# Dynamic Ambulance Relocation

Abderrahim Akhnikh

Begeleider: Rob van der Mei

BWI werkstuk

*vrije* Universiteit
Faculteit der Exacte Wetenschappen
Studierichting Bedrijfswiskunde en Informatica
De Boelelaan 1081a
1081 HV Amsterdam

# Inhoudsopgave

# 1. Inleiding

Het is algemeen bekend dat ambulancediensten belangrijk werk doen. In Nederland voeren ambulancebroeders en zusters jaarlijks circa 970.000 ritten uit. De ritten zijn onder te verdelen in spoedeisende meldingen (A1- en A2-ritten) en planbare ritten (B-ritten).

A1-ritten zijn ritten waarbij de patiënt in direct levensgevaar verkeert of waarbij blijvende invaliditeit dreigt. De ambulances rijden in die gevallen met zwaailichten en sirenes. De Nederlandse ambulancesector hanteert de norm dat bij alle A1-meldingen de ambulances binnen 15 minuten bij de patiënt aanwezig moeten zijn. In 91% van de gevallen wordt die norm ook gehaald.[1]

A2-ritten zijn ritten waarbij de patiënt niet in direct levensgevaar verkeert, maar waarbij wel ernstige gezondheidsschade dreigt. In de meeste A2-gevallen rijden de ambulances zonder optische en geluidssignalen. Voor A2-meldingen wordt de norm gehanteerd dat de ambulances binnen 30 minuten aanwezig moeten zijn.

B-ritten zijn ritten waarbij geen spoed vereist is en worden vaak vooraf gepland. Het betreft veelal patiëntenvervoer tussen ziekenhuizen; patiënten die opgenomen of ontslagen worden, maar liggend vervoerd moeten worden etc. B-ritten vereisen weliswaar geen spoed, ze moeten vaak wel op tijd uitgevoerd worden, omdat de patiënt bijvoorbeeld een afspraak met een specialist heeft. Het grote voordeel is dat ze planbaar zijn; men weet vooraf dat de rit uitgevoerd moet worden.

Van de 970.000 ritten die de ambulancediensten jaarlijks uitvoeren is 62% spoedeisend (A1 of A2)[2]. Dit werkstuk zal zich focussen op de spoedeisende en niet planpare meldingen. Omdat die vanuit een stochastisch oogpunt het interessantst zijn.

Omdat ambulancepersoneel vaak onder grote tijdsdruk staat is het belangrijk om in ieder geval de planning goed te hebben. Maar hoe kun je plannen als je niet weet wanneer of waar incidenten gebeuren? Is het mogelijk om vooruit te kijken? Misschien zijn er in de winter meer meldingen dan in de zomer en met oud en nieuw weet je bijna zeker dat het druk wordt. Forecasting is dus weldegelijk mogelijk.

Locaties vormen een andere belangrijke dimensie van ambulanceplanning. Hoeveel ambulanceposten zijn voor een stad als Amsterdam nodig? En waar moeten die dan staan? En als er eenmaal een melding is en er moeten ambulances gaan rijden, kunnen de andere ambulances dan hun plek innemen om de dekking te handhaven? En hoe bepaal je dan in real-time welke ambulance naar welke post moet? Ambulance logistiek kan opgedeeld worden in drie “disciplines”:

1. Planning en forecasting.

[1] Bron: Ambulancezorg Nederland, *Ambulances in-zicht, 2007*, (2008)
[2] Zie voetnoot 1.

2. Dispatching.
3. Ambulance Relocation.

Planning en forecasting hebben betrekking op het vooraf inschatten van het aantal meldingen en hun locaties. Dit kan door bijvoorbeeld data van ambulancemeldingen te analyseren en een statistische schatting te maken van het aantal te verwachten meldingen op een bepaalde (week)dag[3].

Dispatching is het proces dat ambulances aan meldingen toewijst. Met andere woorden: bepalen welke ambulance naar welke melding moet.

Ambulance Relocation behelst het aanpassen van de planning gedurende de dag om in te spelen op de real-time situatie en de totale dekking hoog te houden. Als in een bepaalde regio meer meldingen plaatsvinden dan verwacht, dan kan de dekking te laag worden om op hand zijnde meldingen tijdig te kunnen bedienen. Als dan in een andere regio ambulances stand-by staan kunnen ze wellicht naar de drukkere regio's gestuurd worden om ondersteuning te bieden. Dit kan gezien worden als het aansluiten van de vooraf gestelde verwachting (planning) met de realisatie.

Dit werkstuk zal zich toespitsen op Ambulance Relocation. Het doel is om de dynamiek achter het probleem te onderzoeken. In het eerste deel wordt een literatuuronderzoek gepresenteerd om enkele voorgestelde benaderingen te bekijken. Daarna wordt een modelsituatie van een kleine stad geformuleerd dat het probleem benaderd. Vervolgens wordt het probleem opgelost om zo meer inzicht te krijgen in de beste beslissingen.

# 2. Literatuur

Ambulance Relocation is een relatief nieuwe discipline in ambulancelogistiek. Er is dan ook een beperkte hoeveelheid wetenschappelijke artikelen beschikbaar. Toch zijn er artikelen die interessante gedachtegangen volgen. In dit hoofdstuk zullen die behandeld worden. Het doel is om een beeld te schetsen van de bestaande benaderingen en technieken.

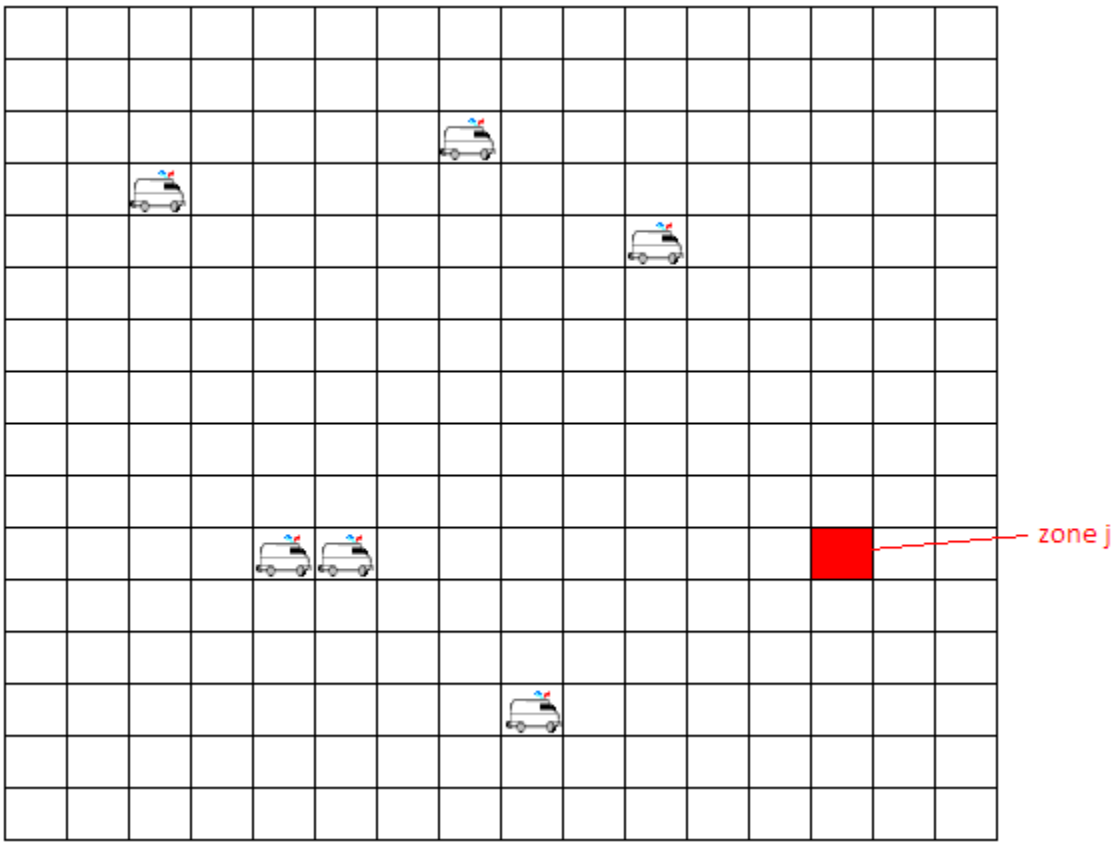
## 2.1. "Decision support tools for ambulance dispatch and relocation."[4]

Dit artikel behandelt zowel het ambulance dispatching probleem als het ambulance relocation probleem. Het begint met het formuleren van een kwantificering van de dekkingsgraad en behandelt vervolgens een methode die de dekkingsgraad zo hoog mogelijk moet houden.

[3] S. Ait Haddou Ou Ali, *Data analyse en modellering van het aantal ambulance oproepen per dag*, BWI-werkstuk Vrije Universiteit, (2009)
[4] Tobias Andersson Granberg and Peter Värbrand, *Decision support tools for ambulance dispatch and relocation*, 2007, Journal of the Operational Research Society, (58), 195-201.

## 2.1.1. Het probleem.

De schrijvers van het artikel delen een stad op in *N* zones die samen een rooster vormen:




**Figuur 1. De stad wordt opgedeeld in een rooster van *N* zones.**

Er zijn in totaal $A$ ambulances. Elke zone wordt in een bepaalde mate gedekt door een aantal ambulances. Dat zijn de ambulances die binnen afzienbare tijd in die zone kunnen zijn. De 'vraag' naar ambulances in zone $j$ wordt weergegeven door een gewicht $c_j$. Het aantal ambulances dat bijdraagt aan de dekking in zone $j$ is $L_j$ ($\leq A$). De reistijd van ambulance $l$ naar zone $j$ is $t_j^l$. De mate waarin ambulance $l$ bijdraagt aan de dekking in zone $j$ wordt weergegeven door $\gamma^l$, waarbij geldt dat:

$$t_j^1 \leq t_j^2 \leq t_j^3 \leq \ldots \leq t_j^{L_j}$$
$$\gamma^1 > \gamma^2 > \gamma^3 > \ldots > \gamma^{L_j}$$

Dat betekent dat hoe groter de reistijd, hoe kleiner de bijdrage.

Het artikel formuleert voor zone $j$ de dekking $p_j$ als volgt:

$$p_j = \frac{1}{c_j} \sum_{l=1}^{L_j} \frac{\gamma^l}{t_j^l}$$

De dekking wordt dus berekend door de $L_j$ ambulances die zich het dichtst bij zone $j$ bevinden te laten bijdragen aan de dekking in zone $j$. Hun bijdrage is omgekeerd evenredig met de reistijden naar zone $j$. Hoe dichter een ambulance bij een zone komt, des te hoger wordt de dekking. Daarnaast kan opgemerkt worden dat hoe hoger de vraag $c$ is, des te lager de dekking wordt.

De meldingen die binnenkomen krijgen prioriteit 1, 2 of 3 (vergelijkbaar met A1, A2 en B). Het spreekt voor zich dat meldingen met een hoge prioriteit voorrang hebben op meldingen met een lage prioriteit. Echter, meldingen met een lage prioriteit die al onredelijk lang wachten kunnen een hogere "pseudo" prioriteit krijgen om alsnog bediend te worden.

De schrijvers stellen een minimale dekking $P_{min}$, de $\gamma$'s en per zone $c_j$ vast. Vervolgens berekenen ze voor iedere zone de dekking. Het doel is om in elke zone waar $p < P_{min}$ de dekking zo snel mogelijk naar $P_{min}$ te verhogen. Dit kan alleen door ambulances dichter bij zone $j$ te brengen. De schrijvers formuleren een model, DYNAROC, dat het probleem beschrijft:

$$\min \quad z$$

$$z \geq \sum_{j \in N^k} \tau_j^k x_j^k \qquad k = 1,\ldots, A$$

$$\sum_{j \in N^k} x_j^k \leq 1 \qquad k = 1,\ldots, A$$

$$s.t. \quad \sum_{k=1}^{A} \sum_{j \in N^k} x_j^k \leq M$$

$$\frac{1}{c_j} \sum_{l=1}^{L_j} \frac{\gamma^l}{t_j^l(\boldsymbol{x})} \geq P_{min} \qquad j = 1,\ldots, N$$

$$\boldsymbol{x} \in \{0,1\}$$

De doelfunctie is om $z$, de maximale reistijd van de verplaatste ambulances, te minimaliseren. Met andere woorden: de tijd totdat de dekking in alle zones weer boven $P_{min}$ is, wat in een van de restricties wordt afgedwongen, moet geminimaliseerd worden.
De "beslissingsvariabele" $x_j^k$ is 1 als ambulance $k$ naar zone $j$ verplaatst wordt. Daarnaast bestaat de restrictie dat ten hoogste $M$ ambulances verplaatst kunnen worden. Een ambulance kan naar maximaal 1 zone verplaatst worden.

### 2.1.2. De oplossing.

De schrijvers stellen een oplosmethode voor dat gebruik maakt van een zoekboom:

1. Let the current (infeasible) solution, i.e. $x_j^k = 0 \ \forall j,k$, be the root of the tree.
2. Let $j$ = the zone with the lowest preparedness.
3. REPEAT
4. Find the $n$ ambulances, with the minimum travel times, that can be relocated in a way that ensures that $p_j \geq P_{min}$. Save a maximum of $m$ zones for each of these ambulances that satisfy the conditions above. The ambulances must not have been relocated once already (earlier in the tree) and not more than $M-1$ ambulances must have been moved.
5. Each of the moves in Step 4 gives a potential solution.
   FOR all new solutions
6. Check if $p_i \geq P_{min} \ \forall i = 1, \ldots, N$ and check the longest travel time against the best solution found so far if this is true.
   If the new solution is not feasible, create a new node and connect it to its parent solution.
7. Pick a new node and let $j$ = the zone with the lowest preparedness in the new solution.
8. UNTIL there are no nodes left to examine, or some other stop criterion triggers

Het komt erop neer dat voor iedere zone waar $p < P_{min}$ de $n$ beste verplaatsingen worden geëvalueerd waarmee zone $j$ weer boven het minimum zit. Daarna wordt gecontroleerd of na de verplaatsing $p \geq P_{min}$ geldt voor de andere zones. En het algoritme wordt herhaald totdat alle mogelijkheden geprobeerd zijn, $M$ verplaatsingen toegepast zijn of een ander stop criterium bereikt is. De rekentijd voor een modelstad gebaseerd op de data van Stockholm is gemiddeld ongeveer 2 seconden en maximaal 6 seconden.

### 2.1.3. De resultaten.

Het aantal verplaatsingen per dag zijn dan hieronder weergegeven:

**Tabel 1. Het aantal verplaatsingen per dag voor een gegeven aantal meldingen per dag.**

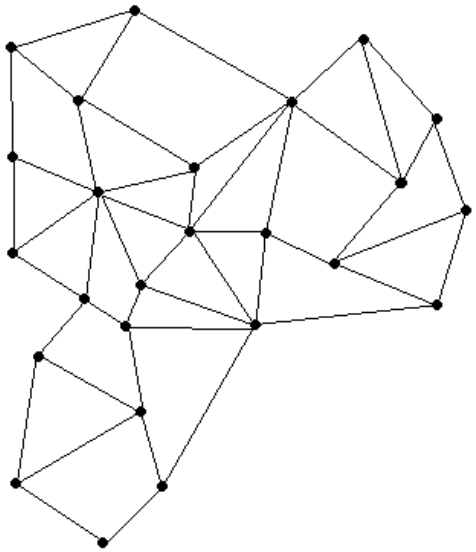
| nr of calls / day | 200 | 250 | 300 | 350 | 400 | 450 | 500 | 550 | 600 | 650 | 700 | 750 | 800 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_{min} = 0.277$ | 4.4 | 6.7 | 9.3 | 10.2 | 14.7 | 19.6 | 25.1 | 33.9 | 51.6 | 65.1 | 101.5 | 146.2 | 223.6 |
| $P_{min} = 0.923$ | 105.5 | 148.7 | 194.0 | 253.4 | 331.4 | 412.5 | 500.4 | 578.5 | 662.2 | 700.9 | 752.3 | 778.4 | 819.6 |

Het is te zien dat het aantal verplaatsingen per dag vrij laag begint maar al snel tot zelfs boven het aantal meldingen groeit. Dit zou, indien geïmplementeerd, ambulancepersoneel zwaar belasten.

Het aantal meldingen dat binnen de norm behandeld wordt stijgt wel ten opzichte van een situatie zonder verplaatsingen.

## 2.2. "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation."[5]

Dit artikel modelleert de stad als een graaf $G = (V \cup W, E)$ waarbij $V = \{v_1, ..., v_n\}$ en $W = \{v_{n+1}, ..., v_{n+m}\}$ sets van knooppunten zijn. V is de set van vraagpunten waar de meldingen plaats kunnen vinden en W de mogelijke ambulancestandplaatsen. E is de set van verbindingen tussen de knooppunten.



**Figuur 2. Een voorbeeld van een graaf *G*. In werkelijkheid zullen er veel meer knooppunten en verbindingen zijn.**

[5] M. Gendreau et al., *A dynamic model and parallel tabu search heuristic for real-time ambulance relocation,* (2000).

## 2.2.1. Het probleem.

De vraag naar ambulances in een knooppunt is $\lambda_i$. Elke verbinding heeft een reistijd van $t_{ij}$. De schrijvers hanteren de volgende prestatienorm:

Alle meldingen moeten binnen $r_2$ minuten bediend zijn. Daarvan moet een fractie ter grootte $\alpha$ binnen $r_1$ minuten bediend zijn ($r_2 > r_1$).

Voor een meldingspunt $v_i$ en ambulancelocatie $v_j$ definiëren de schrijvers:

$$\gamma_{ij} = \begin{cases} 0, & t_{ij} > r_1 \\ 1, & t_{ij} \le r_1 \end{cases} \qquad \delta_{ij} = \begin{cases} 0, & t_{ij} > r_2 \\ 1, & t_{ij} \le r_2 \end{cases}$$

Het aantal beschikbare ambulances is $p$. Het maximale aantal ambulances dat op een ambulance locatie $v_j \in W$ kan wachten is gelijk aan $p_j$. De schrijvers definiëren een boete van $M^t_{jl}$ voor een verplaatsing van ambulance $l=1,\ldots,p$ naar locatie $v_j \in W$. Die boetes zijn niet alleen gebaseerd op de locatie van de ambulance op tijdstip $t$, maar ook op historische informatie. De boetes zijn bijvoorbeeld hoger als een ambulance al meerdere keren verplaatst is. De schrijvers onderstrepen dat de boetes na elk tijdstip $t$ geüpdate moeten worden. Het doel van de boetes is om bepaalde verplaatsingen te ontmoedigen.

Vervolgens definiëren de schrijvers de beslissingsvariabelen; $y_{jl}$ is 1 als ambulance $l$ zich in $v_j \in W$ bevindt. $x_i^k$ is 1 als $v_i \in V$ door minimaal $k$ ambulances gedekt wordt, i.e. er zijn $k$ ambulances die binnen $r_1$ minuten bij $v_i$ kunnen zijn. Merk op dat $x$ zelf geen beslissingsvariabele is, maar afhangt van $y$. De schrijvers gaan uit van $k = 2$ en formuleren het volgende probleem dat ze $RP^t$ noemen:

$$(RP^t) \qquad \text{maximize} \sum_{i=1}^{n} \lambda_i \, x_i^2 - \sum_{j=1}^{m} \sum_{\ell=1}^{p} M^t_{j\ell} \, y_{j\ell} \qquad (1)$$

$$\text{subject to :} \qquad \sum_{j=1}^{m} \sum_{\ell=1}^{p} \delta_{ij} \, y_{j\ell} \ge 1 \qquad \forall v_i \in V \qquad (2)$$

$$\sum_{i=1}^{n} \lambda_i \, x_i^1 \geq \alpha \sum_{i=1}^{n} \lambda_i \qquad (3)$$

$$\sum_{j=1}^{m} \sum_{\ell=1}^{p} \gamma_{ij} \, y_{j\ell} \geq x_i^1 + x_i^2 \qquad \forall \, v_i \in \mathrm{V} \qquad (4)$$

$$x_i^2 \leq x_i^1 \qquad \forall \, v_i \in \mathrm{V} \qquad (5)$$

$$\sum_{j=1}^{m} y_{j\ell} = 1 \qquad \ell = 1, \ldots, p \qquad (6)$$

$$\sum_{\ell=1}^{p} y_{j\ell} \leq p_j \qquad \forall \, v_j \in \mathrm{W} \qquad (7)$$

$$x_i^1 = 0 \text{ or } 1 \qquad \forall \, v_i \in \mathrm{V} \qquad (8)$$

$$x_i^2 = 0 \text{ or } 1$$

$$y_{j\ell} = 0 \text{ or } 1 \qquad \forall \, v_j \in \mathrm{W} \text{ and } \ell = 1, \ldots, p \qquad (9)$$

- Restricties 2 en 3 stellen de dekkingseisen vast, i.e. alle meldingspunten moeten binnen $r_2$ bereikbaar zijn waarvan een fractie α binnen $r_1$.

- Restricties 4 en 5 bepalen $x_i^1$ en $x_i^2$.

- Restricties 6 en 7 stellen respectievelijk dat iedere ambulance aan een mogelijke ambulance locatie toegekend is en dat maximaal $p_j$ ambulances aan locatie $j$ toegekend zijn.

### 2.2.2. De oplossing.

De oplossingsmethode die de schrijvers beschrijven is gebaseerd op de *taboe zoek heuristiek* van Glover en Laguna[6]. Dat houdt in dat eerst een verplaatsing voorgesteld wordt; als dan de dekkingsrestricties geschonden zijn, die met andere verplaatsingen herstellen, bewegingen die de restricties schenden taboe verklaren; en ten slotte de doelfunctie verhogen door niet-taboe verplaatsingen toe te passen. Merk op dat dit algoritme niet altijd tot het optimum leidt, maar hopelijk een goede benadering geeft.

De schrijvers structureren de heuristiek op een manier die het geschikt maakt voor *parallel computing*, i.e.: het probleem kan door meerdere computer tegelijk opgelost worden om rekentijd te besparen.

[6]F. Glover, M. Laguna. *Tabu Search.* Kluwer, Boston, 1997.

### 2.2.3. De resultaten.

De schrijvers gebruiken data van Urgences Santé in Montreal Canada om een test te simuleren. Men gebruikt 2521 meldingspunten (i.e. $n = 2521$). Verder neemt men voor $r_2$ een waarde van 15 minuten en voor $r_1$ een waarde van 7 minuten met $\alpha = 0{,}9$. Met andere woorden, alle meldingen moeten binnen 15 minuten bereikt worden en 90% van de meldingen moet binnen 7 minuten bereikt worden.

Daarnaast maken de onderzoekers gebruik van een benadering van rijtijden waarbij ze aannemen dat de rijtijden afhankelijk zijn van het tijdstip en stadsdeel:

**Tabel 2. De rijtijden die de schrijvers hanteren.**

| Period *t* | Number of ambulances | Speed (km/h) | | |
|---|---|---|---|---|
| | | Center | East | West |
| 5h-6h | 40 | 45 | 50 | 50 |
| 6h-7h | 50 | 40 | 45 | 50 |
| 7h-8h | 60 | 35 | 40 | 50 |
| 8h-9h | 60 | 35 | 40 | 50 |
| 9h-10h | 60 | 35 | 40 | 50 |
| 10h-11h | 58 | 35 | 40 | 50 |
| 11h-12h | 51 | 35 | 40 | 50 |

De schrijvers hebben een simulatie uitgevoerd van 7 ochtenduren (5.00-12.00) en komen tot de volgende resultaten:

- Alle meldingen zijn binnen 15 minuten bereikt.
- 98% van de meldingen zijn binnen 7 minuten bereikt.
- Het algoritme was in staat om in 95% van de gevallen met succes (en op tijd) een relocation te berekenen. Het lukte alleen niet als twee meldingen binnen 32 seconden achter elkaar binnenkwamen.
- 38% van alle melding vereist een relocation.
- Gemiddeld zijn er per relocation 2,08 ambulances verplaatst.
- De berekende relocation strategieën bleken binnen 2% van het optimum te liggen.

## 2.3. "Approximate Dynamic Programming for Ambulance Redeployment."[7]

De schrijvers van dit artikel kiezen voor een andere benadering. Ze modelleren het probleem als een Markov Decision Process (MDP). Dat houdt in dat het probleem zich volgens een Markov keten ontwikkeld waarbij beslissingen van buitenaf invloed hebben op de transities.

### 2.3.1. Het probleem.

Iedere MDP bestaat uit een state space, een action space en een policy die de mapping is van de action space op de state space, i.e. de policy vertelt voor elke state welke actie ondernomen moet worden. Om de policy te optimaliseren moet wel een kostenfunctie geminimaliseerd (of een winstfunctie gemaximaliseerd) worden. We beschrijven het probleem daarom in termen van die elementen.

#### *2.3.1.1. De state space.*

De schrijvers beginnen met het formuleren van een state space. Allereerst definiëren ze de vector $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ als de status van de $N$ ambulances en de vector $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ als de status van de meldingen. Er kunnen maximaal $M$ meldingen tegelijk in het systeem zitten.

De volledige status van een ambulance wordt gegeven door $\mathbf{a} = (\sigma_a, o_a, d_a, t_a)$. Daarin is $\sigma$ de status van de ambulance, $o$ en $d$ zijn de beginlocatie en eindbestemming van de rit die een ambulance maakt, en $t$ is het tijdstip waarop die rit begon. De status $\sigma$ kan 7 vormen aannemen:

1. Buiten dienst.
2. Beschikbaar op standplaats.
3. Onderweg naar melding.
4. Ter plaatse bij melding.
5. Onderweg naar ziekenhuis.
6. Bij het ziekenhuis.
7. Onderweg naar standplaats.

De tijdselementen (vertrektijd, begin en eindbestemming) van de rit zijn nodig om de non-Markovian aspecten (reistijden) van het probleem toch in de Markov keten op te nemen.

De volledige status van een melding wordt gegeven door $\mathbf{c} = (\sigma_c, o_c, t_c, p_c)$. Waar $\sigma$ de status van de melding is, $o$ de locatie van de melding is, $t$ het tijdstip waarop de melding

[7] Mateo Restrepo, *Approximate Dynamic Programming for Ambulance Redeployment*, (2008).

binnengekomen is en $p$ de prioriteit van de melding is. De status $\sigma$ kan voor een melding maar 2 vormen aannemen:

1. Toegewezen aan ambulance.
2. In de wachtrij.

De totale state van het systeem wordt weergegeven door $s = (\tau, e, \mathbf{A}, \mathbf{C})$. Daarin is $\tau$ de huidige tijd, en $e$ de huidige event (e.g. "melding komt binnen").

### *2.3.1.2. De action space.*

De action space bevat alle acties die tot de beschikking van de manager (of de meldkamer) staan. Dat wil zeggen, in ons geval zijn dat alle mogelijk verplaatsingen van vrije ambulances naar ambulanceposten. Laat $R(s)$ de verzameling van vrije ambulances zijn op het moment van state $s$. Daarnaast is $u_{ab}$ een binaire variabele die 1 is als ambulance $a$ naar ambulancepost $b$ verplaatst wordt en anders 0. $B$ is de verzameling van ambulanceposten. Meer formeel geformuleerd komt de action space neer op:

$$U(s) = \left\{ u(s) \in \{0, \quad 1\}^{|R(s)|\times|B|} : \sum_{b\in B} u_{ab}(s) \le 1 \quad \forall a \in R(s) \right\}$$

De sommatie legt de eis vast dat een ambulance naar ten hoogste één ambulancepost verplaatst kan worden.

### *2.3.1.3. De kostenfunctie.*

Om het proces te kunnen optimaliseren is het noodzakelijk een kostenfunctie te definieren. Dat houdt in dat er voor elke transitie die het proces maakt er kosten gegenereerd worden die ook afhankelijk zijn van de genomen actie. De schrijvers van dit artikel definiëren de volgende kostenfunctie:

$$g(s_k, u_k, s_{k+1}) = \begin{cases} 1 & e(s_{k+1}) \text{ is een ambulance aankomst bij de patiënt de melding had een hoge prioriteit en de reistijd was groter dan } r. \\ 0 & \text{anders.} \end{cases}$$

Deze kostenfunctie telt dus het aantal meldingen dat dringend een ambulance nodig heeft maar waarbij de ambulance het niet gered heeft om binnen $r$ minuten aanwezig te zijn.

### *2.3.1.4. De optimale policy.*

Een policy is een mapping van de action space op de state space. Dat wil zeggen dat het voor elke state vertelt welke verplaatsing gedaan moet worden. De schrijvers noteren de policy $\mu$, waarbij $\mu(s)$ de actie is die door de policy $\mu$ wordt voorgeschreven als het proces zich in state $s$ bevindt. Om te kunnen optimaliseren is het nodig een expressie te formuleren die de toekomstige kosten weergeeft, de zogenaamde value function:

$$J^{\mu}(s) = \mathbb{E}\Big[\sum_{k=1}^{\infty} \alpha^{\tau(s_k^{\mu})}\, g(s_k^{\mu}, \mu(s_k^{\mu}), s_{k+1}^{\mu}) \mid s_1^{\mu} = s\Big]$$

Vervolgens kan de actie gekozen worden die de minste toekomstige kosten oplevert:

$$J(s) = \min_{u \in \mathcal{U}(s)} \left\{ \mathbb{E}\Big[ g(s,u,f(s,u,X(s,u))) + \alpha^{\tau(f(s,u,X(s,u)))-\tau(s)}\, J(f(s,u,X(s,u)))\Big] \right\}$$

De bovenstaande optimality equation kiest dus de goedkoopste actie, terwijl het rekening houdt met de toekomst.

De schrijvers merken echter op dat het probleem te hoogdimensionaal en de state space te groot is om bovenstaande optimality equation met de gewone MDP theorie (policy iteration) op te lossen. Ze maken daarom gebruik van Approximate Dynamic Programming.

Approximate Dynamic Programming houdt in dat de value function benaderd wordt door een andere eenvoudigere expressie:

$$J(s,r) = \sum_{p=1}^{P} r_p \phi_p(s)$$

Daarin is $r_p$ een parameter en $\phi_p$ een basis functie. Die functies worden met nattevingerwerk vastgesteld. De parameters worden zodanig getuned dat de benadering overeenkomt met simulaties van gerealiseerde kosten. De schrijvers nemen $P = 6$ en formuleren dus 6 basis functies die de dynamiek van het probleem (hopelijk) nabootsen. Vervolgens voeren ze simulaties van de value function uit. Dat wil zeggen, we gaan van state naar state tot $\tau = T$ en houden steeds de kosten bij. De parameters $r$ worden vervolgens iteratief zo getuned dat het verschil tussen $J(s,r)$ en de simulaties zo klein mogelijk. De simulaties kunnen dus worden gezien als een steekproef uit alle mogelijk realisaties. De schrijvers merken daarom terecht op dat de methode gevoelig is voor steekproeffouten.

### 2.3.2. De resultaten.

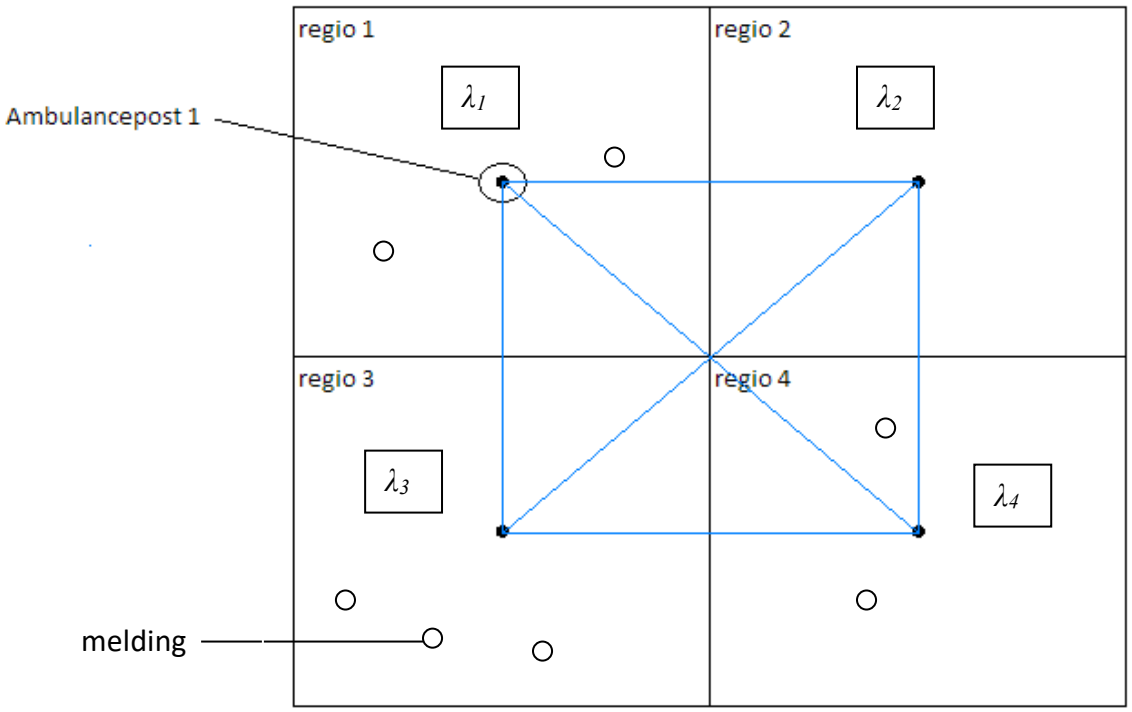
De schrijvers laten het model los op de stad Edmonton in Canada. De stad heeft 730.000 inwoners en is ongeveer 40 × 30 km$^2$. Er zijn 16 ambulances 11 ambulanceposten en 5 ziekenhuizen. Het onderzoek vergelijkt de resultaten van ADP met die van een statische oplossing: Met ADP zouden 21,9% van de meldingen niet op tijd bereikt zijn, waar de statische oplossing 25,3% mist. Het verschil (3,4%) is aanzienlijk; op jaarbasis zou het voor Nederland om vele duizenden meldingen kunnen gaan.

# 3. Case study.

Nu we de literatuur onder ogen gehad hebben, kunnen we aan de slag met een eigen onderzoek om de dynamiek, mogelijkheden én beperkingen van Ambulance Relocation beter te begrijpen. We beginnen met het beschrijven van een modelstad. Vervolgens formuleren we een Markov Decision Process met bijbehorende value function en optimality equation. Het resultaat is een policy die de kosten (nader te definiëren) minimaliseert.

## 3.1. Een situatieschets.

We hebben een stad met $N$ ambulances en 4 ambulanceposten. De stad is opgedeeld in 4 even grote kwartieren. De ambulanceposten zitten in het centrum van de 4 stadsdelen. De meldingen komen volgens een Poisson proces binnen met aankomstintensiteit $\lambda_i$ voor kwartier $i$. De behandeltijd van de ambulances is $1/\mu$, i.e. een ambulance kan $\mu$ meldingen per uur afhandelen. De state space is dus vrij eenvoudig; state $s = (\mathbf{x}, \mathbf{a})$, waarbij $\mathbf{x}$ en $\mathbf{a}$ vectoren zijn van respectievelijk het aantal meldingen in de regio's en het aantal ambulances in de regio's.




**Figuur 3. De stad wordt opgedeeld in 4 regio's. Iedere regio heeft een eigen ambulancepost. De blauwe lijnen zijn de verbindingen waarlangs ambulances verplaatst kunnen worden. Iedere regio heeft een aankomstintensiteit $\lambda$.**

De kosten die we maken zijn het aantal meldingen waarvoor niet onmiddellijk een ambulance beschikbaar is in dezelfde regio:

$$C(\mathbf{x},\mathbf{a}) = \sum_{i=1}^{4} (x_i - a_i)^+$$

We willen na elke transitie van de keten het aantal ambulances zo positioneren dat de toekomstige kosten *C* geminimaliseerd worden. Na de herpositionering wordt de ambulanceverdeling **b** (i.e. **a** wordt **b**). Merk op dat onze state space geen vertrek en aankomsttijden bijhoudt en dus ook geen reistijden kan bijhouden. Dit leidt tot de irreële implicatie dat een verplaatsing onmiddellijk effect heeft. Om die implicatie te corrigeren definiëren we een penalty $r(\mathbf{x},\mathbf{a},\mathbf{b})$.

## 3.2. De value function.

Laat *g* de verwachte lange termijn kosten per tijdseenheid zijn en γ een normalisatieconstante. Voor de zojuist beschreven modelstad komen we tot de volgende value functie:

$$\begin{aligned} g + \gamma V_{n+1}(\mathbf{x},\mathbf{a}) = C(\mathbf{x},\mathbf{a}) &+ \sum_{i=1}^{4} \lambda_i H_n(\mathbf{x}+e_i,\mathbf{a}) + \\ &+ \sum_{i=1}^{4} \mu_i \min(x_i,a_i)\ H_n(\mathbf{x}-e_i,\mathbf{a}) + \\ &+ \left(\gamma - \sum_{i=1}^{4} \lambda_i - \sum_{i=1}^{4} \mu_i \min(x_i,a_i)\right) V_n(\mathbf{x},\mathbf{a}) \end{aligned}$$

Waarbij

$$H_n(\mathbf{x},\mathbf{a}) = \min\left\{ V_n(\mathbf{x},\mathbf{b}) + r(\mathbf{x},\mathbf{a},\mathbf{b}) \,\middle|\, b_i \geq \min(x_i,a_i)\ \forall i, \sum_{i=1}^{4} b_i = N \right\}$$

$H_n$ optimaliseert de acties. Het kiest de actie die de minste kosten oplevert. Omdat onze state space geen vertrek en aankomsttijden bijhoudt impliceert de value functie dat een ambulance na verplaatsing onmiddellijk op de plaats van bestemming is. De penalty functie $r(\mathbf{x},\mathbf{a},\mathbf{b})$ is een correctie die het effect van reistijden nabootst:

$$r(\mathbf{x},\mathbf{a},\mathbf{b}) = l \sum_{i=1}^{4} \min[(x_i - a_i)^+, (b_i - a_i)^+] + d \sum_{i=1}^{4} (b_i - a_i)^+$$

Voor iedere ambulance die verplaatst wordt, worden *d* extra "verloren" meldingen gerekend en voor iedere melding waarvoor een ambulance uit een andere regio moet uitrukken worden *l*+*d* extra "verloren" meldingen gerekend. Zonder die penalty's zou het model geen verplaatsingen toepassen.

De justificatie van $d$ is dat de ambulance niet onmiddellijk na de verplaatsing op zijn bestemming is maar dat een verplaatsing altijd tijd kost. Gedurende die rit is een ambulance niet in staat meldingen te bedienen. Het valt dus te verwachten dat in sommige gevallen al een melding binnenkomt terwijl de ambulance nog verplaatst wordt. Die melding wordt dus vanuit het oogpunt van de kostenfunctie als verloren beschouwd. De verwachting van extra gemiste ambulances wordt door $d$ nagebootst. De keuze van $l$ en $d$ is een kunstvorm die relatief weinig invloed heeft op de policy. De structuur van $r(\mathbf{x}, \mathbf{a}, \mathbf{b})$ is belangrijker.

Met behulp van value iteration kan dan de optimale policy die de kosten minimaliseerd worden verkregen. Het value iteration algoritme is geïmplementeerd in Matlab. Omdat de wachtrijen in principe oneindig groot kunnen worden, kan de state space zo groot worden dat het model onoplosbaar is. Er is daarom voor gekozen bij de implementatie het aantal wachtende meldingen in regio $i$ te limiteren tot $W$. Daarnaast is ervoor gekozen het maximaal aantal verplaatsingen dat per ronde plaats kan vinden te limiteren tot $M$.

Bij de implementatie van het model zijn de genoemde variabelen als volgt ingevuld:

- Het aantal ambulances $N = 3$
- Het maximale aantal wachtende meldingen $W = 3$ (per regio).
- Het maximale aantal verplaatsingen per keer is 3.
- $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 1$.
- $\mu = 3$, voor alle ambulances.

Het algoritme ziet er grofweg als volgt uit:

1. $V_0 = 0$.
2. Initieer $n = 1$.
3. Bereken voor iedere state $s$ $V_n(s)$ (of $V_n(x,a)$).
4. Houd bij welke acties optimaal waren (de vector $b$).
5. Hoog $n$ met 1 op en doe stap 3 opnieuw.
6. Stop als het verschil tussen $V_n$ en $V_{n+1}$ convergeert naar $g$. Dit is het geval als over alle states het hoogste verschil en het laagste verschil hetzelfde zijn, i.e. span($V_{n+1}$ - $V_n$) $\rightarrow 0$.

## 3.3. De resultaten.

Het script resulteert in een optimale policy. Die definieert voor iedere state wat de optimale actie is. De volledige policy is een tabel van meer dan 5000 regels en is in de bijlage bijgevoegd. In Tabel 3 is de policy weergegeven voor 4 willekeurige states.

**Tabel 3. Een subset van de policy.**

| Aantal meldingen | Aantal ambulances | Relocation |
|---|---|---|
| 0 0 0 0 | 0 0 0 3 | 0 1 1 1 |
| 0 0 0 0 | 0 0 1 2 | 0 1 1 1 |
| 1 0 0 0 | 0 0 0 3 | 1 0 1 1 |

| 0 0 0 2 | 0 2 1 0 | 0 0 1 2 |
|---|---|---|

In de bovenstaande 4 states zijn de eerste vier kolommen de openstaande meldingen in regio's 1 t/m 4. De volgende 4 kolommen zijn de ambulances die zich in die regio's bevinden. De laatste 4 kolommen zijn de ambulance verdelingen die het model voorstelt.

Er zijn in de eerste state (eerste rij) geen openstaande meldingen en alle drie de ambulances bevinden zich in regio 4. Het model stelt voor de ambulances met zo min mogelijk verplaatsingen te spreiden over 3 ambulanceposten. Hetzelfde geldt voor de tweede state. Het maakt niet uit naar welke regio de ambulances verplaatst worden, omdat alle regio's dezelfde $\lambda$ hebben.

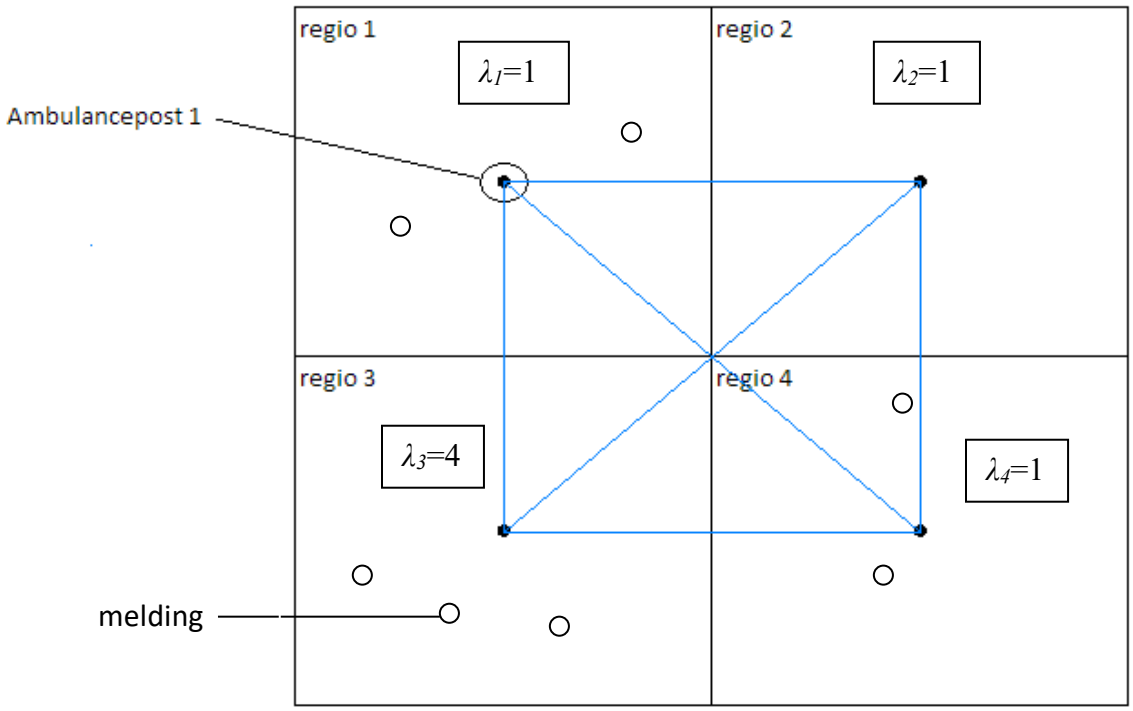
In de volgende state staat een melding open in regio 1. Echter, alle drie de ambulances bevinden zich in regio 4. Het model stelt voor de melding te bedienen en één van de overgebleven ambulances te verplaatsen.

Het is evident dat het model zeer gesimplificeerd is. Echter, we kunnen toch in enkele variabelen variëren om de dynamiek aan te tonen.

## 3.4. Variatie.

We nemen nu de volgende variabelen:

- Het aantal ambulances $N = \mathbf{5}$ (in plaats van 3)
- Het maximale aantal wachtende meldingen $W = 3$ (per regio).
- Het maximale aantal verplaatsingen per keer is **4** (in plaats van 3)
- $\lambda$ is een vector **[1 1 4 1]**. Regio 3 is dus een druk stadscentrum waar meer meldingen vandaan komen.
- $\mu = 3$, voor alle ambulances.




De verwachting is dus dat vrije ambulances meer naar regio 3 verplaatst worden. De resultaten wijzen uit dat dit in sommige gevallen voordelig is maar niet in alle gevallen:

| Aantal meldingen | Aantal ambulances | Relocation |
|---|---|---|
| 0 0 0 0 | 0 0 0 5 | 1 1 1 2 |
| 0 0 0 0 | 0 0 1 4 | 1 1 1 2 |

In de bovenstaande states in dus te zien dat het model liever 2 ambulances in regio 4 houdt dan nog een ambulance naar regio 3 te verplaatsen. Dit is opmerkelijk, je zou namelijk verwachten dat het model meer ambulances in regio 3 zou willen. In de volgende state kiest het model wel voor meer ambulances in regio 3:

| | | |
|---|---|---|
| 0 0 0 0 | 0 2 2 1 | 1 1 2 1 |

In deze state resulteert een verplaatsing van regio 3 naar 1 en een verplaatsing van regio 2 naar 1 in dezelfde directe kosten. Het lijkt er dus op dat alleen in gelijke gevallen de voorkeur naar regio 3 gaat. We zouden verwachten dat het model ook ambulances verplaatst als dit nu wat meer kosten oplevert maar in de toekomst weer kosten bespaard.
Die anomalie is te wijten aan onze keuze van penalty functie $r(\mathbf{x}, \mathbf{a}, \mathbf{b})$. Die stelt namelijk dat een verplaatsing altijd extra verloren meldingen genereert (omdat er reistijden zijn), wat dus niet reëel is. Dat is alleen het geval als er (bijna) geen ambulances meer zijn in de regio waarnaar een ambulance verplaatst wordt. Door onze keuze van $r(\mathbf{x}, \mathbf{a}, \mathbf{b})$ zijn verplaatsingen duurder dan nodig. Een aanpassing van de penalty functie $r(\mathbf{x}, \mathbf{a}, \mathbf{b})$, waarbij niet iedere verplaatsing even duur is, zou dus een meer realistische policy kunnen opleveren.

# 4. Conclusie

Ons model heeft veel weg van het model dat Restrepo gebruikt. Ons model is echter sterk gesimplificeerd om het oplosbaar te houden. Restrepo kiest ervoor het probleem complex te houden maar het niet met backward recursion op te lossen maar met Approximate Dynamic Programming. Het model levert dan toch in aan nauwkeurigheid, het gaat uit van simulaties die benaderd worden door een combinatie van basis functies. De keuze van basis functies is zelf ook een kunstvorm en niet zozeer een wetenschap. Wij daarentegen beginnen met een simpel model en maken het vervolgens complexer door elementen als reistijd toe te voegen in de vorm van penalty's (of beloningen). De keuze van die penalty's is ook een kunstvorm, maar kan naar mijn mening toch vrij nauwkeurig en effectief gedaan worden. Het grote voordeel van de penalty functies is dat de state space zowel laagdimensionaal als klein en aftelbaar blijft en het probleem dus vrij eenvoudig op te lossen is.

Uit onze policy kan ik enkele inzichten afleiden. Het model probeert ervoor te zorgen dat in elke regio in ieder geval 1 vrije ambulance aanwezig is om de volgende melding op te vangen. Als ergens de vrije ambulance dan bezet raakt, zal het model er opnieuw een ambulance naar toe sturen. Het model kiest er niet voor op voorhand al extra ambulances te verplaatsen omdat het nog niet bekend is waar die ambulances nodig zullen zijn. Om de verplaatsingskosten te besparen kiest het model er dus voor te wachten tot alle ambulances in een regio bezet zijn. Door de penalty functie $r(\mathbf{x}, \mathbf{a}, \mathbf{b})$ te verbeteren kunnen we het effect van reistijden realistischer nabootsen. Het model zal dan naar verwachting meer preventieve verplaatsingen toepassen naar drukkere regio's.

# 5. Bijlagen

## 5.1. Bijlage 1: Optimale policy scenario 1.

- Het aantal ambulances $N = 3$
- Het maximale aantal wachtende meldingen $W = 3$ (per regio).
- Het maximale aantal verplaatsingen per keer is 3.
- $\lambda = 1$, voor alle regio's.
- $\mu = 3$, voor alle ambulances.

De tabel is te groot om in dit document op te nemen. Het is daarom in onderstaand Excel bestand opgeslagen:

Optimal_Policy_Scenario_1.xls

## 5.2. Bijlage 2: Optimale policy scenario 2.

- Het aantal ambulances $N = \mathbf{5}$ (in plaats van 3)
- Het maximale aantal wachtende meldingen $W = 3$ (per regio).
- Het maximale aantal verplaatsingen per keer is **4** (in plaats van 3)
- $\lambda$ is een vector **[1 1 4 1]**. Regio 3 is dus een druk stadscentrum waar meer meldingen vandaan komen.
- $\mu = 3$, voor alle ambulances.

De tabel is te groot om in dit document op te nemen. Het is daarom in onderstaand Excel bestand opgeslagen:

Optimal_Policy_Scenario_2.xls

## 5.3. Matlab code.

### 5.3.1. MDP.m:

```
global statespace;
global values;
global aantalstates;
```

```
global aantalambulancecombis;
global statespace2;
global aantal_ambus;
global maxclients;
global lambda;
global mu;
global gamma;
global ambulances;
global z;
global argminimum;
global maxmoves;
global reistijd;
global reiskosten;
global premium;
global death;
global argmini;

z = 0;

reistijd = 0.25;
premium = 0.007;
death = 4;
maxmoves = 4;
maxclients = 3;
aantal_ambus = 5;
lambda = [1 1 4 1];
mu = 3;

reiskosten = reistijd*0.0007/aantal_ambus;
statespace = make_statespace(maxclients);
ambulances = make_actionspace(aantal_ambus);
statespace2 = zeros(0,0);

for i = 1:aantalstates;
    a = (i-1)*(aantalambulancecombis)+1;
    b = i*aantalambulancecombis;
    statespace2(a:b,1) = statespace(i,1);
    statespace2(a:b,2) = statespace(i,2);
    statespace2(a:b,3) = statespace(i,3);
    statespace2(a:b,4) = statespace(i,4);
    for j = 1:aantalambulancecombis;
        statespace2(a+j-1,5) = ambulances(j,1);
        statespace2(a+j-1,6) = ambulances(j,2);
        statespace2(a+j-1,7) = ambulances(j,3);
        statespace2(a+j-1,8) = ambulances(j,4);
    end
end

values = zeros(aantalstates*aantalambulancecombis,10);

argminimum = zeros(aantalstates*aantalambulancecombis,4);

g = 0;
gamma = sum(lambda) + mu * aantal_ambus;
n = 2;
span = 10^10;
stop = 10^-1;
```

```
while span > stop;
    for i = 1:aantalstates*aantalambulancecombis;
        z = i;
        values(i,n) = vxa(statespace2(i,1:4),statespace2(i,5:8),i, n);
    end
    values2 = values(:,2:n) - values(:,1:n-1);
    span = max(values2(:,n-1)) - min(values2(:,n-1))
    g = mean(values2(:,n-1))
    n = n + 1
end

optPolicy = statespace2;

for i = 1:aantalstates*aantalambulancecombis;
    z = i;
    hxa(statespace2(i,1:4),statespace2(i,5:8), n-1);
    optPolicy(i,9:12) = argmini;
end
```

### 5.3.2. make_statespace.m:

```
function [statespace] = make_statespace(maxclients)

statespace = zeros(0,0);
m = maxclients+1;

for l = 0:m-1;
    e = m^3*l+1;
    f = m^3*(l+1);
    statespace(e:f,1) = l;
    for k = 0:m-1;
        c = m^3*l+m^2*k+1;
        d = m^3*l+m^2*(k+1);
        statespace(c:d,2) = k;
        for j = 0:m-1;
            a = m^3*l+m^2*k+m*j+1;
            b = m^3*l+m^2*k+m*(j+1);
            statespace(a:b,3) = j;
            for i = 0:m-1;
                   statespace(m^3*l+m^2*k+m*j+i+1,4) = i;
            end
        end
    end
end
temp = size(statespace);
global aantalstates;
aantalstates = temp(1,1);
```

### 5.3.3. make_actionspace.m:

```
function [ambulances] = make_actionspace(aantal_ambus)
```

```
ambulances1 = zeros(0,0);
ambulances = zeros(0,0);

for l = 0:12;
    e = ((aantal_ambus+1)^3)*l+1;
    f = ((aantal_ambus+1)^3)*(l+1);
    ambulances1(e:f,1) = l;
    for k = 0:12;
        c = ((aantal_ambus+1)^3)*l+((aantal_ambus+1)^2)*k+1;
        d = ((aantal_ambus+1)^3)*l+((aantal_ambus+1)^2)*(k+1);
        ambulances1(c:d,2) = k;
        for j = 0:12;
            a = ((aan-
tal_ambus+1)^3)*l+((aantal_ambus+1)^2)*k+(aantal_ambus+1)*j+1;
            b = ((aan-
tal_ambus+1)^3)*l+((aantal_ambus+1)^2)*k+(aantal_ambus+1)*(j+1);
            ambulances1(a:b,3) = j;
            for i = 0:12;
                   ambulanc-
es1(((aantal_ambus+1)^3)*l+((aantal_ambus+1)^2)*k+(aantal_ambus+1)*j+i+1
,4) = i;
            end
        end
    end
end

q = 1;

for z = 1:length(ambulances1);
    if sum(ambulances1(z,:)) == aantal_ambus
        ambulances(q,:) = ambulances1(z,:);
        q = q+1;
    end
end
temp = size(ambulances);
global aantalambulancecombis;
aantalambulancecombis = temp(1,1);
```

## 5.3.4. vxa.m (value function):

```
function [val] = vxa(x, a, j, n)
e = eye(4,4);
c = zeros(1,4);
val = 0;

global maxclients;
global lambda;
global mu;
global gamma;
global ambulances;
global values;
global death;

for i = 1:4;
    if (x(i)-a(i))>0
```

```
        c(i) = (x(i)-a(i))*death;
    else
        c(i) = 0;
    end
end

rmu = 0;

for i = 1:4;
    if x(i)-1 >= 0
        xmu = x-e(i,:);
    else
        xmu = x;
    end
    if x(i)+1 < maxclients;
        xl = x+e(i,:);
    else
        xl = x;
    end
    rmu = rmu + mu * min(x(i),a(i));
    val = val + lambda(i) * hxa(xl,a,n-1) + mu * min(x(i),a(i)) *
hxa(xmu,a,n-1) + c(i);
end

val = (val + (gamma - sum(lambda) - rmu)*values(j,n-1))/gamma;
end
```

### 5.3.5. hxa.m (optimality function):

```
function [minimum] = hxa(x,a,m)

minimum = 10^100;
s = 1;

global lambda;
global mu;
global aantalstates;
global aantalambulancecombis;
global statespace;
global statespace2;
global values;
global maxclients;
global ambulances;
global z;
global argmini;
global argminimum;
global maxmoves;
global reiskosten;
global pen;
global premium;

argmini = zeros(1,4);
k = maxclients + 1;
index = 0;
```

```
for i = 1:aantalambulancecombis;
    if (sum(ambulances(i,:) < min(x,a))==0) &&
(0.5*sum(abs(ambulances(i,:)-a)) <= maxmoves)
        actions(s,:) = ambulances(i,:);
        index(s,1) = i-1;
        s = s+1;
    end
end

    begin = (x(1)*k^3+x(2)*k^2+x(3)*k+x(4))*aantalambulancecombis+1;
    pen = zeros(1,4);

for b = 1:(s-1);
        for i = 1:4;
            if (x(i)-a(i)>0)
                l = x(i)-a(i);
            else
                l = 0;
            end
            if (actions(b,i)-a(i)>0)
                r = actions(b,i)-a(i);
            else
                r = 0;
            end
            pen(i) = min(l,r)*premium;
        end
            if (values(begin+index(b), m) + 0.5*sum(abs(actions(b,:)-a))
* reiskosten + sum(pen) < minimum)
                minimum = values(begin+index(b),m) + 0.5 *
sum(abs(actions(b,:)-a)) * reiskosten + sum(pen);
                argmini = actions(b,:);
            end

end
argminimum(z,1:4) = argmini;

end
```

# 6. Literatuurlijst.


Tobias Andersson Granberg and Peter Värbrand, *Decision support tools for ambulance dispatch and relocation*, 2007, Journal of the Operational Research Society, (58), 195-201.

M. Gendreau et al., *A dynamic model and parallel tabu search heuristic for real-time ambulance relocation,* (2000).

Mateo Restrepo, *Approximate Dynamic Programming for Ambulance Redeployment*, (2008).

Ger Koole, *Lecture notes Stochastic Optimization*, http://www.math.vu.nl/obp/edu/so , (2006).

Ambulancezorg Nederland, *Ambulances in-zicht, 2007*, (2008).