Approximation Methods for Large-Scale Spatial Queueing Systems

Burak Boyacı[a,b], Nikolas Geroliminis[a,*]

[a] *Urban Transport Systems Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland*
[b] *Department of Management Science, Lancaster University, UK*

Abstract

Different than the conventional queueing systems, in spatial queueing systems (SQS) the service rate for each customer-server pairs differs and the server that intervenes for a specific customer is not known a priori, depending on the availability of servers at the moment a request was made. These features make the SQS computationally expensive (almost intractable for large scale) but at the same time more suitable for real-life problems with high reliability expectations. Emergency response and on-demand transportation systems are two similar systems that can be modeled with the SQS.

In this research, we aim to solve facility location problems as SQS with stochastic demand and service time. The stochasticity concerned here is temporal and spatial, that emerges from the uncertainty in the demand and service time. In order to tackle this problem Larson (1974)'s 2^n hypercube queueing model (HQM) is extended to 3^n HQM. In this model, there are two different possible service types for each server: (i) service for locations in the proximity of a server (area of responsibility) and (ii) service for other locations where the first responsible server is busy during this event. In addition, to decrease the dimension of the problem, which is intractable due to their size, a new 3^n aggregate hypercube queueing model (AHQM) is developed that treats group of servers (bins) in a similar manner by considering interactions among bins. An efficient graph partitioning algorithm is proposed to cluster servers in groups with an objective to minimize the interactions among groups. Both exact and approximate approaches are integrated inside two optimization methods (i.e. variable neighborhood search and simulated annealing) to find server locations that improve system performance. Computational experiments showed that both models are applicable to use inside optimization algorithms to find good server locations and to improve system performance measures of SQS.

Keywords: spatial queues, hypercube queueing models, emergency response, approximation algorithms

1. Introduction

Location-allocation of *emergency response systems* is one of the oldest problems in the operations research literature. Locating ambulances, fire brigades and police-beats were the pioneer problems mathematically modeled and solved. Although there are quite a few number of works on the subject, many of them disregards the stochastic nature of the problem and find solutions with deterministic assumptions. However, this specific property is the one that differs emergency response system location-allocation problems from the other types of location-allocation problems. This randomness (in demand rates, service times and servers' intervention) creates unexpected congestion and eventually causes losses. While stochasticity in demand and service rates have been included in many researches, the choice of the server based on the state of the system (location of request and availability of other servers) has been addressed in only a few instances for small-scale systems.

There are different methods in the literature dealing with locating emergency response systems. One of the models that was proposed by Larson (1974) models this problem as a spatial queueing model which is also known as 2^n hypercube queueing model (HQM). In 2^n HQM, each emergency response unit is regarded as a server on an Euclidean space and each of them has two states, available and busy generating 2^n possible states for the system (where n is the number of servers); these are the vertices of a hypercube.

Larson (1974) assumed that since the time spent on the way to scene is negligible compared to the service time on scene, the region that is served has no effect on the service time, i.e. for a specific server, service time is the same for any region. This may be acceptable for some systems like fire brigades but not for ambulances. In this research, our aim is to alter Larson (1974)'s 2^n HQM in such a way that enables the model to use different service rates for different server-region pairs. For this purpose a new type of 3^n HQM is proposed. In this 3^n HQM, each server has three states: available, busy inside primary service area (intradistrict) and busy outside primary service area (interdistrict), which creates an intractable hypercube for even medium size problems (with more than 8 servers). In order to tackle larger problems an aggregate method, namely 3^n aggregate HQM (AHQM) is also developed. Instead of estimating each server state separately, 3^n AHQM keeps the number of servers at each of the 3 states (i.e. available, busy with intradistrict, busy with interdistrict) at each *bin* (i.e. set of servers). To identify bins, AHQM is integrated inside mix aggregate hy-

*Corresponding author
Email addresses: b.boyaci@lancaster.ac.uk (Burak Boyacı), nikolas.geroliminis@epfl.ch (Nikolas Geroliminis)

percube queueing algorithm (MHQA). In a nutshell, MHQA has three steps. First, the whole problem area is bi-partitioned to have solvable subproblems. Then, performance measures for each of these partitions are calculated with 3^n HQM. Then these partitions are merged and performance measures are calculated with 3^n AHQM. Both methods are used to find better locations for emergency vehicles to improve a real and an experimental regions with the help of two optimization algorithms: variable neighborhood search (VNS) and simulated annealing (SA).

The remainder of the paper is organized as follows: In the next section, Section 2, we describe significant literature about locating emergency response systems. We start with the early location-allocation models and extend it to very recent emergency response systems literature. Section 3 describes the two models, i.e. 3^n HQM and 3^n AHQM, and their comparison with Larson (1974)'s 2^n HQM. Section 4 contains the definition and steps of MHQA with the mathematical model of the partitioning algorithm. In Section 5, we share the computational results of our two algorithms, 3^n HQM and MHQA. This part contains both the accuracy of the two models compared to the simulation of the real system and the results from the optimization algorithms, VNS and SA. In the last section, we discuss the conclusions and the future research directions. Appendices A1-A3 describe in details the application of the methodologies of sections 3 and 4 in a numerical example.

2. Literature Survey

The earliest models dealing with the location of emergency response systems assume deterministic demand and service. They disregard the stochastic nature of the problem and model the problem by median and coverage models.

The first *median problem* was created by Fermat in the 17th Century: Given a triangle, find the median point in the plane such that the sum of the distance from each point of the points to the median point is minimized. Weber (1909) extended the problem with more than three points with weights and objective minimizing total weighted distance. Both Weiszfeld (1937) and Hakimi (1964) proposed methods to solve the *Fermat-Weber problem* optimally, the former gives the optimal location in the Euclidean space whereas the later for networks. Cooper (1963, 1964, 1972) modeled the existing Fermat-Weber problem with more than one facilities and proposed efficient heuristic methods. Calvo and Marks (1973) grouped the population into groups and introduced facilities of these groups. Weaver and Church (1985) proposed the *vector assignment p-median problem* which aims to minimize total transportation cost while forcing the demand nodes to have service from k closest facilities with predefined ratios.

Recently, p-median models dealing with facility disruptions have started to gain attention (Qi et al., 2010). Cui et al. (2010) proposed a continuum approximation model with custom designed Lagrangean relaxation algorithm. Li and Ouyang (2010) deals with the case of correlated demand with a similar model. Wang and Ouyang (2013) considers the effect of facility disruption risk under competition. A leader-follower Stackelberg competition model is developed to find optimal facility location design. An et al. (2014) propsed a two-stage robust optimization model for the same problem. They applied two different approaches and observed that column-and-constraint generation method outperformed Bender decomposition drastically.

Coverage models are used to locate facilities (i.e. emergency response systems) in such a way to maximize coverage and/or minimize number of facilities. The first two models, the *location set covering problem* (LSCP) aims to minimize number of facilities to cover all demand (Toregas et al., 1971) and the *maximal location set covering problem* (MCLP) aims to maximize total coverage with limited number of facilities (Church and ReVelle, 1974). Schilling et al. (1979) proposed a model that aims to have multiple coverages with different types of facilities. Daskin and Stern (1981) promoted the multiple coverage as a secondary objective. Aly and White (1978), Hogan and Revelle (1986), ReVelle and Hogan (1989), Marianov and ReVelle (1992), Ball and Lin (1993), Gendreau et al. (1997) have extended the works given above in different aspects.

In addition to the two main types of models given above, there are also dynamic models proposed in the literature. The main idea in these models is to relocate the facilities (e.g. ambulances, fire brigades) when one or more of the facilities are dispatched for an incident. Kolesar and Walker (1974) proposed a model for the fire brigades. Recently, with the increase in computational power, relocation model applications for ambulances have also emerged. Gendreau et al. (2001) proposed a parallel tabu search heuristic to solve relocation model efficiently. Gendreau et al. (2006) and Schmid and Doerner (2010) are the two recent models on dynamic facility location problems for emergency response systems where the latter is the multistage approach of the former one. Andersson and Värbrand (2007) proposed a decision support tool for a similar aim. Rajagopalan et al. (2008) extended the queueing set covering problem (Marianov and ReVelle, 1996) with the help of Jarvis (1985)'s generalized approximation for loss systems. Schmid (2012) solved relocation and dispatching problem with approximate dynamic programming. Alanis et al. (2013) proposed a model which utilizes a compliance table. A Markov chain is modeled with two state variables: the number of busy servers and a Boolean variable showing if the system is in compliance or not.

Larson (1974) proposed a *hypercube queueing model* (HQM) which is the first model that embeds the queueing theory in facility location-allocation literature. This model analyzes systems such as emergency services, door-to-door pickup and delivery services, neighborhood service centers and transportation services which has response district design and service-to-customer mode (Larson and Odoni, 1981). The solution of the model provides state probabilities and other related system performance measures (e.g. workload, average service rate, loss rate) for any given server locations. Nevertheless, it is a descriptive model that only allows to analyze scenarios (Galvão and Morabito, 2008). HQM models the current configuration as a continuous-time Markov chain and does not determine the optimal configuration.

The first model proposed by Larson (1974) assumed that the service time is not a function of the locations of the calls for service and the dispatched unit. This argument was supported

with the claim that the time spent on the way to scene is negligible compared to the service time on scene. Since, even with this simplification, as the number of servers (n) is increased, number of states grows exponentially; Larson (1975) proposed a heuristic method to deal with the above problem. Jarvis (1985) altered this heuristic for the systems with both server and customer dependent service times. Larson and McKnew (1982) proposed a 3^n model for patrol cars with three states: (1) busy with a call for service, (2) busy with a patrol activity and (3) free on a patrol. They implement exact method for small instances with not more than 5 servers. Larson and McKnew (1982) also developed an approximate approach similar to Larson (1975) for instances with more servers. Atkinson et al. (2008) and Iannoni et al. (2009) proposed partial 3^n models in which each region takes service from two servers with different service rates. Budge et al. (2009) proposed an approximation algorithm which works on systems with multiple vehicles at stations that computes station (i.e. set of servers at the same location) specific steady state probabilities.

Takeda et al. (2007) showed the benefits of decentralization of emergency response systems with the help of hypercube models in Brazil. Iannoni and Morabito (2007), Iannoni et al. (2008) embedded hypercube in a genetic algorithm framework to locate emergency vehicles along a highway for small instances (only 5 servers). They extended the problem to enable multiple dispatch (i.e. more than one server may be needed for an incident). Iannoni et al. (2009) developed a hybrid genetic algorithm approach which adds facility locations to districting decisions with a local search on a genetic algorithm. Iannoni et al. (2011) replaced the exact method with a hypercube approximation algorithm which enables them to solve instances with 100 servers. Geroliminis et al. (2009) extended the hypercube model and developed the spatial queueing model (SQM) to optimize locations emergency response vehicles for systems with up to 10 servers. This model explicitly considers that (i) service rates are not identical and vary between servers (non-homogeneous servers of this type are also analyzed in Morabito et al., 2008) and (ii) for a given server the service rates depend on the incidents characteristics (interdistrict or intradistrict response). Thus, this model links districting and dispatching to the location problem and proposes a hybrid formulation for coverage and mean response time to locate servers and simultaneously identify their areas of responsibilities (rather than analyzing performance measures for a given system configuration). Despite its theoretical elegance and flexibility, the model is computationally difficult to solve for large instances. Later, Geroliminis et al. (2011) extended this work to deal with larger instances of the problem. They propose an approximate solution to the symmetric hypercube model with spatially homogeneous demand. In the first step, the service area is partitioned into sub-areas (called *superdistricts*) while, in parallel, necessary number of units is determined for each superdistrict. A genetic algorithm is combined with the approximate hypercube model for obtaining the number of servers that should be located at each superdistrict (similar to a *bin*) of the system. While this approach can deal with larger instances, interactions (interdistrict responses between regions) are not considered. As we will show later in the paper, these interactions are significant in accurately estimating the performance of the system and the AHQM formulation integrates these aspects.

As it is stated before, there is an extensive literature on location and coverage literature. The more interested readers should see Hale and Moberg (2003), Owen and Daskin (1998), Brotcorne et al. (2003) and Laporte et al. (2009) for broader survey for the related topic. The former two are surveys generally on location and coverage literature whereas the latter two focus more on the ambulance location-relocation models.

3. Hypercube Queueing Models

This section provides a description of the existing 2^n hypercube model of Larson (1974) and formulates two new models to deal more accurately with interactions of servers, (i) a general 3^n HQM, where each server has three possible states: available, busy inside primary service area (intradistrict) and busy outside primary service area (interdistrict), and (ii) a new 3^n aggregate hypercube queueing model (AHQM) that treats group of servers (bins) in a similar manner by considering interactions among bins.

3.1. A Note on 2^n Hypercube Queueing Model

The HQM proposed by Larson (1974) includes *hypercube* in the name since the transition graph of the continuous time Markov chain representing this queueing structure has a hypercube structure when the number of servers is more than three. The state variables contain n binary variables for n servers showing if server i is available (0) or busy (1). Each state is a number in base 2 and each digit shows the state of the corresponding server. For each region, which is called *atom* (j) in HQM literature, there exists a priority list of servers. Requests at each atom are dispatched to the available server with the highest priority for this atom. If there does not exist any available server that can serve the atom, either the call is lost (i.e. call for ambulance may be dispatched by a backup system) or joins a queue to be served (i.e. customers are asked to wait until there is an available server) depending on the assumptions. Service requests arrive from each atom according to an independent Poisson process with rate λ_j and servers have exponential service rates of μ_i for any atom served. The transition graph of 2^n HQM with three servers can be seen in Fig. 1. It is seen on that simple graph that as the system gets congested, the burden on the free server(s) increases. For instance in state "011" all the servers but the third are busy. That is why the next incident in any region will be served by the third server and transition rate from "011" to "111" is $\lambda_1 + \lambda_2 + \lambda_3$. Such a model does not consider different service rates for inter and intradistrict responses.

3.2. 3^n Hypercube Queueing Model

In this research, we use a 3^n HQM described in the previous section. If different rates for intra and interdistrict responses are applied, each server has three possible states: available (0), busy with intradistrict (1) and busy with interdistrict (2). Fig.

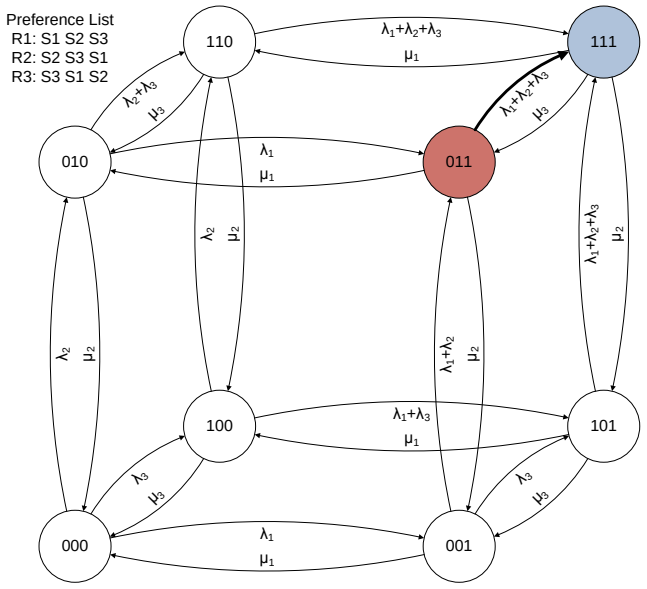


Figure 1: Larson (1974)'s 2^n HQM for three servers with equal intra and interdistrict service rates (μ_i). State "011", "111" and the transition connecting them is shown with different colors.

2 is a transition graph of a system of the same type with three servers. Given the large number of transitions and states the illustration for a system with more servers would be difficult to visualize. For instance "210" represents the state in which first server is available, the second intervenes an incident inside its own region and the third intervenes an incident outside its own region (state reads from right to left). Since there are three possibilities for each server, the number of states also increases to 3^n.

Note that, the server has always priority for the requests inside its own intradistrict area. When the system is empty, the first assignment should be an intradistrict assignment. However, this does not prevent having states such as "222". Although, practically rare for lightly congested systems, it has a non-negative probability in all systems.

The general transition equation for the states of 3^n HQM can be seen in Eq. 2. In this equation, q and r are states which can be regarded as numbers in base 3 (e.g. 120 in which first server (0) is available, second server is busy with interdistrict response and third (1) server is busy with intradistrict response), $\mathbb{P}_q$ is the steady-state probability of state q. i and j represent servers and atoms respectively, $T(q,i)$ is the condition of server i in state q (i.e. i^{th} digit of q, e.g. $T(120,1)=0, T(120,2)=2, T(120,3)=1$), R_i is the set of atoms in the intradistrict area of server i, $S(q,i)$ is the set of atoms that have interdistrict response from server i if there is an incident during state q which is generated by priority lists. $\mathbb{1}(*)$ is an indicator function. It is equal to 1, if $*$ is true and 0 otherwise.

$D(q,r,i)$ is another function defined as:

$$D(q,r,i) = \begin{cases} c, & \text{if } d(q,r)=1, T(r,i)=0, T(q,i)=c \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where $d(q,r)$ is the Hamming distance between states q and r (i.e. minimum number of transitions to reach from q to r). $D(q,r,i)$ simply shows state pairs with the same server conditions except server i (e.g. $D(120,100,2)=2$). If server i is available in state r and busy in state q, condition of the server in state q is the output of the function. Memoryless arrivals and service rates are simplifying the size of transitions as only states with Hamming distance equal to 1 are connected. Such an approximation is reasonable for different types of queueing systems. Real data can further investigate this assumption.

$$\mathbb{P}_r \left[\overbrace{\mathbb{1}\left(\exists i : T(r,i)=0\right) \sum_j \lambda_j}^{\text{arrival}} + \overbrace{\sum_{i:T(r,i)=1} \mu_i + \sum_{i:T(r,i)=2} \mu'_i}^{\text{departure}} \right]$$
$$= \overbrace{\sum_{q,i:D(q,r,i)=1} \mathbb{P}_q \mu_i + \sum_{q,i:D(q,r,i)=2} \mathbb{P}_q \mu'_i}^{\text{upper transitions}} \quad (2)$$
$$+ \underbrace{\sum_{q,i:D(r,q,i)=1} \mathbb{P}_q \sum_{j \in R_i} \lambda_j + \sum_{q,i:D(r,q,i)=2} \mathbb{P}_q \sum_{j \in S(r,i)} \lambda_j}_{\text{lower transitions}}$$

In building Eq. 2, LHS is equal to the rate of leaving state r whereas RHS is equal to the rate of entering state r. To keep the formulation simpler, we assume that, every atom is reachable by any server. In addition, we are formulating a system without queue, i.e. request from any atom is lost if the system is full.

On the LHS of Eq. 2, the total rate leaving state r is multiplied with the steady state probability of r (i.e. $\mathbb{P}_r$). State r can be left either by an arrival (if all servers are not under service and there exists an available server) or a departure (from a server giving either intra or interdistrict service). First term on the LHS stands for arrivals to state r. The term $\mathbb{1}\left(\exists i : T(r,i)=0\right)$ exists in front of the term $\sum_j \lambda_j$ to show that arrival is possible only if an available server exists. Since every atom is reachable by any server, every arrival changes the system state, if there is an available server. This is the reason that there is no condition on j in the first summation. Summation on the first term is equal to the total arrival rate to the system.

The second and third summations multiplied with $\mathbb{P}_r$ in the LHS of Eq. 2 are for departures from state r. The transition rate differs if a server is in intradistrict (first term) ($T(r,i)=1$) or interdistrict (second term) ($T(r,i)=2$) response.

The RHS of Eq. 2 is composed of transitions to state r from the states that are one Hamming distance away. The first two summation terms are from upper transitions (i.e. transitions to a state with one less busy server). They show transitions from an upper state q after an intradistrict (first term) or interdistrict (second term) service. The former is multiplied with μ_i and the latter is with μ'_i.

The last two summations on the RHS of Eq. 2 are from lower transitions (i.e. transitions to a state with one more busy server) to state r. If the condition of server i in state r is intradistrict service ($D(r,q,i)=1$), then there exists a state q that forms a transition to state r. Note that in state q and r, all servers have the same condition except state i. Server i is available in state q and busy with intradistrict in state r. The rate of the transition is the total arrival rate for the atoms in the intradistrict

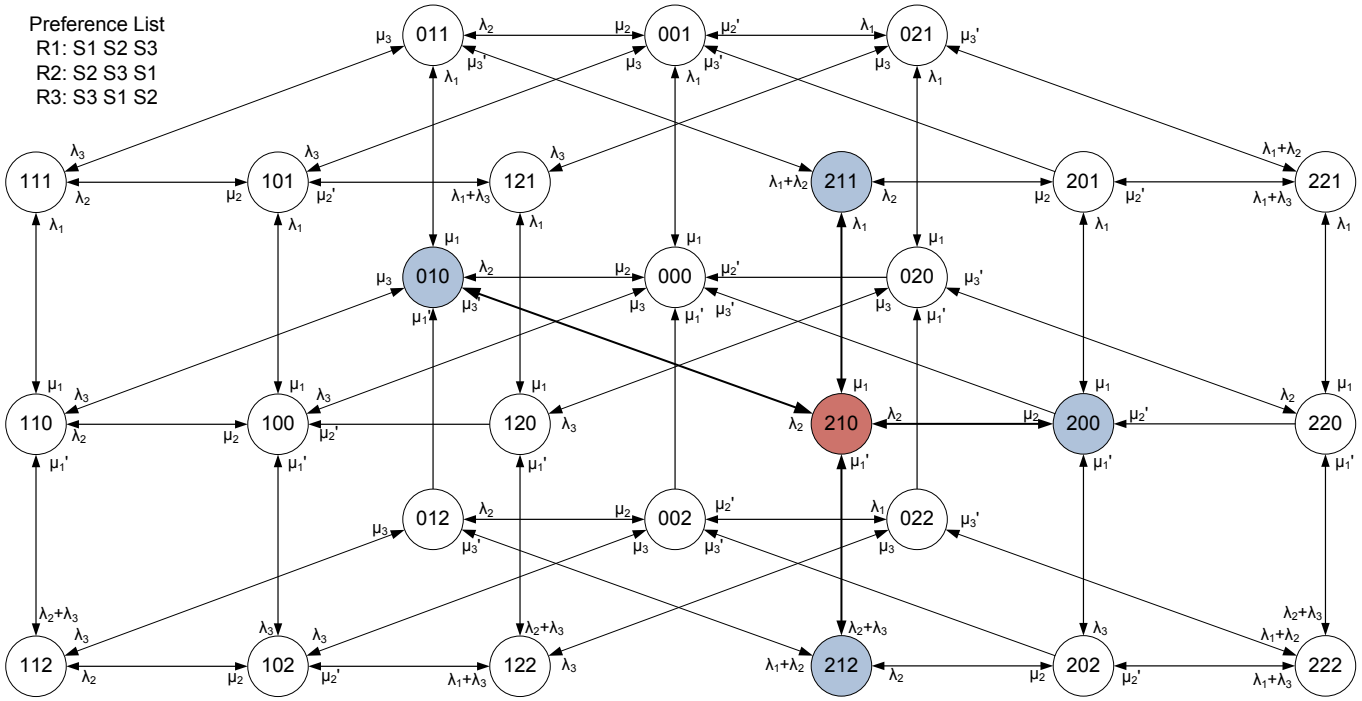


Figure 2: 3^n HQM model for two servers with different intra (μ_i) and interdistrict (μ'_i) service rates. State "210", states directly connected to it and transitions are colored differently.

area of server i. This is formulated with the third summation on the RHS of Eq. 2. Similarly, if the condition of server i in state r is interdistrict service, then we can state there is a transition between state q and r in which the only difference is the condition of server i (i.e. available in state q and busy with interdistrict in state r). The fourth summation deals with the cases in which server i is the available server with the highest priority for atom j in state q and atom j is not in the intradistrict area of server i.

To give an illustrative example, the transition equation for red painted state "210" in Fig. 2 can be written as:

$$\begin{aligned}\mathbb{P}_{210}\left(\lambda_1+\lambda_2+\lambda_3+\mu'_3+\mu_2\right)\\ =\mu_1\mathbb{P}_{211}+\mu'_1\mathbb{P}_{212}+\lambda_2\mathbb{P}_{200}+\lambda_2\mathbb{P}_{010}\end{aligned} \tag{3}$$

Appendix A provides a more detailed description of 3^n HQM through a numerical example with each step of the approach.

3.3. 3^n Aggregate Hypercube Queueing Model

Although 3^n HQM is more accurate than 2^n, the increase in the number of states is more and not applicable for real life cases. For instance a system with 20 servers needs more than three billion states in 3^n HQM. In order to cope with that, we develop a 3^n aggregate HQM (AHQM). In this new model, a new concept called *bin* is used to represent servers. It is assumed that, each bin (b) has a capacity as it consists of a group of servers and each state consists of 2 values for each bin, which show the number of busy servers with intra and interdistrict responses at each bin. For intra and interdistrict service rates μ_b and μ'_b for bin b and demand rate λ_j for atom j, transition diagram for two bins with two servers in each bin can be depicted as seen in Fig. 3. Note that, each row of the state name shows condition of each bin. Given that the number of servers per bin (capacity) is known, the state of each bin is described with only two values. The values on the left and right are number of servers occupied by intra and interdistrict responses respectively.

The general transition equation for 3^n AHQM can be seen in Eq. 5. In this equation, in addition to the definitions used in Eq. 2, b is an index for bins. $\tilde{T}(r,b,*)$ shows number of servers in bin b in the condition of $*$ (i.e. "inter", "intra", "free") in state r and $\tilde{D}(q,r,b,*)$ can be defined as:

$$\tilde{D}(q,r,b,*)=\begin{cases}1, & \text{if } d(q,r)=1, \tilde{T}(q,b,*)=\tilde{T}(r,b,*)+1\\ 0, & \text{otherwise.}\end{cases} \tag{4}$$

where $*$ stand for conditions "inter" and "intra".

Eq. 5 has similar assumptions and formulations as Eq. 2. The sum between square brackets is the rate of leaving state r. The first summation is for arrivals to state r. Since it is assumed that every atom is reachable by any server, every arrival will change the state if there is an available server in state r.

The second and third summation on the LHS are for intradistrict and interdistrict dispatches (i.e. departures) from state r to other states respectively. The rate of leaving state r to another state by departure is directly proportional to the sum of the products of intra and interdistrict service rates and their counts under dispatching respectively. Since each bin may have more than one servers, in Eq. 5, we need to write equations with the number of servers in intra and interdistrict responses. This is one of the main differences of Eq. 5 from Eq. 2.

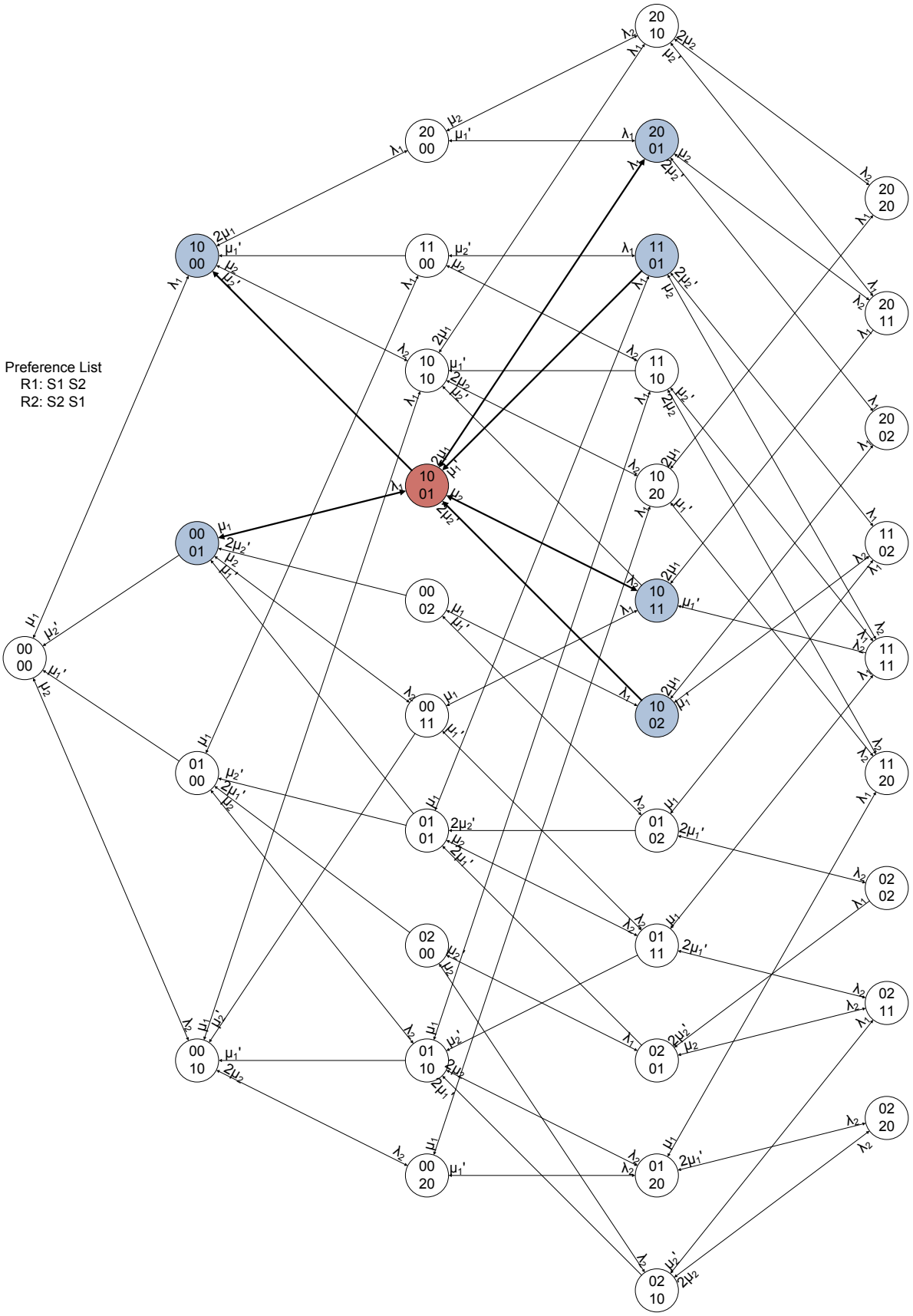


Figure 3: 3^n AHQM for two bins containing two servers in each bin with different intra (μ_b) and interdistrict (μ'_b) service rates, and primary demand areas (λ_j). State "10|01" and states connected to it are filled with different colors to show an example of transition equations.

$$\mathbb{P}_r \Bigg[\overbrace{\mathbb{1}\left(\exists n : \tilde{T}(r,b,\text{free}) \neq 0\right) \sum_j \lambda_j}^{\text{arrival}} + \overbrace{\sum_b \tilde{T}(r,b,\text{intra})\mu_b + \sum_b \tilde{T}(r,b,\text{inter})\mu'_b}^{\text{departure}} \Bigg]$$

$$= \overbrace{\sum_{\substack{q,b:\\ \tilde{D}(q,r,b,\text{intra})=1}} \mathbb{P}_q \tilde{T}(q,b,\text{intra})\mu_b + \sum_{\substack{q,b:\\ \tilde{D}(q,r,b,\text{inter})=1}} \mathbb{P}_q \tilde{T}(q,b,\text{inter})\mu'_b}^{\text{upper transitions}} + \underbrace{\sum_{\substack{q,b:\\ \tilde{D}(r,q,b,\text{intra})=1}} \mathbb{P}_b \sum_{j\in R_b} \lambda_j + \sum_{\substack{q,b:\\ \tilde{D}(r,q,b,\text{inter})=1}} \mathbb{P}_b \sum_{j\in S(r,b)} \lambda_j}_{\text{lower transitions}} \tag{5}$$

The RHS of Eq. 5 is equal to the sum of the products of the probability of the states one hamming distance away from state r and their entering rates to the same state. The first two summations are from upper transitions and the last two are from lower transitions to state r. Each of these summations are formed in a similar fashion that they are formed in Eq. 2. The only difference is that they are multiplied with the number of servers in intra and interdistrict responses in the same bin. Also note that the difference in the definition of intradistrict responses for a server (3^n HQM) and a bin (3^n AHQM). Intradistrict response of a server is an intervention outside its area of responsibility, while the intradistrict response for a bin is considered only when a server intervenes in an atom outside the whole bin.

We can see how transition equations are generated on a specific state. In Fig. 3, in state "10|01", there is a busy server in bin 1 with intradistrict response which is shown with the first line of the state name. In bin 2 there is also a busy server which is in interdistrict response. The transition equation for this state can be written as Eq. 6. Note that, in Fig. 3, state "10|01" and its neighbor states are colored with red and blue respectively:

$$\begin{aligned}\mathbb{P}_{\substack{10\\01}}(\lambda_1+\lambda_2+\mu_1+\mu'_2) \\ = 2\mu_1\mathbb{P}_{\substack{20\\01}} + \mu_2\mathbb{P}_{\substack{10\\11}} + \mu'_1\mathbb{P}_{\substack{11\\01}} + 2\mu'_2\mathbb{P}_{\substack{10\\02}} + \lambda_1\mathbb{P}_{\substack{00\\01}}\end{aligned} \tag{6}$$

As for the case of 3^n HQM, a visualization of all states in a figure for a large number of servers is difficult. To further clarify with a more complex example, in Fig. 4, some specific states of a 3^n AHQM with two bins of 6 servers each can be seen. The complete transition diagram of the system in Fig. 4 has 784 states. We just depict two states "32|41" (green), "42|41" (red) and their neighbor states (blue) to show how transition equations are calculated. For states "32|41" and "42|41", transition equations can be written as follows:

$$\begin{aligned}\mathbb{P}_{\substack{32\\41}}(\lambda_1+\lambda_2+3\mu_1+4\mu_2+2\mu'_1+\mu'_2) \\ = 4\mu_1\mathbb{P}_{\substack{42\\41}} + 5\mu_2\mathbb{P}_{\substack{32\\51}} + 3\mu'_1\mathbb{P}_{\substack{33\\41}} + 2\mu'_2\mathbb{P}_{\substack{32\\42}} + \lambda_1\mathbb{P}_{\substack{22\\41}} + \lambda_2\mathbb{P}_{\substack{32\\31}}\end{aligned} \tag{7}$$

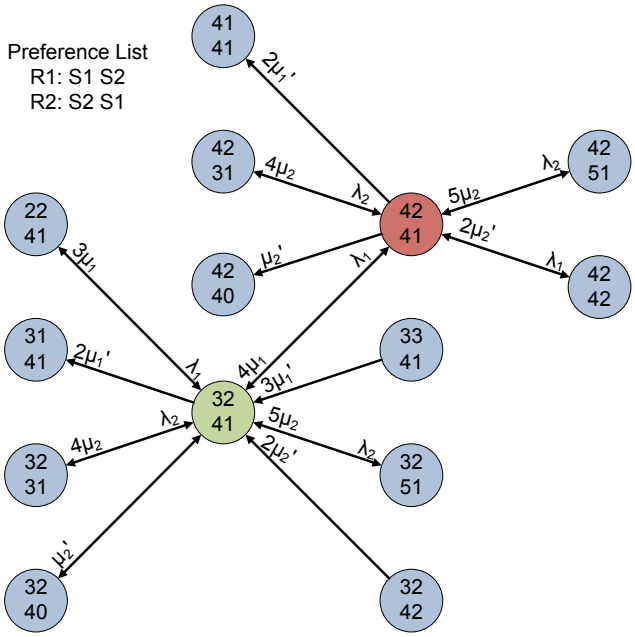


Figure 4: Part of the transition diagram of a 3^n AHQM for two bins containing six servers in each bin with different intra (μ_b) and interdistrict (μ'_b) service rates, and primary demand areas (λ_j). States "32|41" (green), "42|41" (red) and their neighbor states (blue) are depicted to help to visualize transition equations of the former two states.

$$\begin{aligned}\mathbb{P}_{\substack{42\\41}}(\lambda_1+\lambda_2+4\mu_1+4\mu_2+2\mu'_1+\mu'_2) \\ = 5\mu_2\mathbb{P}_{\substack{42\\51}} + 2\mu'_2\mathbb{P}_{\substack{42\\42}} + \lambda_1\mathbb{P}_{\substack{32\\41}} + \lambda_2\mathbb{P}_{\substack{42\\31}}\end{aligned} \tag{8}$$

For a 3^n AHQM, if C_b is the maximum number of servers in bin b, total number of states equals $\prod_b \frac{(C_b+2)(C_b+1)}{2}$. For this estimation, each bin can be regarded independently in this calculation. The number of servers at each bin is separated into three groups: available, busy with intradistrict, busy with interdistrict. Since the number of states each bin is independent, the total number of states of each bin can be multiplied with each other.

As described above, 3^n AHQM is a solution for larger spatial systems. It applies different service rates for intra and interdistrict responses. We can define 3^n HQM as a special case of 3^n AHQM with one server at each bin only. 3^n AHQM with two bins has less number of states than any 3^n and almost all 2^n HQM (i.e. for the cases with at least 8 servers). For instance, the system with 16 servers has 65536 states in 2^n and over 43 million states in 3^n HQM whereas a 3^n AHQM of two bins with 8 servers each has only 2025 states. Appendix B provides a more detailed description of 3^n AHQM through a numerical example with each step of the approach. In the next section, we describe the MHQA that utilizes the two new 3^n models, i.e. 3^n HQM and 3^n AHQM, defined in the current section.

4. Mix Aggregate Hypercube Queueing Algorithm

The exponential increase in the number of states makes 3^n HQM not applicable to real life instances. For this purpose, we

propose 3^n AHQM which has less states. However, the way 3^n AHQM is applied, is also important for the efficiency of the method and accuracy of the results. In this section, the details of this procedure will be described. Simply, the method we propose contains an iterative approach that partitions the whole problem area into *subregions* which is followed by an iterative solve for each partition and merge scheme for the pairs of partitions. In our approach during the different steps of partitioning and merging, we consider interactions between groups of servers, i.e. "bins", that are important especially in systems with many busy servers. In the following two subsections firstly the iterative solution procedure and then the partitioning algorithm are described. An illustration of the procedure is provided in Fig. 5. Fig. 5a shows the whole region with the location of all servers (red dots). Dark color represents atoms of high demand and lighter color the ones of lower demand. Fig. 5b shows the primary areas of responsibilities of each server estimated with a Voronoi (Aurenhammer, 1991, Okabe et al., 2000) approach based on Euclidean distance. Figs. 5b-d shows the results of the sequential partitioning method which is described later in more details.

After sequential partitioning we end up with a number of core subregions, see for example the outcome of Fig. 5e. These subregions are modeled as 3^n HQMs. With 3^n HQMs, any needed performance measures can be calculated. Service rate for the number of servers, availability of each server, loss rate of each atom and percentage of time each server spends at intra or interdistrict responses are found for the algorithm. Then, in the sequence of merging, an inverse process of partitioning is followed to estimate performance measures for the whole area of study (moving from Fig. 5e to Fig. 5a now).

The core subregions are merged to larger *compounds* subregions, which are modeled as 3^n AHQM. Compound subregions are also merged to larger compound subregions with 3^n AHQM until the whole area is covered. Note that at each merging step, only two subregions (core or compound) are merged to a larger compound region. For intradistrict service rate of each bin, the service rate calculated from the previous step is used. Loss rate of each atom in sibling subregion (i.e. subregions which are the pairs of each other in sequential partitioning) and availability of each servers are used to calculate interdistrict service rates. We assume that servers are busy most of the time within their core region responses. The availability of a server can be assumed to be proportional to one minus the occupancy in its core region. Since, in the next steps, all servers are merged and regarded as servers inside bins, we use the availability of each server in a bin to calculate the probability if an interdistrict call can be served or not. The formulation we have conducted for this purpose can be seen in the next subsection.

4.1. Bin Interactions

If a 3^n HQM is applied for each partition of Fig. 5d without considering any interactions, the system performance measures would be consistently less accurate. In our experiments, we see that the interaction between bins is 5-50% (see Fig. 11). However, this relationship also needs a correction, we cannot assume every atom can be served by any bin. Since, bins are

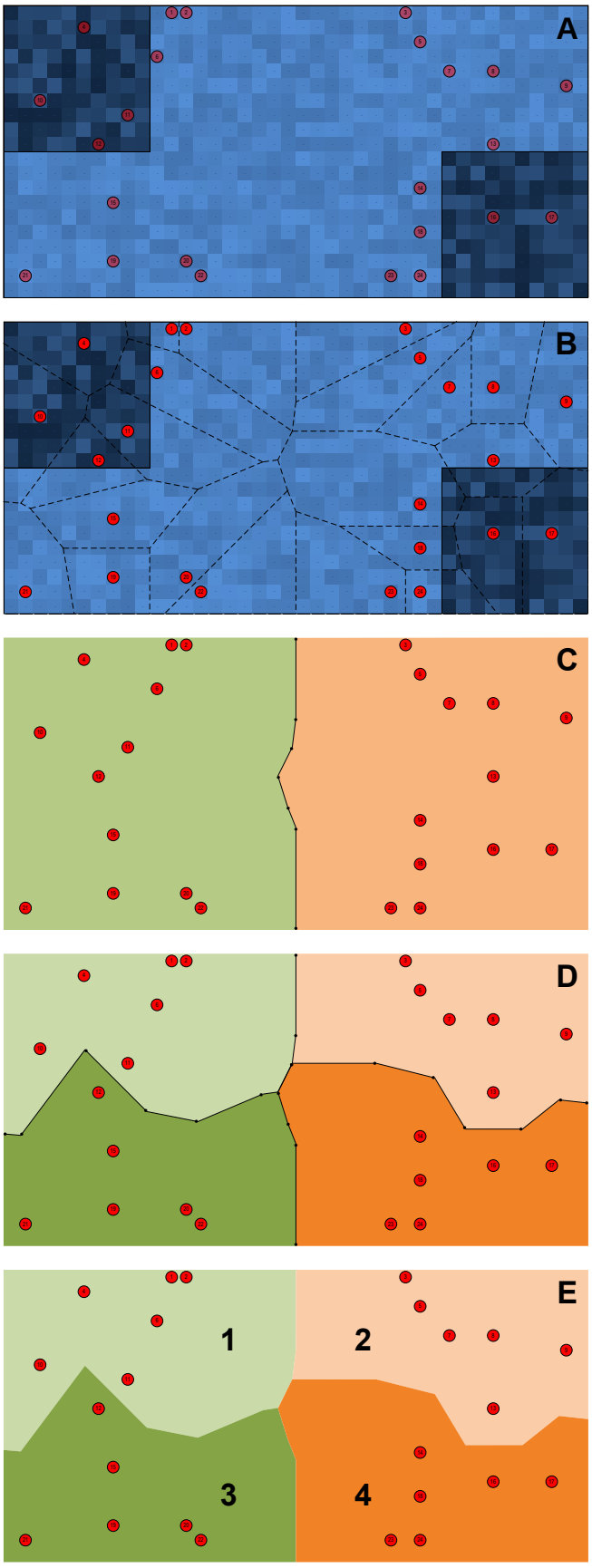


Figure 5: An illustration of the partitioning approach.

not physically at the same location, it is not possible to assign a single location for them. There is no straightforward way to calculate distance between an atom and a bin. However, we need the distance between the bin and the atom to decide if the atom can be served by this bin or not. In order to cope with that, we assume, with some probability, some atoms might lack service even though there is at least some servers available in a bin of the neighbor subregion.

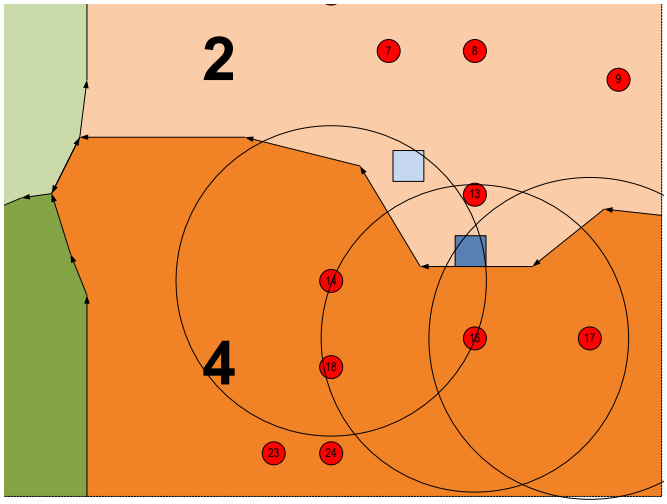


Figure 6: An illustration for bin interactions. Dark and light blue squares represent demand requests and red circles show the server locations in two subregions 2 and 4. Larger circles centering servers show their accessibility area. If there is no available server in region 2, for an incident happening in light and dark blue atoms, we need to use servers in subregion 4.

It is easy to illustrate how we apply methodology on a toy example before a formal mathematical notation is introduced, shown in Fig. 6. Assume that, all servers in region 2 are busy. Consider now 3 servers from region 4 (servers 14, 16 and 17) and the areas within their accessibility distance, i.e. the maximum amount of time (equivalently distance if speed is constant) that a server can travel to serve an atom, (shown with the 3 circles). Let us assume that a new request for service arrives from an atom located in the dark or light blue squares, close to the boundary of the two regions. In such a state, if an incident occurs in light and dark blue atoms, these incidents can either be served with servers in region 4 or they are lost. There is only one server (i.e. server 14) that can serve the light blue atom and three servers (servers 14, 16 and 17) that can serve the dark blue atom from region 4. In the AHQM, servers from the same bin cannot be differentiated after merging. We can only calculate number of servers available or busy with either intra or interdistrict responses at each state. So, in order to approximate the probability of being served, we use the formulation given in Eq. 9. For instance, if all 6 servers available in region 4, we can assume that both light and dark blue atoms can be served by the servers in region 4, as there is always a server available that can reach both atoms. If there are only 4 servers available in region 4, we can still assume dark blue atom is served with probability 1 because in the worst case, one of the three servers that can reach dark blue atom should be available. Nevertheless, if there are not more than 3 servers available, then with some probability both light and dark blue regions cannot be served. But, we can state that, dark blue region can be served with a higher probability because light blue atom can be reached by only server 14, whereas dark blue atom is reachable by servers 16 and 17 in addition to 14.

Let us assume there are n servers in a bin, first m of them can reach to the atom of the sibling subregion and k servers are busy. When $m = 0$, the probability of serving the atom by a server from this bin equals 0. When $k < m$ the request of this atom is served with probability 1, since even in the worst case, there has to be a server available to serve it. However if $k \geq m$, the probability that none of the available servers can reach the atom is approximated as:

$$\mathbb{P}(\text{not served}) = \frac{\prod_{i=1}^{m} \mathbb{P}_i \left[\sum_{\substack{\forall L \in \mathcal{P}(\mathbb{N}_{m+1}^{n}): \\ |L|=k-m}} \left(\prod_{\forall i \in L} \mathbb{P}_i \right) \right]}{\sum_{\substack{\forall L \in \mathcal{P}(\mathbb{N}_{1}^{n}): \\ |L|=k}} \left(\prod_{\forall i \in L} \mathbb{P}_i \right)} \tag{9}$$

where $\mathcal{P}(*)$ represents the power set of $*$ (i.e. set of all subsets of $*$ including the empty set and $*$ itself), $\mathbb{N}_a^b$ is an inclusive sequence of integers starting from a to b and $\mathbb{P}_i$ is the probability that server i is busy. In Eq. 9, the denominator is equal to the sum of the all cases' probabilities with k busy servers. The numerator is equal to the sum of the probability of cases where all servers that can reach the atom are busy. As a result Eq. 9 gives conditional probability that, there does not exist any available server which can reach the atom given k busy servers. Note that, the same combination term ($\mathbb{C}_k^m$) canceled out each other both in the numerator and the denominator.

Eq. 9 is an approximate value for the probability of an incident happening in the selected atom does not get an interdistrict service even though there is an available server in the bin. In order to calculate loss rate caused by this phenomena, the demand that is not served in the previous steps of the algorithm is multiplied with the probability in Eq. 9 and assumed to be lost. Total demand of the atom minus calculated loss rate is assigned as the demand of the atom. This is done for each atom and the new 3^n AHQM is modeled with these demand values. After solving the model, loss rate calculated by 3^n AHQM and assumed loss rates are summed up and assigned as total loss rates of each atom.

An informal pseudocode for our algorithm is given in Fig. 7 where $\bar{T}_l^*(n)$ and $\bar{R}_l^*(n)$ stands for average service time and rate for bin composed of servers of subregion l for the intra and interdistrict responses. $T(i, j)$ is the total service time needed for a response to atom j by server i. J_l are the atoms in subregion l and, intra$_i$ and inter$_i$ are atoms in intra- (i.e. atoms closest to server i) and interdistrict (i.e. atoms serviceable but not closest to server i) area of server i. If a 3^n HQM was solved for each partition without considering any interactions, the system performance measures would be consistently less accurate for semi-congested systems (medium to high demand).

1. Partition the whole problem area into regions. A binary tree structure is created with partitioning. Leaves of the binary tree are core subregions, and the rest of the nodes are compound subregions composed of two (core or compound) subregions.

2. Iterate each node with depth first search /*if a subregion is composed of two smaller subregions, smaller subregions performance measures are needed to have performance measures for the bigger subregion*/

   - If selected subregion l is core subregion /*core subregions are not composed of smaller subregions*/
     (a) Calculate average intra and interdistrict service time $\left(\bar{T}_i^{\text{intra}}, \bar{T}_i^{\text{intra}}\right)$, and calculate average service rates $\left(\bar{R}_i^{\text{intra}}, \bar{R}_i^{\text{intra}}\right)$ for each server i in the selected core subregion:
     $\bar{T}_i^{\text{intra}} \leftarrow \frac{\sum_{j \in \text{intra}(i)} d_i T(i,j)}{\sum_{j \in J_l \cap \text{intra}(i)} d_i}, \bar{T}_i^{\text{inter}} \leftarrow \frac{\sum_{j \in J_l \cap \text{inter}(i)} d_i T(i,j)}{\sum_{j \in \text{inter}(i)} d_i}, \bar{R}_i^{\text{intra}} \leftarrow \frac{1}{\bar{T}_i^{\text{intra}}}, \bar{R}_i^{\text{inter}} \leftarrow \frac{1}{\bar{T}_i^{\text{inter}}}$
     (b) Solve 3^n HQM with $\bar{R}_i^{\text{intra}}$, $\bar{R}_i^{\text{inter}}$ and d_j where the former two stand for inter and intradistrict response rates for each server i and latter for demand rate of each atom j in the selected core subregion l.
     (c) Calculate probability of server i busy (P_i), loss rate of atom j (loss_j), and average service rate for the number of servers busy $\bar{S}_l^{\text{intra}}(n)$ where $n \in \mathbb{N}$.
   - If selected subregion l is a compound subregion with children regions l_1 and l_2 /*Assume bins b_1 and b_2 are composed of servers of subregions l_1 and l_2 respectively.*/
     (a) Calculate average interdistrict service times ($\bar{S}_l^{\text{inter}}$) of bins in child subregions of l: /*use loss rates not demand*/
     $\bar{S}_{l_1}^{\text{inter}} \leftarrow \frac{\sum_{i \in J_{l_1}} \sum_{j \in J_{l_2} \cap \text{inter}(i)} \text{loss}_j T(i,j)}{\sum_{i \in J_{l_1}} \sum_{j \in J_{l_2} \cap \text{inter}(i)} \text{loss}_j}, \bar{S}_{l_2}^{\text{inter}} \leftarrow \frac{\sum_{i \in J_{l_2}} \sum_{j \in J_{l_1} \cap \text{inter}(i)} \text{loss}_j T(i,j)}{\sum_{i \in J_{l_2}} \sum_{j \in J_{l_1} \cap \text{inter}(i)} \text{loss}_j}$
     (b) Calculate average interdistrict service rates of each child subregion l: /*service rate = 1 / service time*/
     $\bar{R}_l^{\text{inter}}(n) \leftarrow \frac{1}{\bar{S}_l^{\text{inter}}} \quad \forall l = \{l_1, l_2\}$ and $n \in \mathbb{N}$.
     (c) Generate a 3^n AHQM with given intra and interdistrict service rates of each subregion (or equivalently bin) $\bar{R}_l^{\text{intra}}(n)$ and $\bar{R}_l^{\text{inter}}(n)$ respectively for $l = \{l_1, l_2\}$.
     (d) For each state, find the loss rate because of server unavailability $\left(\text{loss}'_j\right)$ by multiplying loss_j and Eq. 9. Update demand of each atom with $\left(d_i - \text{loss}'_j\right)$ for corresponding states.
     (e) Solve generated 3^n AHQM.
     (f) Calculate loss rate of each atom j with the equation: $\text{loss}_j \leftarrow \text{loss}_j^{\text{AHQM}} + \text{loss}'_j$.
     (g) Calculate average service rate for the number of servers busy $\bar{S}_l^{\text{intra}}(n)$ for $n \in \mathbb{N}$

3. Return the values calculated for the subregion in the root node (which is equivalent to the whole problem area).

Figure 7: Pseudocode for mix aggregate hypercube queueing algorithm

4.2. Partitioning Model

As stated before, the size of 3^n HQM grows exponentially with the number of servers and is applicable only for problems with limited number of servers. In order to cope with that, we develop an aggregate approach that devices both 3^n HQM and 3^n AHQM. In order to have accurate results in efficient time we need a partitioning algorithm that partitions the whole problem area to subregions with an objective that minimizes the total demand that is served by more than one bins. Even if in Section 4.1 we develop a framework to deal with interactions, given that AHQM does not keep track of individual servers, this estimation contains some level of error. For fixed server locations, by minimizing interaction between subregions' (i.e. minimizing services provided by servers of other subregions), we decrease the error of the approximation algorithm. The partitioning algorithm will also determine if for some instances of specific problems (e.g. with no common regions), HQMs can be solved independently. We should also fulfill the following properties:

1. The number of servers in each partition should be the parameter of the partitioning algorithm. We need to set the number of servers in each subregion. There is a maximum size that is efficiently solvable with both hypercube models and over-partitioning (i.e. using more partitions than needed) decreases the accuracy of the final result.

2. Servers in the same partition should be adjacent to each other. This prevents disconnected atoms and helps to have connected subregions which improves accuracy of the method.

3. Partitioning should be sequential in order to apply the approximation algorithm.

4. Partitioning should be efficient. Our aim in developing an approximation algorithm is to evaluate instances in an optimization framework. To do that, we need efficient algorithms in all steps of the evaluation.

For this purpose, we have developed an algorithm that generates "cuts" on the problem area and creates subregions. This algorithm first utilizes a Voronoi diagram for server locations. Afterwards, one or more network flow problems are solved on the line segments of the Voronoi diagram. Flows in these problems start from and end at the vertices on the borders and flows on the inner edges of Voronoi diagram. The set of flows are regarded as the cuts we need to create subregions. To obtain a partitioning with the objectives stated above, for the following indices and sets:

$i \in I$: servers
$j \in J$: atoms
$l \in L$: partitions
$v \in V$: vertices on Voronoi diagram
$e \in E$: edges on Voronoi diagram
$k \in K$: paths on Voronoi diagram

parameters:

$V/\bar{V}$: all/outer vertices on Voronoi diagram
E_v : edges connected to vertex v
I_e^1/I_e^2 : servers that are separated by edge e
J_i : atoms that are accessible by server i
d_j weight of atom j
G^{kl} $\begin{cases} 1 & \text{if partition } l \text{ is on the arbitrarily selected side of path } k \\ 0 & \text{otherwise} \end{cases}$
V_e : vertices connected to edge e
S^l : number of servers in partition l

and variables:

s_v^k : number of edges of path k touching vertex v
z_e^k $\begin{cases} 1 & \text{if edge } e \text{ exists in path } k \\ 0 & \text{otherwise} \end{cases}$
x_i^k $\begin{cases} 1 & \text{if server } i \text{ is on the arbitrarily selected side of path } k \\ 0 & \text{otherwise} \end{cases}$
q_i^l $\begin{cases} 1 & \text{if server } i \text{ belongs to partition } l \\ 0 & \text{otherwise} \end{cases}$
y_j^l $\begin{cases} 1 & \text{if atom } j \text{ is accessible by one of the servers that belongs to partition } l \\ 0 & \text{otherwise} \end{cases}$
t_v^k $\begin{cases} 1 & \text{if vertex } v \text{ exists in path } k \\ 0 & \text{otherwise} \end{cases}$

we develop a binary integer programming problem (BIPP):

$$\min \sum_l \sum_j d_j y_j^l \tag{10}$$

$$\text{s.t.} \sum_{v \in \bar{V}} s_v^k = 2 \qquad \forall k \in K \tag{11}$$

$$s_v^k = 2t_v^k \qquad \forall k \in K; \forall v \in V \backslash \bar{V} \tag{12}$$

$$\sum_{e \in E_v} z_e^k = s_v^k \qquad \forall k \in K; \forall v \in V \tag{13}$$

$$\sum_i x_i^k = \sum_{l:\ \text{if } G^{kl}=1} S^l \qquad \forall k \in K \tag{14}$$

$$x_{I_e^1}^k - x_{I_e^2}^k \le z_e^k \qquad \forall k \in K; \forall e \in E \tag{15}$$

$$x_{I_e^2}^k - x_{I_e^1}^k \le z_e^k \qquad \forall k \in K; \forall e \in E \tag{16}$$

$$x_{I_e^2}^k + x_{I_e^1}^k \ge z_e^k \qquad \forall k \in K; \forall e \in E \tag{17}$$

$$2 - \left(x_{I_e^2}^k + x_{I_e^1}^k\right) \ge z_e^k \qquad \forall k \in K; \forall e \in E \tag{18}$$

$$x_i^k \le \sum_{l:\ \text{if } G^{kl}=1} q_i^l \qquad \forall k \in K; \forall i \in I \tag{19}$$

$$1 - x_i^k \le \sum_{l:\ \text{if } G^{kl}=0} q_i^l \qquad \forall k \in K; \forall i \in I \tag{20}$$

$$\sum_i q_i^l = S^l \qquad \forall l \in L \tag{21}$$

$$y_j^l \ge q_i^l \qquad \forall l \in L; \forall j \in J; \forall i \in J_i \tag{22}$$

$$\sum_l q_i^l = 1 \qquad \forall i \in I \tag{23}$$

$$t_v^k \in \{0,1\} \quad s_v^k \in \{0,1,2\} \qquad \forall v \in V \backslash \bar{V}; k \in K \tag{24}$$

$$s_v^k \in \{0,1\} \qquad \forall v \in \bar{V}; k \in K \tag{25}$$

$$z_e^k \in \{0,1\} \quad x_i^k \in \{0,1\} \qquad \forall k \in K; \forall e \in E; \forall i \in I \tag{26}$$

$$q_i^l \in \{0,1\} \quad y_j^l \in \{0,1\} \qquad \forall l \in L; \forall i \in I; \forall j \in J \tag{27}$$

In this BIPP, Eq. 10 minimizes the total demand that is covered by servers of each partition. As a result, with this objective, for fixed locations of servers, the total demand that is served by multiple partitions' servers is minimized. The physical reasoning of this objective is to have partitions that minimizes the percentage of interdistrict services for given server locations. Since merging subregions uses some approximation algorithms, if the percentage of interdistrict services can be decreased, more accurate results can be calculated.

Constraints 11 force each path to touch only two different outer vertices with constraints 25. With these constraints, we are creating paths which start and end in the borders of the Voronoi diagram, similar to cuts dividing the whole Voronoi diagram into two pieces. Constraints 12 require that if a path is passing through an inner vertex, two edges from the same path should be touching the vertex with the help of the second part of constraints 24. Note that, the mathematical model selects both edges and vertices, and if an edge is selected, two vertices that belongs to this edge should also be selected. Different than outer vertices, inner vertices should be selected twice, to have a (continuous) path made of edges.

Constraints 13 work for inner and outer vertices differently because of constraints 11 and 12: If an inner vertex v is selected (i.e. $s_v^k = 2$), two of the edges that are adjacent to vertex v have to be selected as well. However if v is an outer vertex, if it is selected (i.e. $s_v^k = 1$) only one of the adjacent edges has to be selected.

Constraints 14 satisfy that the partition sizes that are in the arbitrarily selected side of path k should sum up to the number of servers in the same selected side of path k. Note that when the model is generated, each S_l is calculated in a way that each partition has a predetermined number of servers. For instance,

for the partitioning in Fig. 5, we set $S^l = 6$ where $l = 1, ..., 4$. Binary variables G^{kl} are set in a way that each path has equal number of partitions in both of their sides: $G^{1,1} = G^{1,3} = G^{2,1} = G^{2,2} = 1$ and the rest of the $G^{kl} = 0$, which means that path $k = 1$ (the vertical path) creates partitions $l = 1$ and $l = 3$, and path $k = 2$ (the horizontal path) creates partition $l = 1$ and $l = 2$ in their selected sides. As a result, we end up with 4 partitions with 6 servers that are formed by two paths. If for example, we prefer to make 3 partitions composed of 8 servers for the same example, we again need 2 paths but with three partitions. In this case, we should set $S^l = 8$ for $l = 1, 2, 3$ and $G^{1,1} = G^{1,2} = G^{2,1} = 1$ and the rest of the $G^{kl} = 0$. This way, we will make two paths in a way that, path $k = 1$ has two partitions ($l = 1$ and ($l = 2$) and path $k = 2$ has one partition ($l = 1$) on their selected sides.

If the edge between two servers does not belong to path k (i.e. $z_e^k = 0$), constraints 15 and 16 ensure that the servers on the opposite side of edge e belongs to the same side of path k. Similarly, if the edge between two servers belongs to path k (i.e. $z_e^k = 1$), constraints 17 and 18 forces the servers to be at the opposite side of path k.

Constraints 19 and 20 assign each server into their groups by checking in which side of the paths they belong. In the case depicted in Fig. 5, partition $l = 1$ is in the selected side of both paths. If a server is at the same side, it is assigned to partition $l = 1$. On the other hand, if a server belongs to not selected sides of both paths, it is assigned to partition $l = 4$ because this partition is selected as such by setting $G^{1,4} = G^{2,4} = 0$. In a similar way, the rest of the partitions' servers are assigned.

Constraints 21 sets the number of servers in each partition to the partition's size. Constraints 22 associates atoms with partitions: If an atom can be accessed by a server, one of the servers that belongs to this partition can access to the atom. Constraints 23 place each server into one and only one partition.

Constraints 24-27 sets variables as binary variables except for the variables s_v^k in which v represents an inner vertex of the Voronoi diagram. The reason of this assignment is, as stated above, to satisfy that inner vertices should be selected twice in a continuous path starting from and ending at the outer vertices of the Voronoi diagram. Physical topological boundaries (e.g. rivers, mountains etc.) that do not allow interactions between specific servers and atoms, can be easily integrated in the above formulation as further constraints.

In experiments, we have observed that adding cuts iteratively is much more efficient than solving the whole BIPP at once. As a result, we develop and use a heuristic that generates a single cut at each step. This is also consistent with the step of merging, which will follow inverse iterations of the partitioning. Steps of this heuristic for an instance can be seen in Fig. 5. Fig. 5a shows the problem area: demand intensity (darker color represents higher demand) and locations of servers (red circles). The first step of the algorithm is to generate a Voronoi diagram (Fig. 5b). The steps afterwards are iteratively adding cuts to whole problem area. In our case, first a vertical then a horizontal cut is added (Fig. 5c-d).

Appendix C provides a more detailed description of MHQA through a numerical example to highlight each methodological and algorithmical step of the approach.

5. Computational Results

In this section, we first evaluate the accuracy of the models described in Section 3 and 4 (i.e. 3^n HQM and MHQA) by comparing them with the results of a discrete event simulation. Then, we utilize our models to evaluate instances in two optimization methods with an objective to identify close-to-optimum locations that optimize specific performance measures (loss rates and service times): variable neighborhood search (VNS) (Mladenović and Hansen, 1997) and simulated annealing (SA) (Kirkpatrick et al., 1983). Finally, we provide different performance measures of the optimization results that highlight the importance of the developed models. All of the algorithms in this work are developed under C# .NET environment. For partitioning algorithm IBM ILOG Cplex 12.4 is used through Concert user interface. MATLAB 7.9 through MATLAB Automation Server interface is used for matrix operations. All experiments are conducted on a PC with Intel Core2 Quad 3.00 Ghz processor.

In order to test the method, we use two different networks for demand distribution: Central Athens network with demand for tow-away services for bus operations (taken by Geroliminis et al., 2011) (top) and an experimental network (bottom) which are given in Fig. 8. The model is demonstrated for the case of the Athens (Greece) surface public transportation network which is also utilized in Geroliminis et al. (2011). The network consists of over 3000 buses of different sizes and types with a daily passenger demand of 1.7 million passengers. In an effort to provide high level services, The Athens Public Transport Organization (OASA) uses response units to provide rapid repair services in cases of bus accidents and/or malfunctions, tow-away of illegally parked vehicles, and so on. In order to apply the model to the examined network, we follow the approach of Karlaftis et al. (2004) which divides the network according to a grid of 1km^2 square cells (with each cell corresponding to a hypercube atom). Incident rates per cell are then derived as a function of the total length of bus lines within the cell and 10-year statistics on the average number of incidents per line type and vehicle size within the network. In both figures, each square shows a 1km^2 area. The value inside each square shows 10000 times the ratio of arrival rate to total arrival rate. Euclidean norm is used to calculate distance. Total service time is the sum of on-scene service time and the total travel time to reach incident atom and coming back. We did not test networks with physical boundaries but it can be easily integrated to our hypercube models (by applying exact travel times for each atom-server pairs) and partitioning algorithm (by generating an additional edge in the Voronoi diagram). Different distance metrics (e.g. rectilinear, squared Euclidean) can also be applied to them. The only difference may be in the partitioning algorithm. Voronoi diagrams work for Euclidean distance metric. For another metric, we need different set of vertices and edges but the mathematical model used in the partitioning algorithm is still applicable.

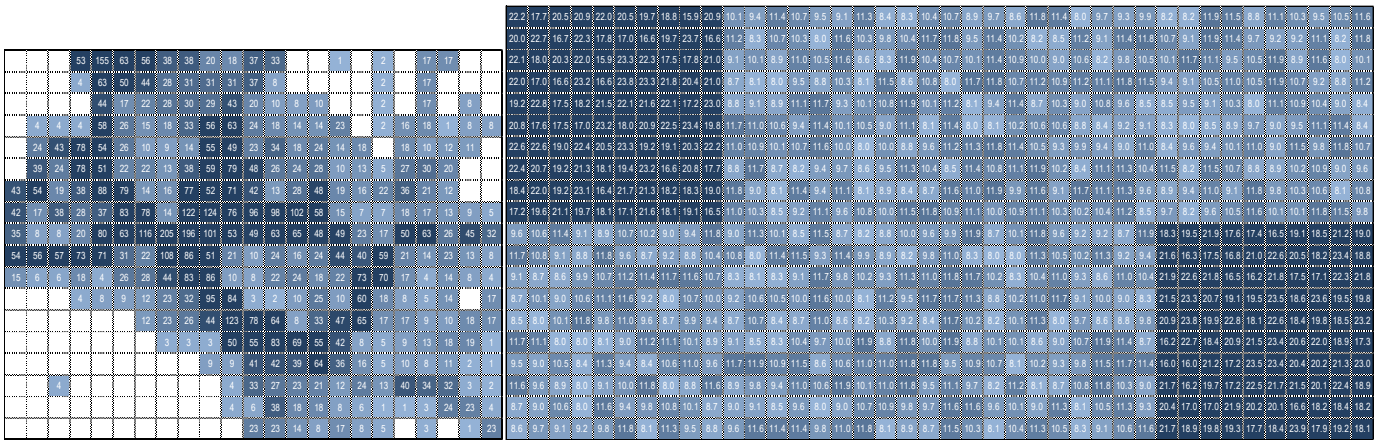


Figure 8: The demand distribution of the two networks used in our experiments: Central Athens (left) and experimental (right).

5.1. Accuracy of 3^n HQM

In this part, the loss rate computed by MHQA is compared with the solution of the discrete event simulation for a case with 12 servers. We tested following instances with three demand (5, 15, 45 requests/hour), average on-scene service time (5, 10, 20 min) and accessibility distance (10, 15, 20 kms) for each demand distribution which makes 27 scenarios in total for each instance. It is assumed that each server travels with a speed of 60 km/h. On-scene service time and inter-arrival times are distributed exponentially. In simulation, a random value is generated whereas in approximation method it is assumed that total service time is exponentially distributed with the sum of travel time and on scene service time. We need such an assumption for the Markovian property. We generated 500 random instances with 12 facilities. In approximation algorithm, the whole problem area is partitioned into two core subregions with 6 servers each. Then the algorithm described in Fig. 7 is applied. Both simulation and our method are run over these networks. The percentage of error in loss rates are reported in Fig. 9. We calculate the errors by comparing the values of MHQA with simulation values.

To ensure that simulations have reached steady states, 25 parallel simulation instances were created with 11 batches simulating length of 50 days each. The first batch of each run was discarded and mean of the rest of the batches of all 25 parallel simulations are reported. Length of the simulations are selected in a way that calculated values have tight enough confidence intervals to guarantee steady state.

For each instance, simulation took around 25 s for both networks whereas our method took around 3 s for Athens network and 7 s for the experimental network on average. The comparison showed that compared to simulation, our method gives results with acceptable error (less than 5% error on average and 10% in the worst case) in less run time (12-28% of simulation run time). This error gets even less for increased server range, for the scenarios with the range of 20 km, average error is less than 1% and in 95% of the cases error is less than 2%. Furthermore, simulation might need longer run times to have accurate results if more detailed performance measures (e.g. loss rate per number of busy servers) for larger instances of the problem should be calculated.

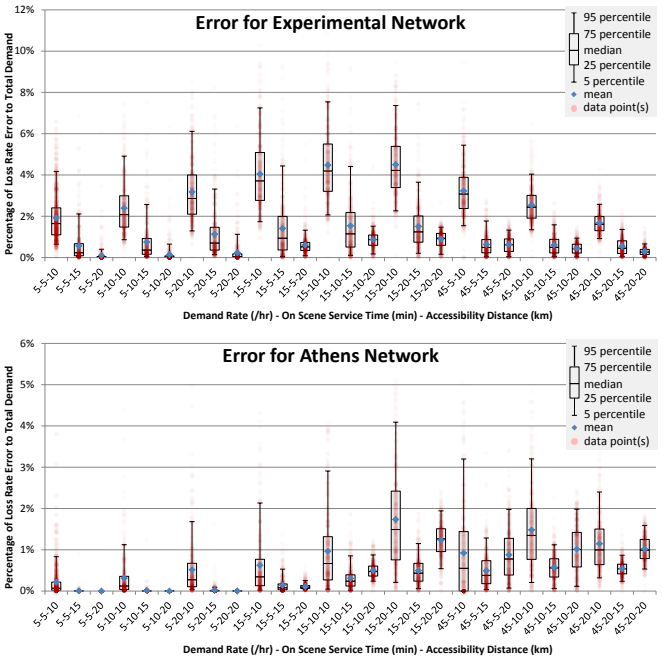


Figure 9: The ratio of difference between the loss rates calculated by simulation and the MHQA.

5.2. Heuristics for Better Location of Servers

In this part, we have tested our exact 3^n HQM (for cases with less than or equal to 8 servers) and mix aggregate hypercube (for cases with more than 8 servers) algorithms inside two heuristic approaches to identify close-to-optimum locations of servers: variable neighborhood search (VNS) (Mladenović and Hansen, 1997) and simulated annealing (SA) (Kirkpatrick et al., 1983). Both methods are initialized with the maximum expected coverage location model (MEXCLP) (Daskin, 1983). MEXCLP is selected because it is fast and suitable to compare with our model. In VNS algorithm, we assume that if two instances' all but one servers are in different locations, they are

neighbors. In other words, in every iteration, a randomly selected server's location is changed. We use the same neighborhood structure in SA algorithm. We set the starting "temperature" coefficient to 1 and increase it by 10% in every 20 iterations. Temperature is assumed to be the division of temperature coefficient with the average loss rate in every iteration. We have applied 3^n HQM for cases with 6,7 and 8 servers for total arrival rates of 5, 8, 10, 15 and 20 requests/hour. For cases with 12, 16, 20 and 24 servers, arrival rates are increased (i.e. 10, 16, 20, 30, 40, 60, 80 requests/hour), and the problems are solved by approximation algorithm with two (for instances with 12 and 16 servers) and four (for instances with 20 and 24 servers) partitions of equal size. Run times for the former three (i.e. 6, 7 and 8-server) cases are set to one hour, whereas the latter two (i.e. 12, 16, 20 and 24-server) cases are run for four hours. We have applied two different on scene service times: 5 and 20 minutes. For all cases, maximum accessibility distance is set to 30 km. Found minimum loss rate and percent loss rate improvements after MEXCLP for Athens and experimental networks (Fig. 8) by both heuristics (i.e. VNS and SA) for cases with realistic loss ratios (ratio of loss rate to the total arrival rate) can be seen in tables 1 and 2 respectively.

For both Athens and experimental networks, it is observed that there is an improvement of more than 35% on average for the loss rates over MEXCLP. Average loss rate improvement gets around 45% for the Athens network with short on scene service times (i.e. 5 minutes). We have not observed any significant difference between VNS and SA. We have observed that VNS performs better in larger instances (e.g. experimental network with 24 servers) than SA. Besides that, they perform very close to each other. Since our primary goal in this research is to test the applicability of hypercube models inside search algorithms, we have not searched for parameters that may give better final results for both VNS and SA.

From the results in tables 1 and 2 one can also observe that the loss rates dramatically increase with the increase of on scene service time. Small increase in demand has also considerable influence on the loss rate. Queueing systems are unpredictably complex and need custom-built algorithms to be tested. Last but not least, careful readers might realize that for the same value we have calculated different percent improvements. This is the consequence of showing results with limited precision. We also noticed (not shown here) that even after 30 minutes (instead of 4 hours) the VNS and SA methods provide similar improvement with a 4 hour run.

# servers	on scene serv. time	demand	MEXCLP	VNS		SA	
				value	% impr.	value	% impr.
6	5	5	0.007	0.004	38.75	0.004	38.75
		8	0.129	0.080	38.28	0.080	37.95
		10	0.422	0.266	37.07	0.268	36.50
	20	5	0.174	0.138	20.61	0.138	20.52
		8	1.197	0.989	17.36	0.989	17.34
		10	2.403	2.048	14.76	2.048	14.76
7	5	5	0.001	0.001	42.5	0.001	41.13
		8	0.031	0.018	41.98	0.018	42.48
		10	0.136	0.079	42.32	0.081	40.78
	20	5	0.059	0.046	22.30	0.045	23.18
		8	0.653	0.521	20.19	0.528	19.12
		10	1.564	1.283	17.95	1.295	17.17
8	5	8	0.008	0.004	53.68	0.005	41.29
		10	0.044	0.021	52.53	0.023	46.62
		15	0.651	0.335	48.60	0.393	39.60
	20	5	0.019	0.013	29.13	0.014	27.18
		8	0.340	0.248	26.87	0.260	23.46
		10	0.987	0.745	24.54	0.795	19.51
12	5	16	0.005	0.002	70.05	0.001	71.86
		20	0.077	0.016	79.45	0.018	76.40
		30	1.824	0.690	62.17	0.692	62.05
	20	10	0.039	0.020	49.23	0.020	48.62
		16	0.993	0.726	26.86	0.691	30.43
		20	3.118	2.307	26.02	2.305	26.07
16	5	20	0.001	<1E-3	42.19	<1E-4	89.56
		30	0.121	0.058	51.88	0.025	79.35
		40	1.666	1.113	33.20	0.412	75.27
	20	16	0.102	0.059	41.74	0.045	55.40
		20	0.645	0.374	42.00	0.368	43.00
		30	5.761	4.359	24.33	4.271	25.86
20	5	30	0.001	<1E-4	95.52	<1E-4	93.33
		40	0.044	0.003	93.71	0.003	92.88
		60	3.362	0.726	78.42	0.741	77.97
	20	20	0.038	0.012	68.57	0.013	65.75
		30	1.68	0.909	45.89	0.927	44.80
		40	7.745	5.801	25.11	5.814	24.94
24	5	40	0.001	<1E-4	96.32	<1E-4	91.69
		60	0.274	0.024	91.23	0.053	80.65
		80	5.574	1.504	73.02	2.183	60.84
	20	30	0.243	0.103	57.43	0.115	52.51
		40	3.024	1.817	39.92	2.129	29.58
		60	18.676	15.81	15.35	16.277	12.84

Table 1: The best loss rate found by MEXCLP, VNS and SA algorithms for the Athens network given in Fig. 8a.

5.3. Performance Measures of Hypercube Models

In this subsection, we conduct further analysis to some location instances improved by the optimization heuristics. In the first analysis we investigate the effect of accessibility range and demand on servers' workload (fraction of time a server is busy) and intradistrict service ratio for fixed locations, with the 3^n HQM for 8 servers. These fixed locations are estimated through the optimization heuristics for some level of demand and they are not recalculated when demand changes. In the second analysis, the effect of demand increase on the number of busy servers and the ratio of the time spend on intradistrict service in bin level is analyzed. A larger instance with 12 servers solved by 3^n AHQM is considered (2 bins of 6 servers each).

In order to see the effect of accessibility range and demand on servers' utilities, ten instances (5 demand levels and 2 accessibility ranges) with 8 servers on the experimental network are analyzed. To have a better insight, the location of the servers

# servers	on scene serv. time	demand	MEXCLP	VNS		SA	
				value	% impr.	value	% impr.
6	5	5	0.076	0.055	26.72	0.055	26.67
		8	0.725	0.594	18.09	0.594	18.09
		10	1.691	1.452	14.1	1.452	14.1
	20	5	0.432	0.377	12.63	0.377	12.67
		8	2.074	1.926	7.11	1.926	7.11
		10	3.626	3.439	5.16	3.439	5.16
7	5	5	0.022	0.013	39.27	0.013	39.27
		8	0.343	0.252	26.29	0.252	26.29
		10	0.994	0.775	22	0.775	22
	20	5	0.207	0.167	19.34	0.168	19.15
		8	1.436	1.267	11.81	1.27	11.62
		10	2.813	2.566	8.78	2.569	8.69
8	5	5	0.005	0.003	47.51	0.003	47.51
		8	0.129	0.098	23.94	0.098	23.94
		10	0.483	0.387	19.82	0.387	19.82
	20	5	0.084	0.065	22.77	0.068	19.71
		8	0.894	0.776	13.21	0.798	10.73
		10	2.035	1.826	10.28	1.848	9.21
12	5	10	0.003	<1E-3	84.54	<1E-3	84.54
		16	0.263	0.054	79.36	0.054	79.36
		20	1.272	0.472	62.90	0.472	62.9
	20	10	0.200	0.095	52.49	0.099	50.18
		16	2.680	2.062	23.03	2.064	22.99
		20	5.714	5.004	12.42	4.989	12.68
16	5	16	0.004	0.001	87.32	0.001	87.32
		20	0.068	0.008	87.92	0.008	87.92
		30	2.423	0.786	67.58	0.786	67.58
	20	16	0.551	0.282	48.85	0.290	47.37
		20	2.203	1.458	33.84	1.448	34.29
		30	9.947	9.058	8.94	9.172	7.80
20	5	30	0.141	0.064	54.95	0.082	41.92
		40	1.139	0.700	38.49	0.741	34.95
		60	10.199	8.085	20.73	8.258	19.03
	20	20	0.41	0.299	27.04	0.325	20.81
		30	3.767	3.192	15.26	3.351	11.03
		40	10.787	9.218	14.54	10.126	6.12
24	5	40	0.149	0.089	40.35	0.139	6.36
		60	3.652	2.721	25.51	3.576	2.10
		80	16.228	13.868	14.54	16.184	0.27
	20	20	0.064	0.044	30.84	0.064	-
		30	1.304	1.084	16.83	1.304	-
		40	6.098	5.443	10.74	6.027	1.16

Table 2: The best loss rate found by MEXCLP, VNS and SA algorithms for the experimental network given in Fig. 8b.

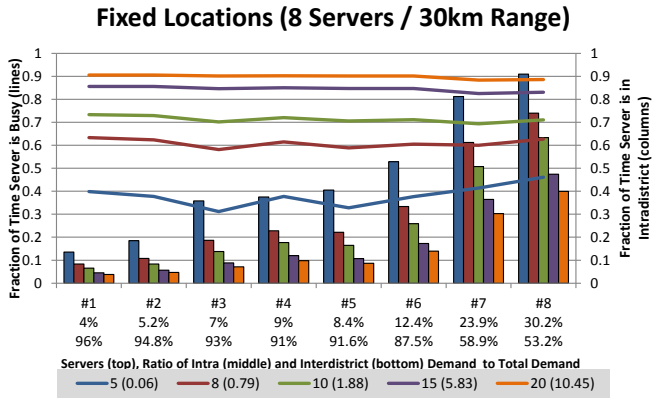
are fixed in all instances. We set the best locations found by the heuristic for demand 5 requests/hour and on scene service time equal to 20 minutes (see Table 2). The fraction of time each server is busy (lines) and the fraction of busy time each server is in intradistrict response (columns) are reported in Fig. 10a and b. Demand levels (5, 8, 10, 15 and 20 requests/hour) are shown with different colors. Two different accessibility ranges (15 and 30 minutes) are used in two separate graphs, 10a and b respectively. The loss rate for different demand levels are reported in the legends of each figure inside the parenthesis. The fraction of the total demand each server has in their intra and interdistrict area are also reported under the x-axis. Note that the first line of percentages sum up to 1, which means that all atoms are reachable by at least one server. The second line has much higher values because it considers the accessibility of interdistrict responses. The locations of each server (#1 to #8 are shown in the x axis of the graphs) can also be seen in the map reported in Fig. 10c. Note that, the intensity of the blue color shows the level of demand; darker the blue, higher the demand.

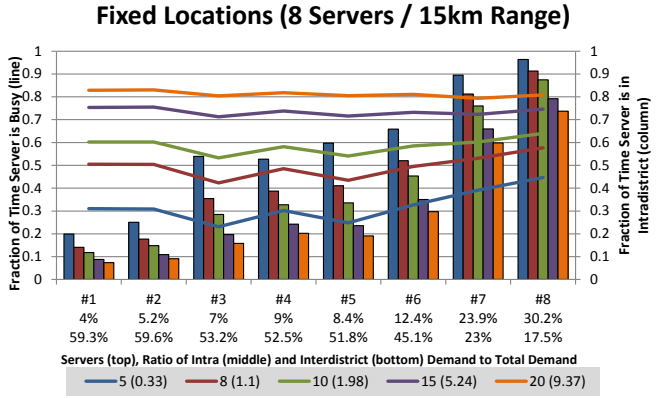
One of the striking outcomes of these graphs is the ratio of the time spend in intradistrict responses. Although, the probability of being busy fluctuates a little among servers (5-10%), the ratio of intradistrict responses is quite variant for different servers (5-95%). This fact shows the importance of using a model, which differentiates intra and interdistrict responses. The main reason to have such fluctuations in intradistrict response ratios is the difference between the responsibility areas of each server. Servers with more intradistrict demand spend more on intradistrict service. This is expected. However, seeing such a huge difference between time spent on intradistrict service among servers is an interesting observation. This high intradistrict fraction motivates to investigate the effect of the accessibility distance in the results.

In addition, by comparing Figs. 10a and b, we can see the effect of accessibility range and the importance of dispatching policy. In this research, advanced dispatching policies are not investigated. In every state, the available server with the minimum service time is dispatched at all times. For low demand levels, higher accessibility range works better than the low one. For demand 5 requests/hour, we get a loss rate of 0.06 /hour with a 30 km accessibility range whereas this value is 0.33 /hour for an accessibility range of 15 km. With the increase in demand, lower accessibility range improves more and gives better results. For demand level 20 requests/hour, lower accessibility range has a smaller loss rate than the higher accessibility range because requesting from a server to intervene in a request far from his responsibility area under high demand conditions, might result in loss demand for intradistrict requests.

There is something more to add to the analysis. Although, compared to high accessibility range, low accessibility range always provides a higher intradistrict ratio for all instances and servers, for low demand levels, it does not give better loss rates. This is probably because of the congestion in the system. By lowering accessibility range, although we encourage the model to have more intradistrict responses, we are also decreasing the secondary service areas of each server. In other words, we are lowering the number of servers that can be dispatched at each incident and obviously specific to our instance, this declines the performance of the system for lower demand and ameliorates for higher demand. Note that, even for small accessibility range, all demand is accessible by at least one server.

In our second analysis, we investigate performance measures from a larger instance with 12 servers solved by the MHQA (2 bins with 6 servers each). We take best instances we have found





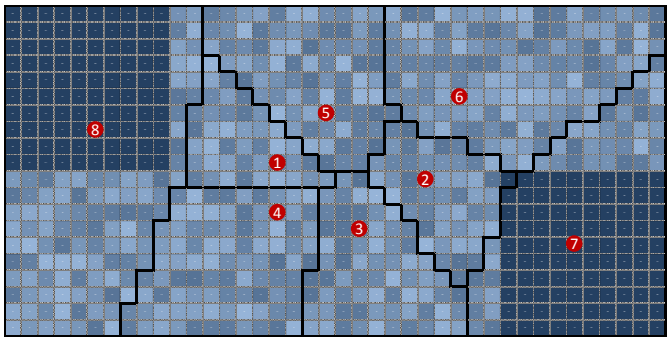


Figure 10: In the top two graphs (a and b), the effect of increased demand and accessibility range is shown for instances of 8 servers (with 3^n HQM). The same server locations are selected in all instances, which can be seen in the map given below the two graphs (c). Both the fraction of time each server is busy (line) and the fraction on intradistrict response (column) are shown in a and b. The total demand at each servers' primary and secondary service area are shown under the x-axis respectively. Note that the values in parenthesis in the legend are the loss rates.

with our heuristics for 12 servers and 20 minutes on scene service time in the experimental network. Different than the previous analysis, we investigate the probability of having specific number of servers busy in the system all of which can be seen in Fig. 11. These values are shown with columns. With lines, the percentage of time servers busy with bin instradistrict responses (i.e. dispatching inside bin) are depicted. We use 5 different demand levels and each of which are shown with a different color. Again, loss rates are shown inside parenthesis in the legend next to the related demand level.

The first interesting result that can be observed from this figure is the effect of increased demand on the system efficiency. The increase in demand results in less time spend in intradistrict (equivalently more time in interdistrict) responses. The increase in the number of busy servers has a similar effect. With the increase in busy server count, intradistrict response ratio decreases.

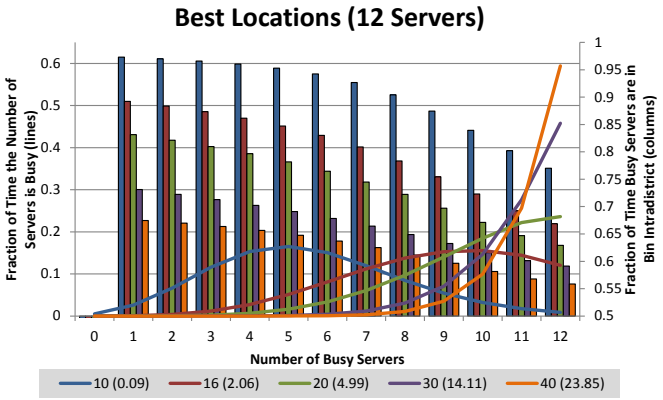


Figure 11: The effect of increased demand on 5 best instances of different demand levels. In all instances, the number of servers is 12, on scene service time is 20 minutes and accessibility range is 30 km. The fraction of time for different busy servers' count (line) and the ratio of servers in intradistrict response (columns) can be seen in the figure. Loss rates for each demand level is given in parenthesis in the legend, next to the related demand level.

One of the other important findings observed in graphs in Fig. 11 is the interaction between subregions. In a simpler approach, we could consider each core subregion as a separate problem and do calculations regarding this assumption (e.g. sum loss rate of each subregions to find total loss rate). However, results in Fig. 11 show that the amount of time spent for interdistrict responses is significant. Because of this, disregarding interdistrict responses between subregions and estimate performance measures with 3^n HQM without the step of merging two compound regions, might create inaccurate results and does not seem to be a right approach.

6. Concluding Remarks

In this research, spatial queueing systems (SQS) are considered where the service rate for each customer-server pairs differs and the server that intervenes for a specific customer is not known a priori. While SQS are computationally expensive (but more appropriate for real-life problems), in this paper, we extend hypercube queueing models that can accurately estimate various performance measures for medium to large scale networks and consider the probability that the nearest server might be unavailable to intervene. To decrease the dimension of the problem, a new 3^n aggregate hypercube queueing model (AHQM) together with a partitioning algorithm are developed that treat group of servers (bins) in a similar manner and also considering interactions among bins. We first compare our approach with the system with real service time and show the accuracy of our approach for different parameters on two different networks: one real (central Athens) and one experimental

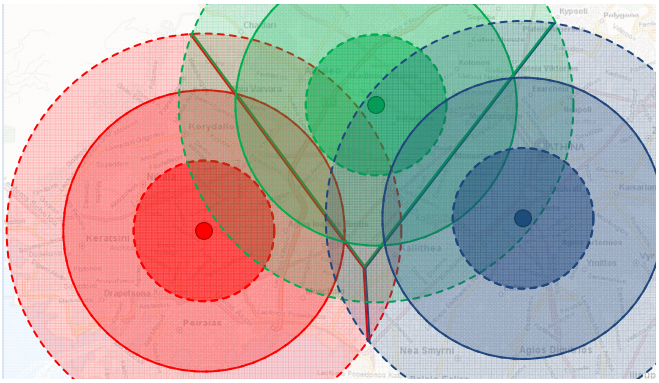


Figure 12: Illustration of a system with three servers and three service range belts for each server

with different spatial distribution of demand. In order to show the applicability of our two algorithms inside an optimization framework, the two methods are implemented with the variable neighborhood search and simulated annealing to identify near optimal server locations that improve system performance. From these experiments, it is seen that, although hypercube queueing models are not initially developed as optimization models, they can be integrated in such a framework. Then, we have investigated different performance measures such as the percentage of time the servers spend on intra and interdistrict responses. The percentage of interdistrict responses is not negligible and should not be omitted from the analysis of location models where fast and reliable response are required.

As future research, our aim is to change the way we define intra and interdistrict response areas. In hypercube queueing models, service rate and service priority are different parameters. We can still define intra and interdistrict areas with service priority, where instances can be served by the closest available server. However, we can arrange different service rates for different distance ranges from the servers. We can create two or more service range belts and we can apply different service rates for each belt (see Fig. 12). Another direction to further investigate is the partitioning algorithm. With some additional cuts, we might create a tighter convex hull which in the end might give a more efficient partitioning algorithm. We are also interested in some efficient heuristic methods to replace the partitioning algorithm. Last but not least, different dispatching algorithms are also worth investigating. For the same server locations, intelligent dispatching schemes may give better results.

Acknowledgement

This research was partially supported by European Union Research Council (ERC) Starting Grant "METAFERW: Modeling and controlling traffic congestion and propagation in large-scale urban multimodal networks". The authors would like to acknowledge the useful comments and direction provided to them by the associate editor and the anonymous referees.

Appendix A. Application of a 3^n HQM

In this section, the approach to calculate the steady state probabilities of an instance of 3^n HQM is shown. The configuration shown in Fig. 10 is used with total demand 10 requests/hour, 10 km service range and 20 minutes on scene service time.

First, primary and secondary service areas of each server are found. In Fig. A.13A-D servers 5-8's (both primary and secondary) service areas can be seen separately in different maps (first 4 maps). The intensity of colour blue shows the arrival rate magnitude to this atom. Server location at each map is marked with a red circle around. Primary service area of each server is the area that is bounded by red borders. Fig. A.13E shows all servers' locations (dark red circles) with their reachable area (light red circles) and the atoms that cannot be served by any of the atoms (blue).

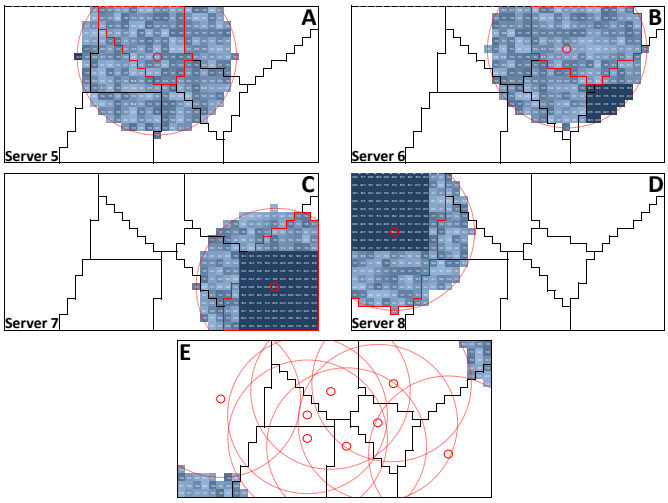


Figure A.13: Primary (area with red border) and secondary (rest of the atoms colored with blue) service area of servers 5-8 in order (A-D) and the atoms that cannot be reached by any server (E). Values on each atom shows 10000 times the ratio of the demand on that specific atom to whole demand.

With the help of these partitioned maps for each server, average intra and interdistrict service times and service rates are calculated. In Table A.3, the values used in calculation and average service rates can be seen. Note that, service duration is the sum of on scene service time (20 minutes) and travel time to and from the service scene. In distance calculation, Euclidean distance is used. Each square represents atom with 1km^2 area and service vehicles travel with a speed of 60 km/hour.

Next step is creating transition diagram which is composed of generating states and calculating transition rates between states. In this step, incoming and outgoing transition rates are calculated for every state (see Eq. 2). Fig. A.14 contains transition diagram for a state of the system given in Fig. 10. $\begin{smallmatrix}0000\\1221\end{smallmatrix}$ shows, the first and fourth servers are busy with intradistrict (represented with "1"s), second and third servers are busy with interdistrict (represented with "2"s) responses whereas the rest of the servers are available for service (represented with "0"s).

For most of the states, specific arrival rates for different busy servers should be calculated. These calculations can only be

#	primary				secondary			
	∑ demand x time	∑ demand	av. ser. time	av. ser. rate	∑ demand x time	∑ demand	av. ser. time	av. ser. rate
1	10.76	0.40	26.73	2.24	103.44	2.99	34.6	1.73
2	13.45	0.52	26.08	2.30	105.56	3.03	34.88	1.72
3	19.48	0.70	27.97	2.15	76.37	2.21	34.59	1.73
4	26.00	0.86	30.08	1.99	75.17	2.18	34.42	1.74
5	23.71	0.84	28.39	2.11	68.89	1.98	34.72	1.73
6	32.70	1.11	29.55	2.03	62.22	1.76	35.26	1.70
7	67.42	2.36	28.62	2.10	27.75	0.77	36.16	1.66
8	84.37	2.81	29.99	2.00	21.02	0.57	36.85	1.63

Table A.3: Sum of demand (columns 2 and 6) sum of products of service duration and demand of each atom (columns 3 and 7), average service time (columns 4 and 8) and average service rate (columns 5 and 9) for each servers' primary (columns 2-5) and secondary (columns 6-9) secondary service areas.

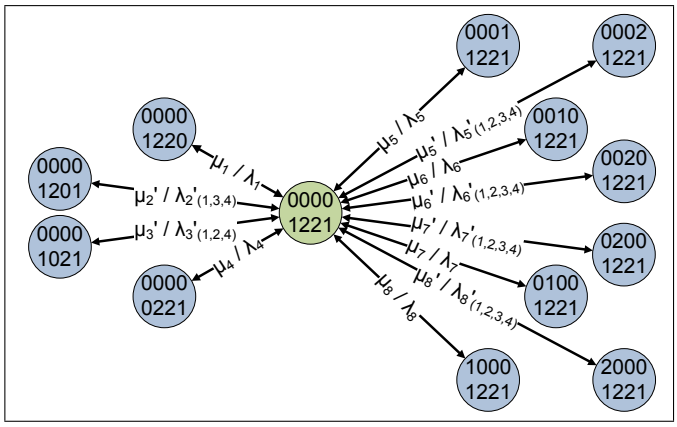


Figure A.14: Part of the transition diagrams of a 3^n HQM for 8 servers for state 00001221. Each server has different intra (μ_i) and interdistrict (μ_i') service rates for different primary (λ_i) and secondary (e.g. λ_i', $\lambda_i^{\{i_1,i_2,i_3\}}$) service areas.

done at the atom level. In other words, for different busy server sets, each atom's preferred available server is different. If in addition to μ_i and μ_i' are intra and interdistrict service rates and λ_i and λ_i' are demand arrival rates from primary and secondary service areas of server i, λ_i^I is the arrival rate of atoms, which are dispatched to server i in first place in case of servers in set I are busy, the transition equation for state $\substack{0000\\1221}$ (see Fig.A.14) is:

$$\mathbb{P}_{\substack{0000\\1221}}\Big[\overbrace{\lambda_5+\lambda_5^{\{1,2,3,4\}}+\lambda_6+\lambda_6^{\{1,2,3,4\}}+\lambda_7+\lambda_7^{\{1,2,3,4\}}+\lambda_8+\lambda_8^{\{1,2,3,4\}}}^{*}$$
$$\mu_1+\mu_2'+\mu_3'+\mu_4\Big]=\mathbb{P}_{\substack{0000\\1220}}\lambda_1+\mathbb{P}_{\substack{0000\\1201}}\lambda_2^{\{1,3,4\}}+\mathbb{P}_{\substack{0000\\1021}}\lambda_3^{\{1,2,4\}}$$
$$+\mathbb{P}_{\substack{0000\\0221}}\lambda_4+\mathbb{P}_{\substack{0001\\1221}}\mu_5+\mathbb{P}_{\substack{0002\\1221}}\mu_5'+\mathbb{P}_{\substack{0010\\1221}}\mu_6+\mathbb{P}_{\substack{0020\\1221}}\mu_6'+\mathbb{P}_{\substack{0100\\1221}}\mu_7$$
$$+\mathbb{P}_{\substack{0200\\1221}}\mu_7'+\mathbb{P}_{\substack{1000\\1221}}\mu_8+\mathbb{P}_{\substack{2000\\1221}}\mu_8' \quad (A.1)$$

$*$ in Eq. A.1 is equal to the arrival rate that can be served by the servers that are available in the system during the state $\substack{0000\\1221}$. When some servers are not available, the service area of each server should be updated as seen in Fig. A.15A-D. With the help of Fig. A.15, the atoms that are served by the last four servers when the first four are busy can be seen. See also in Fig. A.15E that when some servers are not available, there are more atoms that cannot be reached by any of the available servers. Note that, the service rate to that atom still depends on the server-atom service rate (primary or secondary area).

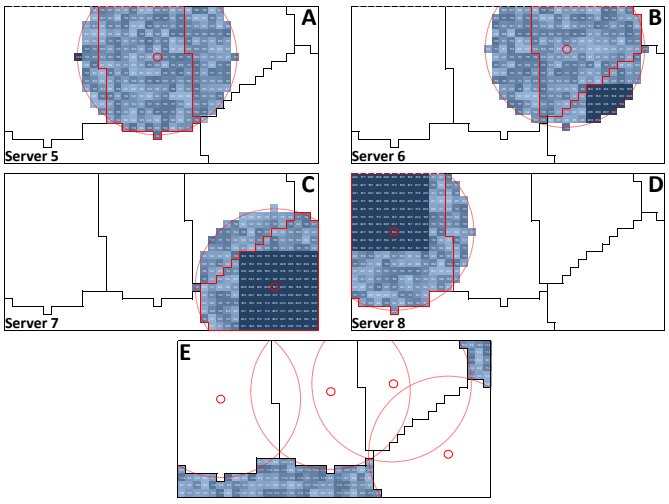


Figure A.15: Service area (with red and black borders) of each available server (5,6,7,8) in order (A-D) and the atoms that cannot be reached by any available server (E) for the state 00001221. Values on each atom shows 10000 times the ratio of the demand on that specific atom to whole demand.

Appendix B. Application of a 3^n AHQM

In this section, the approach to calculate the steady state probabilities of an instance of 3^n AHQM is described. The same characteristics as in Appendix A are utilized. In addition, 8 servers are replaced with 3 bins with 3 servers each. The locations of each bin with their primary and secondary service areas (A-C), and the atoms that cannot be served (D) can be seen in Fig. B.16.

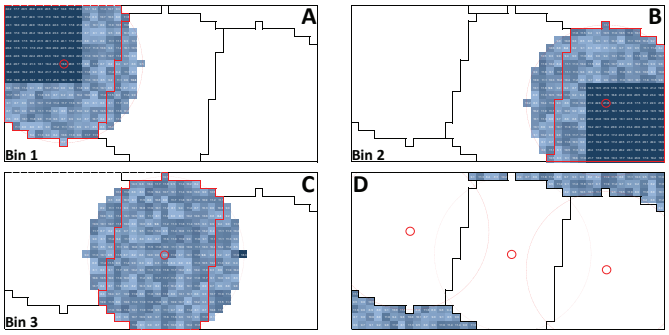


Figure B.16: Bin locations with primary (area with red border) and secondary (rest of the atoms colored with blue) service area of each server in order A-C) and the atoms that cannot be reached by any available server (D) for a system with 3 bins. Values on each atom shows 10000 times the ratio of the demand on that specific atom to whole demand.

Each of the bins' maps can be used to calculate their primary and secondary service rates. The values calculated are average service rates and service times for each server of the bins and in service duration calculation on scene service time and travel time to and from service scene are taken into consideration. These values can be seen in Table B.4.

#	primary				secondary			
	∑ demand x time	∑ demand	av. ser. time	av. ser. rate	∑ demand x time	∑ demand	av. ser. time	av. ser. rate
1	103.94	3.33	31.18	1.92	13.37	0.36	37.50	1.60
2	100.44	3.23	31.07	1.93	17.33	0.47	37.04	1.62
3	78.17	2.43	32.13	1.87	28.60	0.77	37.30	1.61

Table B.4: Sum of demand (columns 2 and 6) sum of products of service duration and demand of each atom (columns 3 and 7), average service time (columns 4 and 8) and average service rate (columns 5 and 9) for each bins' primary (columns 2-5) and secondary (columns 6-9) secondary service areas.

Generating transition diagrams starts with the generation of all states. There are 1000 states when there are 3 bins with 3 servers each. Then for each state, all positive transition rates are calculated. Eq. 5 is used to calculate transition equations with the help of the calculated transition rates.

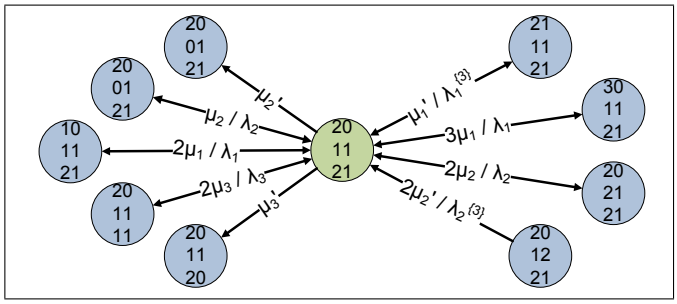


Figure B.17: Part of the transition diagrams of a 3^n AHQM for 3 bins with 3 servers each for state 20|11|21. Servers of each bin (b) has different intra (μ_b) and interdistrict (μ'_b) service rates for different primary (λ_b) and secondary (e.g. λ'_b, $\lambda_b^{\{b_1\}}$) service areas. Values on each atom shows 10000 times the ratio of the demand on that specific atom to whole demand.

If all servers are available in the system (i.e. 00|00|00), if there is a demand from an area that is reachable by one of the bins, this demand has to be in one of the available bins primary service area. When there is no available server (e.g. 30|21|12), there are no upper transitions. Since we model a system without queue, any arrival of demand during states with no available server is a lost.

When the system is partially full, as it is given in state 20|11|21 (see Fig. B.17), there are both upper (e.g. transitions from/to 21|11|21) and lower (e.g. transitions from/to 20|01|21) transitions to be taken care of. In state 20|11|21, bin 1's two servers are busy with intradistrict, bin 2's one server busy with intra and one server busy with interdistrict, and bin 3's two servers are busy with intra and one with interdistrict services. For μ_b and μ'_b are intra and interdistrict service rates for each server, λ_b and λ'_b are primary and secondary service areas of bin b and λ_b^B is the arrival rate of atoms that are served by one of the servers of bin b in first place in case bins in set B have no available servers, the transition equation for state 20|11|21 is:

$$\begin{aligned}\mathbb{P}_{\substack{11\\21}}^{20}\Big[\lambda_1+\lambda_2+\lambda_1^{\{3\}}+\lambda_2^{\{3\}}+2\mu_1+\mu_2+\mu'_2+2\mu_3+\mu'_3\Big]\\ =3\mathbb{P}_{\substack{11\\21}}^{30}\mu_1+\mathbb{P}_{\substack{11\\21}}^{21}\mu'_1+2\mathbb{P}_{\substack{21\\21}}^{20}\mu_2+2\mathbb{P}_{\substack{12\\21}}^{20}\mu'_2\\ +\mathbb{P}_{\substack{11\\21}}^{10}\lambda_1+\mathbb{P}_{\substack{01\\21}}^{20}\lambda_2+\mathbb{P}_{\substack{11\\11}}^{20}\lambda_3\end{aligned}\tag{B.1}$$

If some bins do not have available servers, the service area of each bin with available server(s) should be updated as it is done in Appendix A. Since the third bin has no available server during state 20|11|21, its load should be shared by the first two bins' servers. In Fig. B.18A-B, the service area of the first two bins when the third bin has no available server can be seen. In addition, there are some more atoms that are not reachable by any of the available servers of the bins which can also be seen in Fig. B.18. Similar to the previous section, the service rate to that atom still depends on the bin-atom service rate. If the atom is in the primary service are of the bin, it is served with the primary area service rate, if not it is served with the secondary area service rate of the same bin.

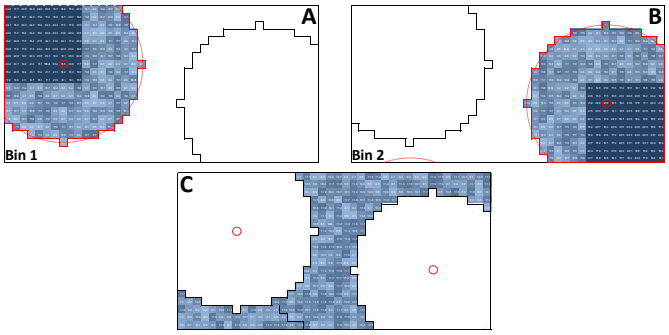


Figure B.18: Service area (with red and black borders) of each available bins (1,2) in order (A-B) and the atoms that cannot be reached by any available server (C) for the state 20|11|21.

Appendix C. Application of an MHQA

In this section, the steps of the MHQA algorithm in Fig. 7 are described on an instance. The configuration shown in Fig. 10 is used with total demand 20 requests/hour, 10 km service range and 20 minutes on scene service time with 12 servers located as shown in Fig. C.19A.

The first step in solving an instance of MHQA is partitioning the problem into subregions (see Fig. 7 Step 1). This partitioning algorithm works with binary steps to create not only core subregions but also compound regions that are different than the whole problem. As described in Subsection 4.2. Partition size is selected as 3. In the runs shown in tables 1 and 2 bins with 5,6 and 8 servers are used since larger partition size increases accuracy of the algorithm. Here, 3 is selected as the bin size to scale down the problem and to show the steps of the algorithm with all of its details without confusing the reader.

The solution of the partitioning algorithm gives the following core and compound regions given in Fig. C.19. In Fig. C.19A, each servers primary service areas are shown. Note also that, straight lines separate core regions whereas dashed lines are for primary service areas of each server within core regions. Fig. C.19B-E shows the service area of each core regions' servers. By merging core regions pairwise we end up with two compound regions shown in Fig. C.19F-G. Compound region 1 is composed of core regions 1 and 2. Compound region 2 is composed of the rest of the core regions, 3 and 4. The whole problem area is also a compound region composed of two compound regions 1 and 2. In Fig. C.19H, the demand in the area

of intersection of at least two partitions' servers service area is shown. Fig. C.19I shows how many times these atoms exist in the service area of partitions' servers. Yellow stands for areas that are covered by only two partitions' servers whereas orange shows the atoms that are covered by the servers' of three different partitions.

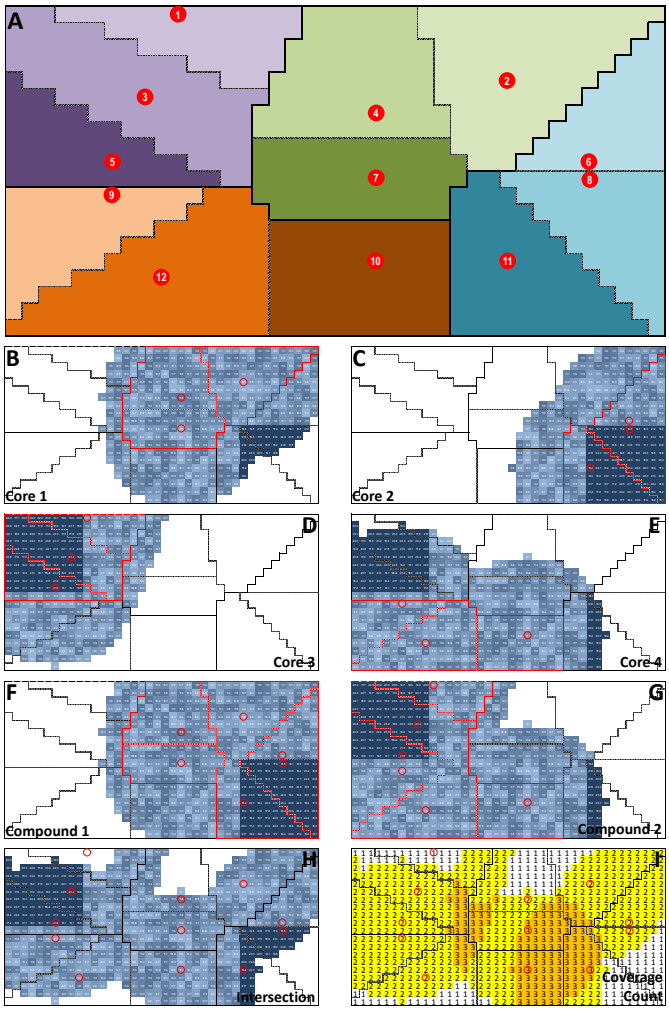


Figure C.19: Primary service area of each server (A), primary service area of each core (B-E) and compound (F-G) regions' servers (red bordered) with their coverage areas (blue), the atoms that are covered by at least the servers' of two partitions (H) and how many partitions covered each atom (I). Values on each atom shows 10000 times the ratio of the demand on that specific atom to whole demand.

The next step in the algorithm is to iterate each region with depth first search (see Fig. 7 Step 2). We start calculations from the core regions since they are the leaf nodes. For each core region, we calculate intra and interdistrict service rates for the primary service areas of the region's servers'.

In solving core regions, we apply the 3^n HQM for primary service areas. For instance for the third core region, we will solve a 3^n HQM for the servers 1,3 and 5 and the atoms that exist in their primary service areas. Note that, we are only taken care of the primary service areas of the three servers, not all the service areas that are accessible. The probabilities of each 27 states and their total service rate are given in Table C.5.

states		probabilities			service rates		
		S1			S1		
S5	S3	0	1	2	0	1	2
0	0	0.093	0.036	0.033	0	2.176	1.737
0	1	0.078	0.033	0.046	2.089	4.265	3.826
0	2	0.025	0.016	0.025	1.893	4.069	3.63
1	0	0.05	0.021	0.023	2.177	4.353	3.914
1	1	0.047	0.026	0.049	4.266	6.442	6.003
1	2	0.026	0.019	0.033	4.07	6.246	5.807
2	0	0.032	0.016	0.025	1.782	3.958	3.518
2	1	0.049	0.035	0.069	3.871	6.047	5.608
2	2	0.027	0.023	0.046	3.675	5.851	5.411

Table C.5: Probabilities (columns 3-5) and service rates (columns 6-8) for all 27 states of the 3^n HQM of core region 3.

The results of the 3^n HQM are used to calculate the average service rates for the number of busy servers, percentage of time the servers are busy and the loss rate of each atom. Average service rate is calculated by summing the products of the probabilities with the service rate of related states for any number of busy servers (for this instance: 0, 1, 2, 3). The average service rates for different bins are given in Table C.6. The average service rates for each bin per server count are used as intradistrict service rates when the related core region becomes one part of a compound region.

service rates		core region			
		1	2	3	4
# servers	1	2.062	2.041	2.015	2.057
	2	4.039	4.003	3.955	4.028
	3	5.963	5.928	5.846	5.985

Table C.6: Average service rates per number of busy servers for each bin.

The percentage of time each server is available is also calculated with the help of the probabilities of each state (Table C.5). See Table C.7 for the availabilities of all servers. Availabilities of servers are used for correcting interdistrict service rates with the help of Eq. 9. When two core (or compound) regions are merged to make a new compound region, we assume that the servers in each of the core (or compound) regions are servers located at the same location. We need this assumption to calculate the performance measures of the new compound region. Further information about this calculation is given below.

core region							
1		2		3		4	
S2	0.47	S6	0.43	S1	0.427	S9	0.527
S4	0.431	S8	0.345	S3	0.33	S10	0.541
S7	0.507	S11	0.381	S5	0.384	S12	0.459

Table C.7: Availability of each server at each bin.

The loss rates of each atom when they are only served with the servers from the core regions are also calculated. These loss rates are used to calculate interdistrict service rates of the regions when they are used to form a compound region.

After calculating performance measures for the core regions, we need to continue with the calculation of performance measures for the compound regions starting from the ones they are located closest to the leafs. We can demonstrate detailed calculations on compound region 1, which is composed of two core regions, 1 and 2.

In compound region 1, the servers of two core regions are regarded as servers of the same bin. Compound region 1 has two bins with three servers: Servers 2, 4 and 7 in bin 1, and 6, 8 and 11 in bin 2. For these bins, we use the service rates per server, which are calculated in the previous step, as intradistrict service rates. Note that, the primary service areas of the first and second bin are the service areas of their servers. Bin 1 takes primary service area of servers 2, 4 and 7 as its primary service area. Secondary service area of bin 1 is composed of atoms of bin 2 that are reachable by the servers of bin 1. Similar assignment applies for bin 2 as well.

This new structure, 2 bins with three servers each, can be formulated as a 3^n AHQM. However, since servers are not exactly at the same location, we need to slightly adjust their service rates and state probabilities. First of all, the values given in Table C.6 are used as intradistrict service rates. For interdistrict service rates, average service time is calculated with the help of loss rates of each atom in core region 2. The locations of atoms of core region 2 and the accessible atoms of core region 2 by the servers of core region 1 (servers 2, 4 and 7), are shown in Fig. C.20.

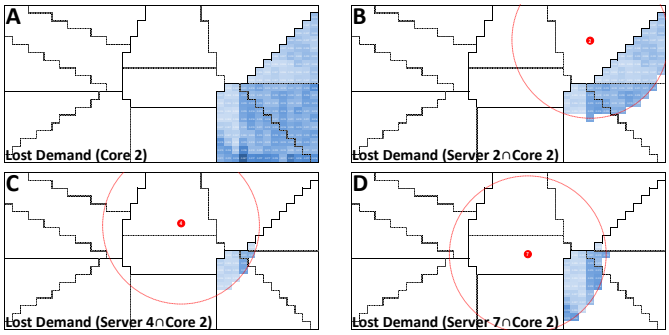


Figure C.20: The loss demand of the atoms of core region 2 (A) and core region 2's reachable loss demand by the servers of core region 1 (B-D).

With the help of these figures, average service time and service rates for the first two compound regions can be calculated. The values specific to this instance can be seen in Table C.8.

service rate		compound region			
		1		2	
	bin	1 (core 1)	2 (core 2)	1 (core 3)	2 (core 4)
# servers	1	1.631	1.689	2.751	1.655
	2	3.292	3.400	4.494	3.421
	3	4.968	5.118	6.292	5.257

Table C.8: Interdistrict service rates of the bins for compound regions 1 and 2.

Note that, these service rates are only applicable when the atom is reachable by the servers of the bin. However, when the servers are condensed to a bin (e.g. servers 2, 4 and 7 to bin 1), these servers are regarded identical to each other to make the model more tractable. However, we use availability of the servers to implement some correction calculations. To do these calculations, for each atom, the probability for lack of service is calculated with the help of Eq. 9. We describe how this probability is calculated on a subset of selected atoms shown in Fig. C.21 for different states.

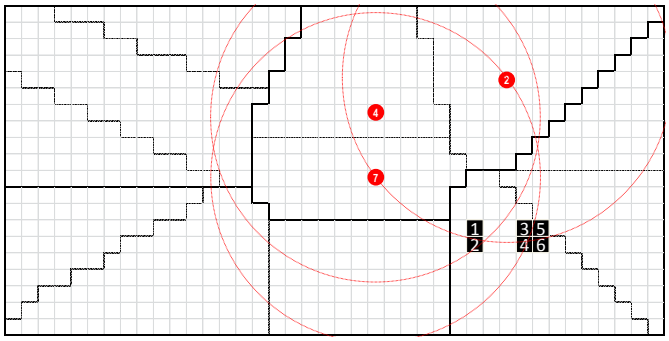


Figure C.21: Atoms selected to calculate their not having service probability.

- Atom 1 is covered by all three servers 2, 4 and 7. So as long as there is an available server in bin 1, it takes service in compound region 1.
- Atom 2 is covered by servers 4 and 7. If there are 2 or 3 available servers in bin 1, there is no doubt it takes service since at least one of the two servers (or maybe both) is available. However, if there is only one server available, with some probability, both 4 and 7 might be unavailable. We can use Eq. 9 use to calculate this probability:

$$\mathbb{P}_{a=2}^{n=2} = \frac{(1-0.507)(1-0.431)}{0.47\times0.431 + 0.47\times0.493 + 0.431\times0.493} = \frac{0.281}{0.780} = 0.360 \tag{C.1}$$

  With probability 0.360, the demand of this atom is not served when there is only one available server in bin 1 and zero in bin 2. We consider that, 0.360 of the demand in atom 2 is not served during states with only one available server in bin 1 and zero in bin 2.
- Probabilities for atoms 3, 4 and 5 are estimated in a similar way.
- Atom 6 is not reachable by any of the servers of bin 1 so all of its demand is lost no matter how many servers exist in bin 1 if bin 2 has no servers to serve.

We use the loss rate to deduce the demand from each atom when 3^n AHQM is applied for compound regions. Then this deduced demand is added on top of the loss rates calculated by 3^n AHQM.

service rate		compound region 3			
		compound region 1		compound region 2	
# servers	1	2.020	1.608	2.019	1.597
	2	3.998	3.283	4.002	3.230
	3	5.939	5.000	5.962	4.887
	4	7.851	6.742	7.909	6.563
	5	9.736	8.499	9.845	8.255
	6	11.607	10.271	11.785	9.961

Table C.9: Intra and interdistrict service rates of the bins of compound region 3.

After estimating the performance measures for the first two compound regions (see Table C.9), we repeat the steps in merging the two compound regions to make a new region: compound

region 3. For a compound region of two merged compound regions, the steps are pretty similar. First the intra and interdistrict service rates are compared for the number of busy servers in both bins. For intradistrict service rates, previous compound regions, in this instance compound regions 1 and 2, service rates are used. For interdistrict service rates, average service times to serve each set of atoms is used as it is done for the core regions.

When 3^n AHQM is solved for the very final compound region, we end up with the expected loss rates of each atom from the results of this queueing model. The loss rate of each atom can be seen in Fig. C.22. The total loss rate can be found by summing up these values which is equal to 4.854 requests/hour in this specific scenario. The loss rate we have calculated would be 6.478 requests/hour if we did not do all these steps of interactions.

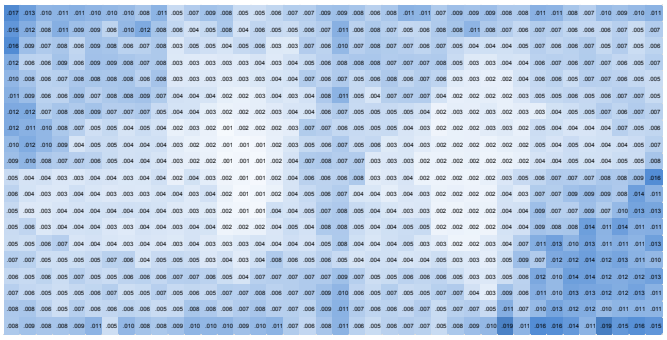


Figure C.22: Expected loss rates of each atom after applying MHQA to the whole problem area.

References

R. Alanis, A. Ingolfsson, and B. Kolfal. A Markov chain model for an EMS system with repositioning. *Production and Operations Management*, 22(1): 216–231, 2013. http://dx.doi.org/10.1111/j.1937-5956.2012.01362.x.

A. A. Aly and J. A. White. Probabilistic formulation of the emergency service location problem. *Journal of the Operational Research Society*, 29(12): 1167–1179, 1978. http://dx.doi.org/10.2307/3009582.

Y. An, B. Zeng, Y. Zhang, and L. Zhao. Reliable *p*-median facility location problem: Two-stage robust models and algorithms. *Transportation Research Part B: Methodological*, 64:54–72, 2014. http://dx.doi.org/10.1016/j.trb.2014.02.005.

T. Andersson and P. Värbrand. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201, 2007. http://dx.doi.org/10.1057/palgrave.jors.2602174.

J. Atkinson, I. 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. http://dx.doi.org/10.1016/j.ejor.2007.08.014.

F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991. http://dx.doi.org/10.1145/116873.116880.

M. O. Ball and F. L. Lin. A reliability model applied to emergency service vehicle location. *Operations Research*, 41(1):18–36, 1993. http://dx.doi.org/10.1287/opre.41.1.18.

L. Brotcorne, G. Laporte, and F. Semet. Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451–463, 2003. http://dx.doi.org/10.1016/S0377-2217(02)00364-8.

S. Budge, A. Ingolfsson, and E. Erkut. Approximating vehicle dispatch probabilities for emergency service systems with location-specific service times and multiple units per location. *Operations Research*, 57(1):251–255, 2009. http://dx.doi.org/10.1287/opre.1080.0591.

A. B. Calvo and D. H. Marks. Location of health care facilities: An analytical approach. *Socio-Economic Planning Sciences*, 7(5):407–422, 1973. http://dx.doi.org/10.1016/0038-0121(73)90039-6.

R. Church and C. S. ReVelle. The maximal covering location problem. *Papers in Regional Science*, 32(1):101–118, 1974. http://dx.doi.org/10.1111/j.1435-5597.1974.tb00902.x.

L. Cooper. Heuristic methods for location-allocation problems. *SIAM Review*, 6(1):37–53, 1964. http://dx.doi.org/10.1137/1006005.

L. Cooper. Location-allocation problems. *Operations Research*, 11(3):331–343, 1963. http://dx.doi.org/10.1287/opre.11.3.331.

L. Cooper. The transportation-location problem. *Operations Research*, 20(1): 94–108, 1972. http://dx.doi.org/10.1287/opre.20.1.94.

T. Cui, Y. Ouyang, and Z.-J. M. Shen. Reliability facility location design under the risk of disruptions. *Operations Research*, 58(4):998–1011, 2010. http://dx.doi.org/10.1287/opre.1090.0801.

M. Daskin. A maximum expected covering location model: Formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983. http://dx.doi.org/10.1287/trsc.17.1.48.

M. Daskin and E. Stern. A hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, 15(2): 137–152, 1981. http://dx.doi.org/10.1287/trsc.15.2.137.

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(5):525–549, 2008. http://dx.doi.org/10.1111/j.1475-3995.2008.00654.x.

M. Gendreau, G. Laporte, and F. Semet. Solving an ambulance location model by tabu search. *Location Science*, 5(2):75–88, 1997. http://dx.doi.org/10.1016/S0966-8349(97)00015-6.

M. Gendreau, G. Laporte, and F. Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27 (12):1641–1653, 2001. http://dx.doi.org/10.1016/S0167-8191(01)00103-X.

M. Gendreau, G. Laporte, and F. Semet. The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society*, 57(1):22–28, 2006. http://dx.doi.org/10.1057/palgrave.jors.2601991.

N. Geroliminis, M. 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. http://dx.doi.org/10.1016/j.trb.2009.01.006.

N. Geroliminis, K. Kepaptsoglou, and M. Karlaftis. A hybrid hypercube - genetic algorithm approach for deploying many emergency response mobile units in an urban network. *European Journal of Operational Research*, 210 (2):287–300, 2011. http://dx.doi.org/10.1016/j.ejor.2010.08.031.

S. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964. http://dx.doi.org/10.1287/opre.12.3.450.

T. S. Hale and C. R. Moberg. Location science research: A review. *Annals of Operations Research*, 123(1-4):21–35, 2003. http://dx.doi.org/10.1023/A:1026110926707.

K. Hogan and C. Revelle. Concepts and applications of backup coverage. *Management Science*, 32(11):1434–1444, 1986. http://dx.doi.org/10.1287/mnsc.32.11.1434.

A. P. Iannoni and R. Morabito. A multiple dispatch and partial backup hypercube queuing model to analyze emergency medical systems on highways. *Transportation Research Part E: Logistics and Transportation Review*, 43 (6):755–771, 2007. http://dx.doi.org/10.1016/j.tre.2006.05.005. Challenges of Emergency Logistics Management.

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. http://dx.doi.org/10.1016/j.ejor.2008.02.003.

A. P. Iannoni, R. Morabito, and C. Saydam. A hypercube queueing model embedded into a genetic algorithm for ambulance deployment on highways. *Annals of Operations Research*, 157(1):207–224, 2008. http://dx.doi.org/10.1007/s10479-007-0195-z.

A. P. Iannoni, R. Morabito, and C. Saydam. Optimizing large-scale emergency medical system operations on highways using the hypercube queuing model. *Socio-Economic Planning Sciences*, 45(3):105–117, 2011. http://dx.doi.org/10.1016/j.seps.2010.11.001.

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

http://dx.doi.org/10.1287/mnsc.31.2.235.

M. Karlaftis, K. Kepaptsoglou, A. Stathopoulos, M. K. Jha, D. J. Lovell, and E. Kim. Genetic algorithm-based approach for optimal location of transit repair vehicles on a large urban network. *Transportation Research Record: Journal of the Transportation Research Board*, 1879(1):41–50, 2004. http://dx.doi.org/10.3141/1879-06.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. http://dx.doi.org/10.1126/science.220.4598.671.

P. Kolesar and W. E. Walker. An algorithm for the dynamic relocation of fire companies. *Operations Research*, 22(2):249–274, 1974. http://dx.doi.org/10.1287/opre.22.2.249.

G. Laporte, V. Louveaux, François, F. Semet, and A. Thirion. Application of the double standard model for ambulance location. In J. A. Nunen, M. G. Speranza, and L. Bertazzi, editors, *Innovations in Distribution Logistics*, volume 619 of *Lecture Notes in Economics and Mathematical Systems*, pages 235–249. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-92943-7. http://dx.doi.org/10.1007/978-3-540-92944-4_12.

R. Larson. Approximating the performance of urban emergency service systems. *Operations Research*, 23(5):845–868, 1975. http://dx.doi.org/10.1287/opre.23.5.845.

R. Larson and A. Odoni. *Urban Operations Research*. Prentice-Hall, Englewood Cliffs, N.J., 1981.

R. C. Larson. A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research*, 1(1):67 – 95, 1974. http://dx.doi.org/10.1016/0305-0548(74)90076-8.

R. C. Larson and M. A. McKnew. Police patrol-initiated activities within a systems queueing model. *Management Science*, 28(7):759–774, 1982. http://dx.doi.org/10.1287/mnsc.28.7.759.

X. Li and Y. Ouyang. A continuum approximation approach to reliable facility location design under correlated probabilistic disruptions. *Transportation Research Part B: Methodological*, 44(4):535–548, 2010. http://dx.doi.org/10.1016/j.trb.2009.09.004.

V. Marianov and C. ReVelle. The queueing maximal availability location problem: A model for the siting of emergency vehicles. *European Journal of Operational Research*, 93(1):110–120, 1996. http://dx.doi.org/10.1016/0377-2217(95)00182-4.

V. Marianov and C. ReVelle. The capacitated standard response fire protection siting problem: Deterministic and probabilistic models. *Annals of Operations Research*, 40(1):303–322, 1992. http://dx.doi.org/10.1007/BF02060484.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997. http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

R. Morabito, F. Chiyoshi, and R. D. Galvão. Non-homogeneous servers in emergency medical systems: Practical applications using the hypercube queueing model. *Socio-Economic Planning Sciences*, 42(4):255–270, 2008. http://dx.doi.org/10.1016/j.seps.2007.04.002.

A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations Concepts and Applications of Voronoi Diagrams*. John Wiley, 2nd edition, 2000.

S. H. Owen and M. S. Daskin. Strategic facility location: A review. *European Journal of Operational Research*, 111(3):423–447, 1998. http://dx.doi.org/10.1016/S0377-2217(98)00186-6.

L. Qi, Z.-J. M. ShenShen, and L. V. Snyder. The effect of supply disruptions on supply chain design decisions. *Transportation Science*, 44(2):274–289, 2010. http://dx.doi.org/10.1287/trsc.1100.0320.

H. K. Rajagopalan, C. Saydam, and J. Xiao. A multiperiod set covering location model for dynamic redeployment of ambulances. *Computers & Operations Research*, 35(3):814–826, 2008. http://dx.doi.org/10.1016/j.cor.2006.04.003.

C. S. ReVelle and K. Hogan. The maximum availability location problem. *Transportation Science*, 23(3):192–200, 1989. http://dx.doi.org/10.1287/trsc.23.3.192.

D. Schilling, D. J. Elzinga, J. Cohon, R. Church, and C. ReVelle. The team/fleet models for simultaneous facility and equipment siting. *Transportation Science*, 13(2):163–175, 1979. http://dx.doi.org/10.1287/trsc.13.2.163.

V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621, 2012. http://dx.doi.org/10.1016/j.ejor.2011.10.043.

V. Schmid and K. F. Doerner. Ambulance location and relocation problems with time-dependent travel times. *European Journal of Operational Research*, 207(3):1293–1303, 2010. http://dx.doi.org/10.1016/j.ejor.2010.06.033.

R. A. Takeda, J. A. Widmer, and R. Morabito. Analysis of ambulance decentralization in an urban emergency medical service using the hypercube queueing model. *Computers & Operations Research*, 34(3):727–741, 2007. http://dx.doi.org/10.1016/j.cor.2005.03.022.

C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971. http://dx.doi.org/10.1287/opre.19.6.1363.

X. Wang and Y. Ouyang. A continuum approximation approach to competitive facility location design under facility disruption risks. *Transportation Research Part B: Methodological*, 50:90–103, 2013. http://dx.doi.org/10.1016/j.trb.2012.12.004.

J. R. Weaver and R. L. Church. A median location model with nonclosest facility service. *Transportation Science*, 19(1):58–74, 1985. http://dx.doi.org/10.1287/trsc.19.1.58.

A. Weber. *Über den standort der industrien*. JCB Mohr, 1909.

E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Math. J*, 43(2):355–386, 1937.