An Efficient Simulation-based Approach to Ambulance Fleet Allocation and Dynamic Redeployment

Yisong Yue and **Lavanya Marla** and **Ramayya Krishnan**
iLab, H. John Heinz III College
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
{yisongyue, lavanyamarla, rk2x}@cmu.edu

Abstract

We present an efficient approach to ambulance fleet allocation and dynamic redeployment, where the goal is to position an entire fleet of ambulances to base locations to maximize the service level (or utility) of the Emergency Medical Services (EMS) system. We take a simulation-based approach, where the utility of an allocation is measured by directly simulating emergency requests. In both the static and dynamic settings, this modeling approach leads to an exponentially large action space (with respect to the number of ambulances). Futhermore, the utility of any particular allocation can only be measured via a seemingly "black box" simulator. Despite this complexity, we show that embedding our simulator within a simple and efficient greedy allocation algorithm produces good solutions. We derive data-driven performance guarantees which yield small optimality gap. Given its efficiency, we can repeatedly employ this approach in real-time for dynamic repositioning. We conduct simulation experiments based on real usage data of an EMS system from a large Asian city, and demonstrate significant improvement in the system's service levels using static allocations and redeployment policies discovered by our approach.

1 Introduction

Emergency Medical Services (EMS) comprise an important component of public services, and involve allocation of scarce resources as critical events occur. The life and death nature of the events involved and the generally limited availability of resources necessitate the use of sophisticated analytical tools to design and operate EMS systems. Within this domain, one important resource allocation challenge is how to allocate and dynamically reposition ambulances to best serve a given set of requests.

Due to their computational convenience, conventional approaches typically employ relatively simple models that do not fully capture the dynamics of how particular ambulance allocations provide service to some distribution of emergency requests (Larson and Stevenson 1972; Brotcorne, Laporte, and Semet 2003; Budge, Ingolfsson, and Erkut 2007; Erkut, Ingolfsson, and Erdogan 2008) or capture them approximately (Restrepo, Henderson, and Topaloglu 2009). For example, how any given request is served depends on which ambulances are available when that request arrives – this can give rise to congestion and other interdependencies. Such dependencies typically can only be captured via simulation, which consequently are often used for final evaluation (Restrepo, Henderson, and Topaloglu 2009).

In this paper, we propose an optimization approach that directly incorporates a simulator as a subroutine. By leveraging structural properties, we show that, for a large class of fitness measures,[1] the problem can be efficiently solved to near-optimality using a simple and efficient greedy approach. Furthermore, due its efficiency, we can repeatedly apply our approach in real-time for dynamic repositioning. This allows us to employ a two-tiered optimization approach that combines static allocation with real-time repositioning to deal with congestion and other "hotspots" that arrive.

We applied our approach to an EMS provider in Asia, where such a system is being established to meet emergent demand for service. The setting is highly resource-constrained due to the relatively small number of ambulances. We demonstrate that, by directly leveraging a simulator for optimization, our approach can significantly improve the performance of the system along several metrics.

2 Related Work

The study of vehicle allocation (or deployment) for Emergency Medical Services (EMS) enjoys a rich history. The general setting can be described as computing an allocation (and possibly a reallocation strategy) for a set of ambulances such that some measure of fitness of the system is optimized. As such, the two main technical challenges that arise are that of system modeling and allocation optimization.

From the perspective of this work, the end goal of system modeling is to accurately measure or evaluate the fitness of the system. Conventional methods typically employ mathematical models (cf. (Larson and Stevenson 1972; Brotcorne, Laporte, and Semet 2003; Gendreau, Laporte, and Semet 2005; Andersson and Värbrand 2006; Restrepo, Henderson, and Topaloglu 2009)), and the desired fitness measure can then be computed via fitting the model's parameters to a particular system. Afterwards, optimization

[1] E.g., the number of requests served within 15 minutes.

or local search techniques are employed to identify a well-performing allocation or set of allocations.

However, such mathematical programming approaches often fail to fully characterize the dynamics of ambulance dispatch and emergency response in general. In particular, they fail to capture features such as time-dependent travel times, congestion patterns and high variability in travel time to hospital specified by patient. To that end, simulation-based evaluation is often used.

Indeed, many recent modeling approaches employed simulations for final evaluation (Restrepo, Henderson, and Topaloglu 2009; Maxwell et al. 2010). This implicit preference for simulation-based evaluation as the evaluation method of choice motivates directly optimizing via simulation during ambulance allocation.

Two natural areas for resource allocation are in allocation of ambulances to bases, and dispatching of ambulances to requests. We restrict ourselves to the former, since it typically offers greater gains (which we observe empirically in our experiments). In contrast to (Andersson and Värbrand 2006; Bjarnason et al. 2009) which optimizes over both allocations and dispatching, our problem setting and algorithmic approach yield provable aposteriori guarantees on optimality. In contrast to (Maxwell et al. 2010), we focus on allocating and repositioning entire fleets of ambulances rather than individual ambulances.

Our approach is closely related to a line of coverage-based optimization approaches known as submodular optimization (Nemhauser, Wolsey, and Fisher 1978; Feige 1998; Khuller, Moss, and Naor 1999; Leskovec et al. 2007). Applications include optimizing information gain in sensor placements (Leskovec et al. 2007; Krause, Singh, and Guestrin 2008; Streeter, Golovin, and Krause 2009), conservation planning (Golovin et al. 2011), maximizing cascading behavior in viral marketing (Kempe and Kleinberg 2003), and maximizing information coverage in content recommendation (El-Arini et al. 2009; Yue and Guestrin 2011). Submodular optimization is an attractive formalism due to greedy forward selection achieving an $(1 - 1/e)$ approximation guarantee.

Our fitness function is essentially a notion of coverage over a distribution of requests. We show that our optimization objective is "close" to submodular in a data-dependent way, which motivates a greedy optimization approach. We then leverage this property to derive data-dependent bounds on the optimality gap. Furthermore, we show how to compute even tighter data-dependent bounds by more directly leveraging the structure of our problem setting. Thus, unlike many simulation-based multi-resource allocation problems that are provably hard (e.g., (Sheldon et al. 2010)), our setting lies within a regime where worst case instances that admit no approximation guarantee can be shown to not arise through a posteriori analysis.

3 Data-driven Simulation

We first present a data-driven simulation approach for evaluating ambulance allocations. After simulating how a call center assigns ambulances to a sequence of emergency requests, we can then measure any evaluation metric of interest (e.g., the number of requests serviced within 15 minutes).

Algorithm 1 SIMULATOR: Data-driven Simulator Method

```
1: input: (R, A, π, t_d), DISPATCH
2: W ← A    //keeps track of which ambulances are free
3: R̂ ← ∅    //keeps track of active requests
4: initialize Y = {y_r}_{r∈R} such that y_r ← ⊥
5: initialize events E ← R sorted in arrival order
6: insert redeployment events spaced every t_d minutes to E.
7: while |E| > 0 do
8:    remove next arriving event e from E
9:    if e = new request r then
10:      y_r ← DISPATCH(r, W, R)    //dispatch policy
11:      if y_r ≠ ⊥ then
12:         R̂ ← R̂ + r(y_r)    //updating active requests
13:         W ← W − y_r    //updating free ambulances
14:         insert job completion event at time t̄_r(y_r) into E
15:      end if
16:   else if e = job completion event t̄_r(y_r) then
17:      R̂ ← R̂ − r(y_r)    //updating active requests
18:      W ← W + y_r    //updating free ambulances
19:   else if e = redeployment event then
20:      W ← π(W, R̂)    //redeploying free ambulances
21:   end if
22: end while
23: return: Processed assignments of ambulances to requests Y
```

Whenever a new emergency request arrives, the dispatch officer assigns the best available ambulance to service the request (or none if no good ambulances are available). We define the following inputs to our simulator:

- A request log $R = \{r_1, \ldots, r_N\}$.
- An allocation A of ambulances to bases.
- A redeployment policy $\pi(W, \hat{R})$ that takes as input the current allocation of free ambulances A and the currently active requests $\hat{R}$, and outputs a repositiong of A to potentially different base locations.
- A dispatch policy, DISPATCH, which emulates how ambulances are dispatched to service emergency requests. In our experiments, we employ the "closest available ambulance" myopic dispatch policy used by the EMS operator.

We assume that ambulances are exchangeable or identical up to base allocation. We use y_r to denote the base of the ambulance dispatched to service request r (or $\perp$ if no ambulance was dispatched). Let $r(y_r)$ denote a request r to which ambulance y_r is dispatched, and $\bar{t}_r(y_r)$ denote the corresponding completion time of r. Our event-driven simulator processes emergency requests in first-come first-served order as they arrive, and is described in Algorithm 1. There are three types of events that are processed:

- Request arrival (Lines 10-15). The new request r is processed using DISPATCH. If an ambulance is assigned ($y_r \neq \perp$), then three book-keeping steps are performed: r is added to the set of active requests with assignment y_r (Line 12), the assigned ambulance is removed from the set of free ambulances (line 13), and a request job completion event is added to the event queue (Line 14).
- Request job completion (Lines 17-18). A currently active request r has been completed. Two book-keeping steps

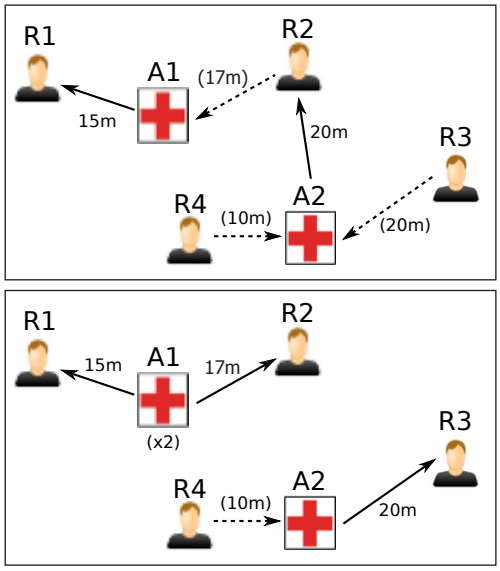

Figure 1: Examples of simulated dispatch of four requests and two bases. The top example has one ambulance allocated per base. The bottom example has two ambulances allocationed to A_1 and one to A_2.

are performed: r is removed from the set of active requests (Line 17), and the assigned ambulance is added to the set of available ambulances (Line 18).

- Ambulance redeployment (line 20). All available ambulances are considered for redeployment by the redeployment policy π. Note that these events are only used when simulating dynamic redeployment policies.

Figure 1 describes two examples with four requests and two base locations. Requests $R_1, \ldots, R_4$ arrive in index order. Figure 1 top shows dispatch behavior with one ambulance allocated per base. R_1 arrives first, and is assigned its closest ambulance (from A_1). When R_2 arrives, the dispatcher assigns it its closest free ambulance (from A_2). When R_3 and R_4 arrive, all nearby ambulances are busy, so both requests are not serviced. Figure 1 bottom shows a similar example with two ambulances allocated to A_1.

3.1 Sampling Requests

In order to evaluate ambulance allocations and redeployment policies, we also require a sample of emergency requests R. In this paper, we assume requests are sampled from a generative model $\mathbf{P}(R)$ (with parameters estimated from historical data). We begin by stating an independence assumption.

Assumption 1. *Sampling a request depends only on exogenous factors (e.g., location, time of day, road conditions), and is independent of other requests and the allocation and dispatch behavior of the EMS.*[2]

[2]E.g., poor road conditions can increase the probability of accidents, but each accident event is sampled independently. This assumption is not suitable for modeling cascading behavior such as epidemics, since modeling how epidemics spread can depend on previous incidents as well as the service level of the EMS.

Algorithm 2 SAMPLER: Request Sampler

1: **input**: t_{start}, t_{end}
2: $R \leftarrow \emptyset,\ t \leftarrow t_{start}$
3: **while** $t < t_{end}$ **do**
4: Sample $r \leftarrow P(r|t)$ *//sampling request starting at time t*
5: $t \leftarrow t_r$ *//incrementing current time*
6: $R \leftarrow R \cup \{r\}$ *//adding sampled request to collection*
7: **end while**
8: **return**: R

This independence between request arrival patterns and EMS behavior will be important when developing our optimization approach and analysis. When simulating dispatch behavior (and evaluating EMS service levels), we can first pre-sample a call log R (e.g., one week's worth), and then evaluate service quality of an allocation A by running SIMULATOR described in Algorithm 1. On the other hand, if request arrival depends on the EMS allocation and dispatch behavior (and thus violates Assumption 1), then one can no longer pre-sample future calls for simulation.

Assumption 1 also implies that R can be incrementally sampled according to a memory-less stochastic process.[3] Thus, for any time interval (e.g., one week), we can incrementally sample requests using Algorithm 2. Here, $P(r|t)$ denotes the distribution of the next arriving request starting at time t, and t_r denotes the arrival time of request r.

4 Static Allocation Problem Formulation

We begin by describing the static allocation setting, which we later build upon for the dynamic setting. Let A denote an allocation of ambulances to bases $\mathcal{A}$ (there can be more than one ambulance at a base). We represent A as a multiset of elements in $\mathcal{A}$. Let $M(\mathcal{A})$ denote the multi-powerset of $\mathcal{A}$. We measure the utility of an allocation A using a real-valued objective $F : M(\mathcal{A}) \rightarrow \Re$. We focus on penalty reduction formulations, where we can write $F(A)$ as

$$F(A) = L(\emptyset) - L(A), \tag{1}$$

where $L : M(\mathcal{A}) \rightarrow \Re$ measures the penalty (or cost) of an ambulance allocation over some period of time (e.g., one week). For example, $L(A)$ may correspond to the fraction of requests whose service time is above some target threshold, and is a tunable component of the framework. In that case, $F(A)$ would then correspond to the reduction of such requests after adding A to the empty allocation $\emptyset$.

We define L using the outcomes of simulated requests. Ideally, our goal is to maximize the expected gain in performance over some (known) distribution of requests $\mathbf{P}(R)$. Let $Y = \{y_r\}_{r \in R}$ denote the output of Algorithm 1 for request log R. Then we can write the expected penalty as

$$L(A) = \mathbf{E}_{R \sim \mathbf{P}(R)} \left[\sum_{r \in R} L_r(y_{r,A}) \right], \tag{2}$$

where $L_r(y)$ is the penalty of assigning request r with y (e.g., whether or not assigning ambulance y to r results in a

[3]In our experiments, we use a Poisson process (Ross 1983).

Algorithm 3 Greedy Ambulance Allocation

input: F, K
$A \leftarrow \emptyset$
for $\ell = 1, \ldots, K$ **do**
  $\hat{a} \leftarrow \text{argmax}_a \, \delta_F(a|A)$ *//see* (5)
  $A \leftarrow A + \hat{a}$
end for
return: A

Table 1: Comparing incremental gain when adding an ambulance to A_2 w.r.t. the two allocations shown in Figure 1. $F(A)$ is defined as the #requests serviced within 15 minutes. The left and right sides of the table below correspond to the top and bottom allocations, respectively, of Figure 1. Submodularity is violated since the gain on the left side is smaller than the gain on the right side.

A_1	A_2	$F(A)$	A_1	A_2	$F(A)$
1	1	1	2	1	1
1	2	1	2	2	2
Gain		0	Gain		1

service time above a target threshold), and $y_{r,A}$ denotes the assignment that the simulator (Algorithm 1) gives to r.

In practice, we resort to optimizing over a collection of request logs $\mathcal{R} = \{R_m\}_{m=1}^{M}$, where each $R_m \in \mathcal{R}$ is sampled i.i.d according to $\mathbf{P}(R)$. We thus approximate the expectation with the sampled average,[4]

$$L_{\mathcal{R}}(A) = \frac{1}{M} \sum_{m=1}^{M} \sum_{r \in R_m} L_r(y_{r,A}). \tag{3}$$

Given a budget of K ambulances, the static allocation goal then is to select the ambulance allocation A (with $|A| \leq K$) that has maximal utility $F(A)$. More formally, we can write our optimization problem as

$$\underset{A \in M(\mathcal{A}): |A| \leq K}{\text{argmax}} \quad F(A). \tag{4}$$

5 Greedy Allocation Algorithm and Analysis

At first glance, it may seem difficult to compute good solutions to (4), since $M(\mathcal{A})$ is exponentially large in $|\mathcal{A}|$, and $F(A)$ is evaluated using a simulator that appears complicated to analyze. Nonetheless, we show that a simple greedy algorithm can compute provably good solutions.

Let $\delta_F(a|A)$ denote the gain of adding a to A,

$$\delta_F(a|A) = F(A \cup a) - F(A). \tag{5}$$

The greedy algorithm is described in Algorithm 3, and can be used to solve both the static allocation problem (4) and the redeployment problem (11) to be defined later. The algorithm iteratively selects the ambulance a that has maximal incremental gain relative to the current solution until K ambulances have been allocated. Note that each evaluation of $\delta(a|A)$ requires running the simulator to evaluate $F(A+a)$.

[4]In our experiments, we use Sample Average Approximation (Verweij et al. 2003) to bound the difference between our sample average objective and the optimal expected performance.

5.1 Non-submodularity

Optimization problems that are well-solved by greedy algorithms are often *submodular*. Formally, a set function $F(A)$ is submodular if and only if

$$\forall A \subseteq B, \, \forall a: \; \delta_F(a|A) \geq \delta_F(a|B).$$

When F is monotone and submodular, it is known that the greedy algorithm returns a solution that achieves $F(A) \geq (1 - 1/e)OPT$ (Nemhauser, Wolsey, and Fisher 1978).

In our setting, this notion of diminishing returns can be interpreted as "the gain of adding an ambulance decreases with larger allocations." Unfortunately, one can show that our setting is not submodular.

Consider the two examples in Figure 1. Table 1 left and right show the incremental gains of adding an ambulance to A_2 with respect to the top and bottom allocations, respectively, of Figure 1. Notice that the first allocation is a subset of the second allocation. Submodularity is violated since the incremental gain with respect to the second allocation is greater than the first. This is due to request R_4 only being serviced in the allocation $(A_1 = 2, A_2 = 2)$. In fact, one can construct pathological cases with arbitrarily bad submodularity (and monotonicity) violations.

We will present a method for measuring how close F is to monotone submodular. We then derive data-driven bounds on the optimality gap. Our experiments show this gap for Algorithm 3 to be reasonably tight.

5.2 Submodular Upper Bound

In order to analyze our objective F, (1), we first consider the behavior of a simulator using an onmiscient dispatch policy. In particular, for any sequence of requests R and allocation A, the omniscient dispatcher computes an assignment Y_A^* with minimal penalty

$$\bar{L}_R(A) = \sum_{r \in R} L_r(y^*_{r,A}) = \min_{Y_A} \sum_{r \in R} L_r(y_{r,A}).$$

We can now define a new objective $G_R(A)$ based on omniscient dispatching,

$$G_R(A) = \bar{L}_R(\emptyset) - \bar{L}_R(A). \tag{6}$$

Figure 2 shows how one can compute $G(A)$ by solving a relatively simple integer linear program. Note that Assumption 1 allows us to pre-sample the requests R and then compute $G(A)$ knowing all of R in advance.

Observation 1. *The objective G,* (6)*, as measured by simulating an omniscient dispatcher, is monotone submodular. Furthermore, for any A and R, we have $G_R(A) \geq F_R(A)$.*

Simulating with an onmiscient dispatcher yields an idealized objective G that is both monotone submodular and a rigorous upper bound on F (which results from simulating the myopic dispatcher). Define

$$OPT_R(K) = \max_{A: |A| \leq K} F_R(A),$$

and define $OPT_R^G(K)$ analagously for G. By leveraging properties of monotone submodular functions (Leskovec et al. 2007), we arrive at a data-dependent bound on the optimality gap of any allocation A, $OPT_R(K) - F_R(A)$.

Computing Omniscient Dispatching Utility:

$$G(A) = \bar{L}(\emptyset) - \min_{\mathbf{x}} \sum_{i \in R} \sum_{s \in Q_i} x_{is} L_{is} \tag{8}$$

s.t.

$$\sum_{s \in Q_i} x_{is} = 1, \qquad \forall i \in R$$

$$x_{is} + \sum_{j \in P_i^s} x_{js} \leq a_s, \qquad \forall i \in R$$

Notation:

- R: the request set .
- $\perp$: denotes the null assignment for a request.
- Q_i: the set of feasible bases for request i (including $\perp$).
- x_{is}: (binary variable) defined for $s \in Q_i$ and $i \in R$, is 1 iff assignment to i is from base s (s can be $\perp$).
- P_i^s: set of parents of request i via base s ($j \in P_i^s$ iff j arrives before i, j completes after i arrives, and s is in both Q_j and Q_i).
- L_{is}: the cost of assigning ambulance from base s to request i.
- a_s: #ambulances allocated to base s ($A = \{a_1, \ldots, a_{|\mathcal{A}|}\}$).

Figure 2: ILP formulation for computing utility of a static allocation using the omniscient dispatcher.

Theorem 1. *For any allocation A with $|A| = K$ and request log R, define*

$$\delta_a = G_R(A + a) - G_R(A).$$

Then

$$OPT_R(K) \leq F_R(A) + (G_R(A) - F_R(A)) + K \max_a \delta_a. \tag{7}$$

Proof. Applying Theorem 4 in (Leskovec et al. 2007) yields

$$\max_{A' : |A'| = K} G_R(A') \leq G_R(A) + K \max_a \delta_a.$$

Hence,

$$\begin{aligned} OPT_R(K) &= \max_{A' : |A'| \leq K} F_R(A') \\ &\leq \max_{A' : |A'| \leq K} G_R(A') \\ &\leq G_R(A) + K \max_a \delta_a \\ &= F_R(A) + (G_R(A) - F_R(A)) + K \max_a \delta_a \end{aligned}$$

□

Upon termination of Algorithm 3, we can upper bound OPT via (7). Note that the δ_a in Theorem 1 are defined with respect to $G(A)$ and not $F(A)$. Thus $G(A) - F(A)$ is a measure of how close F is to monotone submodular. Note also that if $G(A) - F(A)$ is small, then one would expect little to be gained from smarter dispatching (since $G(A)$ is the value of the omniscient dispatcher).

5.3 Omniscient-Optimal Upper Bound

Rather than appealing to submodularity, we can compute an even tighter bound on the optimality gap by leveraging the structure of G directly. In particular, we can extend the ILP formulation of Figure 2 to optimally solve G,

$$OPT_R^G(K) = \max_{A \in M(\mathcal{A}) : |A| \leq K} G_R(A). \tag{9}$$

Theorem 2. *For any A with $|A| = K$,*

$$F_R(A) \leq OPT_R(K) \leq OPT_R^G(K).$$

Proof. Define $A_F^* = \operatorname{argmax}_{A \in M(\mathcal{A}) : |A| \leq K} F_R(A)$. The result follows immediately from

$$OPT_R(K) = F_R(A_F^*) \leq G_R(A_F^*) \leq OPT_R^G(K).$$

□

6 Dynamic Redeployment

In the dynamic setting, free ambulances can be redeployed to nearby bases to improve service levels under changing conditions. This is motivated by the intuition that a temporary repositioning of available ambulances offers better (expected) service to incoming requests until the busy ambulances are freed.

We consider redeployment at regular intervals t_d (e.g., 30 minutes). Let s_t denote the state of the system at time step t. The utility of redeployment policy π on requests R is

$$F_R(\pi) = \mathbf{E}\left[\sum_{t=1}^{T} F(\pi(s_t)|s_t)\right], \tag{10}$$

where the expectation is over the randomness of π. The expected utility of π is then

$$F(\pi) = \mathbf{E}_{R \sim \mathbf{P}(R)}\left[F_R(\pi)\right],$$

or its sample mean approximation.

Due to randomness, requests can cluster in time and space, which results in patterns of congestion (e.g., Figure 1). Due to Assumption 1, the distribution of future requests do not depend on the past. As a consequence, the *main benefit* of redeployment is to better cover the distribution of requests $\mathbf{P}(R)$ until the busy ambulances become free again.

Since ambulances take time to redeploy to another base, we incorporate this cost into the service time of dispatching a redeploying ambulance. In particular, dispatching a redeploying ambulance to a request incurs an additional service time equal to the redeploy travel time remaining. We incorporate the additional service time of redeploying ambulances into $F_{\hat{R}_t}(\cdot|s_t)$.

6.1 Myopic Redeployment

We consider myopically redeploying ambulances to optimize the expected utility of the next time interval. Assumption 1 implies that myopic redeployment is essentially equivalent to (a smaller version of) the static allocation problem. By sampling requests $\hat{R}_t$ from $\mathbf{P}(R)$ up to t_d into the future (e.g., using Algorithm 2), as well as sampling when currently busy ambulances become free, the myopic redeployment problem is

$$\pi(t) = \operatorname*{argmax}_{A \in M(\mathcal{A}, W_{s_t}), |A| \leq |W_{s_t}|} F_{\hat{R}_t}(A|s_t), \tag{11}$$

where s where denotes the current state, and W_{s_t} denotes the currently free allocation, and $M(\mathcal{A}, W_{s_t})$ denotes the set of possible redeployments. Like in the static setting, we solve (11) approximately using a greedy approach (Algorithm 3).

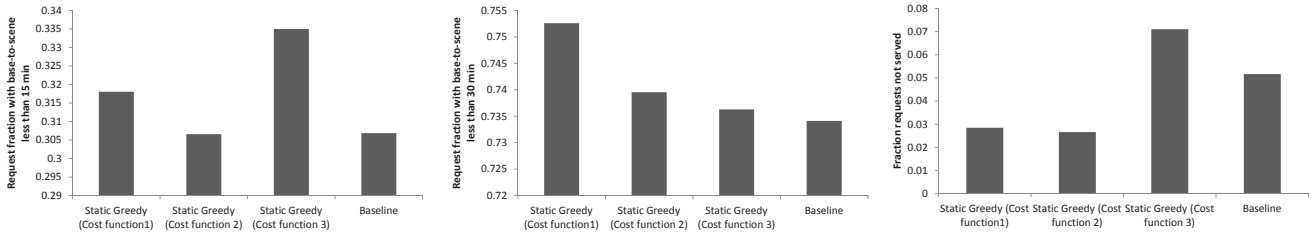

Figure 3: Comparing static allocation solutions to uniform baseline on the test set ($K = 58$).

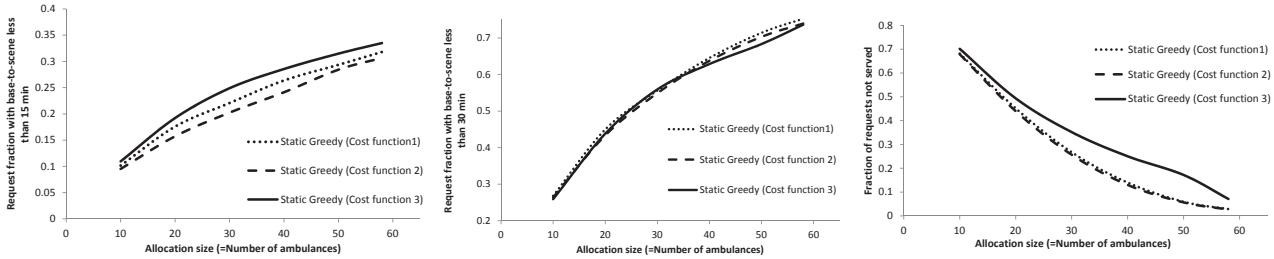

Figure 4: Comparing static allocation solutions on test set as allocation budget is varied.

6.2 Discussion on Dynamic Upper Bounds

Similar to the static upper bound (9), in principle, one can further extend the ILP of Figure 2 to optimally solve the *omniscient* dynamic redeployment problem for a set of requests R. In other words, an omniscient algorithm with perfect knowledge of future requests (which can be sampled exogenously due to Assumption 1) can jointly optimize redeployment and dispatch. We defer a detailed evaluation of dynamic upper bounds to future work.

7 Experiments

We conducted simulation experiments derived using real usage data from an EMS system in a large Asian city. The usage data contained approximately ten thousand logged emergency requests over the course of one month. Each logged request contains the type and location of the request, the ambulance (if any) that was dispatched, and the various travel times (e.g., base to scene, scene to hospital, etc). The request arrivals and service times both fit into a Poisson distribution per zone and base, respectively. Request arrivals and service times all appear statistically independent, lending support for Assumption 1. Using the parameters of the fitted Poisson distributions, we built a generative Poisson process model $\mathbf{P}(R)$ (Ross 1983) for sampling emergency requests.[5]

Our action space contains 58 bases and 58 ambulances. We evaluate our methods over a period of one week. Rather than using Algorithm 3, we employ a lazy variant (called CELF in (Leskovec et al. 2007)) which greatly reduces the number of function evaluations and provides almost identical solutions. Computing an allocation takes only a few seconds, easily allowing for real-time redeployment.

7.1 Sample Average Approximation (SAA)

We use Sample Average Approximation (SAA) (Verweij et al. 2003) for model selection and for bounding the deviation of our sample average (3) to (2). Furthermore, we also wish to bound the optimality gap of our greedy approach (with respect to the true distribution of requests $\mathbf{P}(R)$).

For the static allocation setting, we compute M allocations $A_1, \ldots, A_M$ using M collections of N_{train} sets of requests. We select the allocation $\hat{A}$ with the highest validation performance and on separate collection of N_{valid} sets of requests. Finally, we report the test performance of $\hat{A}$ on another separate collection of N_{test} sets of requests.

A simple modification of Theorem 1 in (Mak, Morton, and Wood 1999) yields the optimality gap to be bounded by the difference between the sample average upper bounds, (7) and (9), and the test performance $F_{test}(\hat{A})$.

[5] An anonymized data sample and software are available at `projects.yisongyue.com/ambulance_allocation`.

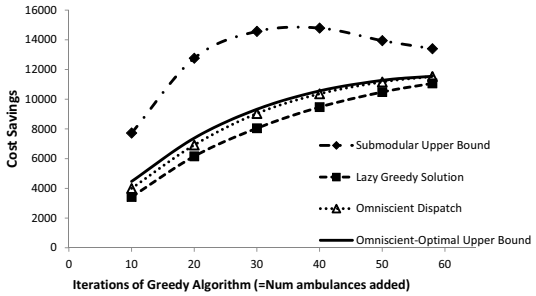

Figure 5: Comparing greedy static allocation versus data-dependent bound on OPT when optimizing for **Cost 1**.

7.2 Static Allocation Results

We consider three penalty functions in our experiments.

Cost 1.

$$L_r^{(1)}(y) = \begin{cases} 0 & \textit{if service time } \leq \textit{ 15 min} \\ 1 & \textit{if service time } \leq \textit{ 30 min} \\ 2 & \textit{if service time } \leq \textit{ 60 min} \\ 5 & \textit{otherwise} \end{cases}$$

Cost 2.

$$L_r^{(2)}(y) = \begin{cases} 20 & \textit{if not served} \\ L_r^{(1)}(y) & \textit{otherwise} \end{cases}$$

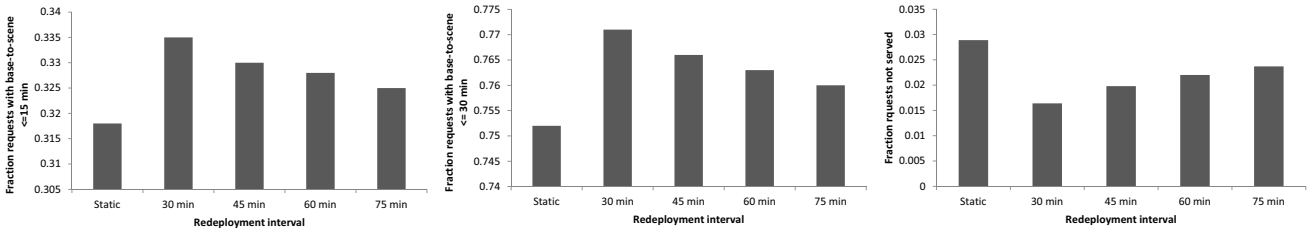

Figure 6: Comparing static allocation versus dynamic redeployment solutions on test set when optimizing for **Cost 1**.

Cost 3.

$$L_r^{(3)}(y) = \begin{cases} 0 & \textit{if service time } \leq \textit{ 15 min} \\ 1 & \textit{otherwise} \end{cases}$$

Cost 1 can be interpreted as a blend of different threshold functions. **Cost 2** is similar except that it more severely penalizes unserviced requests. **Cost 3** is a simpler threshold function that does not discriminate between requests with service time longer than 15 minutes. We performed SAA to select the best allocation A for each penalty function using $M = 50$ samples of $N_{train} = 10$ request logs, $N_{valid} = 500$ validation logs, and $N_{test} = 500$ test logs.

Figure 3 shows the comparison with the default EMS allocation (one ambulance per base) for three metrics of interest. The choice of penalty function affects which metrics are improved. Not surprisingly, optimizing for **Cost 3** achieves the best performance for requests serviced within 15 minutes, albeit with an increase in requests not serviced at all. Optimizing for **Cost 2** achieves the greatest reduction in requests not serviced, but does not greatly improve the other metrics. Optimizing for **Cost 1** offers a compromise between **Cost 2** and **Cost 3** and significant improves upon the baseline allocation for all three metrics, including an almost 50% relative reduction in requests not serviced. Figure 4 shows a comparison as the budget is varied. We see that the patterns in relative performance are consistent.

Figure 5 compares our algorithm's preformance (for **Cost 1**) with the upper bounds. Note that all the upper bound guarantees are computed using the training set. We see that our objective is "close" to submodular since the gap between the myopic dispatcher and the omniscient one is small. This implies that there is little to be gained from smarter dispatching. We further observe that the omniscient-optimal upper bound is almost identical to the omniscient dispatcher objective using the greedy allocation. This implies that the optimality gap is indeed quite small.

7.3 Dynamic Redeployment Results

In our dynamic redeployment experiments, we start with the static allocation (for **Cost 1**), and then dynamically reposition ambulances every t_d minutes. Each window of t_d minutes is equivalent to a (smaller) static allocation problem. For each redeployment, we sample 100 requests logs t_d minutes into the future to use for optimization. We estimated inter-base travel times using road-network data.

Figure 6 compares the static allocation and myopic redeployment policies with varying redeployment windows. We observe significant improvements in all three metrics. For example, with a redeployment window of 30 minutes, the myopic redeployment policy achieves almost a 50% reduction in requests not serviced. Table 2 shows the average relocation rate (per hour). For all redeployment windows, only a small fraction (less than a third) of the ambulances are selected for relocation at any point in time.

Table 2: Number of relocated ambulances per hour via dynamic redeployment for varying redeployment windows.

30 min	45 min	60 min	75 min
17.8	12.0	8.7	6.6

8 Discussion

Our approach offers the flexibility of using a similar approach to both planning (static allocation) and real-time operations (redeployment), facilitating real-world implementation. Additionally, our approach offers flexibility in balancing between competing performance metrics. However, such flexibility also implies the need to tune the penalty function via simulation for different operational goals.

One potential limitation is the need for a reliable simulator (and data sampler), which typically requires a substantial amount of historical data as well as model engineering. However, simulation is typically used for evaluation in these types of settings, implying the ready availability of such simulators in practice to use for optimization.

Because our setting lies within a regime where our optimization objective is "close" to submodular, we can employ greedy algorithms to arrive at an efficient and effective algorithm for real-time redeployment. Nonetheless, it may be beneficial to develop more sophisticated algorithms that can offer further improvements, especially for objectives that are far from submodular.

Our modeling and optimization approach is applicable in other resource allocation settings such as disaster response, humanitarian service logistics and facility positioning. While the static allocation problem is well studied in such settings, significant gains are possible via efficient and effective dynamic redeployment. One caveat arises when applying our approach to other settings: we require the request (or event or job) arrival distribution to not depend on the behavior of the emergency response system (see Assumption 1). In other settings, it may be more appropriate to employ non-myopic planning approaches (e.g., for disrupting cascading disasters such as wildfires (Konoshima et al. 2008)). One possibility is embedding our approach within a Markov Decision Process framework (Maxwell et al. 2010; Lagoudakis and Parr 2003; Bertsekas and Tsitsiklis 1996).

9 Conclusion

We have presented an efficient and effective approach to ambulance fleet allocation and dynamic redeployment. In simulation experiments based on a real EMS system in Asia, our approach significantly improved several performance metrics (including over 50% reduction in requests not serviced). We further show that a simple myopic redeployment algorithm can be effective when the randomness of the future depends purely on exogenous factors.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments. This work was funded in part by gifts from industrial partners and other donors to the iLab at Heinz College, Carnegie Mellon University. Yisong Yue was also partly supported by ONR (PECASE) N000141010672 and ONR Young Investigator Program N00014-08-1-0752.

References

Andersson, T., and Värbrand, P. 2006. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society* 58(2):195–201.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Bjarnason, R.; Tadepalli, P.; Fern, A.; and Niedner, C. 2009. Simulation-based optimization of resource placement and emergency response. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

Brotcorne, L.; Laporte, G.; and Semet, F. 2003. Ambulance location and relocation models. *European journal of operational research* 147(3):451–463.

Budge, S.; Ingolfsson, A.; and Erkut, E. 2007. Approximating vehicle dispatch probabilities for emergency service systems with location-specific service times and multiple units per location. *Operations Research*.

El-Arini, K.; Veda, G.; Shahaf, D.; and Guestrin, C. 2009. Turning down the noise in the blogosphere. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Erkut, E.; Ingolfsson, A.; and Erdogan, G. 2008. Ambulance deployment for maximum survival. *Naval Research Logistics* 55:42–58.

Feige, U. 1998. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)* 45(4):634–652.

Gendreau, M.; Laporte, G.; and Semet, F. 2005. The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society* 57(1):22–28.

Golovin, D.; Krause, A.; Gardner, B.; Converse, S. J.; and Morey, S. 2011. Dynamic resource allocation in conservation planning. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Kempe, D., and Kleinberg, J. 2003. Maximizing the spread of influence through a social network. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Khuller, S.; Moss, A.; and Naor, J. 1999. The budgeted maximum coverage problem. *Information Processing Letters* 1:39–45.

Konoshima, M.; Montgomery, C.; Albers, H.; and Arthur, J. 2008. Spatial-endogenous fire risk and efficient fuel management and timber harvest. *Land Economics* 84(3):449–468.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)* 9:235–284.

Lagoudakis, M., and Parr, R. 2003. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)* 4:1107–1149.

Larson, R., and Stevenson, K. 1972. On insensitivities in urban redistricting and facility location. *Operations Research* 595–612.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N. 2007. Cost-effective outbreak detection in networks. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Mak, W.-K.; Morton, D. P.; and Wood, R. K. 1999. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* 4:47–56.

Maxwell, M.; Restrepo, M.; Henderson, S.; and Topaloglu, H. 2010. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing (INFORMS)* 22(2):266–281.

Nemhauser, G.; Wolsey, L.; and Fisher, M. 1978. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming* 14:265–294.

Restrepo, M.; Henderson, S.; and Topaloglu, H. 2009. Erlang loss models for the static deployment of ambulances. *Health care management science* 12(1):67–79.

Ross, S. 1983. *Stochastic processes*, volume 23. John Wiley & Sons New York.

Sheldon, D.; Dilkina, B.; Elmachtoub, A.; Finseth, R.; Sabharwal, A.; Conrad, J.; Gomes, C.; Shmoys, D.; Allen, W.; Amundsen, O.; et al. 2010. Maximizing the spread of cascades using network design. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Streeter, M.; Golovin, D.; and Krause, A. 2009. Online learning of assignments. In *Neural Information Processing Systems (NIPS)*.

Verweij, B.; Ahmed, S.; Kleywegt, A.; Nemhauser, G.; and Shapiro, A. 2003. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications* 24(2):289–333.

Yue, Y., and Guestrin, C. 2011. Linear submodular bandits and their application to diversified retrieval. In *Neural Information Processing Systems (NIPS)*.