Server Positioning and Response Strategies for Spatially Arriving Jobs with Degradation: Light and Medium Traffic Cases

Fatemeh Aarabi and Rajan Batta,

Department of Industrial and Systems Engineering, University at Buffalo, Buffalo, NY 14260, USA

May 2021

Abstract

This paper studies server positioning and response strategies for spatially arriving jobs with degradation, for situations of light and medium traffic. For the light traffic case it is shown that the p-median solution provides the optimum server positioning, and the optimum response strategy involves no server cooperation. To analyze the medium traffic case, an extended Hypercube queuing model tailored to handle spatially distributed jobs with degradation rate is formulated. The steady state probabilities from this model are used to obtain system performance measures.An appropriate set of preferred server assignments are developed for each region. Results are presented for different problem-size instances. To demonstrate applicability of the model, a case study is presented based on an emergency response application. The main findings for the medium traffic case are that the degree of server cooperation is strongly related to the rate of job degradation and to the cost of assigning jobs that find all servers busy to a backup server—higher job degradation rates and higher cost of assigning a job to a backup server result in lower server cooperation.

Keywords: Hypercube model, Spatial queues, Degradation rate, Emergency response

1 Introduction

Job scheduling has an extensive range of application that range from scheduling jobs in industry to public-sector applications such as responding to medical emergencies. In most of the previous studies, the processing time of jobs is considered to be a fixed amount of time, but in reality, there are situations that processing time is a function of the "start" time. In many applications, there are two system characteristics that should be considered simultaneously: (a) jobs have degradation rate, and (b) jobs are distributed spatially. We aim to solve different categories of spatially distributed jobs with degradation. Common examples of these problems are fire fighting, pothole repair, and humanitarian relief to victims after a disaster. In scheduling problems with spatially distributed jobs with degradation, time is a critical aspect. The importance of time in time dependent problems is well understood to everybody when it comes to responding to an emergency incident. The objective is to minimize the average response time to incidents, while responding to as many of them as possible.

There are three aspects that show the importance of time dependent problems.

- Life protection: In emergency incidents, every minute has a precious value. For instance in a heart attack, the first 60 minutes are decisive and it is called as a golden hour. So treatment during that time is critical.
- Financial: A quick and scheduled response time to a fire incident can prevent significant financial loss. A small fire can turn into a highly destructive one which may result in irreparable economic damage to the residential and commercial buildings as well as urban infrastructure.
- Environmental: In natural disasters like earthquake or in fire fighting there is always a risk of damage to environmental resources.

All the above aspects highlight the significance of a rapid response to time dependent events. These problems are better understood by categorizing them based on queue traffic volume. For a **heavy** traffic queuing system with a large number of jobs, servers are not idle till all of the jobs have been completed. Repairing potholes of a network of roads in a city is an example of a heavy traffic queuing system. This problem is studied previously in (Aarabi and Batta, 2020). In **medium** traffic situations, there are some instances of queuing and many instances of non-preferred server assignment. An emergency response system is an example of a medium traffic queuing system. In such emergency response applications, queues are typically not allowed and queued calls are responded to by a backup service unit. In **light** traffic situations, the number of jobs are small and calls are almost always responded to by their most preferred server (who is almost always available when needed). Discrete location problems are examples of this category. As mentioned before, the heavy traffic case has already been studied. This paper studies the light and medium traffic cases. For the light traffic case, we prove that the p-median solution provides an optimum location for servers. To tackle the medium traffic case, an extended Hypercube queuing model is developed to calculate transition probabilities in steady state situations—the key extension is to capture the effect of job degradation, making the model much more complex than variants of the standard Hypercube situation developed by other researchers. Steady state probabilities are computed and used to generate appropriate performance measures for the system. A computational study is performed for small and large instances. A case study is also presented to illustrate how the model could be be used in an emergency service application.

Before moving forward it is important to point out some key characteristics of a system in which jobs are both spatially distributed and have a degradation effect. Because jobs are spatially distributed, a server needs to travel from their idle location to the scene of the job to deliver service. The decision of which server to send to the call is paramount, and is usually based on travel time. The service time of a job (which is the time interval measured from when the job arrives in the system till when its service is completed) is the sum of the travel time from the current location of the assigned server to the job scene and the on-scene service time at the job scene. The "start" time of the job in such a case is in fact the travel time of the assigned server to the scene of the job. The degradation effect is captured in the changed on-scene service time (which now becomes a function of the travel time). After completion of a job's service, the assigned server returns back to its idle location. The total "service" time for the server from the perspective of the queuing system is the sum of the travel time from its idle location to the scene of the job, the on-scene service time at the job scene, and the travel time back from the scene of the call to its idle location. Another system characteristic that is especially important is the understanding of situations where all servers are busy when a job arrives. In this case, the job is serviced by a backup server at a specified cost (i.e. its servicing is outsourced).

The main contributions of this paper are as follows:

1. A queuing framework is developed to analyze job scheduling for situations that simultaneously have spatially distributed jobs and degradation effects.
2. Three situations relative to the queuing framework are highlighted, light traffic, medium traffic and heavy traffic, with solutions presented for the light and medium traffic cases.
3. It is shown that the p-median problem provides the optimal set of server locations for the case of light traffic; no server cooperation is needed in this case.
4. An extended Hypercube queuing model tailored to handle spatially distributed jobs with degradation rate is formulated for the medium traffic case. The main finding is that higher job degradation rates and higher cost of assigning a job to a backup server result in lower server cooperation.

The rest of this paper is organized as follows: Section 2 contains a literature review. Section 3 contains the analysis of the light traffic case. Section 4 presents the medium traffic case. In section 5 we present results of a set of computational experiments. Section 6 presents a case study for our model. Section 7 provides model limitations and presents other possible modeling schemes. Finally, section 8 presents our conclusions and future research directions.

2 Literature review

Queuing theory is a diverse research topic. Most queuing models consider servers fixed. Since emergency response situations require server travel to customers, the literature review is confined to queuing systems with distinguishable servers that move to customer locations to serve them.

The first paper in this stream of research is that by (Larson, 1974), who proposed the Hypercube queuing model, in which servers travel to the spatially distributed demands. The classic version of the Hypercube model assumes Poisson arrival rates, exponential service times that are independent of service start time, and one server assigned to each call request. After (Larson, 1974), many papers relaxed several key assumptions made in this base model. Several of these papers are now discussed. (Jarvis, 1985) formulates a heuristic model for the Hypercube that allows the service time to be a function of the server and the customer they are serving. In the original Hypercube model each server can be either free or busy so the number of states is 2^n (where n is the number of servers). (Larson and Mcknew, 1982) proposed a model with 3^n states that considers three statuses for each server: free, busy and waiting for dispatch. Later (Atkinson et al., 2008) and (Iannoni et al., 2009) proposed models with 3^n states in which atoms get service from two servers with different service rates. Another example of a model in which regions get service from two servers with different service rates is by (Mendonça and Morabito, 2001), as applied to emergency response on a highway. (Geroliminis et al., 2009) extended the Hypercube model to consider service rates that are independent of the incident's characteristics. Another 3^n version of the Hypercube queuing model is proposed by (Boyacı and Geroliminis, 2015); in their model each server has three statuses: free, busy with intradistrict calls, and busy with inter-district calls.

To model service time degradation in the context of the Hypercube queuing model, this paper takes the following approach: If the most preferred server is assigned, the service time is an Exponential random variable. If the second most preferred server is assigned, the service time is the sum of two Exponential random variables (each with different mean values, implying that it is not an Erlang random variable). This same logic applies if the k^{th} preferred server is assigned.

Another line of work that is relevant to our efforts are probabilistic coverage models. Some major contributions in this area are recapped. (Daskin, 1983) assumes that all servers share a common busy probability and that ambulances operate independently. (Goldberg et al., 1990) extends the model that was first presented by (Daskin, 1983). By using stochastic travel and service times to compute ambulance busy probabilities, (Goldberg et al., 1990) allows for dispatch policies through a preference list, and considers prioritized calls for service. Readers are also pointed to a survey paper on probabilistic location models that by (Galvão and Morabito, 2008). It is noted that only a few papers in this area, however, balance cooperation between the servers.

To study the cooperation between servers, we need to study methods to optimize the preference list for any atom. It is a key component of dispatching strategy that has been studied in the literature. The main concern of emergency systems is to provide suitable and immediate response to arriving requests; however in some situations not sending a free server due to the large distance between the server and request locations leads to a more desirable outcome. This fact can affect the dispatching policy. This dispatching policy is also known in the queuing literature as a cutoff service discipline, meaning a refusal of immediate service to lower priority customers when the number of available servers is below a threshold number (cutoff level)(Iannoni et al., 2015). Usually, highest priority customers are served immediately unless all servers are occupied, in which case they can wait in queue or be lost to the system, depending on the dispatching strategy (Sacks and Grief, 1994). The server reservation policy has been also referred as "idle-server-based threshold-priority" when applied to call center systems, since it allows that low priority customers are queued even if there are servers available (Gans and Zhou, 2003; Gurvich et al., 2008). (Schaack and Larson, 1986) develop an approximated optimization procedure that combines the cutoff with other policies into a procedure that determines the number of servers required in order to minimize staffing costs.

3 Light traffic case

In a light traffic queuing system, the number of servers is large enough so that first preferred server will always be sent to each response request call. The objective is to find the best location of servers that minimizes the average job completion time.

3.1 Motivating example

Suppose there are 10 spatially distributed jobs with degradation rate and three servers that are assigned to serve them. Each job has to be assigned to one of the servers. When moving to or between jobs, both servers travel at a fixed speed of 1 mile per hour. The objective is to find the idle locations for the servers that minimizes the summation of the service times for all the jobs. Let us define x_j as location of server j. The locations of the jobs are shown in Figure 1, with travel distance being measured using the Euclidean metric.

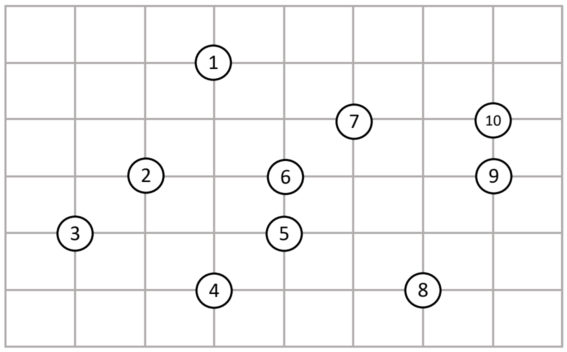

Figure 1: Example for the light traffic queuing system case

Let λ_i denote the arrival rate for job j, and λ denote the total arrival rate for all jobs. The objective function is defined as the summation of service time of all the jobs. Because of degradation consideration the processing time of the job i is defined as $(a_i + t(x_j, i)\alpha_i)$, where $t(x_j, i)$ is the travel time between server j and job i if server j is located in x_j, a_i is the fixed component of the service time of job i and α_i is the degradation rate of job i. Using this notation, the objective function is defined as below:

$$F(x_1, x_2, x_3) = \sum_i \lambda_i/\lambda(a_i + min\{t(x_1, i), t(x_2, i), t(x_3, i)\} * \alpha_i) \tag{1}$$

Given that initial service times (a_i) are fixed, we can equivalently seek to minimize the second part of the objective function:

$$F(x_1, x_2, x_3) = \sum_i (\lambda_i/\lambda * min\{t(x_1, i), t(x_2, i), t(x_3, i)\} * \alpha_i) \tag{2}$$

Let w_i be the product of request rate of job i and its degradation rate, i.e. $w_i = \lambda_i/\lambda * \alpha_i$. It follows that this is equivalent to the well-known p-median problem, with nodal weights w_i. Note that the p-median problem was introduced by (Hakimi, 1964), and its objective is to minimize the total cost of traveling between jobs and the p opened facilities (servers), when each job is assigned to its closest server. See Figure 1. For the case of infinitesimally small degradation rates (i.e. $\alpha_i \approx 0$) the optimal location of the servers are at coordinates $(2, 3), (4, 2), (7, 4)$. For the case with degradation rates as is presented in Table 1 the optimal location of servers are on coordinates $(2, 3), (4, 3), (6, 1)$, i.e. the servers are more spread out.

To study the impact of (dramatically) increasing the degradation rate of a single job, we increase the degradation rate for job 4 to 0.9, now the new locations for the servers are (1,2), (4,3) and (6,1), i.e. the servers are even more spread out.

Table 1: Coordinates and degradation rates of 10 jobs

Jobs	1	2	3	4	5	6	7	8	9	10
Degradation rate (α)	0.02	0.03	0.001	0.01	0.01	0.1	0.02	0.3	0.01	0.01
Coordinate	(3,5)	(2,3)	(1,2)	(3,1)	(4,2)	(4,3)	(5,4)	(6,1)	(7,3)	(7,4)

4 Medium traffic case

In the light traffic case there is no queuing of jobs and the closest server is always available. Furthermore, given the travel time of the closest server to the job, the processing time of the job is assumed to be deterministic. A different approach is taken for the medium traffic case. The whole system is considered as a spatially distributed queuing system, which implies that arrival rates and service times for jobs need to be appropriately defined. Server idle locations also need to be defined, along with server preference lists for jobs. Also, jobs that arrive to find all servers busy need to be handled by a backup server at a cost.

We study a model in which a set of servers $U = \{1, 2, 3, .., N\}$ are assigned to an area. The area of interest is partitioned into M distinct atoms. Each atom generates job requests according to a Poisson process, independently from other atoms, with known rate λ_j. The N servers are stationed at their designated idle locations when available. Each atom has a server preference list, which specifies the order in which servers are sought for assignment. Jobs arrive continuously over time and are either served by the first idle server in the server preference list or by a backup system (with a penalty cost) if all servers in the preference list are busy.

The total "service" time for the server from the perspective of the queuing system is the sum of the travel time from its idle location to the scene of the job, the on-scene service time at the job scene, and the travel time back from the scene of the call to its idle location. Job degradation needs to be incorporated in the service time distribution in a way that can be captured by suitably expanding the states space in the Hypercube queuing model. To do this, we take the following approach: If the first preferred server (server i) is dispatched to a request from atom j, service time is assumed to be an Exponential random variable with parameter μ_i. When the second preferred server (server i) is dispatched to a request, the service time will increase due to job degradation as it takes longer for the server to reach the atom location. The impact of degradation is modeled by assuming that the service time is now given by the summation of two Exponential random variables, the first one with a parameter of μ_i and the second one with a parameter of μ_i/α_j. We extend this modeling framework for the case of more than two servers. If server i is the n^{th} preferred server which is sent to serve the request from atom j, the service time is the sum of n exponential random variables with parameters of μ_i, μ_i/α_j,...,$\mu_i/\alpha_j * (n-1)$. Since the Exponential random variables being summed to arrive at the service time have different parameter values, the resultant service time random variable does not have an Erlang distribution.

The degradation effect on average service time based on which server in the preference list is assigned is now calculated. Towards this end, define f_{ij} as the average service time for jobs originating at atom j which are serviced by server i. This average service time is the sum of (i) the average travel time from the idle location of server i to atom j, (ii) the average on-scene time spent to serve a job at atom j, and (iii) the average travel time from atom j to the idle location for server i. A function is introduced to capture server preferences, a_{ij}, where $a_{ij} = k$ means that server i is the k^{th} preferred dispatch unit for atom j. Using this notation, we get:

$$f_{ij} = \begin{cases} 1/\mu_i & a_{ij} = 1 \\ 1/\mu_i + (1/\mu_i) * \alpha_j & a_{ij} = 2 \\ \vdots & \vdots \\ 1/\mu_i + (1/\mu_i) * \alpha_j + \ldots + (1/\mu_i) * \alpha_j * (n-1) & a_{ij} = n \end{cases} \tag{3}$$

From equation 3 it is evident that the degradation of average service times is a linear increasing function of the preference number of the first available unit. It is assumed that jobs that arrive when all servers are busy do not queue, but are instead handled by a backup system. Let C be the probability that an arriving job finds all servers busy, i.e. it is the system “loss” probability.

The method for modeling degradation is based on the preference number of the first available unit. The logic for this is based on the premise that unit preferences are based on proximity of the unit to the atom under consideration. The linear function of degradation with respect to the preference number of the first available unit assumes that the travel time from the idle location of the n^th preferred unit to the atom under consideration is n times the travel time from the idle location of the first preferred unit to the same atom. This is an approximation to what might be encountered in practice, but allows degradation modeling without explicit consideration of travel time.

Another point that needs to be mentioned is the incorporation of travel time into determination of the service time parameter μ_j of atom j. The well-documented method of mean service calibration can be used to do this.

4.1 Example to illustrate states and transition rate calculations (case of three atoms and two servers)

Consider the simple case of 3 atoms (indicated by circles) and 2 servers (indicated by triangles) shown in Figure 2. Each atom is an independent generator of Poisson service requests. Assume that service rate (μ) of each unit (server) and request arrival rate (λ) as well as the degradation rate of each atom j (α) is as shown in Figure 2. Table 8 presents a full transition matrix for the 2-server example.

Table 2: Dispatch preference for two-server region

Atom number	1	2	3
First preference server	*I*	*II*	*I*
Second preference server	*II*	*I*	*II*

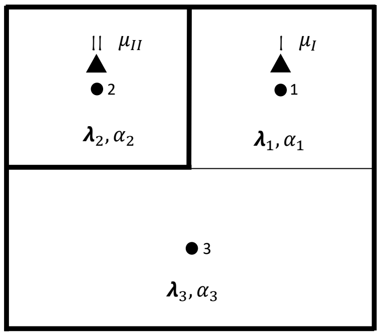

Figure 2: Map of two-server three-atom region

Each atom has a server preference list. The first preferred server for each atom is the closest server to it, i.e. the preference list for each atom is built by considering the closest server to each atom. The server preference for atom 1 is 1,2, for atom 2 is 2,1 and for atom 3 is 1,2. These are shown in Table 2.

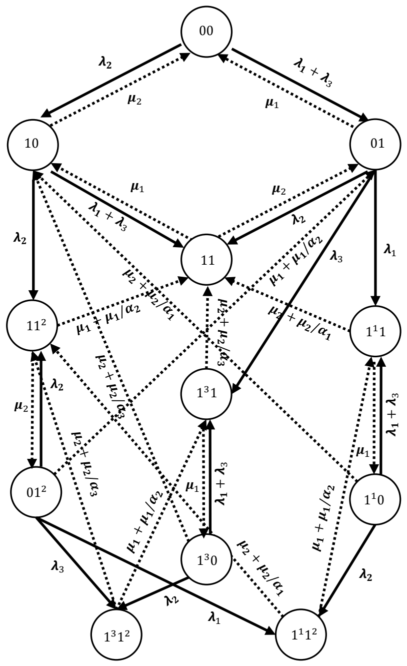

Figure 3: Extended hypercube state space for two-server problem

The expansion of the state space presented for the Hypercube model is motivated by the presentation in (Halpern, 1977). In the traditional Hypercube model we simply distinguish between servers by keeping track of whether a server is busy or idle. For a two server case, this results in four states. For the situation being modeled, when a server is busy servicing an atom for which it is the first preferred server it has a known exponential service time, but when a server is busy servicing an atom for which it is the second preferred server it is necessary to know the atom number it is servicing, as the mean service time of the Exponential random variable is a function of the atom it is servicing. For the situation being modeled, server 1 can be in one of three possible states: 0, corresponding to idle, 1 corresponding to busy when serving a request from an atom for which server 1 is the first preferred server, i.e. either atom 1 or 3, 1^2 corresponds to busy when serving a request from atom 2, for which server 1 is the second preferred server. Server 2 can be in one of four possible states: 0, corresponding to idle, 1 corresponding to busy when servicing a request from an atom 2, for which server 2 is the first preferred server, 1^1 corresponding to busy when serving a request from atom 1 and 1^3 corresponding to busy as serving a request from atom 3, both of which are atoms for which server 2 is the second preferred server. Therefore, there are 12 (3*4) system states as is shown in Table 3 in Appendix A. Figure 3 shows the state space of a two server and three atoms case.

Table 3: System states

00	Both servers are free.
01	Server 1 is busy servicing atom 1 or 3 and server 2 is free.
10	Server 1 is free and server 2 is busy servicing atom 2.
11	Server 1 is busy servicing atom 1 or 3 and server 2 is busy servicing atom 2.
01^2	Server 1 is busy servicing atom 2 and server 2 is free.
11^2	Both servers are busy servicing atom 2.
$1^1 0$	Server 1 is busy servicing atom 2 and server 2 is free
$1^1 1$	Server 1 is busy servicing atom 1 or 3 and server 2 is busy servicing atom 1.
$1^3 0$	Server 1 is free and server 2 is busy servicing atom 3.
$1^3 1$	Server 1 is busy servicing atom 1 or 3 and server 2 is busy servicing atom 3.
$1^3 1^2$	Server 1 is busy servicing atom 2 and server 2 is busy servicing atom 3.
$1^1 1^2$	Server 1 is busy servicing atom 2 and server 2 is busy servicing atom 1.

So far the states in the system have been presented. To illustrate the transition between system states, all the transitions from and to state (11^2) are studied.

Transitions to state (11^2)*:* When the system is in state (10) server 2 is busy, a request from atom 2 will result in server 1 to be sent to atom 2. Since server 1 is the second preferred server for atom 2, the transition will take the system to state (11^2). Being in state (01^2) when a request for service arrives from atom 2, in server 2's primary area, and server 2 is free, server 2 would be sent to answer the request and system will transfer to state (11^2) by rate λ_2. When system is in state $(1^3 1^2)$ and server 2 completes serving atom 3, by rate $\mu_2 + \mu_2/\alpha_3$ transitions to state (11^2). State $(1^1 1^2)$ indicates that server 1 is busy servicing atom 2 and server 2 is busy servicing atom 1. When server 2 completes servicing atom 1, the system transitions to state (11^2) by rate $\mu_2 + \mu_2/\alpha_1$.

Transitions from state (11^2)*:* When the system is in the state (11^2) when server 1 finishes serving atom 2, the system transitions to the saturated state (11) with service rate $\mu_1 + \mu_1/\alpha_2$. In state (11^2), since server 2 is the first preferred server for atom 2, when server 2 finishes servicing atom 2, the transition will take the system to state (01^2) with rate μ_2.

Table 4: Dispatch preference for three-server region

Atom number	1	2	3	4	5	6	7	8	9
First preference unit	*III*	*III*	*II*	*III*	*III*	*II*	*I*	*I*	*II*
Second preference unit	*I*	*II*	*III*	*I*	*II*	*III*	*III*	*III*	*I*
Third preference unit	*II*	*I*	*I*	*II*	*I*	*I*	*II*	*II*	*III*

4.2 Example with 3 servers and 9 atoms

4.2.1 Steady state probability calculations

This sub-section illustrates computation of steady state probabilities for a problem with degradation rate, 3 servers and 9 atoms. In this case, each server can have at most $2 + 3M$ different statuses (where M is the number of atoms), 2 statuses correspond to 0 (free) and 1 (busy), M for the first stage of the second preferred server, and $2M$ for the first and second stages of the third preferred server. For the example shown in Figure 4 with preference list indicated in Table 4, server I can be free (0) or busy (1) while servicing any of atoms 7 or 8 which have server I as their first preferred server (2 statuses). Server I can be busy as well while servicing the first stage of atoms 1, 4 or 9 which have server I as their second preferred server (3 statuses). Finally, server I can be busy while servicing the first and second stages of atoms 2, 3 ,5 or 6 (4*2 statuses). In general, let A_i indicate the number of possible stages for server i, then A_i = 2+ number of atoms that have server i as their second priority + 2$*$ number of atoms that have server i as their third priority. Then the total number of all states are as below:

$$NS = \prod_{i=1}^{n} A_i \tag{4}$$

It is instructive to note that that all of the achieved states from above formulation are not feasible. For instance, in Figure 4, the number of possible states are NS = (2+3+8)$*$(2+2+8)$*$(2+4+2) = 1248.

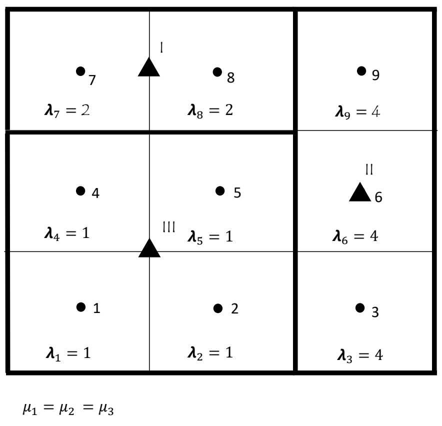

Figure 4: Map of 3-server 9-atom region

As discussed previously, any server can have more than 2 statuses. Each possible state of the system is presented with a vector of three numbers, each one shows the state of the servers. States of server one are represented with a number in the vector [0, 1, 2,..., 9], server two with a number of [0, 1, 10, 11, 12, ..., 19], and server three with a number of [0, 1, 20, 21, 22, 23,..., 28]. To compute performance measures, the first step is to compute the steady-state probability of being in any of the 1248 possible states. Equilibrium probability equations are used to compute the steady-state probabilities. They are defined supposing that the system attains steady state. Here, as an example, the steady state equation for state 100 is presented, which is one of the states of the system.

$$
\begin{aligned}
[(\lambda_1 + \lambda_2 + \lambda_4 + \lambda_5) + (\lambda_3 + \lambda_6 + \lambda_9) + \lambda_2 + \lambda_5 + \lambda_1 + \lambda_4 + \mu_3]p_{100} = \\
\mu_1 p_{101} + \mu_2 p_{110} + \\
+ (\mu_3 + \mu_3/\alpha_6)p_{7,0,0} \\
+ (\mu_3 + \mu_3/\alpha_7)p_{8,0,0} \\
+ (\mu_3 + \mu_3/\alpha_7)p_{8,0,0} \\
+ (\mu_3 + \mu_3/\alpha_8)p_{9,0,0}
\end{aligned}
$$

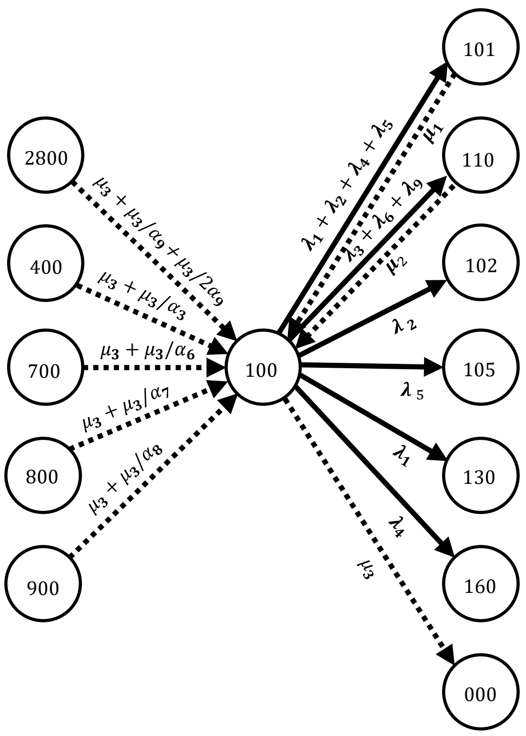

Figure 5: Steady state equation for state 100

4.2.2 Performance measurement

An essential input for calculating coverage and other response-time performance measures is the probability that an incoming call at a particular location is served by a particular server. When these dispatch probabilities are known, many performance measures can be calculated by conditioning on the location of the call and the location of the server, and then using the law of total probability. The calculation of some key system wide performance measures are now presented.

Workload: A useful system performance measure is workload of any individual servers. Workload p_n of server n, can be define as the fraction of time that server n is busy. It is equal to the sum of all steady-state probabilities that server n is busy plus the fraction of time that the system is saturated with all its servers busy.

Fraction of total dispatches interresponse area: Given that we can calculate the p_{nj}'s from the steady state probabilities, we are able to obtain other performance measures such as fraction of total dispatches interresponse area. Let us consider p_{nj} as the fraction of all dispatches that send server n, when available, to atom j and t_{nj} as the associated travel time. The fraction of total dispatches interresponse area can be computed as:

$$f_I = \sum_{n=1}^{N} \sum_{\substack{j \notin \text{response} \\ \text{area n}}} f_{nj} \tag{5}$$

f_{nj} is the fraction of all calls that take server n to serve atom j, that can be computed by $f_{nj} =$

$E_{nj} * \lambda_j/\lambda$, wherein λ_j is the call arrival rate for atom j, λ is the summation of all atom's call arrival rates. E_{nj} is the probability that server n answer requests from atom j.

Average travel times: One of the main performance measures that can be computed in a Hypercube model is average travel time. Different travel times are obtained by the model starting from the origin-destiny matrix of travel times (τ_{ij}). (Larson, 1978) presents the average travel time performance in his paper. For example, the mean travel time of the system (T) is estimated by:

$$T = \sum_{n=1}^{N} \sum_{j=1}^{M} (f_{nj} t_{nj} + P'_Q) \tag{6}$$

We are interested in finding the impact of degradation rate of regions on the performance measurements. The following plots shows the variations in two performance measures with respect to degradation rate. As shown in Figure 6, by increasing the degradation rate the fraction of inter-response area dispatches decreases and the workload of each server increases. It is due to the fact that a higher degradation rate will result in a higher service time, therefore the probability that a second or third preferred server dispatch to serve the atom is low.

4.2.3 Server cooperation

To study the cooperation of servers for a system of jobs with degradation, three kinds of cooperation for the example of 3 servers and 9 atoms are defined: complete cooperation (CC), incomplete cooperation (IC), and no cooperation (NC). In a CC scenario all three servers fully cooperate in serving atoms, i.e. all three servers appear in each atom's preference list. In the IC scenario, two of the three servers service each atom. In the NC scenario, only the first preferred unit is send to service each atom.

To show the effect of degradation rate on server cooperation, rseults from CC, IC, and NC are plotted for different amount of degradation rates in Figure 7. Since a higher degradation rate results in larger service times, by increasing cooperation, the mean travel time for CC increases when compared to IC. Degradation rate is not an effective parameter on mean travel time in the case of NC. In Figure 7 a comparison between mean travel time of CC and IC is made. One can see that mean travel time for CC is always higher than that for IC. This is due to the fact that in a CC system, the second and third preferred units are located farther away from service request location, thus increasing the number of trips that are inter-district.

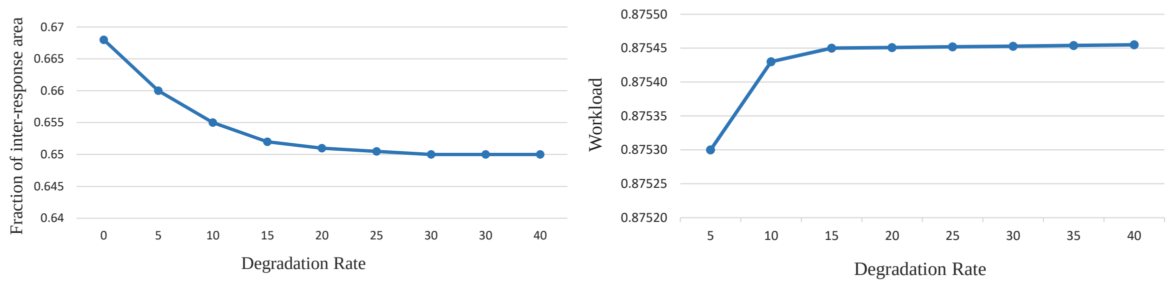

Figure 6: The impact of degradation on the performance measurements

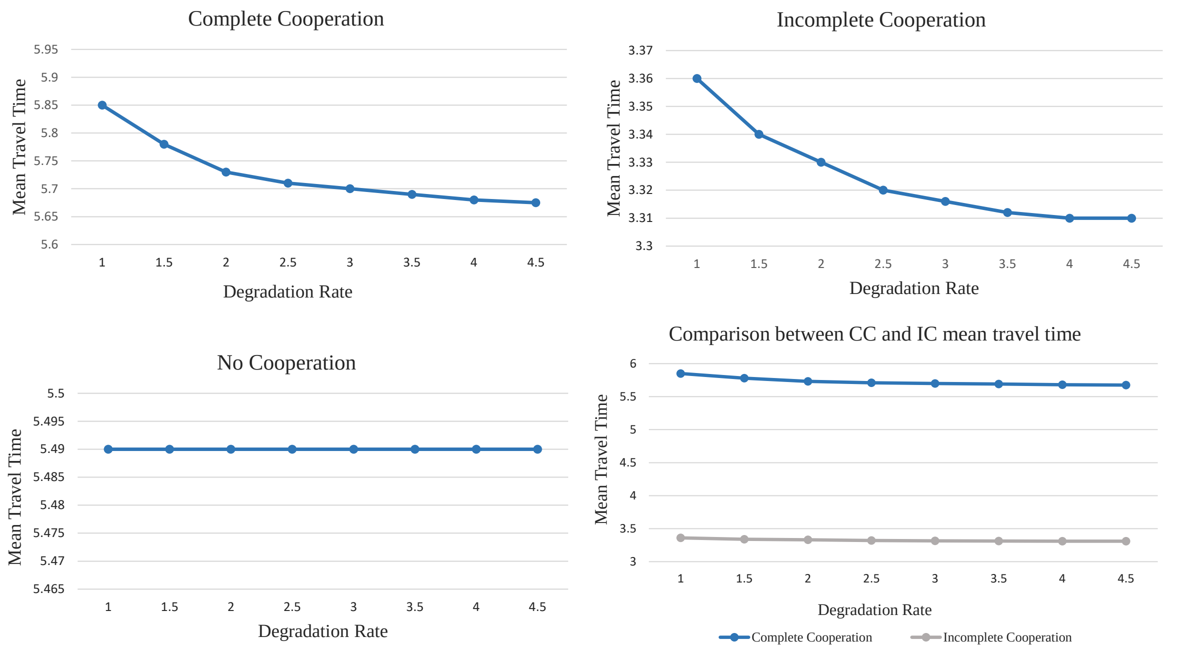

Figure 7: Impact of different levels of degradation rates on the complete, incomplete and no cooperation

Table 5: Test results for small instances

Servers (N)	2														
Atoms (M)	4					5					6				
States (NS)	12					14					21				
Degradation Rate (α)	1	2	10	50	100	1	2	10	50	100	1	2	10	50	100
Run Time	3.08	2.94	2.80	2.77	2.76	2.87	2.68	2.49	2.45	2.44	3.80	3.60	3.39	3.35	3.34
Mean Travel Time	0.19	0.19	0.20	0.21	0.19	0.37	0.35	0.57	0.44	0.37	0.72	1.21	1.04	0.61	0.62
Mean Travel Time (no-queuing)	0.39	0.39	0.40	0.43	0.38	0.88	0.85	1.39	1.06	0.77	1.68	2.84	2.44	1.44	1.47
Servers (N)	2														
Atoms (M)	7					8					9				
States (NS)	28					35					42				
Degradation Rate (α)	1	2	10	50	100	1	2	10	50	100	1	2	10	50	100
Run Time	4.59	4.35	4.10	4.04	4.03	11.29	7.09	13.60	14.29	8.31	11.14	11.35	11.21	11.45	12.05
Mean Travel Time	2.36	2.36	2.57	2.74	2.34	4.70	4.42	4.13	4.06	4.05	5.50	5.16	4.81	4.73	4.72
Mean Travel Time (no-queuing)	4.80	4.82	5.27	5.62	4.80	9.87	9.29	8.69	8.55	8.54	11.35	10.65	9.93	9.76	9.74
Servers (N)	3														
Atoms (M)	4					5					6				
States (NS)	162					231					396				
Degradation Rate (α)	1	2	10	50	100	1	2	10	50	100	1	2	10	50	100
Run Time	285	287	288	289	285	913	934	904	1325	1328	8066	5529	5598	5721	8273
Mean Travel Time	1.23	1.18	1.14	1.13	1.13	1.01	0.95	0.89	0.87	0.87	1.46	1.37	1.28	1.26	1.26
Mean Travel Time (no-queuing)	2.53	2.44	2.35	2.33	2.33	2.44	2.29	2.14	2.11	1.79	3.41	3.21	3.01	2.98	2.97
Servers (N)	3														
Atoms (M)	7					8					9				
States (NS)	648					936					1287				
Degradation Rate (α)	1	2	10	50	100	1	2	10	50	100	1	2	10	50	100
Run Time	24084	24006	24192	23691	23824	22815	26182	26120	26425	25957	55878	55878	55882	55878	55878
Mean Travel Time	1.76	1.63	1.51	1.49	1.48	1.62	1.46	1.32	1.28	1.28	1.35	1.32	1.21	1.18	1.18
Mean Travel Time (no-queuing)	3.58	3.33	3.10	3.06	3.05	3.41	3.08	2.77	2.70	2.70	2.79	2.72	2.50	2.44	2.44

5 Computational results

5.1 Problem instances with small number of atoms

In this section the enhanced Hypercube model is implemented for a small region with two atoms and three number of servers. A 9 (1 mile * 1 mile) atoms region, as is shown in Figure 8, is considered for situations with both 2 and 3 servers. The optimal location of servers are specified using the p-median solution. Arrival rates are considered to be one for all atoms. The μ values for servers 1, 2 and 3 are set as 2.5, 3.5 and 4.5 respectively. The model is run for two and three servers, and different number of atoms and degradation rates as well. For any instance, degradation rate for all atoms are similar. As described in section 4.2.1, the number of states is a function of the number of atoms and the number of servers. To see the effects of these two factors, mean travel time for different combination of server and atom sizes are presented in Table 5, with run times provided in secs.

Two types of travel time statistics are contained in Table 5: (i) mean travel time for the case where calls are queued and are responded to when a server becomes available, and (ii) mean travel time for the case where no queuing of calls is permitted, and lost calls incur a travel time that is equivalent to the maximum travel time in the region (the assumption here is that lost calls are responded to by a backup unit). By examining the last two rows of Table 5 it is evident that increasing degradation rate results in reducing the mean travel time. This is due to the fact that higher degradation rate leads to a larger service time, therefore the cooperation between servers decreases and make travel time smaller. By comparing different instances it is clear that problem instances with larger number of servers have a smaller value of mean travel time. One explanation is that an increase in the number of servers will increase the availability of servers, which contributes to a smaller mean travel time. Moreover, it is evident from Table 5 that there is a significant trade-off between the number of servers and size of the problem.

By comparing two instances of $N = 2, M = 9, \alpha = 100$ (2 servers, 9 atoms, and a degradation rate of 100) and $N = 3, M = 9, \alpha = 100$, it is evident that larger number of servers has a significant effect on the size of problem and consequent run time. The instance with 3 servers has a run time 2220 times greater than that with 2 servers.

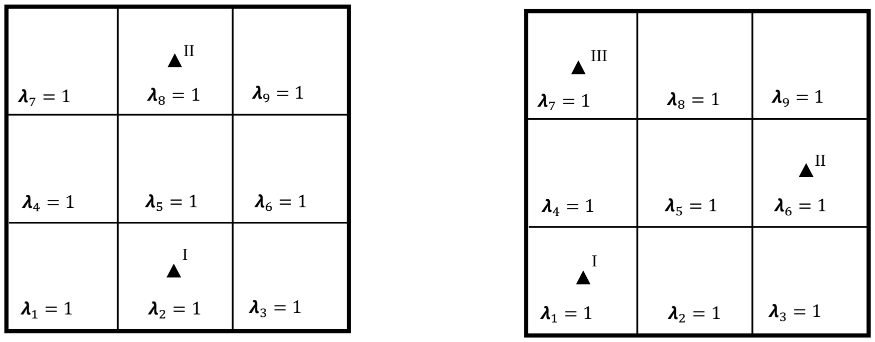

Figure 8: Map of 9-atom problem instances. Left: 2-server, Right: 3-server

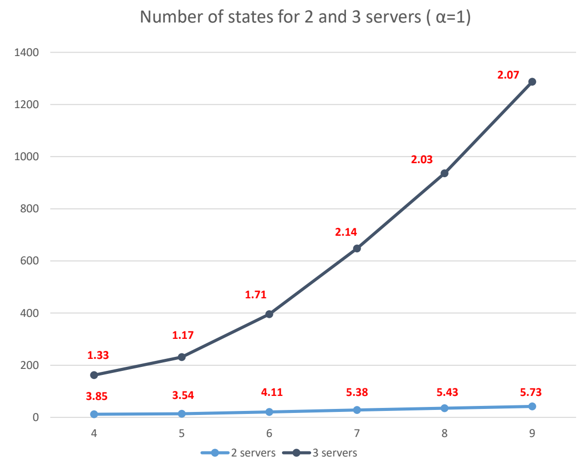

Figure 9: Impact of different number of servers on the number of states

In Figure 9 the result of number of states for different number of atoms with $\alpha = 1$ is presented. Red numbers show the associated mean travel time to each point. The number of servers has a dominant impact on the size and run time of the problem.

5.2 Problem instance with large number of atoms

To address the functionality of our new hypercube model on problems with large number of atoms and servers, a region with 30 atoms and 10 servers is considered, as shown in Figure 10. The servers are located optimally using the p-median algorithm. We divide the region into four sub-regions to make it possible to be solved in an acceptable amount of time. Table 6 contains the mean travel time associated with each sub-region. It is evident that the mean travel time for a larger number of atoms is greater than that of smaller number of atoms.

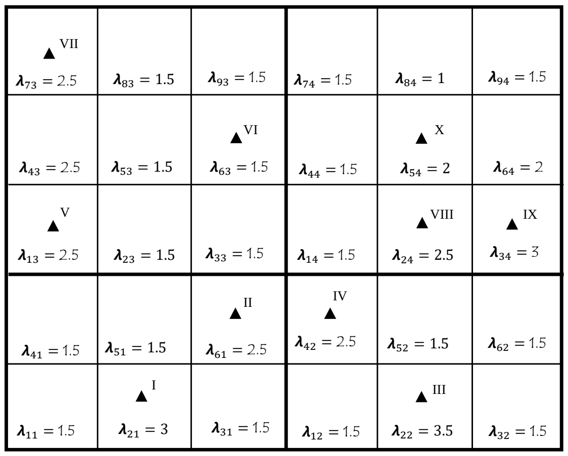

Figure 10: Map of 10 servers and 30 atoms

Table 6: Fixed service time and degradation rates of 10 jobs

Subset	Mean travel time	Number of states	Run time
1	3.45	21	1.37
2	4.36	25	1.77
3	1.42	396	1684.40
4	1.55	396	1650.61

6 Emergency response case study

6.1 Context

The Hypercube queuing model represents an important planning tool that can be used in a variety of applications, including an emergency response system. Response time is one of the system primary performance measures for public safety that can be measured using the Hypercube model. The importance of studying emergency response time stems from the fact that delay in a response may have significant consequences. The goal in emergency response system is to answer requests in a short amount of time and cover as many call as possible. In this section, the extended Hypercube model is applied to this situation after preparation of an appropriate data set. For this purpose, a case study is presented for an application of the proposed framework to determine the response strategies in the City of Fort Collins, CO, when service time is affected by degradation. The characteristics of the region is presented in Section 6.2, and then results are summarized in the analysis in Section 6.3. Furthermore, in Section 6.4, the case study is used to present a convincing argument increased degradation rate necessitates a decrease server cooperation.

6.2 Data used for generation of problem instances

To illustrate the functionality of our model, an experiment was conducted using real world data from Fort Collins city, Colorado, which is the urban zone of Poudre county. The City of Fort Collins is 58.1 square miles and contains about 70,962 housing units. Fort Collins has an estimated population of 172,653. The county has 12 stations that 9 of them are located in the urban area. Each station contains one engine unit and the area is divided into 402 atoms. The deployment of stations is shown in the map by a triangle. Historical call data captures call arrival rates and service rates for each atom of the region. Each atom has two attributes: workload or call requests (λ), and latitude and longitude coordinates. Data of travel time between any two atoms of the area is available for this study as well. The urban area of the City of Fort Collins is divided into nine sub-regions. Each atom of any of sub regions are assigned to the associated server, on the other words, the first preferred server of the atoms of any sub region is the center server of that region. On average there are 15.97 calls each day for this urban area. For testing it was assumed that degradation rate is a factor greater than one. We have assigned degradation rate based on the distance of the atoms from server locations, with atoms further from stations having a lower degradation rate.

6.3 Results

As discussed in section 5, the number of servers has a significant effect on the size of the Hypercube model. It can be understood from Figure 9 that three server models have higher computational run time in comparison to two server models. Due to this, for the case study two servers are assumed to service each sub-region—one is the central server of the sub-region, the other is the closest server to the center of that sub-region. Every atom in the sub-region has a preference list which includes these two servers. Any server which is closest to the atom is its preferred server.

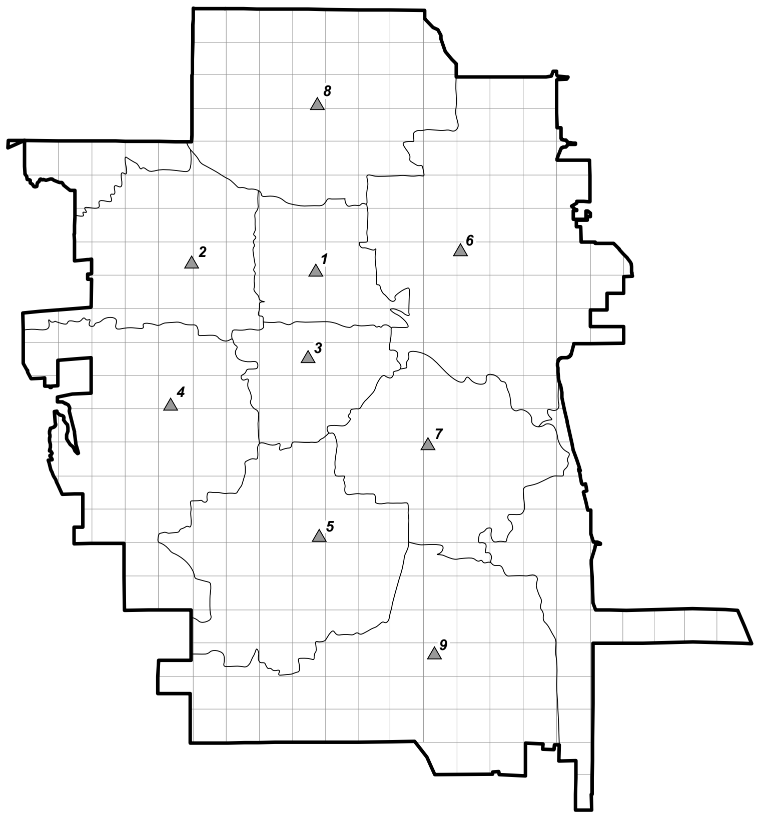

Figure 11: Fort Collins City, Colorado map used for case study

The model was run separately for each sub-region. Results of solving the model for the nine sub-regions are presented in Table 7. The first column indicates the sub-region number for each instance. The second column is the number of atoms in the associated sub-region. The third column shows the number of states in the associated Hypercube model. In column four, the degradation rate (α) is presented. Column five and six show the mean travel time and CPU run time (in secs), respectively.

Each sub-region is solved for four different degradation rates. Here k is a coefficient that get multiplied by the degradation rate, implying that the degradation rate related to $k = 2$ is twice larger than that of $k = 1$. By comparing mean travel times for different amounts of degradation in the Table 7, it is clear that by increasing the degradation rate in the model, mean travel time decreases. The reason why this is is true is because for a higher degradation rate, second preferred servers will

Table 7: Numerical result for case study

Sub region	Number of atoms (M)	Number of states (NS)	Degradation rate (α)	Mean travel time* (MTT)	Run time
1	38	80	$k = 1$	0.026	975
			$k = 2$	0.025	989
			$k = 10$	0.024	854
			$k = 100$	0.022	834
2	35	108	$k = 1$	0.028	999
			$k = 2$	0.026	985
			$k = 10$	0.025	1048
			$k = 100$	0.023	983
3	26	81	$k = 1$	0.024	325
			$k = 2$	0.022	625
			$k = 10$	0.021	324
			$k = 100$	0.020	322
4	56	171	$k = 1$	0.032	6182
			$k = 2$	0.030	6383
			$k = 10$	0.029	12237
			$k = 10$	0.027	6284
5	48	100	$k = 1$	0.035	1144
			$k = 2$	0.035	1143
			$k = 10$	0.032	1129
			$k = 100$	0.029	1161
6	85	174	$k = 1$	0.038	156266
			$k = 2$	0.038	160646
			$k = 10$	0.035	157284
			$k = 100$	0.032	159601
7	53	110	$k = 1$	0.036	1670
			$k = 2$	0.034	1685
			$k = 10$	0.032	1705
			$k = 100$	0.030	1702
8	49	196	$k = 1$	0.031	7287
			$k = 2$	0.029	7449
			$k = 10$	0.027	7546
			$k = 100$	0.025	7225
9	50	104	$k = 1$	0.044	89167
			$k = 2$	0.041	89519
			$k = 10$	0.038	91171
			$k = 100$	0.035	94065

not be dispatched to the scene and first preferred servers will always respond to calls. Since the first preferred server is the one that is closest to each atom, mean travel time is decreased. Mean travel time for larger sub-regions is higher, which is due to the fact that they have more atoms and therefore higher travel time is needed for servers to service them.

6.4 Analyzing the impact of server cooperation

To further study the impact of cooperation between servers the workload of different servers based on the concept of workload measurement presented in section 4.2.2 is compared. The workload can be defined as the fraction of time that server n is busy.

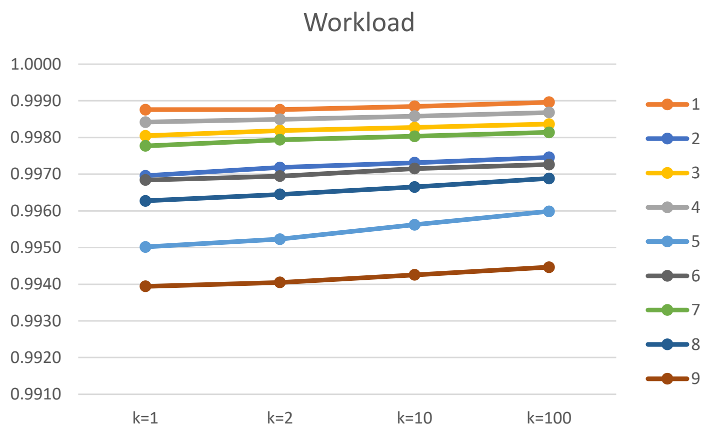

Figure 12: Workload of the servers in terms of degradation rate

Numerical experiments for the case study of 402 atoms for 9 regions is used to compute the workload of any server. The same instances are presented in Table 7, and compute the workload of each server in terms of degradation rate. To better compare the results, a diagram based on the result is presented in Figure 12, which contains the workload of each server with regard to degradation rates. One can see from Figure 12 that by increasing the degradation rate, workload of all individual servers increases as expected. For higher degradation rate, service time increases, therefore to avoid high travel time and to respond to requests in a short amount of time, cooperation between servers should be decreased. Low cooperation between servers means that most calls will be answered by the first preferred server. Hence the workload of the first preferred server in each region increases by increasing degradation rate. Looking more closely, one can see that servers 1 and 4 have the greatest workloads, respectively. This is likely due to the fact that regions 1 and 4 are central regions of the urban area of the Fort Collins city and higher request calls arrive to these centers.

7 Discussion

This paper is an attempt to capture the relationship between server cooperation in a spatially distributed queuing system and the rate of job degradation. It is intuitively clear that the higher the rate of job degradation, the lower the extent of server cooperation, since server cooperation tends to put units “out of position” and results in longer travel times, which in turn result in higher service times for jobs in a job degradation situation. What is not so clear is the quantification of this interactive effect. The spatial queuing model which we analyze allows us to quantify this relationship. Nevertheless, there are other viable ways to approach this problem. One possible method could be to develop a simulation model for an emergency service system in which jobs are subject to job degradation effects and study the system under different levels of server cooperation. Another method could be to develop a mathematical approximation for the statistics of interest and validate this approximation versus the queuing model or the simulation. Then, these statistics could be used directly to answer the questions related to the desired level of server cooperation to match the job degradation that is prevalent for the application at hand.

8 Conclusions and Future Research

Our conclusions are as follows:

- The light traffic case can be solved using the solution to the p-median problem; no server cooperation is needed in this case.
- In the medium traffic case, the number and location of servers has a significant effect on the size of problem. Also, the degree of server cooperation is related to the rate of degradation; the higher the degradation rate the lower the cooperation rate.

An important area for future research is to develop approximation methods for the extended Hypercube model. This will enable the solution of problems with a larger number of servers and atoms.

References

F. Aarabi and R. Batta. Scheduling spatially distributed jobs with degradation: application to pothole repair. *Socio-Economic Planning Sciences*, 2020.

J. Atkinson, I. N. Kovalenko, N. Kuznetsov, and K. Mykhalevych. A hypercube queueing loss model with customer-dependent service rates. *European Journal of Operational Research*, 191(1):223–239, 2008.

B. Boyacı and N. Geroliminis. Approximation methods for large-scale spatial queueing systems. *Transportation Research Part B: Methodological*, 74:151–181, 2015.

M. S. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983.

R. Galvão and R. Morabito. Emergency service systems: The use of the hypercube queueing model in the solution of probabilistic location problems. *International Transactions in Operational Research*, 15:525–549, 2008.

N. Gans and Y.-P. Zhou. A call-routing problem with service-level constraints. *Operations Research*, 51(2):255–271, 2003.

N. Geroliminis, M. G. Karlaftis, and A. Skabardonis. A spatial queuing model for the emergency vehicle districting and location problem. *Transportation Research part B: methodological*, 43(7): 798–811, 2009.

J. Goldberg, R. Dietrich, J. M. Chen, M. G. Mitwasi, T. Valenzuela, and E. Criss. Validating and applying a model for locating emergency medical vehicles in tuczon, az. *European Journal of Operational Research*, 49(3):308–324, 1990.

I. Gurvich, M. Armony, and A. Mandelbaum. Service-level differentiation in call centers with fully flexible servers. *Management Science*, 54(2):279–294, 2008.

S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964.

J. Halpern. Journal article the accuracy of estimates for the performance criteria in certain emergency service queueing systems. *Transportation Science*, 11:223–242, 1977.

A. P. Iannoni, R. Morabito, and C. Saydam. An optimization approach for ambulance location and the districting of the response segments on highways. *European Journal of Operational Research*, 195(2):528–542, 2009.

A. P. Iannoni, F. Chiyoshi, and R. Morabito. A spatially distributed queuing model considering dispatching policies with server reservation. *Transportation Research Part E: Logistics and Transportation Review*, 75:49–66, 2015.

J. P. Jarvis. Approximating the equilibrium behavior of multi-server loss systems. *Management Science*, 31(2):235–239, 1985.

R. C. Larson. A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research*, 1(1):67–95, 1974.

R. C. Larson. *Structural system models for locational decisions: an example using the hypercube queueing model*. Massachusetts Institute of Technology: Operations Research Center, 1978.

R. C. Larson and M. A. Mcknew. Police patrol-initiated activities within a systems queueing model. *Management Science*, 28(7):759–774, 1982.

F. Mendonça and R. Morabito. Analysing emergency medical service ambulance deployment on a brazilian highway using the hypercube model. *Journal of the Operational Research Society*, 52: 261–270, 2001.

S. R. Sacks and S. Grief. Orlando police department uses or/ms methodology, new software to design patrol districts. *OR/MS Today*, 21:30–2, 1994.

C. Schaack and R. C. Larson. An n-server cutoff priority queue. *Operations Research*, 34(2):257–266, 1986.

A Steady state matrix

To provide a better understanding of the transition rate between states in our hypercube model, the steady state matrix for 2-server and 3-atom example is presented in Table 8. Our model is programmed in Python 3.6 on a computer running Windows 10 Enterprise (64bit) with a processor Intel(R)Core(TM) i7-6500U CPU @ 3.50GHz and 7.88GB usable RAM.

Table 8: Generator matrix for example of 3 servers and 3 atoms

	0,0	1,0	1^1,0	1^3,0	0,1	1,1	1^1,1	1^3,1	0,1^2	1, 1^2	1^1,1^2	1^3,1^2
0,0	0	λ_2	0	0	$\lambda_1 + \lambda_3$	0	0	0	0	0	0	0
1,0	μ_2	0	0	0	0	$\lambda_1 + \lambda_3$	0	0	0	λ_2	0	0
1^1,0	0	$\mu_2 + \mu_2/\alpha_1$	0	0	0	0	$\lambda_1 + \lambda_3$	0	0	0	λ_2	0
1^3,0	0	$\mu_2 + \mu_2/\alpha_3$	0	0	0	0	0	$\lambda_1 + \lambda_3$	0	0	0	λ_2
0,1	μ_1	0	0	0	0	λ_2	λ_1	λ_3	0	0	0	0
1,1	0	μ_1	0	0	μ_2	0	0	0	0	0	0	0
1^1,1	0	0	μ_1	0	0	$\mu_2 + \mu_2/\alpha_1$	0	0	0	0	0	0
1^3,1	0	0	0	μ_1	0	$\mu_2 + \mu_2/\alpha_3$	0	0	0	0	0	0
0,1^2	0	0	0	0	5	0	0	0	0	λ_2	λ_1	λ_3
1,1^2	0	0	0	0	0	$\mu_1 + \mu_1/\alpha_2$	0	0	μ_2	0	0	0
1^1,1^2	0	0	0	0	0	0	$\mu_1 + \mu_1/\alpha_2$	0	0	$\mu_2 + \mu_2/\alpha_1$	0	0
1^3,1^2	0	0	0	0	0	0	0	$\mu_1 + \mu_1/\alpha_2$	0	$\mu_2 + \mu_2/\alpha_3$	0	0