Decision Support in Emergency Medical Systems: New Strategies for Dynamic Ambulance Allocation

Niklas Meinzer and **Sabine Storandt**
Albert-Ludwigs-Univeristät Freiburg
Institut für Informatik
79110 Freiburg, Germany
storandt@informatik.uni-freiburg.de

Abstract

Most countries have implemented some form of Emergency Medical Services (EMS) in order to help people with urgent medical needs. Typically a number of ambulances serve a specific geographic region. Since the number of ambulances is limited and emergency calls need to be responded to quickly by nature, EMSs have to be thoroughly managed and coordinated, which is conventionally done by operators in special call centres. In this paper, we model this task and compare different strategies which could be employed by a completely automatic ambulance fleet management system or used as decision support tool for operators. We show that continuous optimization of ambulance distribution over the region and dynamic reassignment of ambulances to incoming requests can benefit both the patients and economically the provider of the EMS.

Introduction

Emergency Medical Services (EMS) are a key component of the modern civilization which ensure that people with urgent medical needs quickly receive help. Transporting a patient to a hospital or medical center using ambulances is always a key part in the system. As the number of ambulances in a certain area (e.g. a city) is limited, the ambulance fleet needs to be carefully managed. This is usually done by specially trained operators in emergency dispatch centres. These operators have to decide which vehicle they want to send to a specific emergency, determine the destination hospital and possibly re-distribute the remaining ambulances. The environment in which they operate is characterized by a high degree of uncertainty, as little can be known about future emergency requests at any given time. The main purpose of an EMS system is to provide help in medical emergencies and save as many lives as possible. This leads us to the following optimization problem:

We are given a street graph $G(V, E)$ augmented with travel costs $c : E \to \mathbb{R}^+$, a set of hospitals $H \subset V$ (with $|H| = h$), and an ambulance fleet of size k. Moreover there is a stream of patient requests arriving in an online fashion. Every request is specified by the patient location $v \in V$, the call time $a \in \mathbb{R}^+$ and the deadline $b \in \mathbb{R}^+$, which determines the latest point in time at which the patient must arrive at a hospital in order to be saved. So a patient is considered as rescued if an ambulance at position $w \in V$ is assigned to the request at time $t \geq a$ and $t + c(w, v) + c(v, H^*) \leq b$ with $H^* \in H$ being the target hospital. We refer to the total number of requests as r. The goal is to distribute the k ambulances and assign them to incoming requests such that the number of served requests is as close to r as possible. To reflect the dynamics of real-world EMS even better, we also allow redeployment of free ambulances at any time.

However the operation of an EMS system is also very cost intensive and being a part of traditionally tightly budgeted healthcare systems, economical aspects must also be taken into account. So in the remainder of the paper we will introduce several strategies to carefully position ambulances in their service range and to determine sensible mappings of ambulances to patient requests. In our experimental evaluation we will compare those strategies primarily in terms of the number of patients saved, but we will also analyze the strategies under an economical point of view.

Related Work Ambulance allocation is related to the broad range of dynamic vehicle routing problems where a fleet of vehicles needs to fulfil certain tasks on a graph, as e.g. Dial-A-Ride-Problems (Pillac et al. 2013) or the coordination of a fleet of delivery vehicles (Azi, Gendreau, and Potvin 2012). But strategies for those problems are often focused on maximizing the profit of the system and not the number of served customers. Typical approaches dealing with highly dynamic EMS models are based on maximizing some fitness function of the system as e.g. described in (Restrepo, Henderson, and Topaloglu 2009) for static ambulance allocation. To capture also redeployment of ambulances, simulation-based models and algorithms were proven to be useful. In (Yue, Marla, and Krishnan 2012) a very efficient approach for the dynamic ambulance allocation problem was introduced, which uses simulation as a subroutine. As quality indicator a penalty function was designed, which considers service time and number of requests. But simulation-based algorithms require very precise modelling as well as a huge amount of historical data. In contrast, for our approach no a priori knowledge is necessary. Instead we present event-driven algorithms embedded in a framework which can be easily extended and adapted for varying scenarios.

Contribution In this paper we present strategies to manage an ambulance fleet and examine their performance in different scenarios. We created an event based simulator to test the strategies in simulated environments and discuss several measures by which to evaluate the outcome. These strategies could eventually be implemented in a supervised automated form or as assistance systems to human operators. To visualize the results and to observe the simulation we created a graphical user interface (GUI) using a combination of traditional GUI programming and web tools.

Computing Upper Bounds

The considered problem of ambulance allocation is NP-hard, as it can be interpreted as a variant of scheduling, see (Gary and Johnson 1979). So we will seek to come up with good heuristics to maximize the number of saved patients in an efficient manner. Unfortunately, it might not be possible to rescue all patients, even when using an optimal strategy. This might distort the quality evaluation (saving only 80% of the patients might sound bad, but if we could only rescue 82% with the available resources, the respective strategy would be practical). Therefore we develop now easily computable upper bounds on the number of patients which we later use in our evaluation.

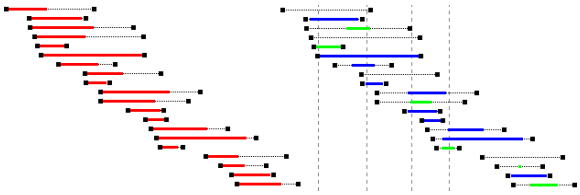

Figure 1: Instance example (left), black boxes indicate call time and deadline, red lines correspond to travel times. Matching duty zones (right) in blue/green. Green zones are the optimal offline solution for $k = 1$. Vertical lines indicate conflicts for $k = 2$ (so ≥ 3 duty zones are intersected).

The idea is to start with the trivial upper bound $U = r$ (all patients can be saved) and then identify points in time where too many intervals (aka requests) have to be served at once. For that we first compute the 'duty zone' for each dynamic interval $[a, a+w] \cap [b-w, b]$, i.e. the interval it overlaps independently of the (valid) start point of its execution (which might be empty). We then search for points in time where more than k duty zones overlap. As at most k of the respective intervals can be processed, the upper bound decreases by the overhead. To make sure that no interval is incorporated in more than one upper bound decrease (which would be invalid), we remove contributing duty zones from the set. After sweeping over all duty zones we receive our upper bound as U plus the number of dynamic intervals without a duty zone. As the set of duty zones also provides us with a non-dynamic instance, we additionally compute the optimal solution for a single ambulance on the duty zone set, which can be done in poly-time (Snoeyink and Hill 2005). This number multiplied with k plus the number of intervals without a duty zone is then a valid upper bound U' as well. Hence our final upper bound is $U^* = \min(U, U')$. In Figure 1 both upper bounding techniques are illustrated.

Allocation and Redeployment Strategies

In our general model we assume that all ambulances are controlled by a single agent, which at any point in time knows about the exact position of each ambulance and whether or not they are occupied by a patient or not. Agents are notified about new requests, ambulances becoming free and ambulances asking which hospital to take a patient to. They then need to issue orders according to their strategies. In addition to theses reactive actions, agents have the possibility to take pre-emptive measures, like redistribution of ambulances, in order to be better prepared for future requests.

Greedy Agents The greedy strategy is to always send the closest ambulance to a new request. Once the patient is in the ambulance it will be sent to the nearest hospital and from there back to its base (or to a new request). The greedy strategy does not involve any preparation such as redistribution of ambulances. The ambulances are simply left in their bases until needed. This makes it very sensitive to the placement of the ambulance bases and hospitals. If there is only one hospital on the edge of the map, response times can be expected to be higher compared to a centred base.

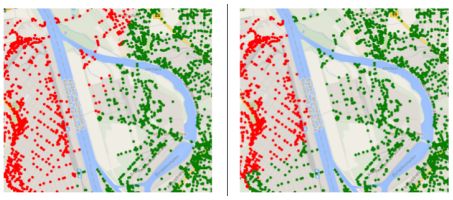

Figure 2: Refinement for k-medoid: Without (left) some nodes on the right hand side of the river are added to the red cluster, although because of the lack of bridges in this area they are closer to the green medoid on the street graph. With the refinement (right) they are added to the green cluster.

k-Medoid Assignment In order to overcome the weaknesses of the greedy strategy, we design k-medoid agents. The main idea is that in areas with dense emergency calls, it might be beneficial for the ambulances to be already enroute and not necessarily be stationed at hospitals. So we try to minimize the average distance of each possible request origin (i.e. each node in the graph) to the nearest ambulance and thus the response time to the next request. So we want to solve the k-medoid problem with k being the actual number of available ambulances and the metric being the shortest path distance in the graph. Classically the optimal positions are determined by a round-based algorithm called Partitioning around Medoids (PAM). It starts with a random positioning and tries then to improve the medoids by swapping. For large input graphs PAM is very time-consuming as in each round a lot of configurations have to be checked. To accelerate the approach, we propose the following heuristic: In all but the last round we do not use shortest path distance in the graph as metric to find the best k-medoids, but simply euclidean distance. This spares us a lot of Dijkstra computations. Only in the last round – after according to euclidean distances the optimal positions are found – we perform one assignment steps with actual shortest path distances (see Figure 2 for an exemplary illustration).

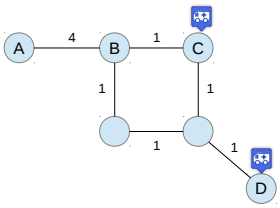

Figure 3: Consider requests from B and A shortly after each other. A non-reassigning agent sends the ambulance at C to B and the one at D to A (max. response time of 7). If C is re-assigned to A and D then sent to B, the max. response time is 5.

Voronoi-based Allocation The k-medoid strategy can position ambulances on all nodes in the graph. In practise, positioning at any point throughout the roadmap is probably not going to be accepted by the ambulance personnel and makes no sense if requests are sparse. To cope with this we use a Voronoi-based strategy. Here free ambulances are only positioned at hospitals or bases. For this we divide the graph into cells resulting in a Voronoi-diagram with the hospitals as seeds. Each node is assigned to the cell belonging to the closest hospital using the actual distance on the road network. Each resulting cell $C \subseteq V$ gets assigned a size value $S(C) = {}^{|V|}/_{|v \in C|}$. Like in the k-medoid strategy, each time the number of free ambulances changes, they are redistributed. Here, for each cell C, $\lfloor |free(A)| \cdot S(C) \rfloor$ ambulances are sent to the seed of C. The remaining free ambulances are iteratively assigned to the cell with the fewest ambulances using cell size as a tiebreaker. Voronoi represents a compromise between Greedy and k-medoid. It is more practical since ambulances are only idle in hospitals or ambulance bases and Voronoi involves less driving around than k-medoid, since if an ambulance stays in the same cell, it does not have to move at all. Like Greedy, Voronoi's performance is not completely independent from the setting, as positions of hospitals and ambulance bases are predefined. It is however more flexible since hospitals can be used as bases and the initial distribution of ambulances can be changed. Although it is computationally more complex than Greedy, the precomputation and computation during redeployment is less expensive than in k-medoid.

Redeployment Usually in EMS an ambulance is never reassigned to a different patient while it is en-route. This behaviour can lead to bad response times in case of disadvantageous request streams (see Figure 3). We therefore propose a redeployment scheme that can be incorporated in all our presented strategies: Any time the number of open requests (i.e. requests where no ambulance has arrived) or the number of free ambulances increases, for example once a new requests comes in, we re-evaluate the assignment of ambulances to requests. We define the set of available ambulances as the free ambulances plus the ones already responding to a request. We then try to find an assignment of the available ambulances to the open requests, which minimizes the total distance of ambulances to requests. To solve this problem we use an implementation of Kuhn-Munkres assignment algorithm (also called Hungarian method (Kuhn 1955)).

Experimental Results

We performed our experiments on a laptop with a 2.4 GHz Intel i5 processor (using only one core) and 8 GByte of RAM. The calculations needed for a single event on graphs with hundreds of thousands of nodes stayed well below a second and are thus suited for real world use. For the road network in our simulation we used Open Street Map data [1]. To evaluate the performance of our strategies, we used an event-driven simulator. The agents can be plugged into the simulator, get notified on certain events and issue orders to the ambulances. Moreover we implemented a GUI to visualize the current state of the system (see Figure 4).

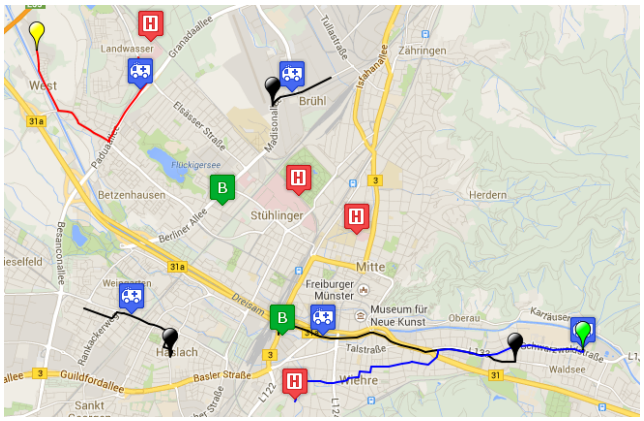

Figure 4: Visualization of the current position and routes of the ambulances (blue markers). Red markers represent hospitals, green markers ambulance bases. Ambulances on red paths respond to a patient request (yellow marker). Ambulances on a blue path are taking a patient to a hospital and ambulances on a black path are currently redeploying.

Evaluation of Allocation Strategies

To compare the performance of our different strategies we measure three different values: most importantly, the *total number of saved patients*, then the *average response time*, i.e. the time it took for an ambulance to arrive at a patient after the call came in, and the *total distance driven* summed over all ambulances. We designed two scenarios with different underlying street graphs and EMS infrastructure and tested our strategies under them.

Scenario 1 First we benchmarked our strategies in a real world example. We use the street graph of Freiburg, a city with about 200,000 inhabitants. The street graph contains 25,000 nodes. The hospitals, ambulance bases and ambulances used in the simulation correspond to those actually present in Freiburg: There are four hospitals and two ambulance bases with five ambulances overall. The hospitals are distributed throughout the city and both ambulance bases are rather close to the city. We simulated a time between requests range from 60s to 180s with 100 patients per run. The results of this benchmark show the following:
Patients saved: With low request frequencies all strategies perform equally bad and are only able to save 50% or less of the patients. With increasing intervals K-Medoids has a slight advantage over the other approaches, saving about two to three patients more on average.

[1] www.openstreetmap.org

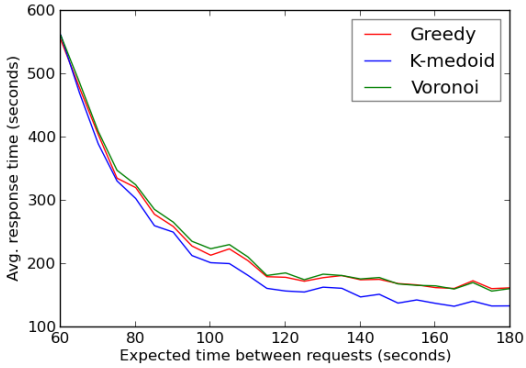

Figure 5: The average response time in scenario 1.

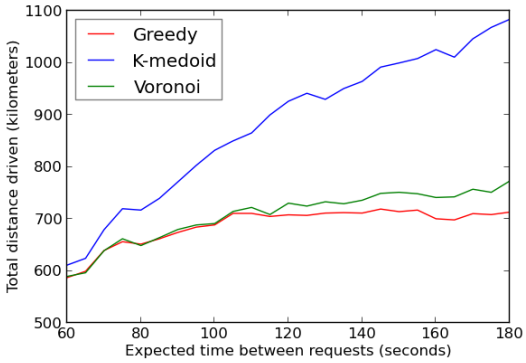

Figure 6: The average overall distance driven in scenario 1.

Response time and time to hospital (see Figure 5): K-medoids is also slightly better in those categories: With increasing request intervals it responds about 30 to 40 seconds faster to requests than the other two approaches.
Total distance driven(see Figure 6): This measure shows the costs involved in the k-medoids strategy. The constant redeployment of ambulances results in almost double the total distance of Greedy with higher request intervals. With lower request frequencies the difference is less significant, because here ambulances have to abort redistribution moves more often and go to the next patient. Voronoi is only slightly worse than Greedy, since often only a few ambulances move during a redeployment.

Scenario 2 For the second scenario we use the street graph of the Rhine-Neckar metropolitan region. A dense city cluster with approximately 2.5 million inhabitants. The region consist of the three cities Ludwigshafen, Mannheim and Heidelberg and surrounding municipalities. The street networks exhibits about 230,000 nodes. We again distributed hospitals relatively evenly on the map, but restricted the ambulance bases to one half of the map. This is to simulate a scenario where half of the ambulance fleet is not operational due to fire or flood in a base or not reachable due to communications outage. While the hospitals are still functional in all areas, ambulances now have to travel longer distances half of the times. The results are the following:
Patients saved: This measure shows a clear advantage of the two redeployment strategies over Greedy. For all tested request interval values k-medoids is able to save about five to ten patients more than Greedy. Voronoi ranks in the middle with two to five patients more than Greedy.

Response time and time to hospital: The ranking from the first measure is confirmed in response time and to hospital time. Here k-medoids performes best, followed by Voronoi and with a greater gap Greedy. With large request intervals k-medoids responds about 2.5 minutes and Voronoi about two minutes faster than Greedy.
Total distance driven: As in scenario 1, ambulances managed by k-medoids travel by far the greatest overall distance. However while the total distance driven by k-medoids was almost double the total distance of Greedy in scenario one, it is only 1.4 times Greedys distance in scenario 2. Since Greedy always sends the ambulances back to the western part of the map when they are idle, they have to travel long distances and thus Greedy is even outperformed by Voronoi.

Reassignment Adding the possibility to reassign ambulances we could drastically improve the performance of all strategies. For example for low intervals between requests the average response time for k-medoid is three minutes shorter with reassignment. Also the number of saved patients improves (up to 30%) for Greedy as well as k-medoid if we perform Munkres algorithm to assign free ambulances. It allowed us to save almost all patients possible according to our upper bound.

Conclusions and Future Work

In this paper we investigated problems related to Emergency Medical Services (EMS) management. We introduced two new ambulance distribution schemes, k-Medoid and Voronoi-based, which outperform the conventional greedy approach in several scenarios as shown in our experiments. Surprisingly, if we allow ambulances on the way to a patient to be reassigned, we can improve the overall service quality significantly. Future work includes the enhancement of our strategies with other redeployment algorithms and the incorporation of detailed emergency plans for larger accidents (which e.g. are likely to occur on certain highways).

References

Azi, N.; Gendreau, M.; and Potvin, J.-Y. 2012. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research* 199(1):103–112.

Gary, M. R., and Johnson, D. S. 1979. Computers and intractability: A guide to the theory of np-completeness.

Kuhn, H. W. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly* 2:83–97.

Pillac, V.; Gendreau, M.; Guret, C.; and Medaglia, A. L. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225(1):1 – 11.

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

Snoeyink, J., and Hill, U. C. 2005. Maximum independent set for intervals by divide-prune-and-conquer. In *CCCG*, 264–265.

Yue, Y.; Marla, L.; and Krishnan, R. 2012. An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment. In *AAAI*.