Benchmarking online dispatch algorithms for Emergency Medical Services

C.J. Jagtenberg[a], P.L. van den Berg[a,c], R.D. van der Mei[a,b]

[a] *CWI, Stochastics, Science Park 123, 1098 XG Amsterdam, The Netherlands*
[b] *VU University Amsterdam, Faculty of Sciences, De Boelelaan 1081a 1081 HV Amsterdam, The Netherlands*
[c] *Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands*

Abstract

Providers of Emergency Medical Services (EMS) face the online ambulance dispatch problem, in which they decide which ambulance to send to an incoming incident. Their objective is to minimize the fraction of arrivals later than a target time. Today, the gap between existing solutions and the optimum is unknown, and we provide a bound for this gap.

Motivated by this, we propose a benchmark model (referred to as the offline model) to calculate the optimal dispatch decisions assuming that all incidents are known in advance. For this model, we introduce and implement three different methods to compute the optimal offline dispatch policy for problems with a finite number of incidents. The performance of the offline optimal solution serves as a bound for the performance of an - unknown - optimal online dispatching policy.

We show that the competitive ratio (i.e., the worst case performance ratio between the optimal online and the optimal offline solution) of the dispatch problem is infinitely large; that is, even an optimal online dispatch algorithm can perform arbitrarily bad compared to the offline solution. Then, we performed benchmark experiments for a large ambulance provider in the Netherlands. The results show that for this realistic EMS system, when dispatching the closest idle vehicle to every incident, one obtains a fraction of late arrivals that is approximately 2.7 times that of the optimal offline policy. We also analyze another online dispatch heuristic, that manages to

Email address: c.j.jagtenberg@cwi.nl +31 20 592 9333 (C.J. Jagtenberg)

reduce this gap to approximately 1.9. This constitutes the first quantification of the gap between online and offline dispatch policies.

Keywords: OR in health services, Ambulances, Emergency medical services, Dispatch, Online Optimization.

---

1. Introduction

In serious life-threatening situations where every second counts, the timely presence of emergency aid at the incident scene can make the difference between life and death. Motivated by this, recently there has been a wide interest in planning of Emergency Medical Services (EMS) and many models and approaches have been developed in order to use ambulances efficiently. EMS planning often revolves around response times: the time between the occurrence of an incident and the time that an ambulance arrives on scene. A key performance metric is the fraction of incidents that have a response time greater than a certain threshold time. EMS call center agents have to make on-the-fly choices about *which vehicle to dispatch* to a newly incoming incident, such that the fraction of arrivals later than a target time is minimized. The classical dispatch policy often used in practice is the *closest idle vehicle* approach. Also, most literature uses the 'closest idle' policy without questioning it, even though it was already shown to be suboptimal in 1972 [2]. Recently, Jagtenberg et al. [8] showed that this policy is in fact quite far from optimal, and developed an algorithm that outperforms the closest idle dispatch method.

The question addressed in this paper is how to benchmark dispatch policies against optimal policies with full information: suppose we would know all incident arrivals and locations *in advance*, then how much better would the performance of the optimal dispatch policy be? What is the potential improvement if we were able to perfectly predict future incidents? The answer to such questions addresses the *value of information* about future incidents, and give insight into how far we are from the optimum under full information and what is the potential for developing accurate forecasting models for emergency incidents.

By analyzing the dispatch process from a new angle, this paper provides a contribution that will be of interest to researchers who develop mathematical models for EMS planning. This new perspective helps to develop a deeper understanding of EMS planning models. Furthermore, we link ambulance

dispatching to the literature on online/offline optimization. For a general introduction to the concept of online versus offline algorithms, see [9].

In the literature, many models are available for ambulance planning. First of all, there are models that deal with planning on the strategic level. Typically, such models determine the best locations for ambulance bases [3], and they sometimes also determine the number of vehicles that should be positioned at each base [5], [6]. The majority of these solutions use mixed integer linear programming to solve the problem. Second, there is previous work on operational ambulance planning. This can be divided in two categories: (1) methods that relocate idle ambulances, and (2) methods that dynamically decide which ambulance to dispatch to incidents.

Dynamic relocations
The vast majority of the papers on dynamic ambulance management focuses on how to redeploy idle vehicles, (e.g., [1], [12] and [18]). Perhaps in order not to overcomplicate things, they assume a basic dispatch rule: whenever an incident occurs, they decide to send the ambulance that is closest (in time). Although this is a common dispatch policy, it was already shown to be suboptimal in 1972 [2]. Regardless, most authors make this assumption without much discussion or justification; for example, Maxwell et al. [12] claim that it is an accurate enough representation of reality; however, they do not address the question of whether it is an *optimal* choice with respect to the objective (which is the fraction of incidents that are reached within the threshold time). Alanis et al. [1] do not address the assumption at all.

Dynamic dispatching
Few papers focus on dynamic dispatch methods. One exception is [16], in which the authors combine both decisions on repositioning and dispatching. They report that adding a dynamic dispatch method improves the average response time from 4.05 to 4.01 minutes (as compared to 'only' dynamic repositioning). It is not mentioned how much dynamic dispatching improves the response time compared to static ambulance planning. Furthermore, their objective differs from ours as they do not consider a response time threshold (RTT).

A paper that explicitly searches for dynamic dispatch methods in combination with a RTT is [8]. It includes an easy to implement heuristic and shows that we can indeed improve the objective - the fraction of late arrivals - by changing the dispatch policy. However, since the topic has been under-

exposed in the literature, it is still unknown how much profit can be expected from an *optimal* dispatch rule.

Bounds for dynamic ambulance planning
The concept of bounds on the performance of EMS systems is relatively new. There is one recent paper that provides a bound for the performance of an optimal ambulance redeployment policy [11]. However, for a bound on the performance of *dispatch* policies, we are not aware of any result.

Offline dispatching
There is previous work on ambulance planning that uses ideas similar to offline dispatching. However, authors typically do not recognize the idea as such. For example, [18] aims to analyze and evaluate repositioning algorithms, and to that end uses optimal offline dispatch policies as an upper bound on the possible performance. Instead of calling it an offline version of an online problem, the authors refer to the offline approach as 'the omniscient observer'. Most importantly, this paper differs from ours because it does not include a comparison with online dispatch methods. Other researchers use offline dispatching to compute the number of vehicles needed to serve all incidents, without noting that this is perhaps a rather optimistic approach [17].

A related problem is the dial-a-ride problem, which deals with online arriving requests for transports between an origin and destination. For an overview of literature on this problem, see [4]. The dial-a-ride problem is similar in the sense that routes are created; however, it typically allows for flexibility in the execution time of each request, whereas the (urgent) ambulance requests require a vehicle to be sent immediately. Furthermore, in dial-a-ride problems the objective is typically either related to efficiency (such as transportation cost or travel time) or based on customers' inconvenience (such as lateness or excess drive time). There is literature that considers dial-a-ride problems specifically in the ambulance context. However, this usually concerns the non-urgent patient transports, see e.g. [13, 15, 14]. Due to the fact that their objectives are *not* related to a response time threshold, we cannot directly use their results or formulations.

Another related problem is the k-server problem [10], which is one of the classical problems in competitive analysis. In this problem, each time step corresponds to a *request* arriving somewhere in a metric space. There is a set of k servers available, and an algorithm prescribes for each request which

server should respond. The objective is to minimize the total distance moved by all servers. The competitive ratio of the k-server problem is currently unknown, although it can be shown that it is at least k, and there exists a conjecture stating that the competitive ratio is exactly k [10]. This problem differs from our ambulance problem in three crucial ways. First of all, in the k-server problem requests do not overlap in time. Second, servers await their next move at the location of their last request, whereas ambulances return to their home base. Third, there is no response time threshold in the k-server problem.

Contribution

The contribution of this paper is twofold: (1) to give a bound for the fraction of late arrivals that can be achieved by *any* ambulance dispatch policy, even if all future incident times and locations would be known in advance, and (2) to benchmark and assess the potential for improvement of existing dispatch algorithms. To this end, we introduce three different methods to compute the optimal offline dispatch decisions in case future incident arrivals are known in advance. The first method is Constraint Programming (CP); to the best of our knowledge, this paper is the first to apply CP to ambulance planning. Next, as an alternative, we also formulate the offline dispatch problem as a Dynamic Programming (DP), and we discuss how this DP provides insight into the problem. We introduce a third method, that is the fastest among the three, using Binary Linear Programming (BLP). We emphasize that all three methods result in the same solution, that is, the optimal solution for the offline problem. Subsequently, we determine the performance of two key online algorithms: the classical 'closest idle ambulance' rule, and the heuristic method described in [8]. These performances are obtained by a discrete event simulation model of an urban EMS region.

Our interest in quantifying the gap between online and offline algorithms is twofold. From a theoretical point of view, we are interested in the competitive ratio of the dispatch problem (i.e., a worst case measure for an optimal online algorithm). Conversely, from a practical point of view, we are interested in the performance ratio between online and offline algorithms for *realistic* incident chains. This gives an indication of how much performance improvement can be obtained by developing better dispatch methods, and simultaneously shows how much one can benefit from developing accurate incident prediction models.

We do a worst case analysis by constructing a toy example that shows

that the so-called competitive ratio (i.e., the worst case performance ratio of the fraction of late arrivals between the optimal online and the optimal offline solution) of the dispatch problem is infinitely large; in other words, the optimal online dispatch algorithms can perform arbitrarily bad compared to the offline solution. We also analyze *realistic* problem instances by performing benchmark experiments for a large ambulance provider in the Netherlands. The results show that for this realistic EMS system, the fraction of late arrivals of the classical 'closest idle' dispatch heuristic is approximately 3.5%, whereas the offline optimum is 1.5%. What is perhaps most surprising, is that our results show there exists an online dispatch heuristic that closes roughly half of this gap between 'closest idle' and the offline optimum. This is the so-called DMEXCLP dispatch heuristic, that results in 2.6% late arrivals (and thereby performs only 1.9 times worse than the optimal offline policy). The remainder of this paper is structured as follows. In Section 2, we give a formal problem definition. In Section 3, we describe the two online policies, and introduce - and analyze - three methods to find optimal offline solutions. In Section 4, we perform a worst case analysis of the problem, and show that the competitive ratio is infinitely large. We end with computational results for the average case in Section 5 and a discussion in Section 6.

2. Problem formulation

We consider the problem of ambulance dispatching. In this problem, incidents occur randomly in time and space, and the task is to determine which ambulance to send to each incident. At first sight, the ambulance problem may seem similar to, e.g., that of a police or taxi crew. However, there is a crucial difference: taxis typically generate *more* requests simply by driving through locations with potential customers. On the other hand, the presence of police cars in an area usually *reduces* crime rates, and thereby the number of requests. For ambulances, however, we assume the following.

Assumption 1. *The occurrence of incidents is independent of previous incidents and the chosen dispatch policy.*

We consider this assumption to be very realistic, and will use it throughout this paper. From Assumption 1 follows that we can generate incidents in a preparatory phase, prior to determining the decisions made by each dispatch policy.

2.1. Model and notation

We generate incidents in time and space as follows[1]. Define V as the set of locations at which incidents can occur. These demand locations are modeled as a set of discrete points. Incidents at locations in V occur according to a Poisson process with rate λ. Let d_i be the fraction of the demand rate λ that occurs at node i, $i \in V$. Then, on a smaller scale, incidents occur at node i with rate λd_i. According to these Poisson processes, we can simulate sequences of incidents.

Let A be the set of ambulances, and $A_{idle} \subseteq A$ the set of currently idle ambulances. When an incident has occurred, we require an idle ambulance to immediately drive to the scene of the incident. The decision which ambulance to send is the main question of interest in this paper. Throughout this paper, we assume the following.

Assumption 2. *There are sufficiently many ambulances, such that at least one ambulance is idle whenever an incident occurs.*

We consider two types of problems: (1) *online* problems, and (2) *offline* problems. In the online problem, the decision which ambulance to send has to be made at the moment the incident occurs; future incidents are unknown and can at best be predicted. In the offline version of the problem, all incidents (i.e., their time stamps and locations) are known *in advance*.

Our objective is formulated in terms of response times, defined as the time between an incident and the arrival of an ambulance at the emergency scene. In practice, incidents have the requirement that an ambulance must be present within T time units. Therefore, we want to *minimize* the *fraction of late arrivals*, defined as the fraction of incidents for which the response time is larger than T.

Assumption 3. *We assume that the travel time $\tau_{i,j}$ between two nodes $i, j \in V$ is deterministic, and known in advance.*

Our objective can be formalized as follows. Recall that incidents are generated according to the Poisson process described above. Let C denote our set of incidents (also known as calls), and let n be the number of incidents, i.e., $n = |C|$. Straightforwardly, $t(c)$ denotes the time that incident c occurs

[1]The call generation described here is equivalent to the system defined in [7] and [8].

($c \in C$). Let furthermore, $h^\pi(c)$ represents the time a vehicle arrives at the scene of incident c, under policy π. Now we can express our objective as:

$$\arg\min_{\pi \in \Pi} \lim_{n \to \infty} \frac{\sum_{c=1}^{n} \mathbb{1}[h^\pi(c) - t(c) > T]}{n}. \tag{1}$$

Sending an ambulance to an incident is followed by a chain of events, such as spending time on scene with the patient, deciding whether the patient needs transport to a hospital (and if so: additional travel time and a drop-off time at the emergency department). In practice, these events will take a random amount of time. However, this makes for a very complex problem, to which both the online *and* offline optimal solution is not known. Thereto, we use a simplified model of the EMS process, which ensures that the optimal offline solution can be computed.

Assumption 4. *The busy time, excluding travel time, is known and deterministic and the same for all calls.*

We define an ambulance to be busy for X minutes after arriving at the scene of an incident. Note that this parameter X is assumed to be independent of the incident location and the base location the ambulance departed from. After these X minutes, the ambulance becomes idle at its (predefined) base location.

We denote the base location of ambulance a by W_a, for $a \in A$. Note that it is possible for multiple ambulances to have the same base location. As soon as an ambulance has reached its base location, it is ready to be dispatched again[2]. An overview of the notation can be found in Table 1.

2.2. Goal

In this paper, we focus on bounding the performance of any online solution to the ambulance dispatch problem. Since the optimal solution to the online problem (in which future incidents are unknown) is not known, we use the optimal solution to the *offline* version of the problem (in which all incidents are known in advance) as a bound.

[2]This problem description is similar to the one defined in [8], with the following two main differences. In [8] vehicles are allowed to be dispatched while returning to their home base (i.e., when they are on the road) and the ambulance service times are modeled as a stochastic process, rather than a constant time X.

V	The set of demand locations.
A	The set of ambulances.
A_{idle}	The set of idle ambulances.
W_a	The base location for ambulance a, $a \in A$, $W_a \in V$.
T	The time threshold.
X	The time an ambulance is busy with one incident, from the moment of arrival at the scene.
λ	Incident rate.
d_i	The fraction of demand in i, $i \in V$.
$\tau_{i,j}$	The driving time between i and j with siren turned on, $i, j \in V$.
n	The number of incidents, for a certain chain of incidents.
$t(c)$	The time that incident c occurs, $c \in C$
$loc(c)$	The location of incident c, $c \in C$, where $loc(c) \in V$.

Table 1: Notation.

Our first goal is to formulate a model that allows us to compute the optimal (offline) dispatch policy. Our second goal is to compare this offline optimum to the performance of existing online (heuristic) methods.

3. Method

We introduce and implement three different methods to find the optimal offline solution for a general instance of the dispatch problem (with a finite number of calls). The first method, constraint programming (CP), has the advantage that it is easy and quick to implement. The second, dynamic programming (DP), is able to find the same solution with somewhat shorter running times, and on top of that allows us to investigate which properties make an instance hard to solve. The downside of this method is that it has the most labour-intensive implementation. The third method, Binary Linear Programming (BLP) solves the problem the fastest. In this section we describe the BLP; the DP and CP model can be found in Appendix A.

We also define the online dispatch policies that we use in our analysis. These solution methods are eventually used to compare the performance on several problem instances.

3.1. The Optimal Offline Solution using Binary Linear Programming

In order to formulate the problem as a BLP, we first introduce parameters $p_{cj} \in \{0, 1\}$, for $c \in C$, $j \in A$. This parameter is the penalty of assigning ambulance j to incident c: it will be set to 0 if ambulance j arrives within threshold time T, and 1 otherwise. Note that the values of p_{cj} can be deduced from the problem specification, using the base locations, driving times between those bases and the incident locations, and the (fixed) parameter T.

Our decision variables will be $x_{cj} \in \{0, 1\}$, for $c \in C$, $j \in A$, which will be 1 if and only if ambulance j is assigned to incident c.

The most important constraint of our problem, is that two incidents handled by the same ambulance may not overlap in time. At first sight, it seems hard to model this in a linear way: recall that the travel time depends on the ambulance that is chosen. However, we can precompute for each combination of incidents c and c', whether or not they overlap in time if they were to be served by ambulance j. Denote this with parameter $o_{cc'j}$, for $c, c' \in C, j \in A$, which is equal to 1 if the incidents overlap in time, and 0 otherwise. If $o_{cc'j}$ equals 1, we add a constraint that at most one incident in $\{c, c'\}$ may be served by ambulance j. Then, the offline ambulance dispatch problem can be modeled as a BLP as follows.

$$\text{Minimize} \sum_{c \in C} \sum_{j \in A} p_{cj} x_{cj}$$

subject to

$$\sum_{j \in A} x_{cj} = 1, \quad c \in C$$

$$o_{cc'j} \cdot (x_{cj} + x_{c'j}) \leq 1, \quad j \in A, \quad c, c' \in C, c \neq c'$$

$$x_{cj} \in \{0, 1\}, \quad c \in C, j \in A$$

3.2. Online Solutions

In this section, we describe two online dispatch methods. The first is often used in practice, and the second was shown to give good performance for our objective (the fraction of late arrivals).

3.2.1. The 'closest idle' dispatch method

When an incident occurs, all idle ambulances are considered. The idle ambulance that is closest[3] to the incident location, is then dispatched. This

[3] closest in time, not necessarily in space

notion can be formally expressed as follows.

$$\arg\min_{a \in A_{idle}} (\tau_{W_a, loc(i)})$$

i.e., the ambulance a for which the travel time τ is the smallest amongst all idle ambulances.

3.2.2. The DMEXCLP dispatch heuristic

In this section, we briefly describe the online dispatch method that we shall refer to as the 'DMEXCLP dispatch heuristic'. This heuristic was first defined in [8], and applied to test data similar to region Utrecht as defined in this paper[4]. This showed that the heuristic reduces the fraction of late arrivals by 18% compared to the 'closest idle' benchmark policy. A mentioned drawback is that this heuristic increases the average response time. Therefore, the authors do not claim that this heuristic is practically preferable over the closest-idle method. However, the mentioned improvement of 18% is considerable, and hence it would be interesting to see how the heuristic performs compared to the offline optimum.

The general idea of the DMEXCLP dispatch heuristic is that we choose an ambulance such that the remaining idle ambulances provide good *coverage* of the region. Coverage can be interpreted as a number that indicates how well we can serve the incidents that might occur in the (near) future.

The definition of coverage for the DMEXCLP dispatch heuristic was borrowed from the well-known MEXCLP model [7], which we briefly describe next. MEXCLP was originally designed to optimize the distribution of a limited number, say $|A|$, ambulances over a set of possible base locations W. Ambulances are modeled to be unavailable with a pre-determined probability q, that is the same for all vehicles. Input parameter q can be estimated by computing the total workload based on expected calls divided by the number of ambulances. For any node $i \in V$, the MEXCLP model determines the coverage based on the number of ambulances (k) that can serve this node within the time standard. Since the travel times $\tau_{i,j}$ $(i, j \in V)$ are assumed to be deterministic, one can straightforwardly determine this number k. Denote the demand at node i by d_i, and define the expected covered demand of this vertex to be $E_k = d_i(1 - q^k)$. Note that the marginal contribution of

[4]In [8], the region considered is also Utrecht. However, the incident rate as well as the number of vehicles used is slightly lower than in the current paper.

the k^{th} ambulance to this expected value is $E_k - E_{k-1} = d_i(1-q)q^{k-1}$. In order to model the problem as an ILP, the authors use binary variables y_{ik} that are equal to 1 if and only if vertex $i \in V$ is within range of at least k ambulances. The objective of the MEXCLP model can now be written as:

$$\text{Maximize} \sum_{i \in V} \sum_{k=1}^{|A|} d_i(1-q)q^{k-1}y_{ik}.$$

The dispatch problem requires us to decide which (idle) ambulance to send, at the moment an incident occurs. Thereto, we compute the *marginal* coverage that each ambulance provides for the region, at this point in time. The ambulance that provides the smallest marginal coverage, is the best choice for dispatch, in terms of remaining coverage for future incidents. However, this does not incorporate the desire to reach the current incident within target time T. We propose to combine the two objectives – reaching the incident in time and remaining a well-covered region – by always sending an ambulance that will reach the incident in time, if possible. This still leaves a certain amount of freedom in determining which *particular* ambulance to send.

If none of the idle ambulances can reach the incident in time, all idle ambulances are eligible for dispatch. To summarize, the DMEXCLP dispatch heuristic chooses the ambulance - within the set of eligible ambulances - that maximizes the coverage provided by the remaining idle ambulances. The calculations are done by brute force, which can easily be performed in real-time for realistic problem sizes.

3.3. Benchmarking solutions

In this section, we describe how we calculated the performance ratio between an online and an offline dispatch policy. By definition, the performance of an online policy must be equal to or worse than the offline optimum. Recall that our objectives are defined as the fraction of late arrivals. Since we are minimizing our objective, we can immediately conclude that the online/offline performance ratio will be ≥ 1.

Given a specific EMS region, we drew a finite sequence of incidents according to the Poisson process defined in Section 2.1. Denote the fraction of late arrivals for a certain policy P and incident sequence s by $FracLate_P(s)$. We repeated this process multiple times, using a large set of incident sequences (S), in order to determine the objective more accurately. Our final

estimate for the performance ratio is then computed as the ratio of the average performances:

$$\text{Performance Ratio} := \frac{\frac{1}{|S|}\sum_{s\in S} FracLate_{Online}(s)}{\frac{1}{|S|}\sum_{s\in S} FracLate_{Offline}(s)} \tag{2}$$

Note that we do *not* compute the performance ratio of each individual incident sequence. The reason for this, is that when the offline optimum results in 0 late arrivals, the performance ratio becomes infinitely large, and this does not lead to a meaningful average performance ratio.

4. Worst case analysis

In this section, we describe a worst case realization of incidents. This example is meant to illustrate to what extent an 'unfortunate' chain of incidents can affect the performance of online dispatch algorithms. The example directly leads to the so-called *competitive ratio* of the dispatch problem.

Consider a region where the time threshold $T = 12$ minutes, and the busy time for an ambulance is $X = 37$ minutes. There are two nodes in which incidents can occur, and the driving time between these nodes is 13 minutes. Each node is the base location for one ambulance. For simplicity, let us say ambulance 1 has base location 1, and ambulance 2 is stationed at location 2. For a graphic representation, see Figure 1. It is easy to see that an ambulance will reach an incident in time, if and only if the ambulance at the location of the incident is available.

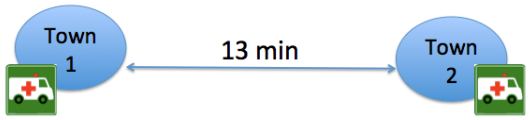


Figure 1: Region with two towns, each being the home base for one ambulance.

Table 2 shows a chain of incident realizations for which the closest idle dispatch policy performs particularly poorly. Typical about this example is that a dispatch algorithm only has a choice for the first incident (at time 0). After that, the sequence of incidents is timed such that there is only one

incidents			**optimal offline**		**closest idle (online)**	
number	time	location	send ambu	in time?	send ambu	in time?
1	0	1	2	no	1	yes
2	5	1	1	yes	2	no
3	51	2	2	yes	1	no
4	56	1	1	yes	2	no
5	102	2	2	yes	1	no
6	107	1	1	yes	2	no
⋮						
n = 2m+1	m·51	1	1	yes	2	no

Table 2: A worst case example of incidents for the region described in Section 4. The corresponding solution of two policies is denoted, as well as whether or not they can serve each incident within the threshold time. Note that the incidents in each location are exactly 51 minutes apart.

ambulance idle at any decision moment[5]. By our problem definition, that ambulance must then be dispatched immediately. So, if an algorithm makes the wrong decision in the first time step - like the closest idle policy does - all following incidents except the first one the ambulance will arrive later than the threshold time. Alternatively, if the correct decision is made in the first time step, only the first incident's ambulance will arrive late.

Note that it is impossible for any online algorithm to know what is the best decision in the first time step. To see this, imagine an (online) algorithm that upon seeing the first incident in location 1, sends ambulance 2 (hence it does the opposite of the closest idle method.) The worst case instance for this algorithm, would have the same incident times as in Table 2, but have the locations of incident 2 ... n *swapped* (i.e. location $1 \leftrightarrow 2$). Then, again, only the first incident would be reached in time. Thereto, we conclude that the performance ratio of any online algorithm can be a factor

$$\frac{(n-1)/n}{1/n} = \frac{n-1}{1} \rightarrow_{n\rightarrow\infty} \infty.$$

[5]Note that incidents in each location are 51 minutes apart, while the busy time of an ambulance is at most 13+37=50 minutes.

larger than the optimal offline policy.[6]

Although this worst case is interesting from a theoretical perspective, we want to clarify that this is not a case that is likely to occur in practice. Since ambulance planning is a topic of practical importance, the rest of this paper focuses on the performance ratio between online and offline algorithms for *realistic* incident chains. More specifically, we are interested in the *expected* performance ratio for incident chains that originate from an incident distribution as described in Section 2.1.

5. Computational results

In this section, we analyze the ambulance dispatch problem based on an EMS system that represents Utrecht, one of the largest EMS regions in the Netherlands. Figure 2 shows a map of the region and the base locations that we used. Utrecht is a densely populated area, with approximately 1.2 million inhabitants and an area of approximately 1,400 square kilometers. The ambulance provider for this region handles more than 100,000 incidents per year - a number equal to roughly 10% of all ambulance demand in the Netherlands.

We chose realistic parameters to model the EMS region Utrecht. For example, the base locations that we used are equal to the ones used in practice (for - at least - the period between 2013 and 2015). Furthermore, we divided the region by postal codes, and model the incident arrivals in each postal code as a Poisson process with a rate proportional to the population. Ambulance travel times were provided by the Dutch National Institute for Public Health and the Environment (RIVM). For the exact parameters used in the implementation, see Table 3.

[6]One might argue that in this particular case the ambulance dispatcher should be allowed to change his mind and send a different ambulance whenever new information becomes available - like a new vehicle becoming idle - as long as the originally dispatched ambulance still has not arrived. That is, the dispatcher might be able to perform better if he is allowed to schedule with preemption during the travel time. However, one can easily see that preemption does not change the worst case: suppose we change the problem instance in Table 2 by increasing the time of incident 2 from 5 to 13, and update the consecutive (odd) incidents accordingly. Then, the new information arrives too late, i.e., the originally dispatched ambulance has already arrived, and hence the infinitely large ratio also holds for the dispatch problem with preemption.

Parameter	Magnitude	Choice
λ_1	$0.9 \cdot \lambda_2$	A 10% lower rate than normal for this region.
λ_2	1/6.4 min	Realistic for urgent calls on a weekday in this region.
λ_3	$1.1 \cdot \lambda_2$	A 10% higher rate than normal for this region.
A	25	Fleet sized such that performance is realistic (near 5% late arrivals).
W	19	Base locations as existing in 2013-2015. We divide the ambulances over the bases according to the static MEXCLP solution.
V	217	4 digit postal codes.
τ_{ij}		Driving times as estimated by the RIVM, rounded to minutes.
d_i		Fraction of inhabitants as known in 2009.
T	12 min	Typical time standard for high priority incidents in the Netherlands.
X	37 min	Realistic average busy time for ambulances[7].

Table 3: Parameter choices for our implementation of the region of Utrecht.

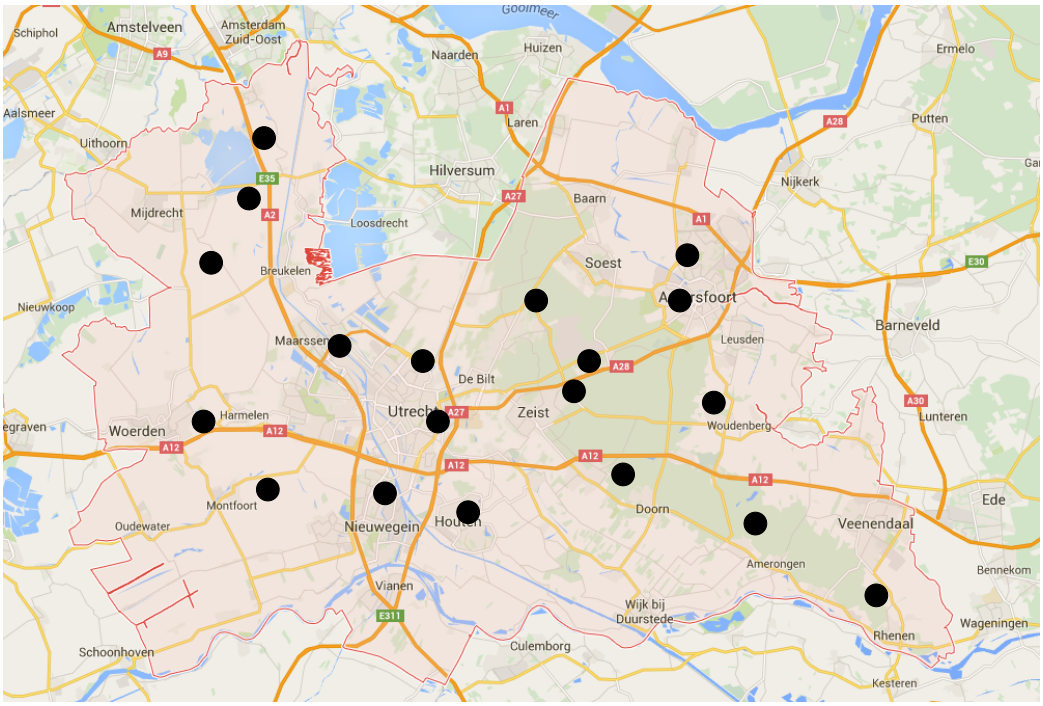


Figure 2: The 19 existing ambulance base locations in the region of Utrecht, The Netherlands. We distribute the 25 available ambulances over the bases according to the MEXCLP solution with busy fraction $q = 0.3$.

It is clear that we can only analyze *finite* incident chains; however, it is not immediately clear what the length of such chains should be. One might argue that longer chains will lead to a larger performance difference between online and offline solutions - simply because the offline solution is able to look further into the future. On the other hand, it seems reasonable to assume that incidents that are *very* far in the future do not greatly affect current decisions. Thereto, we analyzed incident chains of four different lengths: 6, 12, 18 and 24 hours. One might also argue that the result depends on the value of λ, thereto we analyzed three different values (λ_1, λ_2 and λ_3, as described in Table 3).

The parameters described above lead to the analysis of 12 different cases.

[7]This number may seem small, but note that this includes some ambulances that do not need to transport the patient to a hospital

For each of those cases, we drew $|S|$=1000 incident chains according to the Poisson process described in Section 2.1, for the region Utrecht defined in Table 3.

In order to compute the optimal offline performance, we implemented all three methods from Section 3. First, we tried the CP, which we implemented in the MiniZinc modeling language, using its standard G12 finite domain solver. This could only handle very small problem instances. The largest instances we tried to solve with CP had a simulation time of six hours. The computation time varied widely among the different instances, the longest ones taking more than a day. Next, we implemented the DP in C++, which reduced computation times (again for instances of six hours simulation time) to a range of 20 minutes to a few hours. Finally, we implemented the BLP in Java using solver CPLEX 12.6, which solves all instances, including ones for 24 hours simulation time, in a fraction of a second. As stated in Section 3, the performance of the two online dispatch policies is calculated by simulating the EMS system.

Ambulance optimization is a complex topic, and it is often hard to oversee whether stated theoretical results will hold up in practice - even for experts. It is our opinion that in order for results to be meaningful, at least the performance should be close to the performance in practice. In the Netherlands, urgent incidents should be served within the time standard in at least 95% of all cases. The ambulance provider for Utrecht performs slightly better than this 95% on average. Thereto, we decided to use a number of vehicles such that the average fraction of late arrivals for the online dispatch methods is roughly between 3 and 5%. We believe that this choice leads to the most realistic and insightful results.

The obtained fraction of late arrivals for each of the 12 cases is depicted in Figure 3. Recall that we compute the performance ratio as described in Equation 2. The results from Figure 3 then lead to the performance ratios found in Table 4.

A bound on optimal online algorithms

Our offline optimum constitutes the first known bound on the performance of an optimal online ambulance dispatch policy. Above, we have shown that the DMEXCLP dispatch policy performs approximately 1.9 times worse than

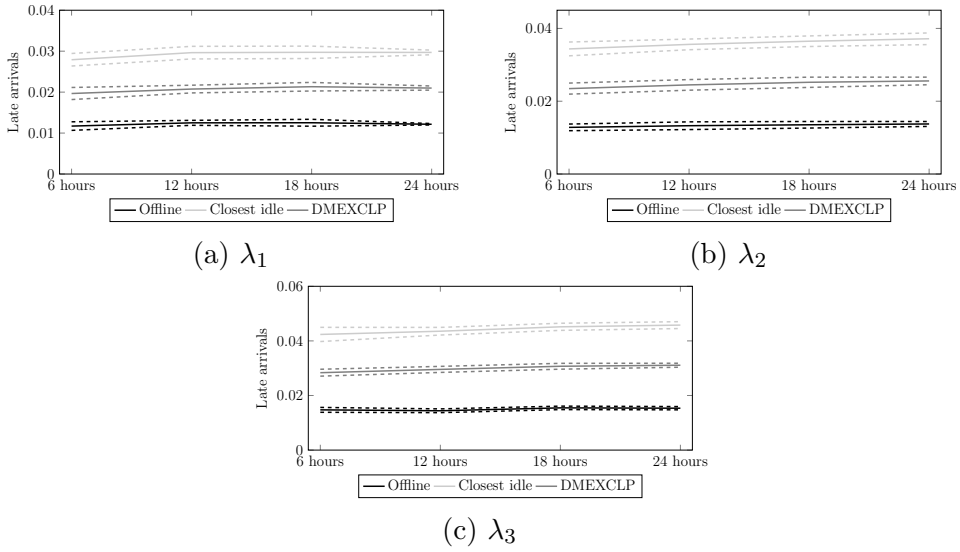


Figure 3: Average fraction of late arrivals for 1000 chains of incidents, with different incident intensities and chain lengths. A 95% confidence interval is displayed.

	λ_1	λ_2	λ_3
DMEXCLP			
6 hours	1.72 ± 0.07	1.86 ± 0.07	1.96 ± 0.12
12 hours	1.67 ± 0.05	1.87 ± 0.07	2.06 ± 0.06
18 hours	1.72 ± 0.07	1.88 ± 0.07	2.00 ± 0.05
24 hours	1.74 ± 0.05	1.87 ± 0.05	2.04 ± 0.06
Closest idle			
6 hours	2.48 ± 0.19	2.73 ± 0.11	2.91 ± 0.17
12 hours	2.39 ± 0.09	2.72 ± 0.14	3.05 ± 0.09
18 hours	2.40 ± 0.11	2.73 ± 0.11	2.94 ± 0.09
24 hours	2.46 ± 0.06	2.72 ± 0.10	3.00 ± 0.09

Table 4: The observed Performance Ratio and 95% confidence interval of online dispatch policies.

the offline optimum[8]. This means that there cannot exist an online dispatch

[8]Note that for this numerical result, we focused on λ_2, since it is realistic for this particular EMS region. Furthermore, Table 4 indicates that the Performance Ratio does not vary greatly between cases of 12, 18 and 24 hours. Therefore we use the 24 hour case as a estimate of the true Performance Ratio.

method that improves the performance of the DMEXCLP dispatch method by more than a factor 1.9 on average. We emphasize that this bound is an optimistic one, since it is obtained using information - on future incidents - that is inaccessible to online policies, but it is a bound nonetheless.

The value of information
Generally speaking, the competitive ratio of a problem shows the importance of knowing the future for this problem. In terms of the ambulance dispatch problem, it gives an indication of 'unfortunate decisions' made by online policies - even an optimal one - that could not have been avoided unless one knew about future incidents. Our results are perhaps surprising: the authors of this paper had previously expected that knowing incidents in advance would have a greater impact on performance. However, our results show that even an omniscient dispatcher will still be left with $\frac{1}{1.9} \approx 53\%$ of the late arrivals, compared to a dispatcher that executes DMEXCLP.

6. Discussion

We have introduced three methods to compute the offline optimal solution to the ambulance dispatch problem. Note that, due to scalability issues, the CP and DP method are not advisable for most numerical work; we recommend the BLP to solve the problem practically.

One may perform the analysis as described in this paper for multiple EMS regions. Different regions typically have different characteristics, such as the average busy fraction of ambulances, or the distance between bases and demand. These differences will most likely result in a different online/offline performance ratio, and it would be interesting to see how these ratios vary over different regions. However, regions are always hard to compare, and therefore instead of simulating different regions we chose to analyze the effect of different arrival intensities.

Table 4 shows that the Performance Ratio between the online policies and the offline optimum increases with λ. This may be explained as follows. A larger λ leads to more incidents within a short time frame. As we have seen in Appendix B, this makes for a more complex problem, because many decisions are now dependent on one another. In particular, an unfortunate choice at some point can have an effect on many incidents after that. It is therefore not surprising that the gap between the online heuristics and the offline optimum increases with λ.

Although it was previously known that the closest idle method is not optimal, it is often assumed to be quite a good policy. In fact, the insight that the 'closest idle' performance is still a factor 2.7 away from the offline optimum, is something that many researchers in the field of ambulance planning may be tempted to attribute to the *value of information*: to the fact that the offline policy has much more knowledge. However, as Figure 3 depicts, the DMEXCLP dispatch heuristic is able to close about half the gap between the 'closest idle' and the offline optimum. We find this observation rather surprising, as it implies that the value of information for the dispatch problem is smaller than we had previously anticipated.

In order to compute the Performance Ratio, we drew random chains of incidents. However, we always started at time 0 with all ambulances idle. This may perhaps be interpreted as the start of the day, for EMS providers that serve few calls at night. However, one might also argue that we should focus more on the system in *steady state*. We conjecture that our result - a Performance Ratio of 1.9 - will roughly hold for steady state as well, since the value did not change much between incident chains of 12, 18 and 24 hours (see Table 4).

Finally, one might suggest to make the model more realistic, e.g., by defining the busy time of an ambulance after arrival at an incident to be a random variable. Then, however, determining the optimal offline policy becomes a very difficult task. We see only one way to overcome this difficulty, and that is to let the offline solution have knowledge of the *realizations* of these random times. Since online policies can only use the busy times in *distribution*, this would increase the gap between information given to the offline and online policies. This deviates further from our main research question, which was how much it helps to have information on when and where incidents will occur. Increasing the gap between what is known in the online and offline case will not help us to gain more insight in this matter. Therefore, we decided not to proceed in this direction.

Acknowledgements

We thank the Dutch National Institute for Public Health and the Environment (RIVM) for giving access to the travel times for EMS vehicles in the Netherlands. The first author of this paper would like to thank Kim Mariott, Mark Carman and Maria Garcia de la Banda from Monash University, Melbourne, for introducing her to Constraint Programming in general, and

MiniZinc in particular. We thank Gerhard Woeginger for his suggestions regarding the dynamic programming solution. This research was financed in part by Technology Foundation STW under contract 11986, which we gratefully acknowledge.

Appendix A. Constraint Programming Formulation

In this appendix, we describe how we found the optimal offline solution with CP. For this purpose, we used the MiniZinc constraint modeling language. We modelled a set of n incidents by the following variables:

- $t(c)$, the c^{th} element of vector $\vec{t}$. This is the time that incident c occurs, $c \in \{1, \ldots, n\}$.
- $loc(c)$, the c^{th} element of vector $\vec{loc}$. This is the location of each incident, where $loc(c) \in V \quad$ for $c \in \{1, \ldots, n\}$.

The input further consists of the base location W_a of each ambulance a, as well as the driving times $\tau_{i,j} \quad \forall i, j \in V$ (in minutes). For each incident c we introduced a variable $\mathscr{A}(c)$, which can take a value between 1 and $|A|$. These variables indicate which ambulance is assigned to each incident.

We aimed to minimize the fraction of arrivals later than threshold time T. Note that since the number of incidents - and therefore the number of arrivals - is known in advance, this is equivalent to minimizing the *number* of late arrivals. In our implementation, we focused on the number of late arrivals, denoted by $\mathscr{N}$.

Finally, we needed to ensure feasibility of the solution. Thereto, we added two constraints[9]. Equation (A.1) makes sure variable $\mathscr{N}$ is set correctly, i.e., it is the number of incidents for which the dispatched ambulance was further than T minutes away. Equation (A.2) ensures that two incidents (c_1 and c_2) assigned to the same ambulance do not overlap in time.

Note that this is not the only CP model one could formulate. In fact, a model similar to the BLP model that we have seen in Section 3.1 is also possible for CP. However, we chose to keep the models diverse.

minimize $\mathscr{N}$

[9]We also added other - redundant - constraints in order to find solutions faster. However, they do not change the result and therefore we do not mention them here.

s.t.

$$\mathscr{N} = \sum_{c \in C} \mathbb{1}(\tau_{W_{\mathscr{A}(c)}, loc(c)} > T) \tag{A.1}$$

and

$$\begin{aligned} &\nexists\ c_1, c_2 \in C \quad \text{such that} \\ &\quad c_1 < c_2 \quad \wedge \quad \mathscr{A}(c_1) = \mathscr{A}(c_2) \quad \wedge \quad t(c_1) + \tau_{W_{\mathscr{A}(c_1)}, loc(c_1)} + X > t(c_2). \end{aligned} \tag{A.2}$$

Note that in the formulation of Equation (A.2), we used the assumption that incidents are ordered chronologically.

One can immediately see the benefit of the flexibility that CP has to offer: we were able to write the problem using just two constraints, which look very similar to the way one might naturally think about the dispatch problem.

Appendix B. Dynamic Programming Formulation

In this section, we describe how we built the dynamic program (DP), and what extra features could be added to it in order to speed up the computation. Note that we only need to make decisions right after an incident has occurred. Therefore, we define states at time steps that coincide with the incidents - and just like the incidents, we denote them c from 1 to n. That way, time step c corresponds to the actual time $t(c)$. Additionally, we add a dummy time step $n + 1$, with $t(n + 1)$ large enough such that all vehicles are idle again, regardless of the dispatch decisions made in the past. The only allowed action at this time step is a dummy action with reward 0.

States, actions and rewards

We define our states to be vectors containing the time in minutes until each ambulance becomes idle. This implies that a state s is a vector of length $|A|$, the number of ambulances in the system. Let $s[a]$ denote the number of minutes until ambulance a becomes idle, for $a \in A$. If this is 0 minutes, that means the vehicle is already idle. At time 0, nothing has happened yet, and all ambulances are idle. Therefore, we start with the zero vector, having a value of 0. In any state s, the allowed actions, i.e., the ambulances that are

eligible for dispatch, are given by: $a \in A$ for which $s_c[a] = 0$. At time step c, $c \leq n$, the penalty corresponding to action a_c ($a_c \in A$) is given by

$$R(s_c, a_c) = \begin{cases} 1 & \text{if } \tau_{W_{a_c, loc(c)}} > T; \\ 0 & \text{otherwise.} \end{cases}$$

Note that the reward for $c = n + 1$ is defined as 0.

To know how to update the states, we can precompute the time differences between the incidents. Thereto, we define:

$$\mathit{diff}_c = t(c + 1) - t(c) \quad \text{for } c \in C,$$

Next is described how to update any state s_c to state s_{c+1}, where a_c denotes the chosen action in time step c. Let Γ be the transition function, that depends on s_c and a_c. Define $s_{c+1} = \Gamma(s_c, a_c)$ such that

$$s_{c+1}[a] = \begin{cases} \max(\tau_{W_{a, loc(c)}} + X - \mathit{diff}_c, 0) & \text{if } a = a_c; \\ \max(s_c[a] - \mathit{diff}_c, 0) & \text{otherwise.} \end{cases}$$

The value of being in state $s_{c'}$ at time step c' can then be defined as:

$$V_{c'}(s) = \min_{\{a_c\}_{c=0}^{c'}} \sum_{c=0}^{c'} R(s_c, a_c)$$

subject to

$$\mathrm{a}_c \in A \text{ and } s_c[a_c] = 0$$

and

$$s_{c+1} = \Gamma(s_c, a_c), \quad \forall c = 0, 1, 2, \ldots n$$

The objective is to minimize the fraction of late arrivals, which - for any fixed number of incidents - is equal to minimizing the *number* of late arrivals. So we are interested in the value $V_{n+1}(\vec{0})$.

Note that decisions made in the past have a large effect on the set of states that we need to analyze in the future. In fact, only a small subset of all states we can think of, will ever be reached. That is, one can obtain s_{c+1} from s_c, but not the other way around. Therefore, a backward recursion does not make sense for this problem; instead, we used a *forward recursion*

to obtain the set of states that we need to analyze. Hence, for each state s, we computed the value at time step c based on the value in the previous time step, as follows.

$$V_{c+1}(s_{c+1}) = \min_{a_{c+1}}\{V_c(s_c) + R(s_{c+1}, a_{c+1})\}.$$

Although this method in theory computes the optimal solution to any instance of the offline ambulance dispatch problem, practical difficulties can occur. Just like in the Constraint Programming approach, the difficulty is that many situations need to be considered (in the Dynamic Programming case, that means many states need to be stored).

Note that it is hard to give an exact formula that describes the number of states that need to be computed in order to find the solution. There are, however, two formulas that both give an upper bound on the number states. The first one follows straightforwardly when one realizes that the time until each ambulance becomes available completely defines a state (and that we should consider this n times). This means there are at most $nM^{|A|}$ relevant states, where M is the maximum driving time between any base location and demand point. Furthermore, there is a maximum on the number of decisions that can be made. Assuming all possible combinations of ambulance assignments are allowed, this leads to a maximum $|A|^n$ decisions, and hence states, to be considered.

There are some ways to reduce the total number of states required to store, which directly lead to shorter computation times. We next describe three ways to accomplish this.

Appendix B.1. DP speed-up

In this appendix, we describe three ways to speed up the computation time of the DP. We illustrate the usefulness of each technique, by the effect it has on the following two problem instances[10]. In both examples, $T = 12$.

Instance 1.

$$\vec{t} = [9, 13, 35, 47, 70, 95, 104, 105, 115, 127, 152, 169].$$
$$\vec{loc} = [34, 54, 23, 159, 81, 81, 39, 10, 142, 146, 140, 156].$$

[10]The cases considered here are for the region Utrecht. Due to the scalability issues of the DP, we used only 10 ambulances in this example.

(For this instance, the closest idle dispatch policy results in one late arrival. The offline optimum is also one late arrival.)

Instance 2.

$$\vec{t} = [1, 1, 30, 33, 34, 43, 43, 62, 63, 81, 103, 114, 124, 135, 138, 139, 168, 174].$$
$$\vec{loc} = [200, 182, 135, 217, 67, 131, 74, 179, 95, 15, 74, 37, 206, 206, 142, 54, 145, 44].$$

(For this instance, the closest idle dispatch policy results in four late arrivals. The offline optimum is equal to three late arrivals.)

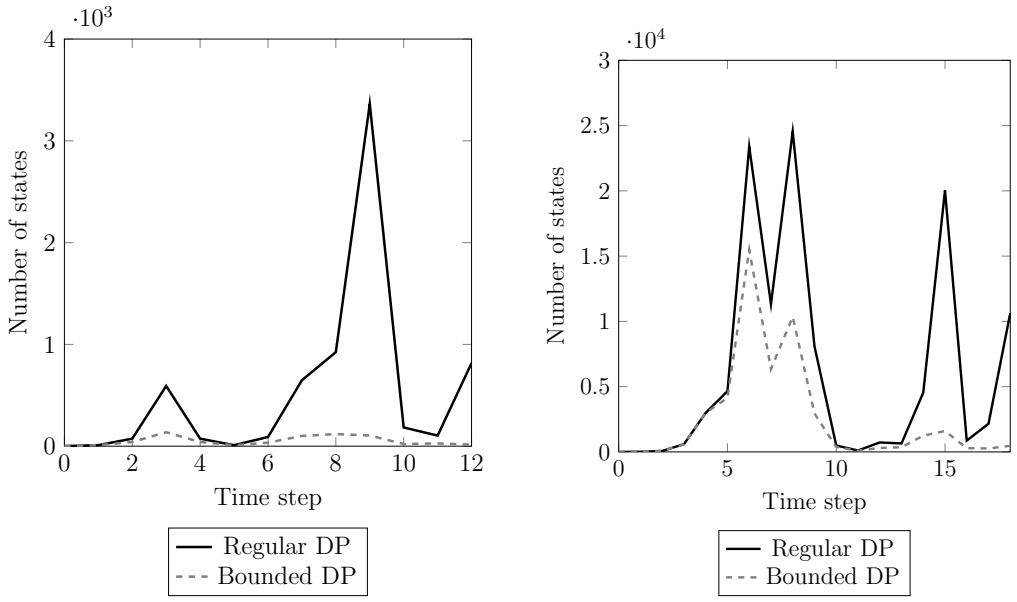
Eliminating dominated states
One well-known way to reduce the number of states, is to eliminate so-called dominated states. We define a state s to be dominated at time c, if there exists another state s', such that:

$$s'[a] \leq s[a] \quad \forall a \in A \quad \text{and} \quad V_c(s') \leq V_c(s).$$

That is, there exists another state for which all vehicles will be idle at earlier (or equal) times, while resulting in fewer (or equal) late arrivals. We iteratively removed dominated states until none are left in our state space.

Bounding the objective
Another way to reduce the time and memory spent on the dynamic program, is to bound the solution by any feasible objective value. For example, we can quickly pre-compute the objective from the 'closest idle' dispatch heuristic and eliminate any state that has a larger value. We show the benefit of this approach by example: Figure B.4 depicts the number of states that we need to analyze at each time step. (Note that Figure B.4b has more time steps than Figure B.4a, simply because more incidents occur.)

Figure B.4 shows that, in the first few time steps, the number of states for the bounded and unbounded DP are more or less equal. Let us explain why this makes sense, by the example of Instance 2. Here, the closest idle method results in four late arrivals. Therefore, bounding the states by the ones with values ≤ 4 does not have any effect before time $c = 5$ (since the



(a) Instance 1 (b) Instance 2

Figure B.4: Comparison of the number of states stored for each time step in the dynamic program, with and without bounding the solution by the value of the 'closest idle' policy.

value can increase by at most 1 per incident).[11]

Also note that the number of states does not always increase over time. So what is it exactly, that causes the need to store many states? A key insight is that an incident that occurs at time t, only has an *indirect* effect on the system[12] after time $t+\tau_{i,j}+37$, for some travel time $\tau_{i,j}, i, j \in V$. That is, the ambulance will be idle by time $t + \tau_{i,j} + 37$, and can be used for any incident after that time, regardless of whether it is dispatched to the incident at time t. However, whether or not this particular ambulance is dispatched at time t, *does* have an effect on which ambulances are eligible for dispatch to incidents between time t and $t + \tau_{i,j} + 37$. Hence, we regard the effect

[11] An alternative, more elaborate way to bound the DP would be to store all states that the (online) heuristic passes through over time. That is, after each incident, store the remaining busy time for each ambulance, as well as the number of late arrivals observed in the past. Then, when computing the offline optimum in the DP, remove all states that are dominated by the states observed in the online solution. Note that we did not implement this idea.

[12] assuming that we never run out of ambulances

as an indirect one. Note that this indirect effect occurs *only* when incidents arise within this time frame. This leads to the following observation.

Observation 1. *Longer inter-arrival times lead to a reduction in the number of states.*

In particular, if the inter-arrival time between two consecutive incidents is larger than 37 minutes plus the response time from any base to the first incident (of the two), then the state space reduces to a single state (all ambulances are idle). This can be viewed as a 'reset' of the system, i.e., all information on past decisions are irrelevant for future decisions. When this occurs, it avoids an explosion of the state space that we would otherwise see for instances with a long horizon.

Roughly speaking, the computation time should scale linearly with the simulation horizon (given a fixed λ). However, what makes an instance harder to solve is the number of incidents that occur quickly after one another. Many incidents in a short time window imply many *dependent* decisions - and precisely this increases the number of states. This effect can be seen in the 'spikes' in Figures B.4a and B.4b (one can manually confirm this using the incidents times given in Instance 1 and 2. Conversely, the time steps with a very small number of states correspond to large inter-arrival times between incidents.

Rounding the numbers in the input
Another way to speed up the computations is by rounding the driving times and incident times. For example, instead of rounding to minutes, we could round the times to multiples of five minutes. The reason why this would result in fewer states, is that more states will be dominated.

Unfortunately, rounding means some accuracy will be lost: we make use of a trade-off between running time and accuracy here. At the very least, one should make sure to also round the times in the input for the heuristic solutions, or else the computed ratio is meaningless. Then, one might argue that the computed ratio will be similar to the unrounded case. Note that we did not implement this method, but instead suggest to use a Binary Linear Programming approach as described in Section 3.1.

References

[1] R. Alanis, A. Ingolfsson, and B. Kolfal. A Markov chain model for an EMS system with repositioning. *Production and Operations Manage-*

ment, 22(1):216–231, 2013.

[2] G. Carter, J. Chaiken, and E. Ignall. Response areas for two emergency units. *Operations Research*, 20(3):571–594, 1972.

[3] R.L. Church and C.S. Revelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32:101–118, 1974.

[4] J. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

[5] M.S. Daskin. A maximum expected location model: Formulation, properties and heuristic solution. *Transportation Science*, 7:48–70, 1983.

[6] J. Goldberg, R. Dietrich, J.M. Chen, and M.G. Mitwasi. Validating and applying a model for locating emergency medical services in Tucson, AZ. *Euro*, 34:308–324, 1990.

[7] C.J. Jagtenberg, S. Bhulai, and R.D. van der Mei. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care*, 4:27–35, 2015.

[8] CJ. Jagtenberg, S. Bhulai, and R.D. van der Mei. Dynamic ambulance dis- patching: is the closest-idle policy always optimal? *Operations Research for Health Care*, pages 1–15, 2016.

[9] R.M. Karp. On-line algorthms versus off-line algorithms: How much is it worth to know the future? *International Federation for Information Processing Congress*, 12(1):416–429, 1992.

[10] M. Manasse, L.A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[11] M.S. Maxwell, E.C. Ni, C. Tong, S.R. Hunter, S.G. Henderson, and H. Topaloglu. A bound on the performance of an optimal ambulance redeployment policy. *Operations Research*, 62(5):1014–1027, 2014.

[12] M.S. Maxwell, M. Restrepo, S.G. Henderson, and H. Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22:226–281, 2010.

[13] E. Melachrinoudi, A.B. Ilhan, and H. Min. A dial-a-ride problem for client transportation in a health-care organization. *Computers and Operations Research*, 34(3):742–759, 2007.

[14] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A Heuristic Two-Phase Solution Approach for the Multi-Objective Dial-A-Ride Problem. *Networks*, 54(4):227–242, 2009.

[15] U. Ritzinge, J. Puchinge, and R.F. Hartl. Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research*, pages 1–18, 2014.

[16] V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219:611–621, 2012.

[17] J.T. van Essen. *Flowing through hospitals, Chapter 3.* PhD thesis, University of Twente, 2013.

[18] Yisong Yue, Lavanya Marla, and Ramayya Krishnan. An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment. In *AAAI Conference on Artificial Intelligence (AAAI)*, July 2012.