A HYPERCUBE QUEUING MODEL FOR FACILITY LOCATION AND REDISTRICTING IN URBAN EMERGENCY SERVICES

PREPARED FOR THE DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT

RICHARD C. LARSON

R-1238-HUD
MARCH 1973
The research and studies forming the basis for this report were conducted pursuant to Contract No. H-1753 with the Department of Housing and Urban Development. Its contents do not, however, purport to represent the official views or policy of the sponsor. Neither the United States nor HUD makes any warranty, expressed or implied, or assumes responsibility for the accuracy or completeness of the information herein.

The New York City-Rand Institute is a nonprofit research institution founded, as its Articles of Incorporation state, "...primarily to conduct programs of scientific research and study, and provide reports and recommendations, relevant to the operations, planning or administration of the City of New York." The Institute was established in 1969 by the City of New York and The Rand Corporation as a center for the continuing application of scientific and analytic techniques to problems of urban life and local government. It is governed by a Board of Trustees appointed jointly by the City and Rand.

Published by The Rand Corporation
A HYPERCUBE QUEUING MODEL FOR FACILITY LOCATION AND REDISTRICTING IN URBAN EMERGENCY SERVICES

PREPARED FOR THE DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT

RICHARD C. LARSON

R-1238-HUD
MARCH 1973

THE NEW YORK CITY RAND INSTITUTE

545 MADISON AVENUE NEW YORK NEW YORK 10022 (212) 758-2244
The work described in this report was supported by the U.S. Department of Housing and Urban Development under a contract to The New York City-Rand Institute to improve allocation procedures in urban emergency services throughout the United States. The report presents a methodology suitable for computer implementation to assist urban administrators in various location, redistricting, and dispatching problems encountered in urban emergency service systems. A forthcoming companion report by the author focuses on computational results from the computer program that incorporates the techniques described in this report. (R. C. Larson, *Computational Results with a Hypercube Queuing Model for Urban Emergency Services.*)

The author is obliged to Grace Carter and Jan Chaiken for their suggestions of changes made in an earlier draft.

Richard C. Larson is Associate Professor of Electrical Engineering and Urban Studies at the Massachusetts Institute of Technology and is a Consultant to The New York City-Rand Institute.
SUMMARY

This report proposes and develops a spatially distributed queuing model that can be useful as an aid to administrators of urban emergency services in locating their vehicles and designing response districts for their cities. Building on the work of Gregory Campbell [16], this report develops computationally efficient algorithms that allow one to evaluate numerically the performance characteristics of systems having up to 12 emergency response units. This requires the formulation and solution of a set of simultaneous linear equations with up to 4096 unknowns.

The report starts with a review of recent literature pertaining to urban facility location and redistricting and indicates at least three recurring weaknesses in previous work that the current model attempts to overcome. Then, after a brief description of terminology and model assumptions, an iterative efficient way of generating the state transition matrix for the associated continuous-time Markov process is discussed, given a geographical description of the region of interest and a dispatch criterion. The state space of the system, when nonsaturated, can be described as the vertices of an N-dimensional unit hypercube.

It is found to be important to "tour" the hypercube vertices in a unit-step fashion, and an algorithm for generating an appropriate sequence is developed. Then the details of determining the optimal units to dispatch while touring the hypercube are provided. An analysis of the algorithm shows somewhat surprisingly that the time required for the touring algorithm, measured in execution time per step (or operations per hypercube vertex), actually decreases as the problem size increases.

After having demonstrated how to generate the matrix of coefficients of the equations of detailed balance, it is shown how to reduce the storage requirements for the coefficients by a factor of 200 or more for problems of practical size.

Next, an iterative solution procedure is developed that exploits many of the special properties of the hypercube queuing model, including the fact that adding the probabilities of states lying along specified
hyperplanes results in the well-known formulas for an $M/M/N$ queuing system. This fact is used to provide an initial set of probability estimates for the iterative solution procedure. By ordering the iterations in a particular way, a Gauss-Seidel method is derived that always keeps the sum of estimation errors along each such hyperplane equal to zero following each iteration. It is shown that the problem structure is such that one can guarantee convergence of the method with minimal round-off error.

The measures of performance computed analytically by the model include the following: region-wide mean travel time, workload imbalance, and fractions of dispatches that are interdistrict dispatches; workloads of each response unit; mean travel time to each geographical atom and to each district; mean travel time of each response unit; fraction of responses into each district that are interdistrict; and fraction of responses of each response unit that are interdistrict. This mixture of performance measures allows one to focus simultaneously on several region-wide objectives while assuring that spatial inequities in the delivery of service are maintained at an acceptable minimum. The user has the option of assuming a zero-line capacity system, which effectively models the case in which "overflows" are treated by special reserve units (perhaps located outside the region of study), or an infinite-line capacity system, in which case delayed calls are handled in a first-come, first-served manner by the regular units within the region.

Illustrative computational results using a PL/I program are presented for up to $N = 12$ response units, and the method appears very feasible for even such large problems.

A concluding section discusses promising extensions and generalizations.

A forthcoming companion paper, "Computational Results with a Hypercube Queuing Model for Urban Emergency Services," compares the methods outlined in this report with those of Gregory Campbell, analyzes several detailed numerical examples, and provides details of a computer program that implements these methods.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
</tr>
<tr>
<td>2. LITERATURE REVIEW</td>
</tr>
<tr>
<td>3. MODEL DESCRIPTION AND TERMINOLOGY</td>
</tr>
<tr>
<td>4. THE STATE TRANSITION MATRIX AND EQUATIONS OF DETAILED BALANCE</td>
</tr>
<tr>
<td>5. GENERATING THE UPWARD TRANSITION RATES</td>
</tr>
<tr>
<td>6. GENERATING A UNIT-HAMMING-DISTANCE BINARY SEQUENCE</td>
</tr>
<tr>
<td>7. DETERMINING THE OPTIMAL UNIT(S)</td>
</tr>
<tr>
<td>8. ANALYSIS OF THE TOUR ALGORITHM</td>
</tr>
<tr>
<td>9. STORING THE STATE TRANSITION MATRIX</td>
</tr>
<tr>
<td>10. REDUCTION TO THE M/M/N ZERO-LINE CAPACITY SYSTEM</td>
</tr>
<tr>
<td>11. ITERATING TOWARD A SOLUTION</td>
</tr>
<tr>
<td>12. COMPUTED MEASURES OF PERFORMANCE FOR A ZERO-LINE CAPACITY SYSTEM</td>
</tr>
<tr>
<td>Individual Workloads</td>
</tr>
<tr>
<td>Region-Wide Workload Imbalance</td>
</tr>
<tr>
<td>Interdistrict Dispatches</td>
</tr>
<tr>
<td>Fraction of Total Dispatches That Are Interdistrict</td>
</tr>
<tr>
<td>Fraction of Dispatches to Unit j That Are Interdistrict</td>
</tr>
<tr>
<td>Fraction of District i Calls That Require an Out-of-District Unit</td>
</tr>
<tr>
<td>Unconditional Mean Travel Time</td>
</tr>
<tr>
<td>Average Travel Time to Atom k</td>
</tr>
<tr>
<td>Average Travel Time to District i</td>
</tr>
<tr>
<td>Average Travel Time of Unit j</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

Only recently have the resource allocation problems of spatially
distributed urban service systems, as exemplified by police, fire, and
ambulance services, been subjected to serious analytical attention.
Much of the recent work is reviewed by Chaiken and Larson [1] and
Revelle, Marks, and Liebman [2].

One important and recurring allocation problem of urban services
is the "districting" problem. Given a region with a certain spatial
distribution of demands for service and given N response units that are
spatially distributed throughout the region, the districting problem
can be stated as follows: "How should the region be partitioned into
areas of primary responsibility (districts)* so that a level or combina-
tion of levels of service is best achieved?"

Most previous analyses of this problem have suffered from at least
three deficiencies. First, they have often failed to incorporate the
urban service system's probabilistic nature, which is caused by the
Poisson nature of the call arrival process and the variability in ser-
tice times. Second, most previous studies focused solely on intradis-
trict responses of units, while ignoring interdistrict responses and
design issues that relate to interdistrict response.† Third, mean
region-wide travel time (or a closely related measure) has been the
single performance measure that most previous studies have utilized to
evaluate alternative district designs. Little has been done to utilize
a combination of performance criteria.

*Throughout this report we use the term "district" to mean the
region of primary responsibility of a response unit. A district may or
may not overlap with other units' districts. In particular applica-
tions, one hears other terms such as "sector" or "beat" (in the case
of police units), "response area," "division," "zone," etc.
†An interdistrict response causes the responding unit to travel
to an incident outside of its usual or "home" district; an intradis-
trict response does not.
2. LITERATURE REVIEW

In order to place the work reported here in perspective, we briefly review the most relevant related work that has been reported within the past 12 years. One of the first efforts in the police area known to the author is the 1961 publication by R. Dean Smith [3], who used a computer gradient search technique to successively redesign sectors (patrol beats) from smaller cells (or atoms) so as to minimize mean intrasector travel time. No intersector interaction was assumed. The method failed to converge for a 16-sector, 144-cell problem after 1 hour and 55 minutes of computation time on an IBM 650 computer. The reason for a lack of convergence is suggested by the recent analytical models of Larson [4, pp. 78-84, 106-113], which reveal surprising insensitivities of mean intrasector travel time to widely differing sector geometries.

In 1968, Gass [5] employed a heuristic technique first used for political redistricting by Hess et al. [6] to design police sectors (beats) in Cleveland. This technique focused on equalizing call rates from each sector and simultaneously keeping to a minimum the spatial irregularity of the sector, as measured by its moment of inertia. The method incorporated only a deterministic depiction of the system and neglected intersector cooperation.

Another study [7] focusing on the design of police patrol sectors also attempted to minimize mean intrasector travel time, but was based on an incorrect methodology. The authors first designed "patrol areas" (geographical cells) of equal "crime potential" (which they measured in call-for-service workload), and claimed the following: "If the patrol areas [cells] are then assigned to cars on a purely random basis [that is, if a car is equally likely to be in any one of a specified number of areas], probability theory predicts that patrol cars will be closer to the point of need when they are requested than under any other system of patrol assignment." It turns out that, on the contrary, a patrol assignment with a fixed-point prepositioning, usually near but not necessarily coincident with the centroid ("center of mass") of a patrol region, is the one that minimizes mean intrasector travel time.
But most police administrators find such prepositioning unacceptable because of lost preventive patrol coverage. Thus, there is a need to develop a location and redistricting methodology that is compatible with mobile location services such as police departments as well as with fixed location services such as fire departments and ambulance services.

A network approach was developed by Santone and Berlin [8] at the National Bureau of Standards for the evaluation of alternative fire station sites. Here the travel-time characteristics of a city are determined by modeling travel time to occur along a network and employing shortest-route algorithms. Again, all responses are assumed to be intra-district, and a fire house's district is assumed to be all points closer to that firehouse than to any other. The computed response time is weighted by the seriousness of a fire reported from each possible location. Santone and Berlin's program, rather than including an algorithm to iterate automatically toward "optimality," is to be used iteratively by a planner to determine the (weighted) response-time effects of alternative proposed sites and districts. Employing the program with a knowledge of political and social constraints, they arrived at recommended new locations and districts for the fire department of East Lansing, Michigan.

A continuation and elaboration of this approach by Schneider [9], and Schneider and Symons [10] relies on man-machine interaction to locate ambulance dispatch centers and district a region. The alternative locations are nodes in a network and travel is accomplished from point to point by following a minimal travel-time path of the network. Again, a district about each located center is defined to be all points closer to that center than to any other, and all responses are assumed to be intra-district. No interdistrict interaction is considered. Here the objectives are to minimize mean travel time (always assuming intradistrict travel) while satisfying a constraint on maximum possible travel time. An encouraging part of this work is that human analysts, while interacting with a computer-driven cathode-ray tube, seemed to be able, in ten or fewer iterations, to locate and design better districts than any heuristic algorithm that was tried.
A recent article by Toregas et al. [11] viewed the location of emergency service facilities as a set covering problem, using a model similar to the one first analyzed by Hakimi [12,13]. Here the objective was to locate facilities on nodes of a network so as to minimize maximum possible travel time (or distance). Again, an emergency facility's district is the set of points closer to that facility than to any other and, reflecting a focus on intradistrict responses only, Toregas et al. state, "It is assumed here that each facility has response capability at all times." They derived a zero-one integer programming formulation to solve the stated problem.

Some recent work for small numbers of districts has overcome many of the objections associated with the above methods. Carter, Chaiken, and Ignall [14] analyze the case of N = 2 fixed-position response units and rigorously derive the optimal districts for the two units, assuming a very general distance metric and a simplified form of interdistrict cooperation. The stochastic and interdistrict behaviors of the system are fully incorporated in their model. In addition, two measures of effectiveness are treated simultaneously: mean travel time and workload imbalance, the latter being measured as the difference in the fractions of time worked by the most-busy and least-busy units. Concurrent work by Larson and Stevenson [15] investigated several insensitivities of these and other location and districting models. But in all of this analytical work, "optimal" districts have proven very difficult if not impossible to find for N > 2. Thus, although these models have provided useful insights into certain aspects of location and districting problems, they have not addressed computational problems that arise in practical situations with many response units.

A different approach was recently taken by Campbell [16] based on suggestions by Richard C. Larson. Here, the multiserver queuing model employed in Refs. 14 and 15, which facilitates the study of probabilistic phenomena and interdistrict interaction, is extended for up to N = 6 units, and the steady-state equations are solved numerically on a computer. Then, in a spirit similar to that represented in Refs. 2, 10, and 17, the user can examine the numerical results and relocate and/or redistrict accordingly. In this iterative fashion, a very...
reasonable set of locations and districts can be found, incorporating a rich mixture of system performance criteria. However, the storage and execution-time requirements of Campbell's method grow enormously with $N$, making computations for $N > 6$ infeasible.

A main concern of this report will be the generation of new algorithms to make the computations for the model described by Campbell feasible for problems whose size is typically encountered in most cities. As a guideline, the average police precinct in New York City contains approximately nine radio-dispatched patrol cars. Thus, our goal is to devise a method that is feasible and practical for at least $N = 10$ or $N = 11$ units.
3. MODEL DESCRIPTION AND TERMINOLOGY

Central to the model is a geographical description of the region under study. This region is assumed to be partitioned into \( K \) cells or geographical atoms. The atoms can be as small as necessary to avoid unreasonable quantization error, and they can assume any geometric shape. Associated with atom \( k \) (\( 1 \leq k \leq K \)) is the fraction of region-wide workload \( f_k \) generated from within the atom. \( \sum_{k=1}^{K} \xi_k = 1. \)

Travel-time statistics also play a major role in the model. All mean travel times are computed from a travel-time matrix whose generic element is \( \tau_{ij} \), which is the mean travel time from atom \( i \) to atom \( j \). In general, \( \tau_{ij} \neq \tau_{ji} \). The numerical values of the \( \tau_{ij} \)'s may reflect complications in travel such as one-way streets, barriers, traffic conditions, etc. If no matrix of \( \tau_{ij} \)'s is given, then one must specify the centroid \( (\bar{x}_i, \bar{y}_i) \) of each atom, and as a default, \( \tau_{ij} \) is assumed to be \( (|\bar{x}_i - \bar{x}_j| + |\bar{y}_i - \bar{y}_j|)/v \), where \( v \) is the effective response speed. *

The geographical depiction of the "location" of a response unit is general enough to model the fixed locations of fire units and ambulances and the mobile locations of police patrol units. This is accomplished by specifying a location matrix \( L = (l_{ij}) \), where \( l_{ij} \) is the probability that response unit \( i \) is located in cell \( j \) while available or idle † (or, equivalently, the fraction of available or idle time that response unit \( i \) spends in atom \( j \)). We require that \( L \) be a stochastic matrix, i.e., for all \( i \), \( \sum_{j=1}^{K} l_{ij} = 1. \) A fixed-location unit would have \( l_{ij} = 1 \) for some (small) ‡ atom \( j \), and \( l_{ik} = 0 \) for \( k \neq j \). A mobile-location unit would most likely have several nonzero \( l_{ij} \)'s. Note that within this

* In the computer program, one can selectively override this default for the case of intra-atom travel (involving \( \tau_{ii} \)'s).

† Here "idle" is a convenient queuing-theory term for reflecting the activity between responses to calls for services. In actuality, the unit may be far from idle, perhaps performing crime-preventive patrol in the case of police units or performing equipment maintenance or field inspections in the case of fire units.

‡ If one wishes to be exact about the fixed location, the atom \( j \) could be defined as a point atom, i.e., one of zero area.
structure it is very natural to allow mobile units to have overlapping
districts or areas of responsibility. For instance, atom \( k \) would belong
to overlapping districts if \( l_{ik} \neq 0 \) and \( l_{jk} \neq 0 \) for some \( i \) and \( j \neq i \).

From a queuing point of view, we assume that customers (calls for
service) are generated from within the region in a Poisson manner at
a mean rate \( \lambda \) per hour, with each atom \( k \) acting as an independent Poisson
generator with mean rate \( \lambda f_k \). We assume there are \( N \) servers, which are
the response units located within the region.

If one is not concerned with the identity of busy servers, the
queuing system is simply the \( M/M/N \) system. The user can specify whether
the system has zero-line capacity or infinite-line capacity. The fol-
lowing assumptions characterize the \( M/M/N \) model: (1) the interarrival
times of calls have a negative exponential distribution and exactly one
of the \( N \) response units is assigned to every call that is serviced;
(2) the service time of any response unit for any call for service has
a negative exponential distribution with mean \( 1/\mu \); (3) the service
time is independent of the identity of the server, the location of the
customer, and the history of the system; (4) for the zero-line capacity
case, any call for service that arrives while all \( N \) response units are
busy is either lost or (more likely in practice) serviced from outside
the region or by special reserve units from within the region; (5) for
the infinite-line capacity case, any call for service that arrives while
all \( N \) response units are busy is entered at the end of a queue of calls
that is depleted in a first-come, first-served manner.\(^\dagger\)

To describe the more complicated state space in which the identities
of busy (unavailable) and idle (available) response units are re-
tained, we need certain definitions and operations pertaining to binary
numbers. A generic, one-digit binary number is denoted as \( b \). A generic
ordered set of \( N \) one-digit binary numbers is given by \( B = \{ b_N, b_{N-1}, \ldots, b_1 \} \).
The weight of \( B \), denoted \( w(B) \), is equal to \( \sum b_1 \), which is the number of
binary "ones" in the set \( B \). For each set \( B \) there is a corresponding
unique numerical value \( v(B) \), given in decimal form by

\[ v(B) = \sum b_i \times 2^{N-i} \]

\* Thus variations in service time that are due to variations in
travel times are ignored.

\( \dagger \) The results of the model are also applicable under several other
queuing disciplines such as last-come, first-served, and random.
\[ v(B) = b_N \cdot 2^{N-1} + b_{N-1} \cdot 2^{N-2} + \ldots + b_1. \]

Conversely, for each positive integer \( k \), there is a corresponding, unique set \( B(k) \), such that \( v[B(k)] = k \). Where the meaning is clear we shall occasionally perform "arithmetic" operations on the set \( B \), recognizing that \( v(B) \) is implied.

For each set \( B = \{b_N, b_{N-1}, \ldots, b_1\} \), there is a corresponding unique point or vertex in \( \mathbb{R}^N \) with the \( i \)th coordinate equal to \( b_i \) \((i = 1, \ldots, N)\). The set \( C_N \) of all these \( 2^N \) vertices is an \( N \)-dimensional unit hypercube. The \( i \)th hyperplane from the origin (or simply the \( i \)th hyperplane) is the set

\[ H_i = \{B \in C_N : w(B) = i\}, \quad i = 0, 1, \ldots, N. \]

Binary set operations are defined as follows:

Or:

\[ B_1 \cup B_2 = \{1\} \quad \text{if} \quad B_1 = \{1\} \quad \text{or} \quad B_2 = \{1\}. \]

And:

\[ B_1 \cap B_2 = \{0\} \quad \text{if} \quad B_1 = \{0\} \quad \text{or} \quad B_2 = \{0\}. \]

Complement:

\[ B' = \begin{cases} \{0\} & \text{if} \quad B = \{1\}, \\ \{1\} & \text{if} \quad B = \{0\}. \end{cases} \]

These definitions carry over in the obvious way to sets having more than one digit.

The Hamming distance between two vertices, \( B_i \) and \( B_j \), is the weight of the symmetric set difference,

\[ d_{ij} \equiv w[(B_i \cap B_j') \cup (B_i' \cap B_j)]. \]

Thus \( d_{ij} \) is equal to the number of elements (binary digits) of the set \( B_i \), which differ from the corresponding elements in the set \( B_j \). Geometrically, the Hamming distance is simply the rectilinear or "right-angle"
distance between two points in $\mathbb{R}^N$. We also find it convenient to define the "upward" Hamming distance,

$$d^+_{ij} \equiv |B_1 \cap B_j|,$$

and the "downward" Hamming distance,

$$d^-_{ij} \equiv |B_i \cap B_j'|.$$

Clearly,

$$d_{ij} = d^+_{ij} + d^-_{ij}.$$
4. THE STATE TRANSITION MATRIX AND EQUATIONS
OF DETAILED BALANCE

The state space of the zero-line capacity queuing model is the \( N \)-dimensional hypercube \( C_N \), where each vertex (state) corresponds to a particular combination of response units, busy and idle. Given a state \( B = \{b_N, b_{N-1}, \ldots, b_1\} \), unit \( i \) is said to be busy if \( b_i = 1 \) and idle if \( b_i = 0 \). This state space is augmented by an "infinite tail," if one wishes to model the infinite-line capacity case. Since the two models are governed by the same equations for unsaturated states (i.e., states with at least one available response unit), we will focus the development on just one of the models: the zero-line capacity model. In Sec. 13, we indicate the necessary model modifications to allow for infinite-line capacity.

A central problem is converting the geographical data into a queuing framework. To do this we define the state transition matrix, \( \Lambda = (\lambda_{ij}) \), where \( \lambda_{ij} \) is the infinitesimal mean rate at which transitions are made from state \( i \) to state \( j \), given that the system is in state \( i \); \( i, j = 0, 1, \ldots, 2^N - 1 \), \( i \neq j \), and \( \lambda_{ii} = -\sum_{j \neq i} \lambda_{ij} \). Here, for convenience, we index the states (vertices) according to their numerical values (i.e., we select \( i \) so that \( v(B_i) = i, i = 0, 1, \ldots, 2^N - 1 \)). Note that \( \Lambda \) is a differential matrix (i.e., \( \sum_j \lambda_{ij} = 0 \)).

There are two classes of transitions on the hypercube: upward transitions that change a unit's status from available to unavailable and downward transitions that do the reverse. For a given vertex \( B_i = \{b_N, b_{N-1}, \ldots, b_1\} \), upward transitions can occur in all "adjacent" vertices \( B_j \) for which \( d^+_{ij} = 1 \). If unit \( j \) is the unit whose status is changed (from idle to busy), then \( B_i = \{b_N, b_{N-1}, \ldots, 0, \ldots, b_1\} \) and \( B_j = \{b_N, b_{N-1}, \ldots, 1, \ldots, b_1\} \). Downward transitions can occur to all adjacent vertices \( B_j \) for which \( d^-_{ij} = 1 \). A key observation is that no transitions occur to vertices that are more than unit-Hamming distance from \( B_i \). This is due to the fact that only one unit is assigned to each call.
Since the service times are all distributed as negative exponential random variables with a mean of $\mu^{-1}$, the transition rate associated with each downward transition is equal to $\mu$. Thus, for all $(i,j)$ for which $d_{ij} = 1$, we have $\lambda_{ij} = \mu$. For convenience, we set $\mu = 1$, thereby equating the unit of time to the mean service time.

The upward transition rates depend in a complicated way on the region's geography, the system state, and the dispatching selection criterion. We will develop a recursive method to generate the set of upward transition rates, first by fixing the geographical atom of the call, then by touring the hypercube in a unit-step fashion. The entire matrix is completed as soon as the hypercube has been toured once for every atom.

Since the model assumes Poisson input and negative exponential service times, thereby making knowledge of past system history irrelevant, the state of the system is fully specified by $B_{i}$. The model is thus a finite-state, continuous-time Markov process*, whose steady-state probabilities are determined from the equations of detailed balance,

$$
P(B_{j})[\lambda_{j} + w_{B_{j}}] = \sum_{\{B_{i} \in C_{N} : d_{1j}^{+} = 1\}} P(B_{i})\lambda_{ij} 
+ \sum_{\{B_{i} \in C_{N} : d_{1j}^{-} = 1\}} P(B_{i}), \quad j = 0,1,\ldots,2^{N}-1
$$

(1)

where

$$
P(B_{j}) \equiv \text{Prob(system is occupying state } j \text{ under steady-state conditions)},
B_{j} \in C_{N}, j = 0,1,\ldots,2^{N}-1,
\lambda_{j} = \{0 \text{ for } j = 2^{N}-1,
\lambda \text{ otherwise}.
$$

Heuristically, these equations require that the steady-state rate of transitions out of state $B_{j}$ (the left-hand side of Eq. (1)) be equal to the steady-state rate of transitions into state $B_{j}$ (the right-hand side).

*See, for example, Ref. 18.
side of Eq. (1)). The latter transitions include \( w(B_j) \) upward transitions to \( B_j \), which result in a new unit becoming busy, and \([N - w(B_j)]\) downward transitions to \( B_j \), which result in a new unit becoming idle. To guarantee a probability distribution and thereby eliminate a degenerate solution, we also require that the probabilities sum to one, i.e.,

\[
\sum_{i=0}^{2^{N}-1} P\{B_i\} = 1, \tag{2}
\]

a condition that makes any one of the balance equations redundant and therefore removable from the set of equations.

Theoretically, the solution to this set of equations requires only a matrix inversion, and thus is relatively straightforward. However, the size of the matrix \( \Lambda \) (measured in total number of elements) is equal to \( 2^{2N} \), thus becoming huge for even moderate values of \( N \). For instance for \( N = 10 \), the matrix contains 1,048,576 elements. Thus, one is confronted with both computational and storage problems of large magnitude.
5. GENERATING THE UPWARD TRANSITION RATES

We develop in this and the next section an efficient method for generating the upward transition rates. For each geographical atom \( k \) (1 \( \leq \) k \( \leq \) K), we shall tour the hypercube in a unit-step fashion. From one step to the next, the status of only one response unit changes, either from busy to idle or from idle to busy. The method for constructing such a tour is given in the next section.

Ignoring "ties" for the moment, it is quite likely that the optimal unit to dispatch in the current state on the tour is either the same unit that was optimal in the state considered just prior to the current one, or the unit whose status has just changed, provided it has switched from busy to idle.

At each vertex we must compute the identities of the optimal units to dispatch. In particular, the following variables must be calculated:†

\[ n_{ik} \equiv \text{total number of response units that are optimal, given state } B_i \text{ and geographical atom } k, \]
\[ r_{lk} = \text{the number of the } l\text{th optimal response unit,} \]

where \( r_{lk} = 1, 2, \ldots, N, \]
\[ l = 1, 2, \ldots, n_{ik}. \]

If \( n_{ik} > 1 \), it is assumed that the unit dispatched to atom \( k \) when the state is \( B_i \) is to be chosen randomly from among the units \( r_1, r_2, \ldots, r_{n_{ik}} \).

Assuming now that the tour is constructed and the optimal units to dispatch are known, the upward \( \lambda_{ij} \) are calculated as follows. The matrix \( \Lambda \) is initialized to zero. Then, at the point on the tour associated with atom \( k \) and vertex \( B_i = \{ b_{N}, b_{N-1}, \ldots, b_{1} \} \), exactly \( n_{ik} \) transition rates will be incremented by \( \lambda f_k / n_{ik} \). In particular,‡

*Various optimality criteria are discussed in Sec. 7.
†In Sec. 7, this calculation is illustrated for one set of assumptions.
‡The operation "a + b" implies that the contents of the register whose value is "a" is replaced by a value "b."
\[
\lambda_{ij} + \frac{\lambda_i}{\eta_{ik}} + \frac{\lambda_f}{\eta_{ik}}
\]  

for all adjacent states \( b_j = \{b_N, b_{N-1}, \ldots, b_r, \ldots, b_1\} \) such that \( d_{ij} = 1 \) and \( b_{r_j} = 1 \neq b_r \). Very frequently there are no ties (i.e., \( \eta_{ik} = 1 \)), in which case only one addition is performed for every vertex for each geographical atom.
6. GENERATING A UNIT-HAMMING-DISTANCE BINARY SEQUENCE

In order to tour the hypercube in a unit-step manner, one is confronted with the problem of generating a complete sequence, \(S_1, S_2, \ldots\), of \(N\)-digit binary numbers, with \(2^N\) unique members in the sequence and adjacent members being exactly unit-Hamming distance apart. Such a sequence represents a complete unit-step tour of the hypercube.

When the binary numbers are identified with their values, the following algorithm generates such a sequence \(S_1, S_2, \ldots, S_{2^N}\):

Step 1: Set \(S_1 = 0, S_2 = 1, M_2 = 2, J = 2\).

Step 2: If \(J > N\), STOP, binary sequence completed.

Step 3: \(M_1 = M_2, M_2 = 2 \cdot M_1, I = M_1\).

Step 4: \(S_{I+1} = M_1 + S_{M_2-I}\).

Step 5: \(I = I + 1\).

Step 6: If \(I < M_2\), go to step 4.

Step 7: \(J = J + 1\).

Step 8: Go to step 2.

Proof (Induction). Clearly \(S_1\) and \(S_2\) represent a unit-Hamming distance complete sequence of 1-digit binary numbers. Next we must show that if a unit-Hamming distance complete sequence of \(k\)-digit binary numbers has been generated \((k = 1, 2, \ldots)\), then steps 3 through 7 with \(J = k + 1\) generate additional (new) members in the sequence to complete a \((k + 1)\)-digit sequence. At step 3, \(M_1 = I = 2^k\) and \(M_2 = 2^{k+1}\). Thus, the \(\ell\)th new member is

\[S_{(2^k+\ell)} = 2^k + S_{(2^{k+1} - 2^k - \ell + 1)}, \quad \ell = 1, 2, \ldots, 2^k.\]
Letting $d_{ij}$ be the Hamming distance between $S_i$ and $S_j$, we clearly have $d(2^k)(2^{k+1}) = 1$. Since by induction the sequence of $k$-digit binary numbers is a unit-Hamming distance sequence, we must have $d(2^{k+1})(2^{k+2}) = 1$, where $k = 1, 2, \ldots, 2^k - 1$, so the new members are unit-Hamming distance apart. Since by induction all $2^k$ members in the $k$-digit sequence are unique, then all $2^k$ new members are unique and are clearly greater than, or equal to, $2^k$ and strictly less than $2^{k+1}$. Thus the new members complete a $(k + 1)$-digit sequence.

As an example, the following sequence is generated for $N = 3$:

```
Step 1:
Initialization

\begin{align*}
S_1 &= \{0,0,0\} \\
S_2 &= \{0,0,1\}
\end{align*}
```

Steps 3-6:
Add "one" in 2nd position and step backwards

\begin{align*}
S_3 &= \{0,1,1\} \\
S_4 &= \{0,1,0\} \\
S_5 &= \{1,1,0\}
\end{align*}

Steps 3-6 again:
Add "one" in 3rd position and step backwards

\begin{align*}
S_6 &= \{1,1,1\} \\
S_7 &= \{1,0,1\} \\
S_8 &= \{1,0,0\}
\end{align*}

STOP.

This tour is depicted on a three-dimensional hypercube in Fig. 1.

It is worth noting that the generated tour is only one of many possible unit-Hamming distance complete tours of the hypercube. Moreover, any complete unit-Hamming distance $k$-digit sequence can be added to the algorithm at step 3 with $j = k + 1$ and $M_2 = 2^k$, and the algorithm will generate complete sequences whose members have any larger number of digits ($k + 1, k + 2, \ldots$).
Fig. 1 — Unit-step tour of the three-dimensional hypercube
\( (b_i = \text{ith binary digit} = 0, 1) \)
Now that we can tour the hypercube in a unit-step fashion, we must determine the identity of the best available unit(s) to dispatch at each vertex. For a certain class of dispatch policies, it turns out that this can be done very efficiently by exploiting the "unit-step" property. This class includes any policy that allows determination of dispatch preferences for units to calls from any particular atom without reference to system state. In other words, for such a policy, one can always say that some unit \( i \), if available, would be the first preference for dispatch to atom \( k \), unit \( j \) would be the second preference, unit \( l \) the third preference, etc. For such fixed-preferences policies, the dispatcher always selects the most preferred available unit, given the system state. Thus, for instance, the dispatch selection criterion may be strict center of mass (SCM), modified center of mass (MCM), expected strict center of mass (SCM), or expected modified center of mass (MCM) \([4]\), and it may include additional complications, such as giving the district's "own" unit first preference regardless of estimated travel times.

More complicated dispatch policies, such as time-average minimization of system-wide travel time (as illustrated by Carter, Chaiken, and Ignall) \([14]\), are not included in the class of policies we are considering. For such policies, the determination of the optimal unit may be a very difficult task that involves many more calculations than illustrated by the techniques in this section and the next one. However, the results of such calculations can be fed directly into the hypercube model, thereby facilitating the computation of steady-state probabilities and system-performance measures.

In order to be concrete (and brief) in the following discussion, we develop the dispatch selection algorithm in the context of one particular fixed-preferences policy, namely the expected modified center of mass (MCM) policy. With this strategy, the dispatcher takes into account the geographical atom of the call and correctly utilizes all the information he has available regarding locations of units. Defining \( q_{ik} \) and \( t_{ik} \) as
the number of the $i$th closest unit to geographical atom $k$, given that all units are available,

time travel from district $i$ to geographical atom $k = \sum_j \beta_{ij} t_{jk},$

the dispatcher using an $\text{MCM}$ policy will select the available unit with minimum $t_{ik}$. If there is a tie, it is broken by random choice.

Suppose we are in the process of touring the hypercube and consider a step from vertex $S_{i-1}$ to vertex $S_i$. We have already determined the identities of the $\eta_{(i-1)k}$ units that are optimal units at vertex $S_{i-1}$, assuming a call from geographical atom $k$. Now, the step can cause any one of the $N$ units to change status. The particular unit whose status is changed is

$$\eta_0 = \log_2 |v(S_{i-1} \cap S_i') \cup \{S_{i-1} \cap S_i\}| + 1.$$  

If $v(S_{i-1}) > v(S_i)$, then unit $\eta_0$ is now available; otherwise it is unavailable.

Consider the former case first. There are three possibilities:

1. Unit $\eta_0$ is now the unique optimal unit.
2. Unit $\eta_0$ is an additional optimal unit.
3. Unit $\eta_0$ is not an optimal unit.

Given the $\text{MCM}$ dispatching criterion, the possibility that applies is readily determined by comparing the travel time of the new unit, $t_{n_0k}$, to $t_{r_{1k}}$, the travel time of a previously optimal unit. If either (1) or (2) applies, certain status variables (e.g., $\eta_{ik}, r_{ik}$) must be changed. If (3) applies, no further computations are required. This completes operations at vertex $S_i$ if unit $\eta_0$ is now available.

Now consider the case in which unit $\eta_0$ is unavailable. There are also three possibilities here:

1. Unit $\eta_0$ was the unique optimal unit.
2. Unit $\eta_0$ was one of the two or more optimal units.
3. Unit $\eta_0$ was not an optimal unit.
Again, the situation that applies is determined by comparing \( t_{n_0k} \) to \( t_{r_1k} \). If (3) applies, nothing further happens. If (2) applies, unit \( n_0 \) is removed from the list \( r_1, r_2, \ldots, r_{\eta_{ik}} \), and the value of \( \eta_{ik} \) is decreased by one. If (1) applies, another optimal unit (or set of optimal units) must be found. This is done by searching the list \( q_{ik} \), starting immediately after the entry for which \( q_{ik} = n_0 \). The first available unit that is found from this list is an optimal unit for vertex \( S_i \). This unit now becomes \( r_1 \). Additional units in the list of \( q_{ik} \) must be examined since the possibility of a tie exists. This additional search is terminated as soon as a \( t_{ik} \) is found that is strictly greater than \( t_{r_1k} \). At this point all the \( r_k \)'s and \( \eta_{ik} \)'s have been computed, and the job is finished.

There exists one further complication for situation (1) above where unit \( n_0 \) is unavailable, and that involves the case for which \( B_i \) is \{1,1,1,\ldots,1,1\}. Occurring once each tour, this is the state in which all \( N \) units are simultaneously unavailable. Obviously, any search for available units will be fruitless. But we want to preserve the generality of the algorithm and avoid a situation in which we would have to test continually for state \{1,1,1,\ldots,1,1\} and invoke special procedures once it is incurred. The problem is solved by defining an artificial unit that is designated as unit \( N + 1 \), which is always available for an \( N \)-unit problem. We make unit \( N + 1 \) particularly unattractive to dispatch by setting \( t_{(N+1)k} = +\infty \) for all \( k \).

Thus, if the algorithm finds \( B_i = \{1,1,\ldots,1\} \), it reluctantly selects unit \( N + 1 \) as the optimal unit. One step later, when it finds that unit \( N + 1 \) was the optimal unit and that some other unit \( i \) (\( i = 1, \ldots, N \)) is available, it immediately designates unit \( i \) as the optimal unit for the new vertex.

The PL/I listing, called TOUR, which implements the above algorithm is given in Ref. 19.

Regarding the generality of the TOUR algorithm, the dispatch criterion enters only at the point of determining which of the three possibilities applies, given either the availability or unavailability of

\(^*\)In actuality, "\(+\infty\)" in a computer program corresponds to a very large yet finite number.
the unit $n_0$. Thus the general structure of the algorithm remains invariant within the large class of fixed-prefereces dispatch policies. To adapt the algorithm to another dispatch policy within this class, the comparison of $t_{n_0k}$ with $t_{r_1k}$ need only be replaced by the analogous comparison or other procedure associated with the new criterion.
8. ANALYSIS OF THE TOUR ALGORITHM

In this section we briefly analyze the efficiency of the TOUR algorithm. As the problem size (reflected by N) increases, we find that each execution of the algorithm actually becomes less time consuming as reflected by the number of operations required at each vertex!

For simplicity in the analysis we ignore the possibility of ties, an assumption that is quite reasonable in most applications. We note that the work required of TOUR is invariant with respect to the particular geography at hand and the values that the state probabilities will assume. The basic tree structure that depicts the execution of the algorithm is given in Fig. 2. There are four possibilities, corresponding to whether the unit whose status has just changed is now available or unavailable and optimal or not optimal. Above each branch in the tree is the conditional probability that that branch will apply. For instance, 50 percent of all unit steps result in a new unit becoming available; thus we have "1/2" above the two branches on the left, which indicate "available" and "unavailable." Supposing a new unit is available, the probability that it is optimal is simply 1/N. Implicit in this computation is the assumption that the unit-step sequence is generated independent of the problem's particular geography, an assumption that appears very reasonable.

Similar conditional probabilities apply for the case in which a unit becomes unavailable. Below each branch in Fig. 2 is the number of operations (computations) required to perform that branch. For instance, one comparison is required to determine whether the new unit is available or unavailable. Similarly, one comparison is needed with MCM (or any other fixed-preferences dispatch policy) to determine whether the new unit is optimal for dispatch. When a change in the optimal unit is found, a fixed number, nUD, of update operations is performed, which is typically two or three simple substitutions.

The one final issue we must investigate in order to complete analysis of the TOUR algorithm is the statistical behavior of the
Fig. 2 — Sequential tree depicting execution of Tour algorithm.
variable $n_S$, the number of operations required in a search for a new optimal unit. (See Fig. 2.) This search occurs in the ordered list $q_{ik}$ and starts immediately after the value of $i$ that corresponds to the unit that was optimal and is no longer available. Assuming one operation for each unit whose status is checked, the value of $n_S$ is equal to one plus the number of successive unavailable units incurred in the list $q_{ik}$.

To determine the statistics of $n_S$ it is convenient to define

\[ q_{i,N} = \frac{1}{2}, q_{2,N} = \frac{1}{4}, \ldots, q_{i,N} = \frac{1}{2^{i}}, \ldots, q_{N,N} = \frac{1}{2^{N}}, q_{N+1,N} = \frac{1}{2^{N}}. \]

Clearly,

\[ q_{1,N} = \frac{1}{2}, q_{2,N} = \frac{1}{4}, \ldots, q_{i,N} = \frac{1}{2^{i}}, \ldots, q_{N,N} = \frac{1}{2^{N}}, q_{N+1,N} = \frac{1}{2^{N}}. \]

Here $q_{N+1,N} = 1/2^N$, corresponding to the vertex $\{1,1,\ldots,1\}$, in which case the artificial $N + 1$st unit is selected as optimal. Now $n_S$ is closely related to the mean depth of the algorithm, as given by

\[ e_N \equiv \frac{1}{N+1} \sum_{i=1}^{N+1} iq_{i,N}. \]

Performing the computation, we find that

\[ e_N = 2 - \left( \frac{1}{2} \right)^N, \]

implying that the mean depth of the algorithm converges exponentially to 2. Loosely speaking, this says that for large $N$, the "average

\[ * \]

In defining $n_S$, the artificial unit, $N + 1$, can be treated in a number of ways. The current definition treats unit $N + 1$ just like any actual unit, assuming the algorithm must check its status even though we know that unit $N + 1$ is always available for an $N$-unit problem.
assignment* is to the second preferred unit. Referring now to \( n_S \), if
the \( i \)th preferred unit was optimal and is no longer available, there
are \( N - i \) actual units and one artificial unit whose status the algo-

rithm may have to interrogate in a search for the next optimal unit.
But the mean depth of this conditional search is simply \( e_{N-1} \). The
probability that the \( i \)th preferred unit was optimal and is no longer
available is proportional to \( q_i N \), the fraction of vertices for which
the \( i \)th best unit is the optimal unit, for all \( i = 1, 2, \ldots, N \). We must
exclude unit \( N + 1 \) from consideration since it never becomes unavail-
able. Thus the average value of the variable of interest is

\[
\bar{r}_S = \sum_{i=1}^{N} \left( \frac{q_i N}{1 - q_{N+1, N}} \right) e_{N-i}
\]

\[
= \sum_{i=1}^{N} \left[ \frac{\frac{1}{2^i}}{1 - \left( \frac{1}{2} \right)^N} \right] \left[ 2 - \left( \frac{1}{2} \right)^{N-1} \right]
\]

\[
= 2 - \frac{N}{2^N - 1}
\]

Thus, the unconditional average value of the number of operations re-
quired in a search for a new optimal unit converges very quickly (from
below) to 2.

We are now in a position to compute the unconditional average total
number of operations required to execute the TOUR algorithm once (i.e.,
at one vertex). Employing the above results and the conditional proba-
bilities given on the tree in Fig. 2, we obtain for this average value

*Here the "average" is taken over the collection of hypercube
vertices and has nothing to do with the state probabilities associated
with those vertices.
\[
\tilde{n}_{\text{TOUR}} = \frac{1}{2} \left[ 1 + \left( 1 - \frac{1}{N} \right) \cdot 1 + \frac{1}{N} (1 + n_{UD}) \right] \\
+ \frac{1}{2} \left( 1 + \left( 1 - \frac{1}{N} \right) \cdot 1 + \frac{1}{N} \left[ 1 + \left( 2 - \frac{N}{2^N - 1} \right) + n_{UD} \right] \right)
\]

\[
= 2 + \frac{n_{UD} + 1}{N} - \frac{1}{2} \frac{1}{2^N - 1}.
\]

What is surprising about this result is the fact that TOUR requires less time to execute for very large problems!

As an example, for a three-unit problem with \(n_{UD} = 3\), \(\tilde{n}_{\text{TOUR}} = 137/42 \approx 3.26\) operations. For a ten-unit problem, again with \(n_{UD} = 3\), \(\tilde{n}_{\text{TOUR}} \approx 2.4\) operations, a reduction in effort per execution of approximately 26 percent. Of course, the total number of vertices grows as \(2^N\), so the reduction per vertex does not dramatically alter the exponential growth of the entire problem size with \(N\).
9. STORING THE STATE TRANSITION MATRIX

The TOUR algorithm speeds execution time and eliminates the need to store very large intermediate matrices [19]. There is one additional storage problem to be confronted, that of storing the state transition matrix $A$ containing $2^{2N}$ elements. This problem is readily solved once certain sparsity and regularity properties of the matrix are exploited. Consider column $j$ of the matrix. It contains exactly $w(B_j)$ terms corresponding to upward transition rates (i.e., $\lambda_{ij}$'s such that $d_{ij} = 1$); $[N - w(B_j)]$ terms that are equal to unity, the service rate; and an on-diagonal term, $\lambda_{jj}$. All other terms, corresponding to transition rates from states greater than unit-Hamming distance away from $B_j$, have a value of zero. Thus, each column contains only $N + 1$ nonzero entries, yielding a matrix density (measured in numbers of nonzero elements divided by the total number of elements) of

$$\gamma_N = \frac{(N + 1)2^N}{2^{2N}} = \frac{N + 1}{2^N}. \quad (6)$$

For $N = 10$, for instance, the density is $\gamma_{10} \approx 0.0107$, reflecting an exceedingly sparse matrix.

Since all the service rates are unity, only the upward transition rates and the on-diagonal terms have to be stored. This results in a further efficiency of nearly 50 percent. The storage procedure is as follows:

1. Designate $MAP(j)$ as the address in array $A$ of the on-diagonal term, $\lambda_{jj}$, corresponding to state $B_j$; i.e., $A[MAP(j)] = - [\lambda_j + w(B_j)]$, where $\lambda_j = \lambda$ for all $j = 0, 1, 2, \ldots, 2^N - 2$, and $\lambda_j = 0$ for $j = 2^N - 1$.
2. Generate the set of adjacent states successively by changing the status of unit 1, then unit 2, ..., unit $j$, ..., and designate $[MAP(j) + w(B_j) + 1 - k]$ as the address in array $A$ of the $kth$ term of the form $\lambda_{ij}$, where $d_{ij}^+ = 1$; $k = 1, 2, \ldots, w(B_j)$. 
The MAP array is generated recursively,

\[ \text{MAP}(j + 1) = \text{MAP}(j) + w(j) + 1, \]

with the initial condition that \( \text{MAP}(1) = 1. \)

Virtually identical techniques could be employed to store the matrix \( \Lambda \) by row rather than by column, if desired. For instance, row \( i \) of the matrix contains exactly \( w(B_i) \) terms corresponding to downward transition rates (which are all unity), \( [N - v(B_i)] \) terms that are upward transition rates, and an on-diagonal term, \( \lambda_{ii} \).

The coefficients stored in the collapsed matrix must be recovered for use in the equations of detailed balance (see Sec. 4). This is easily done since the state \( j \) is fixed for any particular equation. If \( S_j = \{ b_N, b_{N-1}, \ldots, b_1 \} \), one simply starts with \( b_1 \) and the state \( S_1 = \{ b_N, b_{N-1}, \ldots, b'_1 \} \), where \( b'_1 \neq b_1 \). Then \( \lambda_{ij} \) is unity if \( b_1 = 0 \), or equals \( A[\text{MAP}(j) + w(B_j)] \) if \( b_1 = 1 \). One continues in this manner with \( b_2, b_3, \ldots, b_L, \ldots, b_N \) and the associated adjacent vertices \( S_1, S_2, \ldots, S_L \). In general \( S_L = \{ b_N, b_{N-1}, \ldots, b'_L, \ldots, b_1 \} \), where \( b'_L \neq b_L \), and \( \lambda_{ij} = 1 \) if \( b_L = 0 \), or \( \lambda_{ij} = A[\text{MAP}(j) + w(B_j) + 1 - L] \) if \( b_L = 1 \).
10. REDUCTION TO THE M/M/N ZERO-LINE CAPACITY SYSTEM

There is a large-sized problem due to our demand that the busy and idle servers in the system's state description be identified. However, it is clear that if the system is in some state \( B_1 \) with \( w(B_1) \) servers busy, the transition rate downward to a state with one less busy server is \( w(B_1) \) and the transition rate upward to a state with one more busy server is \( \lambda \) (except for \( w(B_1) = N \), in which case the upward transition rate is zero). Thus, if we aggregate states according to the numbers of servers that are busy, we obtain the well-known M/M/N zero-line capacity queuing system. For this system the steady-state probabilities are given as follows:

\[
\text{Prob\{exactly n servers busy\}} = \frac{\lambda^n}{n!} \frac{1}{\sum_{j=0}^{N} \frac{\lambda^j}{j!}}, \quad n = 0, 1, \ldots, N.
\]

In terms of states of the hypercube, the event "exactly n servers busy" corresponds to the set of states on the \( n \)th hyperplane:

\[
H_n = \{ B \in C_N : w(B) = n \}, \quad n = 0, 1, 2, \ldots, N.
\]

Thus, referring back to the equations of detailed balance, we can state the further condition that

\[
P(H_n) = \frac{\lambda^n}{n!} \frac{1}{\sum_{j=0}^{N} \frac{\lambda^j}{j!}}, \quad n = 0, 1, \ldots, N,
\]

implying that the sum of probabilities on each hyperplane is known.
11. ITERATING TOWARD A SOLUTION

We are now ready to derive an iterative procedure for obtaining the steady-state probabilities on the hypercube. Suppose \( \hat{P}_n \{ B_j \} \) is the estimate of \( P \{ B_j \} \) at the \( n \)th iteration \( (n = 0, 1, 2, \ldots) \). Then, in a manner similar to point Jacobi iteration [Ref. 20, p. 57], we use the equations of detailed balance to determine the values at successive iterations:

\[
\hat{P}_n \{ B_j \} = \frac{\sum_{i \in N^+ \cup N^-} \left[ \hat{P}_{n-1} \{ B_i \} \lambda_{ij} + \hat{P}_{n-1} \{ B_i \} \right]}{\lambda + w(B_j)},
\]

\( j = 1, 2, \ldots, 2^N - 2; \ n = 1, 2, \ldots \) (8)

(The probabilities \( P \{ B_0 \} \) and \( P \{ B_{2^N - 1} \} \) are known exactly from the M/M/N model.) In developing the details of the iteration procedure, we are confronted with the problems of (1) providing good initial estimates of the steady-state probabilities, and (2) guaranteeing that the probabilities sum to unity (thereby satisfying Eq. (2)). Both problems can be attacked simultaneously by exploiting the fact that the sum of probabilities along each hyperplane \( H_n \) is known from the M/M/N birth and death model. If one starts with estimated probabilities at iteration step 0 that sum to the exact birth and death values, then without any additional "error correcting," the sum of estimated probabilities along each hyperplane always equals the exact birth and death value, regardless of how deep one is into the iteration procedure (neglecting computer round-off errors). This initialization guarantees that Eq. (2) is satisfied without explicitly including Eq. (2) in the iteration process. It has also been found to yield probability estimates that converge rapidly to the correct values (within the tolerance specified by the convergence criterion).

We prove by induction that the sum of estimated probabilities along each hyperplane maintains its correct value. (It is assumed that at step 0 the procedure is set up with this property.)
Let
\[ P(k) = \frac{\lambda^k}{k!}, \quad k = 0,1,\ldots,N. \]
\[ \sum_{\{B_j \in H_k\}} \hat{P}_{n-1}\{B_j\} = P(k), \quad k = 0,1,\ldots,N. \]

Suppose that for some iteration \( n - 1 \) (\( n > 0 \)), the hyperplane probabilities sum to the correct values, i.e.,
\[ \sum_{\{B_j \in H_k\}} \hat{P}_{n-1}\{B_j\} = P(k), \quad k = 0,1,\ldots,N. \]

Then for iteration \( n \),
\[ \sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = P(k), \quad k = 0,1,\ldots,N. \]

**Proof.** (a) For \( k = 0 \) and \( k = N \), the result is obvious, since \( P\{B_0\} \) and \( P\{B_{2N-1}\} \) are maintained exactly at \( P(0) \) and \( P(N) \), respectively, for all \( n \).

(b) For \( k = 1,2,\ldots,N-1 \), we apply the iteration equation, Eq. (8), and obtain
\[ \sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k} \sum_{\{B_j \in H_k\}} \left[ \sum_{\{B_i \in C_N: d_{ij}^- = 1\}} \hat{P}_{n-1}\{B_i\} \lambda_{ij} \right. \\
+ \left. \sum_{\{B_i \in C_N: d_{ij}^+ = 1\}} \hat{P}_{n-1}\{B_i\} \right]. \]

By interchanging the order of summations between adjacent hyperplanes, we have
\[ \sum_{\{B_j \in H_k\}} \hat{P}_n\{B_j\} = \frac{1}{\lambda + k} \left[ \sum_{\{B_i \in H_{k-1}\}} \sum_{\{B_j \in H_{k+1}\}} \hat{P}_{n-1}\{B_i\} \lambda_{ij} \right. \\
+ \left. \sum_{\{B_i \in H_{k+1}\}} \sum_{\{B_j \in H_{k-1}\}} \hat{P}_{n-1}\{B_i\} \sum_{\{B_j \in H_k\}} (1) \right]. \]
But since the total upward transition rate from each state equals $\lambda$, we have for any $i$ ($i \neq 2^N - 1$),

$$\sum_{\{B_j \in C_i : d_{ij}^+ = 1\}} \lambda_{ij} = \lambda.$$  

Also, since the total downward transition rate from each state in hyperplane $k$ equals $\lambda$, we can write for any $i$ having the property $w(i) = k + 1$,

$$\sum_{\{B_j \in C_i : d_{ij}^- = 1\}} \lambda_{ij} = k + 1.$$  

Thus

$$\sum_{\{B_j \in C_k \}} \hat{P}^n_{n \{B_j \}} = \frac{1}{\lambda + k} \left[ \lambda \sum_{\{B_i \in C_{k-1} \}} \hat{P}^n_{n-1 \{B_i \}} + (k + 1) \sum_{\{B_i \in C_{k+1} \}} \hat{P}^n_{n-1 \{B_i \}} \right].$$

Applying the inductive hypothesis, we can write

$$\sum_{\{B_j \in C_k \}} \hat{P}^n_{n \{B_j \}} = \frac{1}{\lambda + k} [\lambda P(k - 1) + (k + 1)P(k + 1)].$$

Since this last equation is contained in the set of birth and death equations whose solution is

$$P(k) = \frac{\lambda^k}{k!}, \quad k = 0, 1, \ldots, N,$$

we must have

$$\sum_{\{B_j \in C_k \}} \hat{P}^n_{n \{B_j \}} = P(k).$$
Examine the proof, we note that iterations could be performed first for even-numbered hyperplanes, then odd-numbered, then even-numbered, etc. Such a procedure of interleaving even- and odd-numbered hyperplanes provides each computation of Eq. (8) with the most recent available estimates of the unknowns (which, for each equation, are always in adjacent hyperplanes), thereby yielding a Gauss-Seidel procedure that is known to be asymptotically faster than point Jacobi iteration.

To start the iterative procedure, we need to provide initial estimates for all probabilities in odd-numbered hyperplanes, i.e., we need \( \hat{P}_0(B_j) \) for all \( B_j \in H_n \quad n = 1,3,5,\ldots \). For simplicity we assume the probabilities of all states in any given hyperplane are equal and set

\[
\hat{P}_0(B_j) = \frac{\lambda^{w(j)}}{w(j)! \cdot \left( \sum_{i=0}^{N} \frac{\lambda^i}{i!} \right)^N}.
\]

(9)

recognizing the numerator as the appropriate steady-state probability of the corresponding M/M/N system and the denominator as the binomial coefficient, the total number of states in \( H_w(j) \). Then the iterations indicated by Eq. (8) are performed in an oscillating manner, first for even-numbered hyperplanes, then odd-numbered, etc. The iterations are terminated as soon as some convergence criterion is satisfied.

Round-off errors are no problem with the proposed method. As is common with most iterative techniques, round-off errors do not accumulate because the same (exact) matrix of coefficients at each iteration is being used. Thus, any round-off error is due only to the current iteration. Estimation errors, whose values are determined by the convergence criterion, would be more significant.

*It can be shown that the matrix \( \Lambda \) is irreducibly diagonally dominant [see Ref. 20, p. 23, for definition] and that therefore both point Jacobi and point Gauss-Seidel iteration methods are convergent [Ref. 20, p. 73]. Moreover, the matrix \( \Lambda \) is readily shown to have a nonnegative point Jacobi matrix [Ref. 20, p. 57]. In such a case, Stern and Rosenberg's theorem [21] states that point Gauss-Seidel iteration is asymptotically faster than point Jacobi iteration.
12. COMPUTED MEASURES OF PERFORMANCE FOR
A ZERO-LINE CAPACITY SYSTEM

Once the numerical values for the state probabilities are obtained
to a desired accuracy, it is necessary to compute the system's per-
formance characteristics. The present model computes the following:

1. Region-wide mean travel time.
2. Region-wide workload imbalance.
3. Region-wide fraction of dispatches that are interdistrict.
4. Workload of each response unit.
5. Mean travel time to each geographical atom.
6. Mean travel time to each district.
7. Mean travel time of each response unit.
8. Fraction of responses in each district that are interdistrict.
9. Fraction of responses of each response unit that are
   interdistrict.

Such a mixture of performance measures allows one to focus simultane-
ously on several region-wide objectives while assuring that spatial
inequities in the delivery of service are maintained at an acceptable
minimum.

We now develop the mathematical procedures for computing these
performance measures for the zero-line capacity system.

INDIVIDUAL WORKLOADS

The workload of unit \( i \), designated \( \rho_i \), is simply equal to the
fraction of time that unit \( i \) is busy servicing calls. Thus, \( \rho_i \) is
equal to the sum of the steady-state probabilities on the hyperplane
coming to \( b_i = 1 \), i.e.,

\[
\rho_i = \sum_{\{j:b_j=1\}} p_{b_j}. \tag{10}
\]
The unit-step tour derived in Sec. 5 has certain periodicity properties that make the computation of the sum particularly easy.

**REGION-WIDE WORKLOAD IMBALANCE**

There are several measures of workload imbalance.

1. Absolute deviation: \( \Delta W = \max_{i,j} |\rho_{i} - \rho_{j}| \).

2. Variance: \( \sigma_{W}^{2} = \frac{1}{N} \sum_{i=1}^{N} \rho_{i}^{2} - \left[ \frac{1}{N} \sum_{i=1}^{N} \rho_{i} \right]^{2} \).

3. Standard deviation: \( \sigma_{W} = \sqrt{\sigma_{W}^{2}} \).

4. Percentage positive deviation from mean:

\[
\left( \frac{\Delta W}{W} \right)^{+} = \left[ \max_{i} \left( \frac{\rho_{i}}{\frac{1}{N} \sum_{j=1}^{N} \rho_{j}} \right) \right] \cdot 100\% - 100\%.
\]

5. Percentage negative deviation from mean:

\[
\left( \frac{\Delta W}{W} \right)^{-} = 100\% - \left[ \min_{i} \left( \frac{\rho_{i}}{\frac{1}{N} \sum_{j=1}^{N} \rho_{j}} \right) \right] \cdot 100\%.
\]

For the remainder of the system performance characteristics, it is necessary to compute

\( \rho_{jk} = \text{fraction of all dispatches that send unit } j \text{ to geographical atom } k \left( \sum_{j,k} \rho_{jk} = 1 \right) \)

By using standard arguments, * the formula is

---

*See, for example, Ref. 22.
\[ \rho_{jk} = \sum_{\{B_{ik} \in E_k\}} \frac{\lambda f_k P\{B_{ik}\}}{\eta_{ik}} \frac{\eta_{ik}}{\lambda (1 - P\{B_k\} / 2^{N-1})}, \quad (11) \]

where \( E_{jk} \) is the set of states in which unit \( j \) is an optimal unit to assign to a call from atom \( k \). The common factor \( \lambda \) in the numerator and the denominator can be ignored. We compute \( \rho_{jk} \) by retracling the unit-step hypercube tour, once for each atom. If, during the course of the tour associated with atom \( k \), we are at some vertex \( B_i \) and have determined the total number of optimal units \( \eta_{ik} \) and their identities, \( r_1, r_2, \ldots, r_{\eta_{ik}} \), then exactly \( \eta_{ik} \) of the \( \rho_{jk} \)'s are incremented:

\[ \rho_{jk} \leftarrow \rho_{jk} + \frac{f_k P\{B_{ik}\}}{\eta_{ik}} \quad j = r_1, r_2, \ldots, r_{\eta_{ik}} \quad (12) \]

neglecting (for the moment) the denominator of Eq. (11). After computations are completed for all \((j,k)\), we scale the result:

\[ \rho_{jk} \leftarrow \frac{\rho_{jk}}{1 - P\{B_k\} / 2^{N-1}}. \quad (13) \]

Now we can state without further comment the formulas for the remaining performance measures.

**INTERDISTRICT DISPATCHES**

**Fraction of Total Dispatches That Are Interdistrict**

\[ FT = \sum_{j=1}^{N} \sum_{k \neq \text{district } j} \rho_{jk} \quad (14) \]
Fraction of Dispatches to Unit \( j \) That Are Interdistrict

\[
F_{ij} = \frac{\sum_{k \in \text{district } j} \rho_{jk}}{\sum_{k=1}^{K} \rho_{jk}}.
\]  

(15)

Fraction of District \( i \) Calls That Require an Out-of-District Unit

\[
F_{Oi} = \frac{\sum_{i \neq j} \sum_{k \in \text{district } i} \rho_{jk}}{\sum_{j=1}^{N} \sum_{k \in \text{district } i} \rho_{jk}}.
\]  

(16)

UNCONDITIONAL MEAN TRAVEL TIME

\[
\bar{T} = \frac{\sum_{j=1}^{N} \sum_{k=1}^{K} \rho_{jk} \tau_{jk}}{\sum_{j=1}^{N} \rho_{jk}}.
\]  

(17)

AVERAGE TRAVEL TIME TO ATOM \( k \)

\[
\bar{T}_{k} = \frac{\sum_{j=1}^{N} \rho_{jk} \tau_{jk}}{\sum_{j=1}^{N} \rho_{jk}}.
\]  

(18)

AVERAGE TRAVEL TIME TO DISTRICT \( i \)

\[
\bar{T}_{D_{i}} = \frac{\sum_{k \in \text{district } i} \rho_{jk} \tau_{jk}}{\sum_{k \in \text{district } i} \rho_{jk}}.
\]  

(19)

AVERAGE TRAVEL TIME OF UNIT \( j \)

\[
\bar{T}_{U_{j}} = \frac{\sum_{k=1}^{K} \rho_{jk} \tau_{jk}}{\sum_{k=1}^{K} \rho_{jk}}.
\]  

(20)
13. MODIFICATIONS FOR AN INFINITE-LINE CAPACITY SYSTEM

Allowing for the appropriate modifications for the calls that incur a queue delay, most of the performance measures of the previous section can be easily calculated for an infinite-line capacity system. It is clear that the equations of detailed balance (Eq. (1)) still hold on the hypercube, but the total probability (Eq. (2)) must include the "infinite tail" as well:

\[ \sum_{i=0}^{2^N-1} P(B_i) + \sum_{j=1}^{\infty} P_Q(j) = 1, \]  

(21)

where \( P_Q(j) \equiv \text{Prob\{exactly } j \text{ calls in queue, assuming steady-state conditions\}}. \)

The reduced birth and death process now contains a countably infinite number of states with the downward transition rate from all states involving a queue equal to \( N \) (since all \( N \) response units are simultaneously busy). The hyperplane probabilities required for initialization at the first iteration are accordingly reduced:

\[ P_H(n) = \frac{\lambda^n}{\sum_{j=0}^{\infty} \frac{\lambda^j}{j!} + \frac{\lambda^N}{N!} \frac{\lambda/N}{1 - \lambda/N}}. \]  

(22)

The probabilities corresponding to an existing queue are given by

\[ P_Q(j) = \frac{\lambda^n}{\sum_{k=0}^{N} \frac{\lambda^k}{k!} + \frac{\lambda^N}{N!} \frac{\lambda/N}{1 - \lambda/N}}, \quad j = 1, 2, \ldots. \]  

(23)
Since Eq. (1) still holds on the hypercube, the iterative solution technique developed in Secs. 5 through 11 still applies; the only modifications are reflected in Eqs. (21) and (22).

Other more significant modifications are required when computing the performance measures, which are addressed below in the same order as in Sec. 12.

**INDIVIDUAL WORKLOADS**

Let

\[ p_Q = \text{Prob}\{\text{a queue of length one or greater exists, assuming steady-state conditions}\}. \]

Clearly,

\[ p_Q = \sum_{i=1}^{\infty} p_Q(i). \]

Now, since each unit works all of the time that a queue exists, an additional term must be added to Eq. (10) to compute the workload of unit \( i \):

\[ \rho_i = \sum_{\{j:b_j=1\}} p(b_j) + p_Q. \tag{24} \]

**REGION-WIDE WORKLOAD IMBALANCE**

All of the previous equations for workload imbalance still apply without modification, where \( \rho_i \) is now given by Eq. (24).

In order to calculate the fraction of dispatches that send a unit to a particular geographical atom, for the case of a queue, we can split \( \rho_{jk} \) into two components,

\[ \rho_{jk} = \rho_{jk}^{[1]} + \rho_{jk}^{[2]} \tag{25} \]
where $\rho_{jk}^{[1]} = \text{fraction of all dispatches that send unit } j \text{ to atom } k$
and incur no queue delay,
$\rho_{jk}^{[2]} = \text{fraction of all dispatches that send unit } j \text{ to atom } k$
and do incur a positive queue delay.

The first term is given by a modification of Eq. (11):

$$\rho_{jk}^{[1]} = \frac{\sum_{B_i \in E_j} f_k P\{B_i\}}{n_{jk}}.$$  (26)

The term $\rho_{jk}^{[2]}$ is equal to the product of three terms: (1) the probability that a randomly arriving call incurs a queue delay; (2) the conditional probability that the call originated from atom $k$, given it incurs a queue delay; and (3) the conditional probability that the call results in the dispatch of unit $j$, given it originates from atom $k$ and incurs a queue delay. Clearly a queue delay will be incurred by any call arriving while all $N$ response units are busy, and thus the first term is

$$P'_Q = P_Q + P\{B_{2N-1}\}.$$

The second term is equal to the fraction of calls $f_k$ that are generated from atom $k$, and is not dependent on the fact that a queue exists. To obtain the third term, we use the fact that queued calls are handled first-come, first-served, regardless of the call's location or the responding unit's location (at the scene of a previous call). Thus, any of the $N$ busy units is equally likely to be assigned to any particular queued call, yielding a conditional probability of $1/N$. Summarizing, we have

$$\rho_{jk}^{[2]} = \frac{f_k P'_Q}{N}.$$  (27)
INTERDISTRICT DISPATCHES

Given the new definition for $p_{jk}'$, reflected in Eqs. (25), (26), and (27), Eqs. (14), (15), and (16) regarding interdistrict dispatches apply without modification.

UNCONDITIONAL MEAN TRAVEL TIME

To compute the unconditional mean travel time we define the "queued-call travel time," as

$$\bar{T}_Q = \sum_{i=1}^{K} \sum_{k=1}^{K} f_if_k\tau_{ik}. \quad (28)$$

To interpret this expression, consider any particular call which incurs a queue delay. The call is generated from atom $k$ with probability $f_k$. Each one of the $N$ busy response units is equally likely to be dispatched to the call. The probability that the dispatched unit must travel from atom $i$ is $f_i$, which is the fraction of region-wide workload generated from atom $i$; the dispatch assignment is made independently of the particular values assumed by $i$ and $k$. Thus, with conditional probability $f_if_k$, the travel time to a queued call will be $\tau_{ik}'$, which is the travel time from atom $i$ to atom $k$. Hence Eq. (28) is the mean travel time to a call that incurs a queue.

The expression analogous to Eq. (17), giving the region-wide unconditional mean travel time, can now be written

$$\bar{T} = \sum_{j=1}^{N} \sum_{k=1}^{K} \rho_{jk}[1] t_{jk} + p'_Q \bar{T}_Q. \quad (29)$$

AVERAGE TRAVEL TIME TO ATOM $k$

Following reasoning analogous to that above, we write

$$\bar{T}_k = \frac{\sum_{j=1}^{N} \rho_{jk}[1] t_{jk}}{\sum_{j=1}^{N} \rho_{jk}[1]} (1 - p'_Q) + \sum_{i=1}^{K} f_i \tau_{ik}' p'_Q. \quad (30)$$
AVERAGE TRAVEL TIME TO DISTRICT $i$

Again, following the above reasoning,

\[
\overline{TD_i} = \frac{\sum \sum \rho_{jk}^{[1]} t_{jk}}{\sum \sum \rho_{jk}^{[1]} j = 1 \text{ to district } i} (1 - \frac{P'}{Q'}) + \frac{\sum \sum f_k^{[1]} t_{jk}}{\sum \sum f_k j = 1 \text{ to district } i} \frac{P'}{Q'}
\]

(31)

AVERAGE TRAVEL TIME OF UNIT $j$

Unfortunately, the author knows of no exact analytical expression for the infinite line-capacity system, analogous to Eq. (20), for the average travel time of unit $j$. The problem in deriving such an expression arises from the fact that a unit’s position when dispatched for the first time back-to-back during a system saturation period (i.e., a period of positive queue length) is not selected from the probability distribution of call locations $f_1$. Such a unit was most probably assigned to its current call when several units were available, and thus its location tends to be near its home location (or district). As an approximation we compute the mean travel time for unit $j$ as follows:

\[
\overline{T_{u_j}} = \frac{\sum \sum \rho_{jk}^{[1]} t_{jk} + \frac{T_{Q'} P'}{N}}{\sum \sum \rho_{jk}^{[1]} j = 1 \text{ to district } i}.
\]

(32)

We recognize that the term reflecting travel time to queued calls is an overestimate that becomes asymptotically exact as the system utilization factor, $\rho = (\lambda/N) + 1$. 
14. ILLUSTRATIVE COMPUTATIONAL RESULTS

To date we have limited computational experience using a PL/I program that implements the techniques of the previous sections. The program is written with fully variable storage that adjusts array sizes to match the requirements implied by N and K. Further details of the computer program are found in Ref. 19.

As an illustration of the use of the program, consider a nine-district linear command with 18 equal-sized atoms, using an MCM dispatching strategy (see Fig. 3). It is assumed that calls are distributed uniformly over the command and that response units are dispatched from mobile positions, with each unit's position (while idle) selected from a uniform distribution over the unit's district, which has unit length. Comparable runs have been made with a zero-line and an infinite-line capacity system. We found that each run costs less than $3.00 on the M.I.T. IBM 370 model 155. In these runs the convergence criterion was a rather simple one: No state probability shall change by more than $10^{-5}$ between successive iterations; as soon as this criterion is satisfied, the iterations are stopped and summary statistics are calculated. To achieve this convergence objective, the algorithm never required more than 11 iterations. The

![Diagram](image)

**Fig. 3** — Nine-district homogeneous linear command; MCM dispatching; mobile units
average number of iterations was approximately eight. Of course, the required number of iterations would increase if the convergence criterion were made more stringent.

Using the results of these runs, the average command-wide average travel distance versus \( \rho(= \lambda/9) \) is plotted in Fig. 4 for both the zero-line and the infinite-line capacity systems. One sees that this travel distance starts at \( E(D) = 1/3 \) at \( \rho = 0 \) (as expected from elementary considerations [Ref. 4, Chap. 3]) and increases monotonically with \( \rho \).

\[
d_{ij} = \begin{cases} 
1/6 & i = j \\
| i - j | / 2 & i \neq j 
\end{cases}
\]

---

**Fig. 4** — Average command-wide travel distance

\( d_{ij} \equiv \) travel distance between atoms

\( i \) and \( j, i, j = 1, 2 \ldots, 18 \)
The average travel distance of the infinite-line capacity system clearly becomes greater than that of the zero-line capacity starting at $\rho \approx 0.4$. The mean travel distance of the latter system remains relatively smaller because of calls that are lost when the system is saturated. In the infinite-line capacity system, calls that arrive when the system is saturated enter a queue and require a mean travel distance of 3.

The fraction of dispatches to each unit that are interdistrict is shown as a function of $\rho$ in Figs. 5a and 5b. The "end effects" of system behavior are particularly apparent in these figures. For instance, at $\rho = 0.1$, units 1 and 9 are sent out of their respective districts for only about 5 percent of their dispatch assignments, whereas the adjacent units, 2 and 8, respectively, experience a fraction of interdistrict dispatches more than twice as great. It is interesting that the "curves" in Figs. 5a and 5b, which are symmetric about unit 5, are bimodal for small $\rho$ but become unimodal for moderate and large values of $\rho$. Thus, as $\rho$ gets large the center unit (unit 5) experiences the greatest fraction of interdistrict dispatches.

Figures 6a and 6b depict workloads of each unit as a function of $\rho$. One sees that this figure has similar properties to those of Figs. 5a and 5b.

We also have limited computational experience with $N = 10, 11$ and 12, and the cost per run has yet to exceed $15.* Thus the anticipated costs appear to be very reasonable considering the volume of useful information obtained.

*See Ref. 19 for more details.
Fig. 6 — Workload of units -- nine districts, 18 atoms, MCM
15. EXTENSIONS AND GENERALIZATIONS

Since it appears computationally feasible to examine spatially distributed queuing systems with up to 12 response units, it is natural to inquire into certain optimization problems of these systems. One such problem is the district design problem, which, in the sense of the hypercube model, is a decision rule for which unit to dispatch to a call from atom $k$, given a particular system state; such a rule must be determined for each state and each atom $k$. Thus, a "districting" for the problem at hand is actually a set of up to $2^N - 1$ partitionings of the region, one for each nonsaturated system state. The model discussed in this report generates such partitionings, but only for the class of "fixed-preferences" dispatch policies. Carter, Chaiken, and Ignall [14] determined more general rules analytically for the case where $N = 2$ so as to minimize time-average system mean travel time, but larger $N$ appeared analytically intractable. Thus, one direction in which we hope to develop the model is toward algorithmic techniques for generating globally optimal dispatch decision rules, or districtings.

The districting problem assumes a set of initial locations, at least for the case of units assigned to fixed facilities. One would also want to develop the model in the direction of improving the initial locations of units. We also hope to do this, most probably using algorithmic approaches.

There are more fundamental extensions and generalizations of the model, and these relate to the basic model assumptions. For instance, the assumption of identical exponentially distributed service times might be troublesome in some applications, say where one unit consistently takes longer to service calls than others or where travel time is a nonnegligible fraction of the total service time. It is straightforward to replace each of the constant service rates ($\mu$) with service rates that are state-dependent; this only "costs" a doubling of the computer storage required for the state transition matrix. With such a capability, one could, for instance, give each response unit $j$ its "own" mean service time, $\mu_j^{-1}$.
Another problem involves the lack of treatment of call priorities. However, within the hypercube framework, it is possible to allow a limited amount of attention to priorities. For instance, one could define $f_{ij}$ as the fraction of all calls that are generated from atom $i$ and are priority $j$. Then, the dispatch policy could depend on priority level. However, the dispatch policy cannot be of a form that allows calls to be queued while units are still available (otherwise the special structure of the hypercube state space is destroyed). Similarly, one could have queued calls separated by priority and handled first-come, first-served by priority class; but again, analogous to the case of the simple $M/M/N$ infinite-line capacity queue, order of service cannot depend on the identity of the server or the location of the call.

Reference 19 reports on further computational results with the model and progress on the topics above.
REFERENCES


