tooling, what proportion of the holes can be punched? Second, how does the 200 tool limit imposed by the tool rack capacity affect the analysis? Third, how does redesign affect the analysis? Which holes are good candidates for redesign, and how much redesign will be required to punch all holes with standard tools?

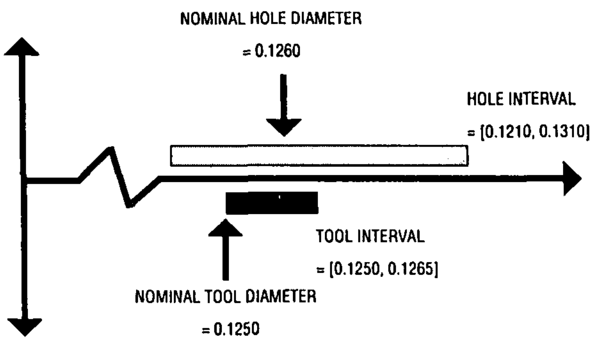

Figure 1. Example hole and tool intervals on the real line.

Fourth, if cost data on tools and out-sourcing options are available, what selection of tools and out-sourcing options can produce the parts at minimum cost?

To proceed, we define the following notation. We assume that there are k distinct parts to be processed and m distinct holes, characterized by nominal diameter and tolerances, to be punched. Furthermore, we assume that there are n distinct tools that can be selected. Let h_{ik} denote the number of times hole i must be punched in each unit of part k, and let p_k denote the annual production rate of part k. Let d_i denote the total number of times hole i is punched per year. We call d_i the annual number of *hits* for hole i. Clearly

$$d_i = \sum_k h_{ik} p_k.$$

H_i will denote the interval corresponding to hole i and T_j will denote the interval corresponding to tool j. Then tool j can punch hole i within the specified tolerances if and only if H_i contains T_j. In this event, we say that tool j *covers* hole i.

We define an $(m \times n)$ 0–1 matrix, hereafter denoted by A, whose rows correspond to holes and whose columns correspond to tools as

$$A_{ij} = \begin{cases} 1 & \text{if tool } j \text{ covers hole } i \\ 0 & \text{otherwise.} \end{cases}$$

We also define a 0–1 column vector of decision variables, x, whose components correspond to tools.

$$x_j = \begin{cases} 1 & \text{if tool } j \text{ is chosen} \\ 0 & \text{otherwise.} \end{cases}$$

We can then define the optimization problem.

$$\begin{aligned} \text{Minimize} \quad & c^t x + b^t z \\ \text{subject to} \quad & Ax + Iz \geqslant e \\ & e^t x \leqslant u \\ & x = 0 \text{ or } 1 \\ & z = 0 \text{ or } 1 \end{aligned}$$

where I denotes the identity matrix, e is a column vector of ones, the superscript "t" denotes transposition, u is the upper bound given by tool-rack capacity, and the definitions of the vectors c and b (discussed below) depend upon the problem being analyzed. Note that for a given vector, x, if the ith row of Ax is equal to zero (none of the tools indicated by the vector x can punch hole i), then z_i must be set to one. We interpret b_i as a positive penalty cost for not punching hole i.

To see how this model can be used to gain insight into the first issue discussed above (the minimum number of tools to punch all holes), let b be a vector of numbers greater than one, let c be a vector of ones, and delete the upper bound constraint. It will now be less costly ($c = 1$) to choose a tool than to not punch the hole ($b > 1$). The optimization problem, therefore, identifies the minimum number of tools needed to cover all holes. Suppose that we restrict tooling to standard tools by using only an appropriate subset of the columns of A, let c be a vector of zeros, set $b_i = d_i$ for each i, and delete the upper bound constraint. The objective of the formulated problem is to minimize the number of annual *hits* that cannot be punched by standard tools, or, equivalently, to maximize the number of annual *hits* that can be punched.

The second issue of analyzing the effects of tool-rack capacity on tooling selection can be investigated by incorporating the upper bound constraint. For example, if c is a vector of zeros and $b_i = d_i$ for each i, then the formulation minimizes the annual number of *hits* that cannot be punched by a fixed number of tools. As before, if we are only interested in selecting standard tools, we can eliminate all columns of A that correspond to custom tools. It is important to note that the upper bound constraint may not be binding. In particular, for the industrial application we have mentioned previously, we shall see that this constraint was not binding, even when custom tooling was permitted. This is somewhat surprising in that the naive approach of selecting a tool that corresponds to each different nominal hole diameter would have selected over 380 tools (and this makes the rather heroic assumption that all parts could be redesigned to use the same material thickness), a considerable violation of the 200 tool limit.

To analyze the third issue of investigating possible redesign options, we make appropriate changes in A. That is, if a hole is redesigned, it will be covered by a different set of tools than before. Although discussed in greater detail in the example of Section 5, our approach is to relax hole tolerances somewhat, thus allowing greater flexibility in choosing tools.

Finally, suppose components of the vector c represent the purchase prices of new tools and components of the vector b represent penalty costs for not being able to punch particular holes. The optimization model would then select a set of tools that minimizes the combined tooling plus penalty costs.

The problem we have formulated is a generalization of the integer programming problem known as the *maximum covering problem* (see, for example, Church and ReVelle 1974). Although this problem is, in general, NP-complete, good heuristics exist. The paper by

Eaton, Church and ReVelle (1977) is a basic reference on heuristics for the maximum covering problem. However, as we shall show in the next section, the special instance of the maximum covering problem we consider in this paper has a totally balanced constraint matrix, and hence, polynomial-time algorithms for determining optimal solutions exist. Thus, we need not rely upon heuristics.

2. TOTALLY BALANCED AND GREEDY MATRICES

In this section, we show that the constraint matrix, A, of the application defined in the previous section, is totally balanced. If A is an arbitrary (0, 1) matrix, the maximum covering problem is NP-complete. Although good heuristics (greedy adding and Lagrangian relaxation) exist, these heuristics are not guaranteed to find optimal solutions. Polynomial algorithms exist, however, to solve several important cases of the problem just identified, provided that A is totally balanced. The problem variants, discussed in the previous section, that delete the upper bound constraint are equivalent to the problem considered by Hoffman, Kolen and Sakarovitch, who provided an $o(mn)$ time algorithm (where m and n are the number of rows and columns, respectively, in A) to determine an optimal solution. If, in analyzing the effects of tool-rack capacity upon tool selection, the upper bound constraint is binding, the problem is equivalent to the maximum covering problem considered by Broin and Lowe who provide an $o(u^2n \min\{m^2, n^2\})$ time algorithm, where u is the upper bound. To proceed, we will need some formal definitions.

Let M_k, $k > 2$, be the family of k by k (0, 1) matrices, whose row and column sums equal 2, and that do not contain the submatrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

A (0, 1) matrix (see Nemhauser and Wolsey 1988) is *totally balanced* if it does not contain a submatrix in M_k for any $k > 2$. A (0, 1) matrix is said to be *standard greedy* (Hoffman, Kolen and Sakarovitch) if it does not contain

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

as a submatrix. (Standard greedy matrices are called *row inclusion* matrices in Nemhauser and Woolsey.)

A standard greedy matrix is totally balanced, and any totally balanced matrix can be converted to a standard greedy matrix in $o(nm^2)$ time (see Hoffman, Kolen and Sakarovitch). The polynomial-time algorithms, referred to previously, require the matrix A to be in standard greedy form. We will show that, by indexing the holes and tools in the proper way, A is in standard greedy form. Clearly, it is advantageous to start with a matrix in standard greedy form.

Recall that each hole has an associated hole interval. Let L_i and R_i denote the left and right limits, respectively, of the hole interval associated with hole i. Similarly, each tool has an associated tool interval. Let TL_j and TR_j denote the left and right limits, respectively, of the tool interval associated with tool j.

We require that holes be ordered so that $R_1 \leq R_2 \leq \ldots \leq R_m$ and tools be ordered so that $TL_1 \leq TL_2 \leq \ldots \leq TL_n$. Later on, we will make the assumption that ordering tools by right interval limits will produce the same ordering—a completely reasonable assumption, since it is equivalent to assuming that no tool interval contains another. For Theorem 1, however, this assumption is not needed. In contrast, one hole interval may contain another, so ordering holes by left interval limits may produce a different ordering.

Theorem 1. *When holes (rows) are ordered by right hole intervals and tools (columns) are ordered by left tool intervals, then A is in standard greedy form.*

Proof. We must show that a submatrix of the form

$$\begin{array}{c|cc} & r & s \\ \hline p & 1 & 1 \\ q & 1 & 0 \end{array}$$

cannot occur. Let the submatrix be formed by rows p and q of A, $p < q$, and columns r and s of A, $r < s$. If a submatrix of this form occurs, then tool s does not cover hole q, but covers hole p. It must, therefore, be the case that either $TL_s < L_q$ or $TR_s > R_q$, or both. The second inequality cannot hold, for the ordering on holes would then imply that $R_p \leq R_q < TR_s$, and hence, tool s could not cover hole p. But the first inequality cannot hold either. For if it did, the ordering on tools ($TL_r < TL_s$) would imply, by an analogous argument, that tool r could not cover hole q. Hence, a submatrix of the form specified above cannot occur.

Observation 1. The validity of Theorem 1 depends upon the particular orderings chosen: if holes are ordered by left instead of right interval limits, the result is no longer true.

Observation 2. Holes can be ordered in $o(m \log m)$ time and tools can be ordered in $o(n \log n)$ time, where m and n are the numbers of holes and tools, respectively. Thus, the matrix A can be put in standard greedy form in $o(p \log p)$ time, where

$$p = \max\{m, n\}.$$

The complexity results reported in the remainder of the paper assume that the ordering of the holes and tools has been accomplished.

3. A LINEAR-TIME ALGORITHM

In this section, we give an $o(m + n)$ time algorithm for an important version of the problem. Specifically, we consider the problem of finding a minimum cardinality subset of tools to cover all the holes that can be covered by *at least one* tool. Three separate procedures, run serially, comprise the algorithm. The first procedure, called COVERABLE, identifies and removes those holes that cannot be covered by any tool. The output from the procedure is the subset of holes that can be covered by *at least one* tool. The output from COVERABLE is the input to the second procedure, called UNDOMINATED, which identifies and removes those holes that are dominated (defined below) by other holes. The output from this procedure (run serially after COVERABLE) is the subset of holes that are undominated and can be covered by at least one tool. Finally, the TOOL SELECTION procedure identifies a minimum cardinality subset of tools to cover the set of holes that remain in the data set after the first two procedures. In effect, COVERABLE finds and removes those rows of A that have all zero entries. Note that such a row could occur, for example, if we were concerned with determining which holes could be covered using only standard tools.

We continue to assume that the holes are ordered so that

$$R_1 \leqslant R_2 \leqslant \ldots \leqslant R_m$$

and the tools are ordered so that

$$TL_1 \leqslant TL_2 \leqslant \ldots \leqslant TL_n.$$

We also assume that ordering tools by their right interval limits produces the same ordering. As discussed previously, this assumption is not particularly stringent for realistic applications.

We say that hole i **dominates** hole λ if $L_\lambda \leqslant L_i$ and $R_\lambda \geqslant R_i$, with strict inequality holding in at least one of the inequalities, or if $L_\lambda = L_i$, $R_\lambda = R_i$, and $i < \lambda$. Note that in this case, the interval corresponding to hole i is properly contained in the interval corresponding to hole λ. Therefore, any tool that covers hole i must also cover hole λ. A hole, λ, is **undominated** if there is no hole, i, that dominates hole λ. The set of undominated holes returned by UNDOMINATED is labeled UNDOM. We note that by the definition of undominated holes and by the initial ordering of holes, if holes u and v are both in UNDOM and $u < v$, then $R_u < R_v$ and $L_u < L_v$.

The concept of dominance and problem reduction in set covering was used by Toregas and ReVelle (1972, 1973) in a location problem. In their application, a facility can "cover" a demand point if the site is within a certain travel distance of the demand location. A demand point is **dominated**, and removed from the problem, whenever there is some other demand point whose "covering facilities" are a subset of the "covering facilities" of the dominated demand.

As mentioned above, the first two procedures, COVERABLE and UNDOMINATED, may remove some holes from the initial data set. In what follows, m denotes the number of holes input to COVERABLE, m' denotes the number of holes input to UNDOMINATED, and m'' denotes the number of holes input to TOOL SELECTION. Since $m \geqslant m' \geqslant m''$, after each procedure we re-index the output set so that the hole indices are consecutive integers, while preserving the ordering by right hole interval limits.

Since the first two procedures perform preprocessing, we state the main procedure, TOOL SELECTION, initially. Then, we establish the validity of the procedure. Statements of COVERABLE and UNDOMINATED conclude the section.

Recall that TOOL SELECTION requires as input a set of holes that can be *covered* and are *undominated.* With this in mind, the best way to understand TOOL SELECTION is to imagine all of the left and right interval limits, for both holes and tools, laid out on the real line (see Figure 1). The algorithm starts with the first (leftmost) hole and finds the largest indexed (rightmost) tool that covers the hole. This tool is chosen, and all holes covered by this tool are removed from the list. We will show that the holes this tool covers are indexed consecutively. The first (leftmost) remaining hole is identified, and we repeat the process until all holes are covered.

TOOL SELECTION

```
{ CURTOOL = an index for tools
  CURHOLE = an index for holes
  M″ = number of holes
  N = number of tools
  PERMTOOL = set of tools selected}
```

```
{INITIALIZATION}
  CURTOOL ⇐ 1
  CURHOLE ⇐ 1
  PERMTOOL ⇐ ∅
{MAIN LOOP}
  DO WHILE CURHOLE .LE. M″
    BEGIN
      DO WHILE CURTOOL .LT. N .AND.
          TR[CURTOOL + 1] .LE. R[CURHOLE]
          BEGIN
        CURTOOL ⇐ CURTOOL + 1
      END
      {CURTOOL points to the rightmost tool
          covering CURHOLE}
      DO WHILE CURHOLE .LT. M″ .AND.
          L[CURHOLE + 1] .LE. TL[CURTOOL]
          BEGIN
        CURHOLE ⇐ CURHOLE + 1
      END
      {CURHOLE points to rightmost hole
              CURTOOL covers}
      PERMTOOL
          ⇐ PERMTOOL ∪ {CURTOOL}
      CURHOLE ⇐ CURHOLE + 1
    END
```

In analyzing TOOL SELECTION, we refer to the outer loop (main loop) and the two inner (do) loops. Each major iteration (execution of the main loop) terminates by selecting a permanent tool. The output set of the procedure, labeled PERMTOOL, is a minimum cardinality subset of tools that covers the input set of holes. The first inner (do) loop finds the rightmost tool that covers CURHOLE. This tool becomes the permanent tool, and the second inner (do) loop finds the rightmost hole covered by the new permanent tool.

Suppose that the algorithm selects exactly p permanent tools (terminates after p major iterations). Let the indices of the permanent tools be denoted by $[1]$, $[2], \ldots, [p]$, where tool $[j]$ is the tool selected on the jth pass through the main loop. Since tools are considered from left to right, $[j] < [k]$ whenever $j < k$. Let Left$[j]$ denote the index of CURHOLE at the beginning of the jth major iteration. Left$[j]$ is the leftmost uncovered hole at the beginning of the iteration. Thus, for example, Left$[j] = 1$ when $j = 1$. The first inner loop finds the rightmost tool that covers hole Left$[j]$. This tool, whose index is $[j]$, is added to the set of permanent tools at the end of the jth major iteration. We show in Lemma 1 that the second inner loop finds the rightmost hole that tool $[j]$ covers. Let Max$[j]$ denote the index of CURHOLE at the end of the second inner loop. Note that for $1 \leq j < p$, Left$[j+1]$ = Max$[j] + 1$; that is, the index of CURHOLE at the beginning of major iteration $j + 1$ is one plus the index of the rightmost hole covered by tool $[j]$. We can now establish two lemmas that we subsequently use to verify the optimality of the procedure.

Lemma 1. Tool $[j]$ *covers every hole k, where*

Left$[j] \leq k \leq$ Max$[j]$.

Proof. By definition, tool $[j]$ covers hole Left$[j]$, so that $L_{\text{Left}[j]} \leq TL_{[j]}$ and $R_{\text{Left}[j]} \geq TR_{[j]}$. Also, CURTOOL = $[j]$ at the end of the first inner loop during the jth major iteration. With Max$[j]$ = CURHOLE at the end of the second inner loop, we have $L_{\text{Max}[j]} \leq TL_{[j]}$. Let k satisfy Left$[j] \leq k \leq$ Max$[j]$. Since the holes are ordered with increasing right hole limits, it follows that $R_k \geq TR_{[j]}$. Furthermore, since the input set of holes is undominated, no hole interval contains another, so the holes are *also* ordered with increasing left hole limits. Thus, it follows that $L_k \leq TL_{[j]}$.

Lemma 2. *Every hole k,* $1 \leq k \leq m''$ *is covered by some permanent tool.*

Proof. Observe that Left$[j+1]$ = Max$[j] + 1$ for $1 \leq j < p$. The result follows by applying Lemma 1 repeatedly.

We can now prove our main result.

Theorem 2. *TOOL SELECTION finds a minimum cardinality set of tools to cover the set of holes labeled UNDOM.*

Proof. Suppose that the algorithm selects p permanent tools. We will show that there are at least p holes, no two of which can be covered by any single tool. Consider the holes indexed by Left[1], Left[2], ..., Left$[p]$ where, again, Left$[j]$ is the index of CURHOLE at the beginning of major iteration j. Let j and k be any two integers that satisfy $1 \leq j < k \leq p$. Since tool $[j]$ is the rightmost tool that covers hole Left$[j]$, no tool with an index larger than $[j]$ can cover hole Left$[j]$. Since $L_{\text{Max}[j]+1} > TL_{[j]}$ and Left$[k] \geq$ Max$[j] + 1$, we have $L_{\text{Left}[k]} \geq L_{\text{Max}[j]+1} > TL_{[j]}$, so tool $[j]$ cannot cover hole Left$[k]$. Furthermore, since $TL_\lambda \leq TL_{[j]}$ for all $\lambda \leq [j]$, no tool with an index less than or equal to $[j]$ can cover hole Left$[k]$. It follows that there is no single tool that can cover both hole Left$[j]$ and hole Left$[k]$. Since j and k are arbitrary, it follows that p tools are required to cover the holes indexed by Left[1], . . . , Left[p]. From Lemma 2, the tools indexed by [1], . . . , [p] cover all holes.

The complexity of TOOL SELECTION is clearly $o(m'' + n)$, because each hole and tool is considered once and only once in the procedure. To conclude this section, we state the preprocessing procedures, COVERABLE and UNDOMINATED.

The input to COVERABLE is the set of m holes, again indexed by nondecreasing right hole limits. Also, the tools are indexed by nondecreasing left (equivalent, by assumption, to right) tool interval limits. For each hole i, COVERABLE finds the largest indexed tool whose right limit does not exceed R_i. Once this tool is identified, the procedure checks whether the left limit of the tool is at least as large as L_i. If it is, the hole is coverable and it is added to the set CVBLE. It is easy to verify that hole i is not coverable by *any* tool if the left limit of the tool is strictly less than L_i. COVERABLE is of complexity $o(m + n)$ since each hole and each tool is considered once and only once.

COVERABLE

```
{ J = an index for tools
  I = an index for holes
  M = the number of holes
  N = the number of tools
  CVBLE = the set of coverable holes}
{INITIALIZATION}
  I ⇐ 1
  J ⇐ 0
  CVBLE ⇐ ∅
{MAIN LOOP}
  FOR I .EQ. 1 TO M DO
  BEGIN
    DO WHILE J .LT. N .AND. TR[J + 1] .LE. R[I]
    BEGIN
      J ⇐ J + 1
    END
    IF TL[J] .GE. L[I] THEN CVBLE ⇐ CVBLE
        ∪ {I}
  END
```

The output set, CVBLE, is the set of holes that can be covered by at least one tool. In what follows, we assume that the cardinality of CVBLE is m'. Also, we assume that the hole indices have been relabeled with consecutive integers, preserving the nondecreasing right hole limit property.

In UNDOMINATED, we always compare two holes, one labeled TEMP, the other CURR. The index of TEMP is always less than the index of CURR. The first inner loop is performed whenever the right hole limits of TEMP and CURR are the same. In this case, note that either TEMP will dominate CURR or vice versa. If CURR dominates TEMP, TEMP takes on the value of CURR, and the value of CURR is increased by one. The second inner loop is performed whenever R[TEMP] < R[CURR]. In this situation, CURR can never dominate TEMP, but TEMP may dominate CURR. UNDOMINATED therefore identifies the set, labeled UNDOM, of holes that are not dominated by other holes. UNDOMINATED is of time complexity $o(m')$.

UNDOMINATED

```
{ UNDOM = the set of undominated holes
  TEMP = an index for holes
  CURR = an index for holes
  M′ = the number of holes}
{INITIALIZATION}
  TEMP ⇐ 1
  CURR ⇐ 2
  UNDOM ⇐ ∅
{MAIN LOOP}
  DO WHILE CURR .LE. M′
  BEGIN
    DO WHILE CURR .LE. M′ .AND. R[TEMP]
        .EQ. R[CURR]
    BEGIN
      IF L[TEMP] .LT. L[CURR] THEN TEMP ⇐
          CURR
      CURR ⇐ CURR + 1
    END
    {at this point, TEMP is undominated}
    UNDOM ⇐ UNDOM ∪ {TEMP}
    IF L[TEMP] .LT. L[CURR] THEN TEMP ⇐
        CURR
    CURR ⇐ CURR + 1
    IF TEMP .EQ. M′ THEN UNDOM ⇐ UNDOM
        ∪ {TEMP}
  END
```

The overall algorithm is to run the procedures COVERABLE, UNDOMINATED and TOOL SELECTION serially. The time complexity of each procedure (in the same order) is $o(m + n)$, $o(m')$ and $o(m'' + n)$. Since $m \geq m' \geq m''$, the overall complexity is $o(m + n)$. Note that the three procedures could certainly be combined into a single procedure. The algorithm is easier to explain, however, presented as three separate procedures.

This section has considered one version of the problem in which the upper bound constraint on the number of tools is deleted. Other, more general versions of the *no-upper-bound* problem (weighted objective function, for example) can be solved using the $o(mn)$ algorithm of Hoffman, Kolen, and Sakarovitch. For problems in which the upper bound constraint is binding, in the next section, we present an algorithm that improves upon previous procedures.

4. DYNAMIC PROGRAMMING TO THE RESCUE

The upper bound on the number of tools selected can be binding for two reasons. First, it is possible that the tool-rack capacity, u, might be exceeded for some problems. Second, it may be of substantial interest to develop *tradeoff* curves that portray, for example, the number of holes punched versus the number of tools allowed. Performing this second analysis requires multiple model runs in which the upper bound is varied parametrically. If we assume that the number of tools, n, is smaller than the number of holes, m, then the algorithm of Broin and Lowe referred to previously finds a solution in $o(u^2n^3)$ time (otherwise, it requires $o(u^2nm^2)$ time). For practical problems, both u and n can become quite large ($u = 200$, $n = 497$ for some of the problems we consider in Section 5), and a faster algorithm is necessary if multiple model runs are to be made in a reasonable amount of time without resorting to exotic hardware. This section provides an algorithm that runs in $o(mn^2)$ time, a substantial improvement. As a by-product of running the algorithm for a specific value of u, we obtain the solution for all values of $u' \leqslant u$.

We assume that no tool interval contains another. Thus, ordering tools by either left or right interval limits will produce an identical ordering. The algorithm also uses the assumption that tools are ordered by their left (or, equivalently, right) interval limits and that holes have been ordered by their right (*not* equivalent to left) interval limits. We proceed by defining a dynamic programming recursion.

We solve this variant of the problem by dynamic programming. The dynamic programming recursion exploits the special structure of A. To develop and justify the recursion, we introduce additional notation. Let $S(p, k)$ denote a set of p tools whose highest indexed (rightmost) tool is tool k. Let $F[S(p, k)]$ be defined as the optimal value of the program

$$\begin{aligned}
\text{minimize}\quad & c'x + b'z\\
\text{subject to}\quad & Ax + Iz \geqslant e\\
& x_j = 0 \quad \text{for } j \notin S(p, k)\\
& x_j = 1 \quad \text{for } j \in S(p, k)\\
& z = 0 \text{ or } 1.
\end{aligned}$$

In words, $F[S(p, k)]$ is the value of selecting the set of tools, $S(p, k)$. Our optimal value function, $V_p(k)$, is defined as the optimal value to the program

$$\begin{aligned}
\text{minimize}\quad & c'x + b'z\\
\text{subject to}\quad & Ax + Iz \geqslant e\\
& e'x = p\\
& x_j = 0 \quad \text{for } j > k\\
& x_k = 1\\
& x_j = 0 \text{ or } 1 \quad \text{for } j < k\\
& z = 0 \text{ or } 1.
\end{aligned}$$

In words, $V_p(k)$ is the best that can be done using exactly p tools, with the rightmost tool being tool k. Additionally, we let

$$V_p = \min_j \{V_p(j)\}, \quad p \leqslant j \leqslant n.$$

Thus, V_p is the optimal value of the covering problem when exactly p tools are chosen. Finally, with V_0^* defined by

$$V_0^* = \sum_i b_i$$

we let, for $1 \leqslant p \leqslant u$

$$V_p^* = \min\{V_{p-1}^*, V_p\}$$

so that V_p^* is the optimal value of the covering problem defined in Section 1 when at most p tools may be selected (i.e., $u = p$).

Let $W(k)$ be the set $\{i\colon a_{ik} = 1\}$, that is, the set of holes tool k covers. The boundary condition is defined by

$$V_1(j) = \sum_i b_i + c_j - \sum_{i \in W(j)} b_i.$$

Our computational recursion may be stated as

$$V_p(k) = \min_{j<k} \left\{ V_{p-1}(j) + c_k - \sum_{i \in W(k)} b_i + \sum_{i \in W(j,k)} b_i \right\}$$

where $W(j, k) = W(j) \cap W(k)$. This recursion needs justification because it implies that the value of selecting as the pth tool, tool k, depends only upon the position, j, of the $p - 1$st tool. To justify this recursion, let $C[S(p, j), k]$ be the contribution made by selecting tool k as the pth tool, given that tools in the set $S(p - 1, j)$ (where $j < k$) have already been selected. That is

$$C[S(p-1, j), k] = c_k - \sum_{i \in W(k)} b_i + \sum_{i \in T} b_i$$

where

$$T = \mathrm{W}(k) \cap \left\{ \bigcup_{h \in \mathrm{S}(p-1,j)} \mathrm{W}(h) \right\}.$$

The first summation in the expression for $\mathrm{C}[\mathrm{S}(p-1, j), k]$ subtracts the total penalty cost for holes covered by tool k. The second summation adds back in penalty costs that have been *double counted* by the first summation. It is then valid to claim

$$\mathrm{V}_p(k) = \min\{\mathrm{F}[\mathrm{S}(p-1, j)] + \mathrm{C}[\mathrm{S}(p-1, j), k]\}$$

where the minimization is over *all* possible sets, $\mathrm{S}(p-1, j)$, of $p-1$ tools to the left of tool k ($j < k$). The keys to validating the recursion are the following two lemmas.

Lemma 3. *Under our assumptions, if tools h and k can punch hole i, then so can tool j, whenever $h \leqslant j < k$.*

Proof. Since tools h and k can punch hole i, it must be the case that $\mathrm{L}_i \leqslant \mathrm{TL}_h \leqslant \mathrm{TL}_k$ and $\mathrm{TR}_h \leqslant \mathrm{TR}_k \leqslant \mathrm{R}_i$. Since $h \leqslant j$, the fact that tools are ordered by their left interval limits implies that $\mathrm{TL}_h \leqslant \mathrm{TL}_j$. Since $j < k$, the assumption that ordering tools by either their right or left interval limits produces the same ordering implies that $\mathrm{TR}_j \leqslant \mathrm{TR}_k$. Combining these inequalities yields the series of inequalities: $\mathrm{L}_i \leqslant \mathrm{TL}_j \leqslant \mathrm{TR}_j \leqslant \mathrm{R}_i$.

Lemma 4. *With*

$$T = \mathrm{W}(k) \cap \left\{ \bigcup_{h \in \mathrm{S}(p-1,j)} \mathrm{W}(h) \right\}$$

and

$$\mathrm{W}(j, k) = \mathrm{W}(j) \cap \mathrm{W}(k)$$

we have $T = \mathrm{W}(j, k)$.

Proof. Since $j \in \mathrm{S}(p-1, j)$, it immediately follows that $\mathrm{W}(j, k)$ is contained in T. Thus, suppose that $i \in T$, so $i \in \mathrm{W}(k) \cap \mathrm{W}(h)$ for some $h < k$. From Lemma 3, this implies that $i \in \mathrm{W}(j)$ for $h \leqslant j < k$, so that $i \in \mathrm{W}(j, k)$. Thus T is contained in $\mathrm{W}(j, k)$ and the result follows.

From Lemma 4, it immediately follows that

$$\mathrm{C}[\mathrm{S}(p-1, j), k] = c_k - \sum_{i \in \mathrm{W}(k)} b_i + \sum_{i \in \mathrm{W}(j,k)} b_i$$

depending only upon tool j in the set $\mathrm{S}(p-1, j)$. Therefore

$$\mathrm{V}_p(k) = \min\left\{\mathrm{F}[\mathrm{S}(p-1,j)] + c_k - \sum_{i \in \mathrm{W}(k)} b_i + \sum_{i \in \mathrm{W}(j,k)} b_i\right\}$$

where the minimization is over all possible sets $\mathrm{S}(p-1, j)$ with $j < k$. Since only the first term in the minimization depends upon tools other than tool j, it must therefore be the case that

$$\mathrm{V}_p(k) = \min_{j<k}\left\{\mathrm{V}_{p-1}(j) + c_k - \sum_{i \in \mathrm{W}(k)} b_i + \sum_{i \in \mathrm{W}(j,k)} b_i\right\}$$

which is the recurrence relation claimed above. As verified before, the function $\mathrm{C}[\mathrm{S}(p-1, j), k]$ depends only upon j and k, and in particular is independent of p (for $p \geqslant 2$). Thus, we can write $\mathrm{C}(j, k)$ for $\mathrm{C}[\mathrm{S}(p-1, j), k]$, and hence, we can express the recursion in the form

$$\mathrm{V}_p(k) = \min_{j<k} \{\mathrm{V}_{p-1}(j) + \mathrm{C}(j, k)\} \quad \text{for } p \geqslant 2.$$

To analyze the complexity of the recursion, note that computing V_p, once the values for $\mathrm{V}_p(k)$ are known, requires $o(n)$ time. Computing $\mathrm{V}_p(k)$, once values for $\mathrm{C}(j, k)$ and $\mathrm{V}_{p-1}(j)$ are known, can be done in $o(n)$ time. Computing $\mathrm{V}_p(k)$ for all values of k requires, therefore, $o(n^2)$ time. With an upper bound of u, this computation must be carried out u times. The time complexity of the recursion, ignoring the work to compute $\mathrm{C}(j, k)$, is thus $o(un^2)$. To complete the complexity analysis, note that there are on the order of n^2 $\mathrm{C}(j, k)$ values to be computed. Each one can be computed in $o(m)$ time as follows.

The preceding lemmas imply that each row of A consists of a string of consecutive ones inserted in a string of consecutive zeros. All relevant information about the matrix can therefore be stored in two vectors of length m, FIRST_TOOL and LAST_TOOL. FIRST_TOOL(i) is the column index of the first "one" in row i, and LAST_TOOL(i) is the column index of the last "one" in row i. The following procedure calculates a value for $\mathrm{C}(j, k)$.

Procedure C(*J*, *K*)

```
C(j, k) ⇐ c_k
FOR i = 1 TO m DO
BEGIN
  IF FIRST_TOOL(i) ≤ k ≤ LAST_TOOL(i) THEN
  BEGIN
    IF FIRST_TOOL(i) > j OR LAST_TOOL(i) < j
    THEN C(j, k) ⇐ C(j, k) − b_i
  END
END
```

Procedure $C(J, K)$ executes in $o(m)$ time. Since it must be executed $o(n^2)$ times, the computational effort for computing all $C(j, k)$ values is $o(mn^2)$.

Once all V_p values are computed, finding all V_p^* for $1 \leq p \leq u$ takes $o(u)$ time. We assume that $u < m$, for otherwise the upper bound constraint would be redundant. It follows that the overall time complexity of the dynamic programming algorithm is $o(mn^2)$ as claimed.

Observation 1. Based on some rather involved arguments regarding column and row indices, it is possible to construct an algorithm that computes, for a fixed j, $C(j, k)$ for all $k > j$ in $o(m)$ time. Thus, the $C(j, k)$ values can actually be computed in $o(mn)$ time instead of $o(mn^2)$ time. The overall time complexity of the dynamic programming algorithm can, therefore, be reduced to $o(\max\{un^2, mn\})$. Computing the $C(j, k)$ values is, however, preprocessing. Including an involved discussion of the most effective preprocessing algorithm would take us too far astray in this paper.

Table I
Summary of Model Inputs

A. Holes
478 round holes
9,243,000 annual hits

B. Tools
1. Standard Tools
117 standard tools with diameters distributed as follows:

Diameter (inches)			
Minimum	Maximum	Increment	Number
$^1/_{16}$	1	$^1/_{64}$	61
$1\,^1/_{32}$	2	$^1/_{32}$	32
$2\,^1/_{16}$	3	$^1/_{16}$	16
$3\,^1/_8$	4	$^1/_8$	8

2. Custom Tools
A custom tool of a diameter equal to the nominal diameter of any hole can be purchased.
3. Tolerances
Every tool can punch holes with tolerances of +0.0015 inch and −0.0000 inch

5. INDUSTRIAL APPLICATION

This section describes our application of these modeling techniques in an actual industrial setting. A Chicago area sheet-metal fabricator provided access to part design and production data. Existing designs required 1,163 different holes as specified by combinations of *size* (diameter for round holes, length and width for other shapes), *tolerance*, and *material thickness*. These holes represented over 11,410,000 annual hits based on current production volumes. Of these 1,163 holes, we analyze only the 927 that are round. These accounted for slightly more than 9,243,000 of the annual hits, approximately 81%. Furthermore, to maintain data confidentiality while still preserving realism, we present results based on the assumption that all parts were redesigned to use the same material thickness. This reduces the problem to 478 round holes with different combinations of diameter and tolerance.

In addition to data regarding the required hole diameters and tolerances, the model requires data on tools. The manufacturer of the laser punch press installed by the local firm produces 117 stock punches for round holes, with diameters ranging from $^1/_{16}$ of an inch up to 4 inches. Table I summarizes the model inputs.

In analyzing the first and second issues discussed in Section 1, the tolerances of tools and holes, along with their nominal diameters, determine whether or not a particular tool can cover a given hole. Analyzing the third issue, possible part redesign, requires redesigning holes. In principle, each instance of every hole could be analyzed to determine the extent of redesign possible given the purpose and use of the hole. For example, a hole in which a bearing is fitted should be designed to accommodate a standard outer bearing diameter. Furthermore, such a hole will require tighter tolerances than one that is subsequently tapped for a bolt. In practice, such a detailed analysis is expensive in terms of both time and other resources of the firm. Indeed, the manufacturer with whom we worked felt that the expense was prohibitive and elected instead to use proxies for redesign limits. We then employed the model in a screening context to assess the degree of redesign needed to achieve certain objectives in reducing tooling requirements.

In the analysis outlined below, we employed three redesign proxies:

a. multiple of the specified tolerance,
b. fraction of an inch, and
c. multiple of the nominal diameter.

To explain these redesign proxies, note that in the original problem statement we said that tool j could cover hole i if the hole interval for hole i (defined by its nominal diameter and tolerances) contained the tool interval for tool j. These redesign proxies, in effect, establish new *pseudo*tolerances for each hole, which, in turn, give rise to a new set of covering relationships. Proxy a, for example, states that the *pseudo*tolerance is a specified multiple of the *real* tolerance. If the multiple is larger than one, the pseudotolerance is *looser* than the real tolerance. Tools that did not cover the hole with its real tolerance might cover the hole with its looser pseudotolerance. Proxy *b* says that the pseudotolerances are ± a specified fraction of an inch, regardless of the real tolerances or nominal diameters. Proxy *c* states that the pseudotolerance is a given multiple of the hole's nominal diameter.

To use any of these redesign proxies, we develop new covering relationships using the pseudotolerances in place of the real tolerances. The algorithms, run on the new problems developed from pseudotolerances, choose tools that can punch holes whose actual diameters lie within the interval specified by the hole's nominal diameter and its pseudotolerances. Using proxy *c*, for example, we might specify that actual diameters of punched holes must be within 2.5% of the original nominal diameter. The resulting pseudotolerance interval limits on a hole whose nominal diameter is 1.00 inch would be 0.975 inch and 1.025 inches.

Table II summarizes the results of seven different model runs. The first five runs were restricted to selecting only standard tools, and the last two runs allowed custom tooling. All runs were performed using the linear-time algorithm developed in Section 3. These results indicate several important points.

First, of the 478 holes, 28 could not be punched within specified tolerances by any tool, stock or custom. This is because the specified tolerances for these 28 holes were tighter than the tool tolerances specified in Table I. As a result, in the runs using the specified tolerances that allowed custom tooling, only 450 holes could be punched.

Second, the number of holes using existing tolerances that are both undominated and coverable is relatively small. Allowing custom tools, 450 holes were coverable, but only 184 were undominated. Using standard tools only, 207 holes were coverable, but only 68 were undominated. These results suggest that substantial benefits can be obtained from the preprocessing procedures alone. To see this, note that in the custom tooling example, using the naive strategy of selecting a tool whose diameter equals the nominal diameter of each coverable hole would require a total of 450 different tools. Applying the same naive strategy to the set of undominated holes obtained after the two preprocessing procedures would substantially reduce the required tooling: only 184 tools would be required. Applying the TOOL SELECTION procedure reduces the required tooling further, to 148 tools. Thus, preprocessing produces substantial benefits, but the TOOL SELECTION procedure makes further, albeit less substantial, improvement.

Third, if no redesign is allowed (i.e., every hole must be punched to within existing design tolerances), 271 of the 478 holes cannot be punched with standard tools. To punch all 450 coverable holes, 148 tools are required, of which 138 must be custom. If only standard tools are allowed, then 52 tools cover as many holes (207) as possible. We also ran a version of the model in which we deleted columns (corresponding to the 52 tools) and rows (corresponding to holes covered by the 52 tools) from A. This model run determined that 120 additional custom tools would be required to cover all coverable holes if we first selected the standard tools that cover as many holes as possible. Fourth, if redesign outside the restrictive existing tolerances is allowed, the number of holes that can be covered using only standard tools increases significantly. For example, if hole diameters can be redesigned so that the new diameter is within 5% of

Table II
Summary of Selected Model Results

Redesign Criterion	No. of Tools	Number of Holes Covered	Number UNDOM Holes	Number of Hits Covered
Specified tolerance	52	207	68	4,911,418
150% of specified tolerance	62	302	102	6,257,028
±1/40th/inch	38	476	375	9,238,881
Nominal diameter × 0.025	57	407	315	8,283,939
Nominal diameter × 0.05	34	464	363	9,153,720
Specified tolerance with custom tools[a]	148 (10)	450	184	9,171,478
Stock tools first[b]	172 (52)	450	184	9,171,478

[a] The number of tools in parentheses indicates the number of stock tools in the solution.

[b] The 52 standard tools chosen in the run displayed in the first row of the table were first loaded into the solution. Custom tools were then used to complete the solution.

the nominal diameter, 34 standard tools can cover 464 of the holes, leaving only 14 holes uncovered.

We also used two heuristics (greedy adding and greedy adding with substitution) to solve the problems summarized in Table II. We made three observations that may be of interest. First, the linear-time algorithm was substantially faster than either heuristic. Second, the greedy adding with substitution heuristic always did at least as well as the simpler greedy adding heuristic, with respect to minimizing the number of tools, and sometimes substantially outperformed it. Third, the greedy adding with substitution heuristic found optimal solutions in some, but not all, cases.

6. CONCLUSION

We have developed models for rationalizing tool selection in a flexible system for sheet-metal manufacturing and have shown that existing polynomial-time algorithms can be used to solve the models. For two important versions of the model, we have provided improved algorithms. Finally, we have applied our modeling techniques to analyze data drawn from an industrial application. Our results indicate that these techniques can yield considerable insight into the problem of selecting tooling and developing standards for redesigning parts. Our approach can be used to determine which holes are good candidates for redesign. Furthermore, that fact that additional tools yield decreasing marginal benefits as measured by hole production suggests that certain parts may be likely candidates for out-sourcing.

Future work can proceed along a number of directions. First, extending the algorithms outlined above to deal with nonround holes would be desirable. Such extensions might be difficult for geometries that must be characterized by more than one dimension. In such cases, the natural orderings of holes and tools employed in this paper cannot be used. Second, extending the models to account for parts production instead of simply punching holes would also be valuable. We are working on such extensions.

ACKNOWLEDGMENT

We thank Professor Wen-Lian Hsu for suggesting several improvements and Professor William Wilson for providing useful information regarding sheet-metal forming. A. J. Hoffman's comments on an earlier version of this manuscript led to substantial improvements in Section 3. The referees' comments led to considerable improvements in the paper. Additionally, Gregory Morrison collected data and helped transform it from engineering drawings into machine readable files. Finally, this research was supported in part by National Science Foundation Presidential Young Investigator Award grant 835-2298 and a matching grant from the General Motors Corporation.

REFERENCES

BOLZ, R. W. 1977. *Production Processes*, 5th ed. Conquest Publications, Winston-Salem, N.C.

BROIN, M. W., AND T. J. LOWE. 1986. A Dynamic Programming Algorithm for Covering Problems With (Greedy) Totally Balanced Constraint Matrices. *SIAM J. Alg. Disc. Meth.* **7,** 348–357.

CHURCH, R., AND C. REVELLE. 1974. The Maximal Covering Location Problem. *Papers Reg. Sci. Assoc.* **32,** 101–118.

EATON, D., R. CHURCH AND C. REVELLE. 1977. Location Analysis: A New Tool for Health Planners. Methodological Working Document 53, Sector Analysis Division, Bureau for Latin America, Agency for International Development.

HOFFMAN, A. J., A. W. J. KOLEN AND M. SAKAROVITCH. 1985. Totally Balanced and Greedy Matrices. *SIAM J. Alg. Disc. Meth.* **6,** 721–730.

NEMHAUSER, G. L., AND L. A. WOLSEY. 1988. *Integer and Combinatorial Optimization.* John Wiley, New York.

TANG, C. S., AND E. V. DENARDO. 1988a. Models Arising From a Flexible Manufacturing Machine, Part I: Minimization of the Number of Tool Switches. *Opns. Res.* **36,** 767–777.

TANG, C. S., AND E. V. DENARDO. 1988b. Models Arising From a Flexible Manufacturing Machine, Part II: Minimization of the Number of Switching Instants. *Opns. Res.* **36,** 778–784.

TOREGAS, C., AND C. REVELLE. 1972. Optimal Location Under Time or Distance Constraints. *Papers Reg. Sci. Assoc.* **28,** 133–143.

TOREGAS, C., AND C. REVELLE. 1973. Binary Logic Solutions to a Class of Location Problems. *Geograph. Anal.* **6,** 145–155.