An Expected Coverage Model with a Cutoff Priority Queue

Soovin Yoon

May 20, 2016

1 Introduction

2 Literature Review

2.1 Queuing Literature

2.1.1 Hypercube Literature

2.1.2 Cutoff Queuing Literature

2.2 Location Literatures with Priorities

3 Approximate Hypercube Model for Cutoff Priority Queue

In this section, we derive a Hypercube model approximation for cutoff priority queue. The Hypercube model can function as a standalone program for analysis of characteristics of cutoff priority queue. Also, the model iteratively interacts with MILP model described in the next section to assist decision-making for deployment and dispatch of ambulances under the uncertainty in their availability. The procedure introduced in this section extends the iterative procedure given by Larson(1975) as well as Jarvis(1985) for approximating equilibrium behavior and performance measures for the Hypercube model. As an extention, two additional features are implemented; calls are distinguished not only by their origins but also by their priorities, and low priority calls are "cut off" to reserve some servers for future high priority call arrivals.

A system with s distinguishable servers and high(low) priority calls from the set of geographical locations J is considered. High(low) priority calls for service arrive in a Poisson manner independent of busy status of servers, at a mean rate $\lambda^H(\lambda^L)$ per hour. The average service time for call from j served by server i is τ_{ij}, which includes the travel time from the station to the scene, the time spent at the scene, transport time to the hospital and return travel time. Servers are distributed among the set of open stations I^{open}, with a

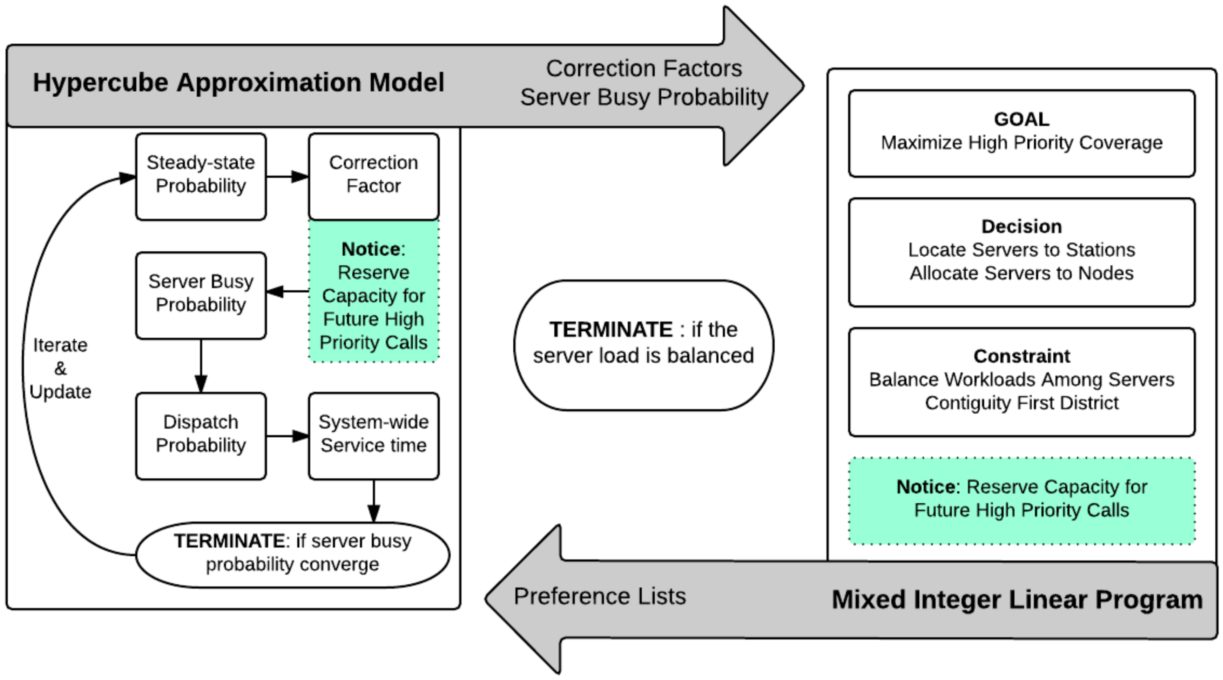
Hypercube Approximation Model
Correction Factors
Server Busy Probability
Steady-state Probability
Correction Factor
Notice: Reserve Capacity for Future High Priority Calls
Server Busy Probability
Iterate & Update
Dispatch Probability
System-wide Service time
TERMINATE: if server busy probability converge
TERMINATE : if the server load is balanced
GOAL
Maximize High Priority Coverage
Decision
Locate Servers to Stations
Allocate Servers to Nodes
Constraint
Balance Workloads Among Servers
Contiguity First District
Notice: Reserve Capacity for Future High Priority Calls
Preference Lists
Mixed Integer Linear Program

Figure 1: Iterative Procedure Flow Chart

server at a station. Exactly one server is assigned to a call immediately according to a fixed preference list, as long as there is any available server. In the preference list $\{b^H_{jk}\}(\{b^L_{jk}\})$,the kth preferred station by the high(low) priority call from node j is given as $b^H_{jk}(b^L_{jk}) \in I^{open}$(for example, if $b^H_{12} = 3$ and $b_L12 = 2$, then the second preferred server by the call from node 1 is 3 if the call is high priority and it is 2 if the call is low priority).

In order to implement the function of reserving resources on the system, a cutoff level of s_1 is given for admission of low priority customers. This means that low priority calls are placed in a queue or lost whenever s_1 or more servers are busy, instead of being served immediately. High priority calls are immediately served by the most preferred available server as long as there are free servers. They are either put in a queue or lost if all servers are busy. This implies that the server cutoff level for high priority customers is set to s by convention(Larson 1986). The service is assumed to be non-preemptive.

We consider the two type of model that differs by the queue capacity. Under the assumption of Loss model $0-$line capacity queue ($M/M/s/s$), high priority calls arriving when all servers are busy or low priority calls arriving when there are more than or equal to s_1 busy servers are "lost to the system", as the queue has zero capacity. Lost calls are considered as being served by external resources. It is common in the EMS literature to assume the system with no queue, and it is also consistent with the real-world operation(Budge

symbol	description
$j \in J$	set of call nodes
$I \in I$	set of potential stations
$i \in I^{open}$	set of open stations, $I^{open} \subset I$
$p \in \{H, L\}$	set of call priorities
s	total number of servers
s_1	cutoff level for low priority calls
ρ	server utilization
r_i	busy probability of server at station i
r	system-wide server busy probability
P_i	steady-state probability
q_k	correction factor
f_{jk}	probability of dispatching kth preferred server to call nodes j
b_{jk}	the station where the server kth preferred by call node j is located
λ_j^p	arrival rate of priority p call from node j
$\lambda^H(\lambda^L)$	arrival rate of high(low) priority calls
λ	system-wide call arrival rate
τ_{ij}	mean service time for calls from node j served by a server from station i
τ	system-wide mean service time
ϵ	server utilization change threshold
δ	server utilization imbalance threshold

Table 1: Summary of Symbols

et al. 2009).

On the other hand, if the queue is assumed to have $\infty-$line capacity($M/M/s/\infty$), calls are put in a queue when they cannot be served immediately. They are served later in a first-come-first-served manner. As the queue is infinitely large, backlogged calls are never lost as they are eventually served in the long run. Note that these calls are counted as lost calls when we compute dispatch probabilities or coverages, since these calls are 'delayed' calls. Here we show the derivation for both queue size assumptions.

Table 1 summarizes symbols used frequently. The hypercube queuing model requires deploying and dispatching policy as inputs, which can be generated either externally(for example, send-the-closest) or by solving the MILP in the next section. There are other parameters needed such as service times and arrival rates, and these can be collected from the data. The outputs of the model are correction factors Q_k for $k = 1, \cdots, s$ and server busy probabilities r_i for $i = 1, \cdots, s$. Then dispatch probability f_{jk}, the probability that a priority p call from node j is served by its kth preferred server, can be approximated as

$$f_{jk}^p = Q_{k-1}^p \Pi_{l=1}^{k-1} r_{b_{jl}^p}(1 - r_{b_{jk}^p}). \tag{1}$$

The correction factors Q_k is needed to approximately correct for the unrealistic assumption of independent servers.

STEP 0 Given:

λ_j^p, τ_{ij} from inputs, open stations I^{open} and preference list b_{jk}

Initialize:

$$\tau^0 = \sum_{j\in J}\sum_{i\in I^{open}}\sum_{p\in\{H,L\}} \tau_{ij}\frac{\lambda_j^p}{\lambda^p} \tag{2}$$

$$\rho = \frac{\lambda\tau}{s} \tag{3}$$

STEP 1 Update steady-state probabilities P_i.

For Loss model, use

$$\begin{aligned} P_i &= P_0\frac{s^i\rho^i}{i!}, \quad 0 \le i \le s_1 - 1 \\ &= P_0\frac{s^i\rho^i}{i!}\big(\frac{\lambda^H}{\lambda}\big)^{i-s1}, \quad s1 \le i \le s \end{aligned} \tag{4}$$

where

$$P_0 = \big(\sum_{i=0}^{i=s1}\frac{s^i\rho^i}{i!} + \sum_{i=s1+1}^{s}\frac{s^i\rho^i}{i!}\big(\frac{\lambda^H}{\lambda}\big)^{i-s1}\big)^{-1}. \tag{5}$$

For Queued model, denoting $\xi_1 = \lambda_H\tau$, $\xi_2 = \lambda_L\tau$, and $\xi = \lambda\tau$, use

$$P_i = P_0\frac{\xi^s}{s!} \quad 0 \le i \le s_1 - 1 \tag{6}$$

$$= P_0(\xi^{s_1}\xi_1^{i-s_1}/i!)s_1/(s_1 - \xi_2 S_Q(s_1, s)) \quad s_1 \le i \le s \tag{7}$$

where

$$P_0 = \{\sum_{i=0}^{s_1-1}\xi^i/i! + (\xi^{s_1}/(s_1-1)!)S_Q(s_1,s)/[s_1 - \xi_2 S_Q(s_1,s)]\}^{-1}, \tag{8}$$

$$S_Q(k,s) = \xi_1^{-k}k![\sum_{i=k}^{s}\rho_H^{i-k}/i! + (\xi_1^s/s!)s/(s-\xi_1)]. \tag{9}$$

STEP 2 Update correction factors Q_k^H and Q_k^L by

$$Q_k^H = \frac{\sum_{j=k}^{s-1} \dfrac{(s-k-1)!(s-j)j!}{(j-k)!s!} P_k}{w^k \rho^k \big(1-(w\rho\big)}, \tag{10}$$

and

$$Q_k^L = \frac{\sum_{j=k}^{s_1-1} \dfrac{(s-k-1)!(s-j)j!}{(j-k)!s!} P_k}{w^k \rho^k \big(1-w\rho\big)}. \tag{11}$$

For Loss model, use

$$w = (1 - P_s \frac{\lambda_H}{\lambda} - \sum_{i=s_1}^{s} P_i \frac{\lambda_L}{\lambda}) \tag{12}$$

and $w = 1$ for Queued model.

STEP 3 Update server busy probabilities r_i as

$$r_i = \frac{V_i}{V_i + 1}. \tag{13}$$

where

$$V_i = \sum_{i \in I^{Open}} \sum_{k \in K} \sum_{p \in \{H,L\}} \lambda_j^p \tau_{b_{jk}^p j} q_k^p (\Pi_{l=1}^{k-1} r_{b_{jl}^p}) \tag{14}$$

STEP 4 Update system-wide service time τ as

$$\tau = \sum_{i \in I^{open}} \sum_{j \in J} \tau_{ij} \frac{\lambda_j^p}{\lambda} \frac{f_{j,a_{ij}^p}^p}{\sum_{k \in K} f_{jk}^p} \tag{15}$$

where f_{jk}^H and f_{jk}^L follows

$$f_{jk}^p = Q_{k-1}^p (1 - r_{b_{jk}^p})(\Pi_{l=1}^{k-1} r_{b_{jl}^p}) \tag{16}$$

STEP 5 Update $\rho = \frac{\lambda\tau}{s}$ and $r = w\rho$.

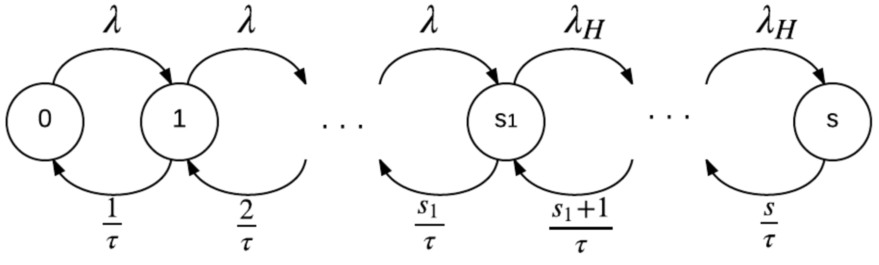

Figure 2: Transition Diagram for 0−line Capacity Queue

Normalize r_i by

$$r_i \leftarrow \frac{r_i}{\sum_i r_i/s} r \tag{17}$$

Check if max change in r_i is below the predetermined threshold.

If yes, terminate with the return value r and Q_k^p. If no, go back to step 1.

First, we need to derive closed-form expressions for steady-state probabilities in cutoff priority queue. For 0−line queue, Figure 2 shows the corresponding transition diagram.

Therefore, by solving the following balance equations, we gain the desired probability.

$$P_1 = \lambda\tau P_0 = \rho s P_0, \cdots, P_i = \frac{\rho^i s^i}{i!} P_0 \qquad i \leq s_1,$$

$$P_{s_1+1} = \lambda_H \frac{\tau}{s_1+1} P_{s_1} = \frac{\rho s}{s_1+1} \frac{\lambda_H}{\lambda} P_{s_1} = \frac{\rho^{s_1}}{s_1+1!} s^{s_1} \frac{\lambda_H}{\lambda} P_0,$$

$$\cdots, P_i = \frac{\rho^i s^i}{i!} (\frac{\lambda_H}{\lambda})^{i-s_1} P_0 \qquad i > s_1$$

In order to derive the steady-state probabilities formulation for the ∞−line queue, SCQQ model in Taylor and Templeton(1980) was referred. In their paper they also provide the formula for the case where high priority customers are lost while only low priority customers are queued (SCQL model). Therefore if one wanted to apply the idea of reserving capacity with low priority queue, that model can be directly used in our procedure too, with updates for steady-state probabilities $P - i$ in STEP 1 and setting the term $w = 1 - P_H P_s$ in STEP 2.

Next, correction factors for low priority in cutoff priority queue is derived. This follows Larson(1975)'s procedure to derive an expression for $P\{B_1b_2 \cdots B_jF_{j+1}\}$ (p852) closely, punctuated by some changes due to cutoff assumption.

Let $B_j \equiv$ event that jth selected is busy (not available), $F_j \equiv B_j^c =$ event that jth server selected is free (or available), $S_k \equiv$ state of the queuing system indicating that exactly k servers are busy, and $P\{B_1B_2 \cdots B_jF_{j+1}\} \equiv$ probability that the first free server is the $j+1$st server selected.

Under the cutoff priority queue assumption, if the arriving call is low priority, we have

$$P\{B_1B_2 \cdots B_jF_{j+1}\} = P\{B_1b_2 \cdots B_jF_{j+1}|S_k, k < s_1\}$$

because we do not select a server to dispatch to a low priority call, if the number of busy server is above the cutoff.

Therefore we wish to derive an expression for $P\{B_1b_2 \cdots B_jF_{j+1}|S_k, k < s_1\}$ that will motivate an approximation procedure for the model in which servers are not identical. (Note: In the hypercube model under a fixed-preference dispatching policy, the dispatcher always assigns the most preferred available server. Therefore the desired probability is the probability that the first j preferred servers are busy and the $j+1$st is free.)

By laws of conditional probability we have

$$P\{B_1b_2 \cdots B_jF_{j+1}|S_k, k < s_1\} = \sum_{k=0}^{k=s_1-1} P\{B_1B_2 \cdots B_jF_{j+1}|S_k\}P_k$$

Where

$$P\{B_1B_2 \cdots B_jF_{j+1}|S_k\} = P\{F_{j+1}|B_1B_2 \cdots B_jS_k\}P\{B_j|B_1B_2 \cdots B_{j-1}S_k\} \cdots P\{B_1|S_k\}.$$

Then we have

$$P\{B_i|B_1B_2 \cdots B_{i-1}S_k\} = \frac{k-(i-1)}{s-(i-1)} \qquad i = 1, 2, \cdots, k+1$$
$$P\{F_{j+1}|B_1B_2 \cdots B_jS_k\} = \frac{s-k}{s-j} \qquad j = 0, 1, \cdots, k$$

Combining above results we have the desired probability

$$\begin{aligned} P\{B_1 b_2 \cdots B_j F_{j+1} | S_k, k < s_1\} &= \sum_{k=j}^{s_1-1} \frac{k}{s} \frac{k-1}{s-1} \cdots \frac{k-(j-1)}{s-(j-1)} \frac{s-k}{s-j} P_k \\ &= \sum_{k=j}^{s_1-1} \frac{(s-j-1)!(s-k)k!}{(k-j)!s!} P_k \end{aligned}$$

with P_k derived in the note above, or,

$$P\{B_1 B_2 \cdots B_j F_{j+1} | S_k, k < s_1\} = Q_j^L r^j (1-r),$$

where

$$Q_j^L = \frac{\sum_{k=j}^{s_1-1} \frac{(s-j-1)!(s-k)k!}{(k-j)!s!} P_j}{(1 - P_s \frac{\lambda_H}{\lambda} - \sum_{i=s_1}^{s} P_i \frac{\lambda_L}{\lambda})^j \rho^j \big(1 - (1 - P_s \frac{\lambda_H}{\lambda} - \sum_{i=s_1}^{s} P_i \frac{\lambda_L}{\lambda})\rho\big)}.$$

Lastly, the derivation of Q_j^H is just analogous to above s_1 replaced by s.

4 Maximum Expected Coverage Model

In this section, a mixed integer linear program model that simultaneously locate servers at stations and dispatch servers to call nodes is introduced. The goal of the model is to maximize expected high priority coverage over the region. A spatial service system with s servers, a set of potential stations I and a set of demand nodes J where calls arrive in a Poisson manner is considered. Exactly s among $\|I\|$ stations are selected as open stations and one server is located for each open station. Also, each node j is assigned a preference list $b_{jk}^H (b_{jk}^L)$, which is an ordered list of open stations for high(low) priority calls. A call is considered to be covered if a server is dispatched to the call immediately upon the arrival and it reaches the call in the predetermined time limit(e.g. 9 minutes).

Define the set of binary variable $y_i = 1(0), i \in I$ if a server is (not) located at station i. This variable set represents deployment decisions that defines the set of open stations. Also define the set of Bernoulli variables $x_{ijk}^p = 1(0), i \in I, j \in J, k = 1, \cdots, s$ if it is (not) the kth preferred station for a priority p call from node j. This variable set captures the dispatch policy. Note that $\{x_{ijk}\}$ is basically a binary representation of the preference list $\{b_{jk}\}$ aforementioned.

The mixed integer linear program is formally stated below.

$$\max \sum_{i=1}^{s} \sum_{j \in J} \sum_{k=1}^{s} c_{ijk} x_{ijk}^{H} \tag{18}$$

subject to

$$\sum_{i \in I} y_i = s, \quad y_i \in \{0,1\} \tag{19}$$

$$\sum_{k=1}^{s} x_{ijk}^{p} = y_i \qquad \forall i, j, p \tag{20}$$

$$\sum_{i \in I} x_{ijk}^{p} = 1 \qquad \forall j, k, p \tag{21}$$

$$\sum_{p,j,k} \lambda_j^p q_k (1-r) r^{k-1} \tau_{ij} x_{ijk}^{p} \le (r+\delta) y_i \tag{22}$$

$$\sum_{p,j,k} \lambda_j^p q_k (1-r) r^{k-1} \tau_{ij} x_{ijk}^{p} \ge (r-\delta) y_i \tag{23}$$

$$x_{ij'1}^{p} \ge x_{ij1}^{p}, \quad \forall p \in \{H, L\}, j \in J, i \in I, j' \in N_{ij} \tag{24}$$

where

$$c_{ijk} = q_{k-1}(1-r) r^{k-1} R_{ij} \frac{\lambda_j^H}{\lambda^H}. \tag{25}$$

The objective function in (18) represents expected coverage over high priority calls. The coefficient set c_{ijk} used in the objective function is detailed in (25), and they represents explicit considerations for uncertainty in the ambulance unavailability and the travel time. Ambulance unavailability is captured by introducing r, the system-wide busy probability. Correction factor q_k, which is the output from previous Hypercube model, yields approximation to actual queuing probabilities that compensate unrealistic independence assumption between the servers. Non-deterministic travel time is also implemented by adopting a parameter R_{ij}, which represents the probability that an ambulance located at station i may reach the node j within the time limit. R_{ij} is naturally a decreasing function of the distance between the station i and node j. Putting the parameters together, $q_{k-1}(1-r)r^{k-1}R_{ij}$ captures the coverage of the server at station i over high priority calls from node j when i is the kth preferred server by high priority call at j. And then this value is weighted by its relative call volume proportion λ_j^H/λ^H to shape the coefficient c_{ijk}, so that the objective function can represents expected coverage over the entire region.

A station is considered to be opened if a server is located, and there are exactly s open stations which is guaranteed by (19). (20) ensures that a server from station i is dispatched only when the station i is open. Also, (21) enforces all call nodes to have an ordered, non-overlapping preference list.

(22) and (23) are balancing constraints that sets upper and lower limits for server utilization. The server

utilization is kept close to average server utilization within the threshold of δ for all servers. This constraint set has two implications. First, the program uses common busy probability r instead of $r_i, i \in I^{open}$ implying the implicit assumption that the busy probability is the same for all ambulances. This assumption is reasonable in an endogenous sense, as the load balancing is approximately guaranteed by (22) and (23) to keep the server utilization equal. Also, this constraints are meaningful in the sense of equity from the service provider's perspective. They ensures that paramedics and emergency medical technicians who staff each server avoid exposure to distress by excessive workload but also have opportunities to practice their skills(McLay and Mayorga 2013).

(24) refers to contiguity constraints, which enforces the program to assign the first priority in a geographically reasonable sense. When the first preferred station for high priority calls from node j is station i, and there is a node j' is a neighborhood of j and closer to i than j, then constraint set (24) enforces i to be the first preferred station for a high priority call from node j.

Finally, note that the cutoff priority queue scheme is brought to the MILP model through the value of q^L_k. q^L_k, which is an output from the Hypercube model, is set to zero for all $k < s_1$. In this program, when i is the kth preferred server by priority p call from node j, the probability of dispatching a server i to priority p call from node j is computed as $q^p_k(1-r)r^{k-1}$. Therefore, the probability of dispatching kth preferred servers to low priority calls are always zero.

Once the MILP problem is solved, the preference list b^p_{jk} is created from the optimal value of x^p_{ijk} accordingly by Algorithm 1.

```
for $j \in J$ do
    for $i \in I^{open}$ do
        $k \leftarrow 1$ while $k \leq s$ do
            if $x^H_{ijk} = 1$ then
                $b^H_{jk} \leftarrow i$;
            end
            if $x^L_{ijk} = 1$ then
                $b^L_{jk} \leftarrow i$;
            end
            $k \leftarrow k + 1$
        end
    end
end
```

Algorithm 1: Pseudo-code for Generating Preference Lists

The system-wide server busy probability r is used as a parameter in objective coefficients as well as

STEP 0 Initialize:

Generate the initial preference lists b_{jk}^p from the *send-the-closest-available-server* rule.

Set correction factors $q_k^p = 1$ for all $k = 1, \cdots, s$, server busy probability $r = \rho = \dfrac{\lambda\tau}{s}$, where system-wide mean service time τ set as $\tau = \lambda^{-1} \sum_i \sum_j \sum_p \lambda_j^p \tau_{ij}$.

STEP 1 Solve the MILP model with inputs r, r_i, q_k^p to update preference lists b_{jk}^p and set of open stations I^{open}.

STEP 2 Solve the Hypercube model with inputs b_{jk}^p and I^{open} to update server busy probabilities r,r_i and correction factors q_k^p.

STEP 3 Check termination criteria. Stop if (1) max change in r_i is less than the imbalance threshold or if (2) MILP was infeasible.

If both termination criteria are not satisfied, go back to **STEP 1**.

Table 2: Iterative MILP-Hypercube Algorithm

load balancing constraints. However, this is an endogenous value in that the decision of deployment and dispatching itself affects the busy probability of servers. Nevertheless, in pursuit of problem tractability, we want to avoid expressing r as a function of decision variable y_i and x_{ijk}^p, because doing so would make the mixed integer program non-linear.

In order to overcome this complication, an iterative scheme is introduced. As aforementioned in the previous section, the Hypercube model is utilized to generate the correction factors q_k and server busy probabilities r used as inputs for MILP model. Then the solutions from the MILP model become the new inputs for the Hypercube model to update the value of correction factors and server busy probabilities. Again, those outputs are utilized as inputs for the MILP model. This iterative procedure continues until outputs converges so that the change in server busy probability drops under the threshold.

Table (2) details the steps in iterative Algorithm that comprehensively describes the iterative use of Hypercube model and MILP model.

5 Computational Results

This section reports analytical and simulation results that illustrates effectiveness of introducing the cutoff priority queue scheme for urban emergency medical service system. A loss system is considered mainly, but computational results for infinite-line queue are also introduced briefly for comparison.

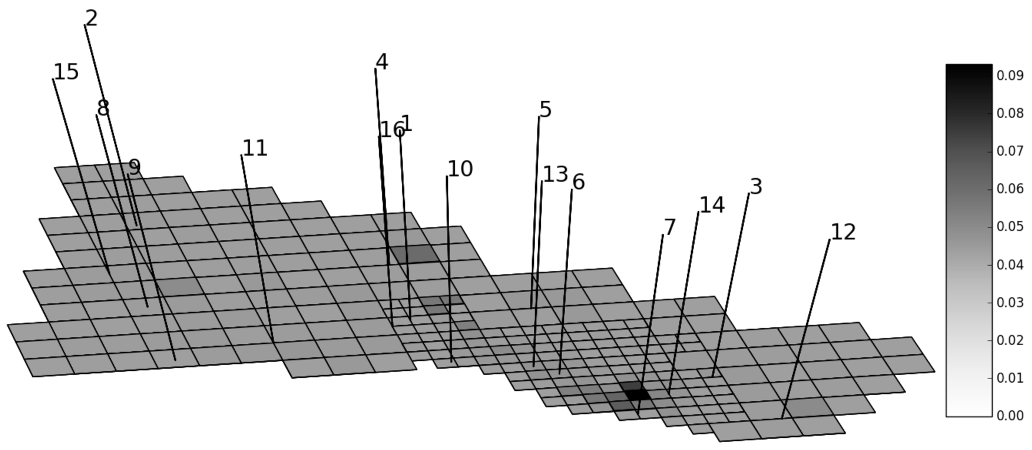

Figure 3: Distribution of Call Demand and Station Location on the Hanover County Map

5.1 Computational Setting

The model is investigated by using real-world data set from Hanover County, Virginia. The data describes a geographical region with 16 potential stations, 270 aggregated demand nodes with 139 2×2 mile cells and 131 1×1 mile cells. Call arrival rate during the weekdays 12PM to 6PM was selected for demand mean. The demand volume over the region is visualized in Figure 3 as a colored cells on a grid that darker color represent higher call volume as it can be seen from the color bar on the right. The call volume over the area is not uniform, as it is concentrated in the middle and south-east area of the county while the call is relatively scarce on the west side. The location of 16 potential stations are also marked in the map.

In pursuit of investigating the effect of reserving capacities in the large fleet system, a scaled-up version of Hanover County dataset was also used. The number of server was increased to 16 from 5, and the demand data was scaled up by a factor of 5.9. Note that the demand wasn't scaled up by the ratio of fleet size(3.2=16/5), but instead the scale parameter was selected so that the system show the similar coverage performance to 5 server case when no server is reserved.

The iterative model was implemented in Python 3.4, and Gurobi 6.5.0 is used as an solver for the Mixed Integer Linear Program. The running time for the iterative MILP-Hypercube procedure using Hanover County data were 27 seconds in average including the time spend for running Hypercube models less than 0.17 second for all cases. The runtime for scaled data with large fleet was – in average. The load imbalance tolerance threshold was set to 10 percent, and server busy probability imbalance tolerance was set to 1 percent.

Throughout the section, analytical results were compared to the result of discrete-event simulation. The purpose of evaluating the result by simulation is to confirm the validity of cutoff model as well as iterative MILP-Hypercube procedure. The simulation used the same demand distribution and geographical data as

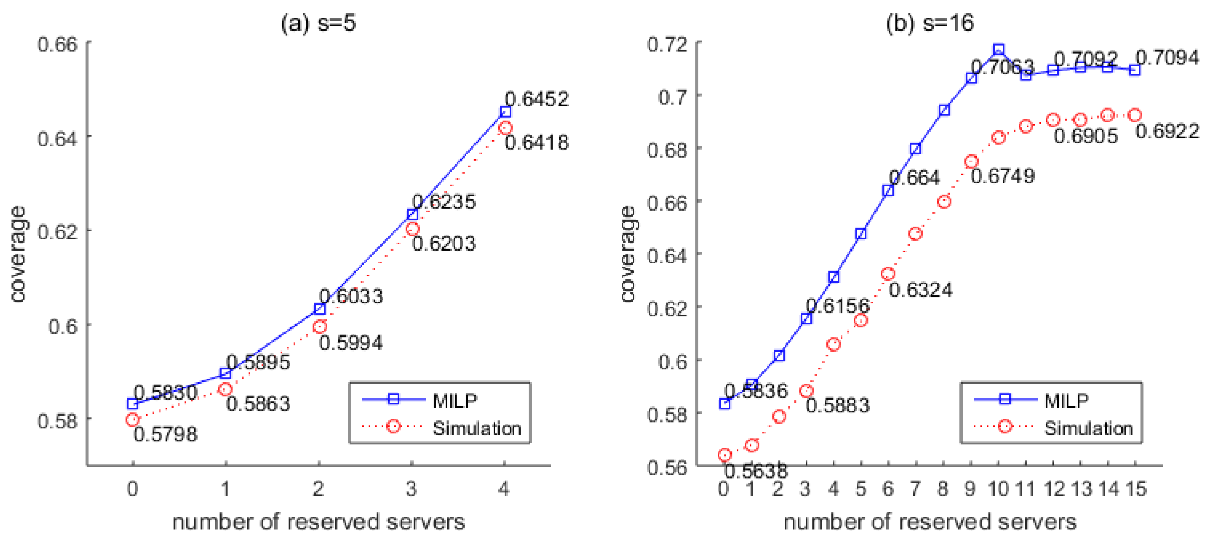

Figure 4: Coverage over High Priority Calls Using Different Cutoff Numbers in (a) Hanover County dataset (b) Scaled Hanover County dataset with Large Fleet

the analytical model, and the resulted deployment and dispatch decisions from the analytical model were also used as inputs for the simulation to set the policy. The program was implemented in Python 3.4, ran until 100,000 call arrives and repeated 30 times to generate summary statistics for dispatch probabilities, server busy probabilities, steady-state probabilities and coverage over the high priority calls. The computational time for the simulation were –,– and – seconds for cutoff values $s1 = 3, 4, 5$ each. Note that the runtime of the simulation is significantly longer than the runtime for Hypercube model that serves the similar function, which is to evaluate the system given the preference lists. In pursuit of precisely getting statistics from rare events with less than 1 percent(P_s from the event that all servers are busy, $f^p(jk)$ for $k = 5$, the simulation should be run for a long time, which highlights the another benefit of developing an analytical model like the Hypercube model.

5.2 Coverage Improvement

The resulted coverages over the high priority calls are displayed in figure 4. The left plot (a) in figure 4 show the resulted expected coverage over high priority calls from MILP model and Simulation with 5 servers. The coverage improvement is monotonic as the system reserve more servers by setting the cutoff number s_1 lower.

The right plot (b) was drawn over a scaled-up version of Hanover County dataset. The effect of capacity reservation on high-priority call coverage is more clear in this large fleet system. Specifically, the marginal improvement was largest when the number of reserved server was increased from 5 to 6, that the decrement of cutoff number s_1 just by 1 resulted the coverage improvement of 1.65 percent. Considering the fact that the maximum achievable coverage of this system is only 81.28 percent (which is gained by assuming that

	Hypercube					Simulation				
	$s_1 = 1$	$s_1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$	$s_1 = 1$	$s1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$
k=1	0.7918	0.7371	0.6980	0.6771	0.6698	0.7887	0.7312	0.6950	0.6721	0.6664
k=2	0.1672	0.1949	0.2033	0.2023	0.2000	0.1628	0.1871	0.1987	0.2011	0.1979
k=3	0.0305	0.0507	0.0656	0.0699	0.0690	0.0356	0.0545	0.0689	0.0712	0.0707
k=4	0.0069	0.0115	0.0220	0.0279	0.0276	0.0087	0.0143	0.0241	0.0311	0.0301
k=5	0.0020	0.0033	0.0062	0.0128	0.0126	0.0028	0.0043	0.0083	0.0142	0.0135
P(lost)	0.0015	0.0025	0.0049	0.0100	0.0209	0.0015	0.0025	0.0050	0.0102	0.0214

Table 3: High Priority Dispatch Probabilities for Different Cutoff Numbers

	Hypercube					Simulation				
	$s_1 = 1$	$s_1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$	$s_1 = 1$	$s1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$
k=1	0.2813	0.5160	0.6296	0.6646	0.6698	0.2833	0.5157	0.6300	0.6605	0.6701
k=2	0.0000	0.0741	0.1503	0.1897	0.2000	0.0000	0.0748	0.1489	0.1941	0.1983
k=3	0.0000	0.0000	0.0280	0.0572	0.0690	0.0000	0.0000	0.0285	0.0568	0.0698
k=4	0.0000	0.0000	0.0000	0.0152	0.0276	0.0000	0.0000	0.0000	0.0152	0.0276
k=5	0.0000	0.0000	0.0000	0.0000	0.0126	0.0000	0.0000	0.0000	0.0000	0.0126
P(lost)	0.7185	0.4093	0.1915	0.0729	0.0209	0.7158	0.4094	0.1923	0.0737	0.0214

Table 4: Low Priority Dispatch Probabilities for Different Cutoff Numbers

every call is served immediately by the server that is located at the most reachable station), the improvement from 64.75 to 66.40 percent by reserving single additional server is significant.

Lastly, in a model validation context, the analytical results were compared to the simulation results. The dotted lines in figure 4 (a) and (b) shows the simulation result that corresponds to the analytical result. The coverage levels for all cutoff numbers were very close for both of analytical MILP model and the simulation model, with differences in coverage result less than – percent in the original data and less than – percent in the large fleet system, for all cutoff number s_1 given.

The resulted effect on the coverage is a collaboration of two factors. First, by reserving more servers for high priority call arrivals, the dispatch probability of sending highly preferred server increases, which results in the improvement in the coverage. Also, with different correction factor and server busy probability given MILP program produces different deployment and dispatch decisions that is optimized for given cutoff number.

Dispatch probabilities for the original Hanover County dataset with 5 servers are shown in left half columns of Table 5.2 and 5.2. Dispatch probabilities for higher k ($k = 1, 2$) for high priority calls increased as the system reserve more servers(lower cutoff number s_1) that result in smaller server busy probability as

	Hypercube					Simulation				
	$s_1 = 1$	$s1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$	$s_1 = 1$	$s1 = 2$	$s_1 = 3$	$s_1 = 4$	$s_1 = 5$
r	0.2082	0.2629	0.3020	0.3229	0.3302	0.2060	0.2618	0.3014	0.3224	0.3300

Table 5: Server Utilization

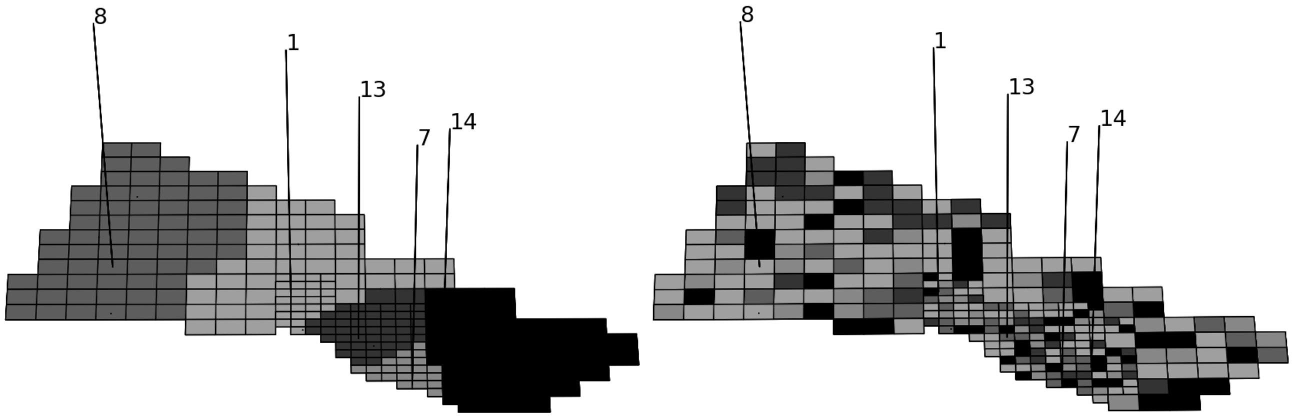

Figure 5: District design for (a) High priority calls (b) Low priority calls in Hanover County dataset

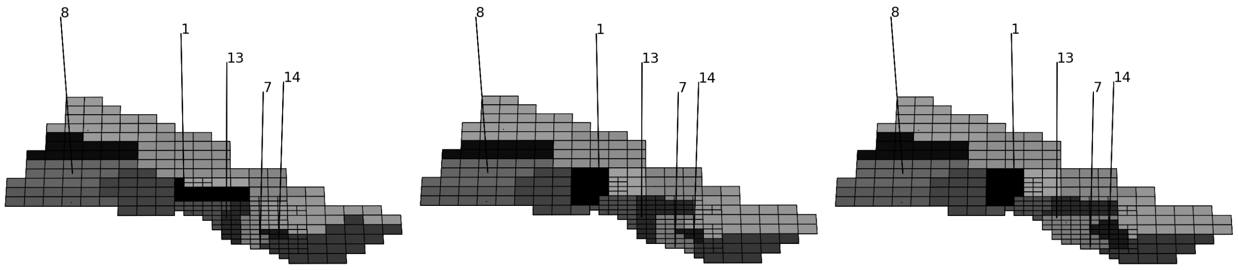

Figure 6: District design for high priority calls with (a) $s_1 = 2$ (b) $s_1 = 8$ (c) $s_1 = 10$ in the scaled Hanover County dataset

seen in Table 5.2. Dispatch probabilities for $k > s_1$ for low priority calls are all zero, which is the direct result of cutoff priority queue discipline.

As we set the cutoff s_1 lower, the probability to dispatch 1st preferred server to high priority calls gets larger, which result in coverage improvement. However, this involves a cost of losing low priority calls more. At the base case of reserving no server, the probability of losing a low priority call is as low as 2.09 percent, and by reserving one server it increases into 7.29 percent, while resulting a coverage improvement of 0.65 percent. As there exist this trade-off between improving high priority coverage by reserving server and staying responsive to low priority calls, it is up to decision maker to choose the level of cutoff with allowable loss for low priority calls.

Finally, dispatch probabilities and loss probabilities gained from simulation under the same setting as MILP are displayed in the right half columns of Table 5.2 and 5.2. Considering the fact the Hypercube model is an approximate model, the discrepancy between the analytical Hypercube model results and simulation results are relatively small enough to conclude that the Hypercube model is valid.

Figure 5 shows the optimal district design for the original Hanover County dataset with 5 servers, cutoff $s1 = 5$. The district design is connected for high priority calls due to contiguity constraint 24, but is not

	High Priority		Low Priority	
	$s_1 = 5$	$s_1 = 4$	$s_1 = 5$	$s_1 = 4$
k=1	0.6606	0.6601	0.6606	0.6432
k=2	0.1983	0.2016	0.1983	0.1846
k=3	0.0688	0.0730	0.0688	0.0559
k=4	0.0276	0.0320	0.0276	0.0149
k=5	0.0127	0.0172	0.0127	0.0000
r	0.3383	0.3376		

Table 6: Dispatch Probabilities for Different Cutoff Numbers With Infinite-line Queue

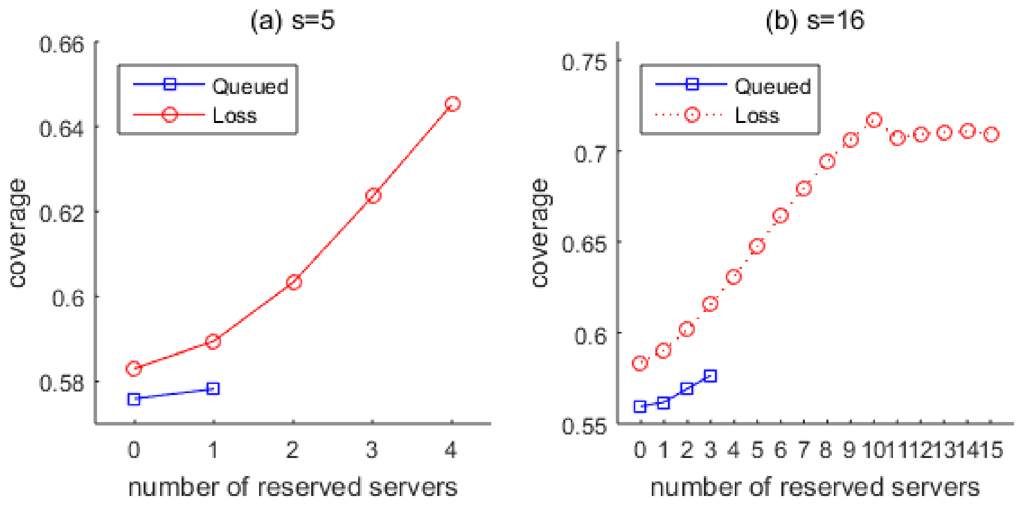

Figure 7: Coverage over High Priority Calls When Calls are Queued, Using Different Cutoff Numbers in (a) Hanover County dataset (b) Scaled Hanover County dataset with Large Fleet. Result for Queued Model is partially presented only for feasible cases.

for low priority calls. In this small dataset, district designs for high priority calls were not affected by the number of reserved servers, while they are in scaled dataset. Figure 6 shows the optimal district designs for the scaled Hanover County dataset with 16 servers. Here district designs for high priority calls differ by the cutoff, which contributes to greater increase in the expected coverage by reserving capacities.

5.3 Infinite-line Queue Systems

In this section we provide result from the system with infinite-line queue.

As it can be seen from Table 6, dispatch probabilities and the system-wide server busy probability r are not changed much when cutoff is modified. This is straightforward as the servers get busy serving the delayed calls in a queue there is minor improvement from reserving capacity.

The coverage result from the Queued model is displayed in Figure7, paralleled with the result from the Loss model. Although reserving capacities result in improved coverage, the amount of improvement is much smaller in the Queued model than the Loss model. This result is straightforward from the previously stated fact that dispatch probabilities are not changed significantly by reserving capacities in the Queued model. Another thing to note is infeasibility of the model regarding load balancing constraint. If delayed calls are queued, it is difficult to balance server workload when cutoff is low. As seen in Figure 7, the Queued model was infeasible for $s_1 \leq 3$ in the original dataset and $s_1 \leq 12$ in the large scaled dataset. Nevertheless, the Queued model has a clear benefit compared to the Loss model that it provides improvements in coverage without abandoning calls or resorting to external services to cover them.

Finally, the computational result for the SCQL model gives similar implications.

TODO: 1. Should I add simulation coverage result here, too? 2. Alternative: Do not put coverage result here, and aim to compare queuing results purely: So put into the same preference lists (send-the-closest) and compare queuing inputs-this way, I can generate tables for al cutoff numbers.