Classificatie van Markovbeslissingsketens

Complexiteit van het multichainclassificatieprobleem

Wendy Ellens

21 augustus 2008

Bachelorscriptie, Mathematisch Instituut, Universiteit Leiden

Begeleider: Prof. Dr. L.C.M. Kallenberg

Abstract

This Bachelor's thesis is about the complexity of the *multichain classification problem*. The problem is to detect whether a given Markov decision process is *unichain* or *multichain*. A Markov decision process is unichain if the corresponding Markov chain contains only one recurrent class (and a possibly empty set of transient states) for every strategy, otherwise it is multichain.

We first show that the general case is NP-complete. A polynomial algorithm is given for Markov decision processes that contain either a state which is recurrent for all strategies or a state which is absorbing under some strategy. The deterministic case is considered to be polynomial, but we only give an outline of the algorithm. We will provide a complete polynomial algorithm to reduce the problem for deterministic Markov decision processes.

At last we will discuss some other polynomial algorithms, including an algorithm that reduces the multichain classification problem for a general Markov decision process in polynomial time to a multichain classification problem for a *communicating* Markov decision process (for every pair of states i, j there is a strategy such that i is reachable from j in the corresponding Markov chain).

Inhoudsopgave

Abstract **2**

1 Inleiding **4**

2 Van Markovketens naar Markovbeslissingsketens en terug **6**
2.1 Inleiding . . . 6
2.2 Markovketens . . . 6
2.3 Markovbeslissingsketens . . . 7
2.4 Conclusie . . . 9

3 NP-volledigheid van het multichainclassificatieprobleem **10**
3.1 Inleiding . . . 10
3.2 Polynomiale omzetting van 3SAT in MCP . . . 10
3.3 Ja-instanties gaan over in ja-instanties . . . 11
3.4 Nee-instanties gaan over in nee-instanties . . . 14
3.5 Conclusie . . . 14

4 Polynomiaal geval 1: Recurrente of absorberende toestanden **16**
4.1 Inleiding . . . 16
4.2 Bereikbare en vermijdbare verzamelingen . . . 16
4.3 Recurrente toestanden . . . 19
4.4 Absorberende toestanden . . . 21
4.5 Conclusie . . . 22

5 Polynomiaal geval 2: Deterministische Markovbeslissingsketens **23**
5.1 Inleiding . . . 23
5.2 Reductie van de geassocieerde graaf . . . 24
5.3 Karakterisering van de gereduceerde intercyclische grafen . . . 28
5.4 Classificatie van een gereduceerde graaf . . . 31
5.5 Conclusie . . . 32

6 Verdere polynomiale gevallen en reducties **34**
6.1 Inleiding . . . 34
6.2 Transiënte toestanden . . . 34
6.3 Communicerende Markovbeslissingsketens . . . 35
6.4 Gecondenseerde graaf . . . 37
6.5 Conclusie . . . 39

7 Conclusie **40**

Bibliografie **44**

1 Inleiding

Deze scriptie is geschreven als verslag van mijn Bacheloronderzoek en dient als afsluiting van de Bachelor wiskunde en als voorbereiding op het Masteronderzoek. In ongeveer vier maanden wordt een wiskundig onderwerp onderzocht, met als doel kennis te maken met wiskundig onderzoek, te leren omgaan met literatuur en de vaardigheden op het gebied van mondeling en schriftelijk presenteren te verbeteren.

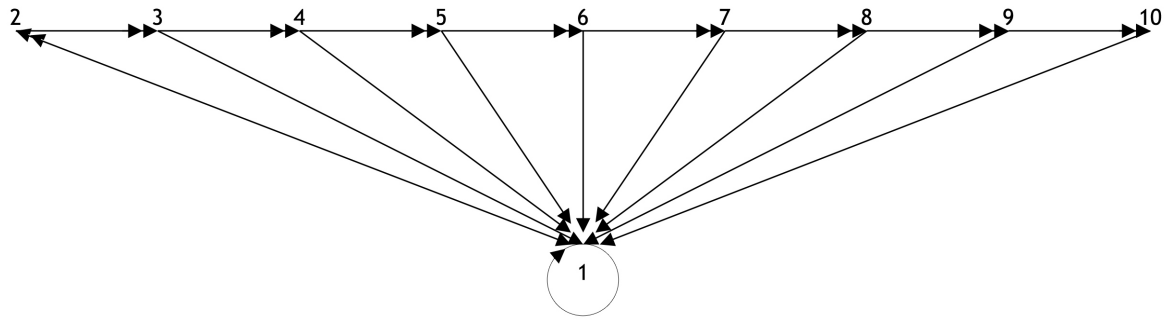
Het onderwerp van deze scriptie zijn *Markovbeslissingsketens*. Een Markovbeslissingsketen kan worden gebruikt als model voor productie- en voorraadvraagstukken, het controleren van wachtrijen, de planning van taken, enz. Bij een Markovbeslissingsketen hoort een verzameling van *toestanden*. In elke toestand hebben we de keuze uit een aantal *acties* en afhankelijk van de gekozen actie zijn er *directe opbrengsten* en *overgangskansen* om naar andere toestanden te gaan, of in dezelfde toestand te blijven. We lichten deze begrippen toe aan de hand van een voorbeeld uit [4].

Voorbeeld 1.1 (vervanging van een vrachtwagen) We nemen als voorbeeld een vrachtwagen van een transportbedrijf. Deze vrachtwagen kan zich in een bepaalde toestand bevinden, laten we zeggen dat de toestanden overeenkomen met de leeftijd van de wagen. Er zijn tien toestanden, toestand één tot en met tien, aangezien de vrachtwagen na uiterlijk tien jaar wordt ingeruild.

Laten we aannemen dat de eigenaar aan het begin van elk jaar beslist of de vrachtwagen vervangen wordt of nog een periode meegaat. In elke toestand zijn er dus twee acties: wel of niet vervangen, behalve in toestand tien, dan is er maar één actie: de vrachtwagen wordt vervangen. Bij elke actie hoort één pijl, deze pijl gaat naar de toestand waarin de vrachtwagen zich na een jaar bevindt.

Als de eerste actie (vervangen) wordt gekozen, gaan we met kans één naar toestand één; na een jaar hebben we een één jaar oude vrachtwagen. Als de tweede actie (niet vervangen) wordt gekozen, komt de vrachtwagen met kans één in de volgende toestand; hij wordt een jaar ouder.

De directe opbrengsten zijn in ons geval negatief, we maken namelijk kosten. Bij keuze voor de eerste actie zijn de kosten: de kosten van een nieuwe vrachtwagen min de inruilwaarde van de oude min de jaarlijkse onderhoudskosten van een nieuwe. Als de tweede actie wordt gekozen bestaan de kosten alleen uit de onderhoudskosten. De inruilwaarde en de onderhoudskosten hangen af van de leeftijd van de vrachtwagen.



Figuur 1.1 Een Markovbeslissingsketen met tien toestanden. De overgangskansen zijn overal 0 of 1. Een positieve overgangskans is aangegeven met één pijl als zij correspondeert met de eerste actie en met twee pijlen als zij hoort bij de tweede actie.

In voorbeeld 1.1 zijn alle overgangskansen nul of één, een dergelijke Markovbeslissingsketen noemen we *deterministisch*. Als we aannemen dat er een kans is dat de vrachtwagen gedurende het jaar onherstelbaar kapot gaat, was dit niet het geval geweest. Bij keuze voor de tweede actie (niet vervangen) zou de volgende toestand afhangen van het toeval. Als de vrachtwagen niet onherstelbaar

kapot gaat, wordt hij een jaar ouder en anders komt er een nieuwe vrachtwagen.

Een *strategie* is een beslisregel die in elke toestand een bepaalde actie voorschrijft. In voorbeeld 1.1 zouden we voor de strategie kunnen kiezen om de vrachtwagen na vijf jaar te vervangen (en direct voor oudere vrachtwagens). Deze strategie schrijft in toestand één tot en met vier actie twee (niet vervangen) voor en in toestand vijf tot en met tien actie één (vervangen).

We kunnen Markovbeslissingsketens classificeren aan de hand van de begrippen *unichain* en *multichain*. Stel dat er een strategie is, zodanig dat voor twee verschillende begintoestanden geldt dat het proces altijd weer terug zal komen in zijn begintoestand en dat deze toestanden niet bereibaar zijn vanuit elkaar. Dit laatste wil zeggen dat we startende in de ene toestand nooit in de andere toestand komen. In dit geval is de Markovbeslissingsketen multichain. Als een Markovbeslissingsketen niet multichain is, noemen we hem unichain.

In voorbeeld 1.1 zullen we altijd weer terug komen in toestand één tot en met vijf als we voor de strategie na vijf jaar vervangen kiezen en toestand één tot en met vijf zijn met elkaar verbonden. Als we een strategie kiezen waarvoor we bij een bepaalde leeftijd voor vervangen kiezen, en niet als de vrachtwagen jonger is, zal het proces alleen in die toestand en in toestanden die overeenkomen met een jongere vrachtwagen terugkomen. Deze toestanden zijn dan ook met elkaar verbonden. De Markovbeslissingsketen uit voorbeeld 1.1 is dus unichain.

De gemiddelde verwachte opbrengst blijkt, bij Markovbeslissingsketens die unichain zijn, niet af te hangen van de gekozen begintoestand. Daarom is het voor unichaine Markovbeslissingsketens eenvoudiger om een optimale strategie (de keuze van acties zodanig dat de verwachte opbrengsten gemiddeld zo hoog mogelijk zijn) te vinden.

De formele definities van de hierboven besproken begrippen vindt u in het volgende hoofdstuk. In dat hoofdstuk worden de begrippen behandeld die in latere hoofdstukken gebruikt worden. De vooronderstelde voorkennis bestaat uit elementaire grafentheorie en complexiteitstheorie.

In hoofdstuk 3 bewijzen we dat het bovengenoemde classificatieprobleem in unichain en multichain NP-volledig is, dat wil zeggen dat het zeer onwaarschijnlijk is dat er een efficiënt algoritme (een recept om het probleem op te lossen) is.

In de daarop volgende hoofdstukken 4 tot 6 wordt een aantal deelproblemen besproken waarvoor wel een efficiënt algoritme bekend is. Voor Markovbeslissingsketen met bepaalde specifieke eigenschappen, zoals voor bijvoorbeeld deterministische Markovbeslissingsketens, zijn er efficiënte algoritmes om te bepalen of hij unichain of multichain is.

2 Van Markovketens naar Markovbeslissingsketens en terug

2.1 Inleiding

In hoofdstuk 1 hebben we een aantal begrippen uitgelegd aan de hand van een voorbeeld. Deze begrippen spelen een belangrijke rol in deze scriptie, daarom zullen we ze in dit hoofdstuk formaliseren. In paragraaf 2.2 leggen we uit wat Markovketens zijn en bespreken de definities van een recurrente en een transiënte klasse. Deze definities gebruiken we om Markovbeslissingsketens te classificeren als unichain of multichain. In paragraaf 2.3 definiëren we Markovbeslissingsketens en bespreken we deze classificatie.

2.2 Markovketens

In deze paragraaf bespreken we een aantal eigenschappen van Markovketens. Ook geven we een classificatie van de toestanden in recurrente en transiënte toestanden. De in deze paragraaf besproken begrippen en beweringen komen uit [3]. Ten slotte geven we een polynomiaal algoritme uit [2] voor deze classificatie.

Een rij stochastische variabelen $\{X_t, t = 0, 1, \ldots\}$ met $X_t \in S$ met S een eindige of aftelbare toestandsruimte heet een *Markovketen* als voor iedere $i_0, i_1, \ldots, i_{t-1}, i, j \in S$ en $t = 0, 1, ...$ geldt dat

$$P(\mathrm{X_{t+1}} = \mathrm{j} \,|\, \mathrm{X_0} = \mathrm{i_0}, \mathrm{X_1} = \mathrm{i_1}, \ldots, \mathrm{X_{t-1}} = \mathrm{i_{t-1}}, \mathrm{X_t} = \mathrm{i}) = \mathrm{P}(\mathrm{X_{t+1}} = \mathrm{j} \,|\, \mathrm{X_t} = \mathrm{i})$$

Als de kansen niet van t afhangen, noemen we de Markovketen *stationair* en noteren we $P(\mathrm{X_{t+1}} = \mathrm{j} \,|\, \mathrm{X_t} = \mathrm{i})$ met p_{ij} en noemen dit de *overgangskansen.* De Markovketen wordt dan volledig beschreven door de *overgangsmatrix* P, waarvoor geldt $[P]_{ij} = p_{ij}$.

Een toestand j heet *bereikbaar* vanuit toestand i als er een tijdstip $t \in \mathbf{N}_{\geq 0}$ is met $p_{ij}^{(t)} > 0$, waarbij $p_{ij}^{(t)} = P(\mathrm{X_t} = \mathrm{j} \mid \mathrm{X_0} = \mathrm{i})$ en $P^{(t)} = P^t$. Als j bereikbaar is vanuit i en andersom, zeggen we dat i en j *communiceren.* Het begrip communiceren introduceert een equivalentierelatie op S, de equivalentieklassen noemen we de *klassen* van de Markovketen.

Een toestand heet *absorberend* als $p_{ii} = 1$.

De *terugkeerkans* f_i is de kans dat het systeem, startend in toestand i, er ooit terugkomt. Als $f_i = 1$ is toestand i *recurrent* , als $f_i < 1$ is i *transiënt.* Een recurrente toestand communiceert met elke toestand die het kan bereiken.

Recurrentie en transiëntie zijn klasse-eigenschappen. We spreken daarom van recurrente en transiënte klassen. De verzameling $Z \subseteq S$ is een recurrente klasse d.e.s.d.a. Z gesloten en communicerend is.

Het volgende algoritme maakt gebruik van deze eigenschap om de recurrente klassen $R_1, R_2, \ldots, R_m$ in een Markovketen te bepalen. In het algoritme wordt de graaf G gebruikt, deze graaf heeft dezelfde toestanden als de Markovketen en een pijl (i, j) als in de Markovketen geldt $p_{ij} > 0$.

Algoritme 2.1 (Bepaling van de recurrente klassen in een Markovketen)

1. (a) Laat $T = \emptyset$.
   (b) Laat $m = 0$.
2. (a) Bepaal de strengsamenhangende componenten in G en noem deze $C_1, C_2, \ldots C_n$.
   (b) Voor i van 1 tot n doe: Als C_i gesloten is doe:
      i. Laat $m = m + 1$.
      ii. Laat $R_m = C_i$.
3. (a) Output: $R_1, R_2, \ldots R_m$.

(b) Stop.

De begrippen die we in deze paragraaf gedefinieerd hebben voor Markovketens, zijn ook van belang voor Markovbeslissingsketens. Hierover spreken we in de volgende paragraaf.

2.3 Markovbeslissingsketens

In deze paragraaf zullen we zien dat we, uitgaande van het begrip Markovketen, een Markovbeslissingsketen kunnen definiëren. Vervolgens laten we zien dat we met behulp van een strategie van een Markovbeslissingsketen weer een Markovketen kunnen maken. Ten slotte bespreken we een aantal manieren om Markovbeslissingsketens te classificeren. We gebruiken de definities uit [2].

Bij een *Markovbeslissingsketen* hoort bij elke toestand $i \in S$ een *actieverzameling* $A(i)$. Als in toestand i actie $a \in A(i)$ gekozen wordt, is er een *directe opbrengst* $r_i(a)$ en is $p_{ij}(a)$ de overgangskans naar toestand j.

Ik ga uit van stationaire Markovbeslissingsketens met een niet-lege eindige toestandsruimte en niet-lege eindige actieverzamelingen.

Een *strategie* f is een functie die aan elke toestand i een actie toewijst, dus $f(i) \in A(i)$. Bij elke strategie f hoort een Markovketen $\{X_t(f)\}$ met overgangsmatrix $P(f)$ met $[P(f)]_{ij} = p_{ij}(f(i))$, we zullen deze Markovketen ook $P(f)$ noemen.

Een Markovbeslissingsketen heet *communicerend* als er voor alle $i, j \in S$ een strategie f is zodat j te bereiken is vanuit i in $P(f)$.

Als er twee disjuncte deelverzamelingen $S_1, S_2 \subseteq S$ zijn met $S_1 \cup S_2 = S$, met S_1 een verzameling van toestanden die onder alle strategieën transiënte zijn en als voor elke twee toestanden $i, j \in S_2$ geldt dat er een deterministische strategie is zodat j bereikbaar is vanuit i, wordt de Markovbeslissingsketen *zwak communicerend* genoemd.

Een andere classificatie van Markovbeslissingsketens is die in unichain en multichain. Een Markovbeslissingsketen is *unichain* als er voor elke deterministische strategie precies één recurrente klasse is (en een mogelijkerwijs lege verzameling van transiënte toestanden).

Een Markovbeslissingsketen wordt *multichain* genoemd als de keten niet unichain is, dat wil zeggen als er een deterministische strategie f is, zodat $P(f)$ ten minste twee niet-lege recurrente klassen heeft.

Als voorbeeld zullen we nu voorbeeld 1.1 uit hoofdstuk 1 formaliseren.

Voorbeeld 1.1 (vervolg) Voor dit model geldt:

1. Voor de toestandsruimte geldt $S = \{1, 2, \ldots, 10\}$.
2. De actieverzamelingen zijn $A(i) = \{1, 2\}$ als $i = 1, 2, \ldots 9$ en $A(10) = \{1\}$.
3. De overgangskansen zijn $p_{i1}(1) = 1$, $p_{i\,i+1}(2) = 1$ en 0 anders.
4. Voor de opbrengsten geldt $r_i(1) = s_i - k - c_0$ en $r_i(2) = -c_i$, waarbij s_i de inruilwaarde na i jaar is, k de kosten van een nieuwe vrachtwagen zijn en c_i de jaarlijkse onderhoudskosten voor een vrachtwagen van leeftijd i.

Zoals we in hoofdstuk 1 al aangaven, is deze Markovbeslissingsketen unichain. Voor elke strategie geldt dat de recurrente toestanden met elkaar communiceren. Ook is de Markovbeslissingsketen communicerend, want als we de strategie kiezen die alleen in toestand tien voor actie één kiest, communiceren alle toestanden met elkaar.

We geven nog een voorbeeld uit [4]. De directe opbrengsten laten we buiten beschouwing, aangezien deze geen invloed hebben op de eigenschap unichain of multichain.

Voorbeeld 2.1 (Gokken) Een gokker met een bedrag van i euro kan een inzet van $a \in \{1, 2, \ldots, i\}$ euro doen. Laten we aannemen dat hij met kans p zijn inzet wint en met kans $1 - p$ zijn inzet verliest. Het doel is om n euro te winnen. Hij stopt als hij zijn doel bereikt heeft of al zijn geld kwijt is. De Markovbeslissingsketen voor dit probleem is als volgt.

1. De toestandsruimte is $S = \{0, 1, 2, \ldots, n\}$
2. De actieverzamelingen zijn $A(i) = \{1, 2, \ldots, min(i, n-i)\}$ voor $1 \leq i < n$ en $A(i) = \{1\}$ voor $i \in \{0, n\}$.
3. Voor de overgangskansen geldt $p_{i\,i+a}(a) = p$ en $p_{i\,i-a}(a) = 1 - p$ voor $1 \leq i < n$, $p_{11}(1) = p_{nn}(1) = 1$ en 0 anders.

De Markovbeslissingsketen in voorbeeld 2.1 is multichain. Aangezien de toestanden 0 en n voor elke strategie absorberend zijn, zijn 0 en n recurrent en communiceren ze niet met elkaar. Daarom zijn er voor elke strategie meerdere recurrente klassen. Uit deze redenering volgt ook direct dat de Markovbeslissingsketen niet zwak communicerend (en dus zeker niet communicerend) is, toestand 0 en n zijn immers voor elke strategie recurrent en communiceren voor geen enkele strategie.

Zoals we in hoofdstuk 1 al vermeldden is de classificatie van Markovbeslissingsketens in unichain en multichain van belang omdat de gemiddelde verwachte opbrengst bij unichaine Markovbeslissingsketens onafhankelijk van de begintoestand is. We tonen dit aan met behulp van beweringen uit [3] en [4].

De gemiddelde verwachte opbrengst over een oneindige horizon is, gegeven een strategie f en een begintoestand i is

$$\lambda_i(f) = \liminf_{T\to\infty} \frac{1}{T} \mathbf{E}\left[\sum_{t=1}^{T} r_{X_t(f)}(f(X_t(f)))|X_0(f) = i\right]$$

en de maximale gemiddelde verwachte opbrengst is $\lambda_i^* = \max_f \lambda_i(f)$. De strategie die de gemiddelde verwachte opbrengst maximaliseert noemen we f_i^*. Er geldt dus $\lambda_i(f_i^*) = \lambda_i^*$.

Stelling 2.1 *In het geval van een unichaine Markovbeslissingsketen hangt voor iedere strategie de gemiddelde verwachte opbrengst over een oneindige horizon niet af van de begintoestand.*

Bewijs Voor de gemiddelde verwachte opbrengst over een oneindige horizon geldt

$$\begin{aligned}
\lambda_i(f) &= \liminf_{T\to\infty} \frac{1}{T} \mathbf{E}\left[\sum_{t=1}^{T} r_{X_t(f)}(f(X_t(f)))|X_0(f) = i\right] \\
&= \liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{j\in S} p_{ij}^{(t)}(f) \cdot r_j(f(j)) \\
&= \sum_{j\in S} p_{ij}^*(f) \cdot r_j(f(j)).
\end{aligned}$$

Waarbij $P^*(f)$ de stationaire matrix is. Hierbij is $p_{ij}^*(f) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} p_{ij}^{(t)}(f)$ de verwachting van de fractie van de tijd (op de lange duur) dat het systeem in toestand j is, gegeven de strategie f en de begintoestand i.

Voor een Markovketen met één recurrente klasse geldt dat de rijen van de stationaire matrix identiek zijn. Dit betekent dat $\lambda_i(f)$ niet afhangt van de begintoestand i. □

In deze paragraaf hebben we een aantal classificaties van Markovbeslissingsketens besproken. Ook hebben we voorbeelden van unichaine en van multichaine Markovbeslissingsketens gezien en de classificatie in unichain en multichain gemotiveerd.

2.4 Conclusie

In dit hoofdstuk hebben we niet alleen de voor deze scriptie belangrijke begrippen uit de theorie over Markovketens en Markovbeslissingsketens gedefinieerd, we hebben ook een aantal voorbeelden uitgewerkt en bewezen dat de gemiddelde verwachte opbrengst niet afhangt van de toestand waarin we starten. Daarmee hebben we een motivatie gegeven om op zoek te gaan naar een polynomiaal algoritme voor de classificatie van Markovbeslissingsketens in unichain en multichain.

In het volgende hoofdstuk tonen we aan dat er zeer waarschijnlijk geen polynomiaal algoritme voor dit probleem is (tenzij P = NP). In hoofdstuk 4 tot 6 behandelen we aantal deelproblemen waarvoor wel een polynomiaal algoritme bestaat.

3 NP-volledigheid van het multichainclassificatieprobleem

3.1 Inleiding

Dit hoofdstuk is gewijd aan de complexiteit van het multichainclassificatieprobleem. We zullen aantonen dat het algemene geval NP-volledig is. Dit is bewezen door Tsitsiklis in [6].

Om over de complexiteit te kunnen spreken, formuleren we classificatie van Markovbeslissingsketens in unichain of multichain als een herkenningsprobleem.

Definitie 3.1 *Multichainclassificatieprobleem (MCP):* Gegeven een Markovbeslissingsketen, is het multichain?

We zullen alvast aantonen dat het probleem tot NP behoort.

Lemma 3.1 *Het multichainclassificatieprobleem zit in NP.*

Bewijs We beschouwen een ja-instantie van het multichainclassificatieprobleem. Er is dan een strategie waarvoor de bijbehorende Markovketen meerdere recurrente klassen heeft. Deze strategie dient als certificaat dat de Markovbeslissingsketen multichain is. Algoritme 2.1 bepaalt in polynomiale tijd de recurrente klassen in een Markovketen, dus het certificaat is polynomiaal verifieerbaar. □

We hebben zojuist gezien dat het multichainclassificatieprobleem in NP zit. In de volgende paragraaf zullen we zien dat MCP tot de moeilijkste problemen behoort. Om aan te tonen dat het probleem NP-volledig is, tonen we aan dat het minstens zo moeilijk is als een van de moeilijkste problemen: het 3-satisfiability-probleem.

Definitie 3.2 *3-Satisfiability-probleem (3SAT):* Een instantie van het probleem bestaat uit m Boolse variabelen $x_1, x_2, \ldots, x_m$ en een zin van n woorden $c_1, c_2, \ldots, c_n$. Elk woord bestaat uit een disjunctie van drie letters, waarbij een letter een variabele x_i of zijn ontkenning $\neg x_i$ is, bijvoorbeeld $c_j = x_i \vee \neg x_h \vee x_g$. Een woord is dus waar als een van de letters waar is. Een zin is de conjunctie van n woorden, $c_1 \wedge c_2 \wedge \cdots \wedge c_n$. Hieruit volgt dat een zin waar is als ieder woord waar is. Het probleem is nu: is er een toekenning aan de variabelen, zodat de zin waar is?

De rest van dit hoofdstuk is gewijd aan het bewijs van de volgende stelling.

Stelling 3.2 *Het multichainclassificatieprobleem is NP-volledig.*

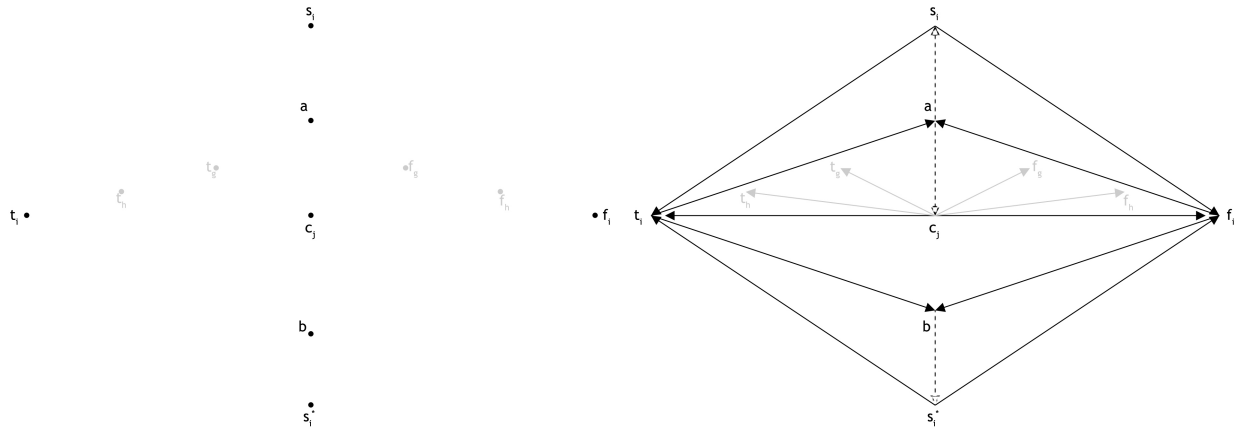
We hebben al bewezen dat het multichainclassificatie in NP zit. Om de NP-volledigheid te bewijzen zetten we een instantie van 3SAT (in polynomiale tijd) om in een instantie van MCP (paragraaf 3.2), zodanig dat ja-instanties overgaan in ja-instanties (paragraaf 3.3) en hetzelfde geldt voor nee-instanties (paragraaf 3.4). We zullen dus laten zien dat MCP NP-volledig minstens zo moeilijk is als het NP-volledige 3SAT.

3.2 Polynomiale omzetting van 3SAT in MCP

In deze paragraaf zullen we een instantie van het probleem het 3-satisfiability-probleem in polynomiale tijd omzetten in een instantie van het multichainclassificatieprobleem.

Laat een instantie van 3SAT met m variabelen en n woorden gegeven zijn. We construeren nu een Markovbeslissingsketen met $4m + n + 2$ toestanden: $a, b, s_i, s_i^*, t_i, f_i, c_j$ met $i = 1, 2, \ldots, m$ en $j = 1, 2, \ldots, n$. (Zie figuur 3.1.)

De acties en de bijbehorende overgangskansen zijn:



Figuur 3.1 Links: schematisch overzicht van de toestanden; rechts: schematisch overzicht van de mogelijke overgangen tussen de toestanden. C_j bevat respectievelijk de variabelen x_i, x_h en x_g.

1. $A(a) = \{1\}$ en $p_{as_i}(1) = \frac{1}{m+n},\ i = 1, 2, \ldots, m;\ p_{ac_j}(1) = \frac{1}{m+n},\ j = 1, 2, \ldots, n.$
2. $A(b) = \{1\}$ en $p_{bs_i^*}(1) = \frac{1}{m},\ i = 1, 2, \ldots, m.$
3. $A(s_i) = \{1, 2\}$ en $p_{s_i t_i}(1) = 1,\ i = 1, 2, \ldots, m;\ p_{s_i f_i}(2) = 1,\ i = 1, 2, \ldots, m.$
4. $A(s_i^*) = \{1, 2\}$ en $p_{s_i^* t_i}(1) = 1,\ i = 1, 2, \ldots, m;\ p_{s_i^* f_i}(2) = 1,\ i = 1, 2, \ldots, m.$
5. $A(t_i) = \{1, 2\}$ en $p_{t_i a}(1) = 1,\ i = 1, 2, \ldots, m;\ p_{t_i b}(2) = 1,\ i = 1, 2, \ldots, m.$
6. $A(f_i) = \{1, 2\}$ en $p_{f_i a}(1) = 1,\ i = 1, 2, \ldots, m;\ p_{f_i b}(2) = 1,\ i = 1, 2, \ldots, m.$
7. $A(c_j) = \{1, 2, 3\}$ en actie a komt overeen met de a-de letter van woord c_j ($j = 1, 2, \ldots, n$). Als de a-de letter van de vorm x_i is, geldt $p_{c_j t_i}(a) = 1$. Als de a-de letter van c_j van de vorm $\neg x_i$ is, geldt $p_{c_j f_i}(a) = 1$. (Zie figuur 3.2.)

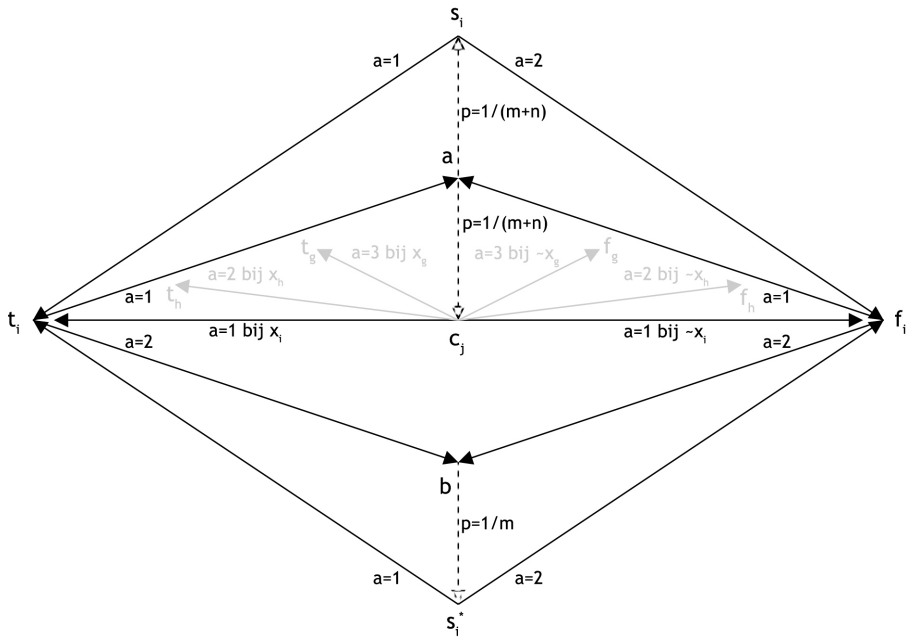
Het is duidelijk dat een instantie van 3SAT op deze manier in polynomiale tijd wordt omgezet in een instantie van MCP.

3.3 Ja-instanties gaan over in ja-instanties

We zullen nu laten zien dat we een ja-instantie voor MCP krijgen d.e.s.d.a. we een ja-instantie van 3SAT hadden.

Stel dat we een ja-instantie van 3SAT hebben, dat wil zeggen dat er een toewijzing voor $x_1, x_2, \ldots, x_m$ is zodanig dat alle woorden c_j waar zijn. We zullen aantonen aan er een strategie is zodanig dat de Markovketen meerdere recurrente klassen bevat. Dit betekent dat de Markovbeslissingsketen multichain is. De volgende strategie voldoet:

1. In toestand a is er maar één actie.
2. In toestand b is er ook maar één actie.
3. In elke toestand s_i kiezen we als volgende toestand t_i als de bijbehorende variabele x_i waar is in de toewijzing en f_i als x_i onwaar is.
4. In elke toestand s_i^* kiezen we als volgende toestand f_i als x_i waar is en t_i als x_i onwaar is.
5. In elke toestand t_i kiezen we als volgende toestand a als x_i waar is en b als x_i onwaar is.



Figuur 3.2 Schematisch overzicht van de acties en overgangskansen. C_j bevat respectievelijk de variabelen x_i, x_h en x_g.

6. In elke toestand f_i kiezen we als volgende toestand b als x_i waar is en a als x_i onwaar is.

7. In elke toestand c_j kiezen we actie 1 als de eerste letter van het woord waar is, actie 2 als de tweede letter waar is en de eerste niet, actie 3 als de derde letter waar is en de eerste en tweede niet. (Zie figuur 3.3.)

Deze strategie leidt tot twee recurrente klassen. Als we in a beginnen, zijn we op het volgende moment in een toestand s_i of een toestand c_j.

1. Stel de volgende toestand is s_i, dan zijn er twee mogelijkheden:

   (a) Als x_i waar is, dan is de volgende toestand is t_i en is de toestand daarna weer a.

   (b) Als x_i onwaar is, dan is de volgende toestand is f_i en is de toestand daarna ook weer a.

2. Stel de volgende toestand is c_j, dan kiezen we de eerste letter uit het woord die waar is, vervolgens zijn er weer twee mogelijkheden:

   (a) Als de letter overeenkomt met een niet-ontkende variabele x_i, is x_i dus waar en is de volgende toestand t_i en de toestand daarna weer a.

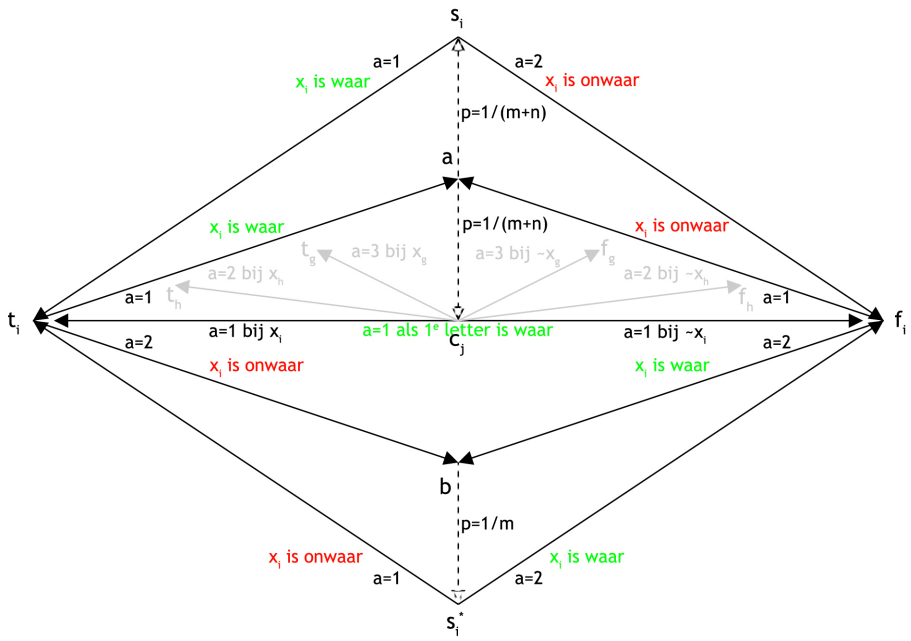
   (b) Als de letter overeenkomt met een ontkende variabele $\neg x_i$, is x_i dus onwaar en is de volgende toestand f_i en de toestand daarna weer a.

We zien dat a een recurrente toestand is en dat b nooit wordt bezocht vanuit a. We zullen nu laten zien dat ook b een recurrente toestand is. Als we in b beginnen, moeten we naar een toestand s_i^*, daar zijn er weer twee mogelijkheden:

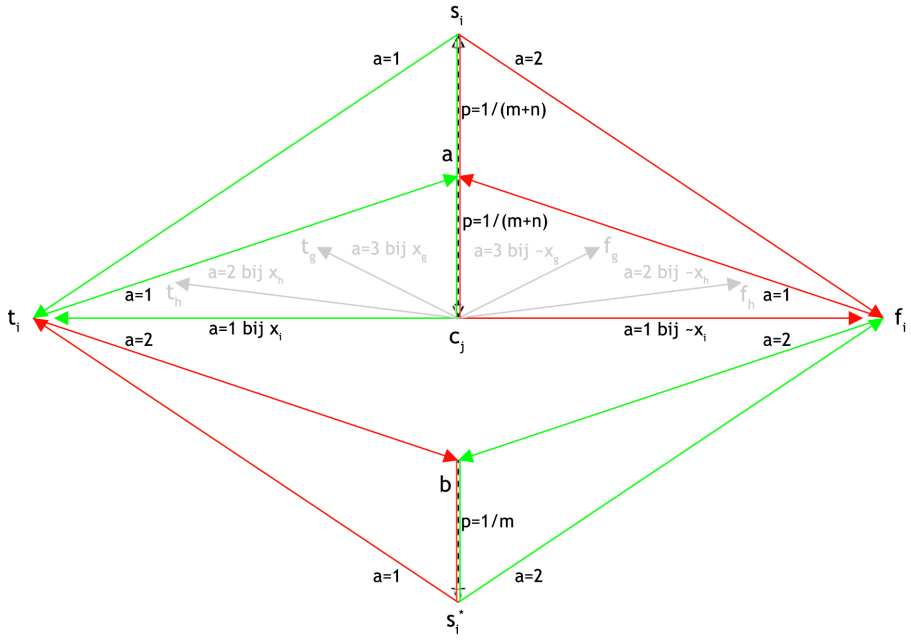
1. Als x_i waar is en de volgende toestand is f_i, dan is de toestand daarna weer b.

2. Als x_i onwaar is en de volgende toestand is t_i, ook dan is de toestand daarna weer b.

De recurrente klassen zijn dus: $\{a, s_i, t_k, f_l, c_j;\ x_k \text{ is waar}, \mathrm{x_l} \text{ is onwaar}, \mathrm{i} = 1, 2, \ldots, \mathrm{m}, \mathrm{j} = 1, 2, \ldots, \mathrm{n}\}$ en $\{b, s_i, t_k, f_l;\ \mathrm{x_k} \text{ is onwaar}, \mathrm{x_l} \text{ is waar}, \mathrm{i} = 1, 2, \ldots, \mathrm{m}, \mathrm{j} = 1, 2, \ldots, \mathrm{n}\}$. (Zie figuur 3.4.)



Figuur 3.3 Schema met een strategie die leidt tot een Markovketen met twee recurrente klassen. We nemen aan dat de eerste letter van c_j waar is.



Figuur 3.4 Schema met de twee recurrente klassen die zijn ontstaan als x_i waar is (groen) en als x_i onwaar is (rood). C_j bevat respectievelijk de variabelen x_i, x_h en x_g. We nemen aan dat de eerste letter van c_j waar is.

De toestanden a en b behoren tot twee verschillende recurrente klassen, dus de Markovbeslissingsketen is multichain. Hiermee hebben we aangetoond dat een ja-instantie van 3SAT overgaat in een ja-instantie van MCP.

3.4 Nee-instanties gaan over in nee-instanties

We zullen laten zien dat nee-instanties van 3SAT overgaan in nee-instanties van MCP door aan te tonen dat ja-instanties van MCP alleen ontstaan als de corresponderende instantie van 3SAT een ja-instantie was.

Stel nu dat we een ja-instantie van MCP hebben, dat wil zeggen dat er een strategie is zodanig dat de Markovketen meerdere recurrente klassen heeft. We geven zo dadelijk een toewijzing voor de variabelen van 3SAT, zodanig dat alle woorden waar zijn.

De structuur van de keten is zó dat we in maximaal drie stappen in toestand a of b zijn. Aangezien er meerdere recurrente klassen zijn, heeft dit tot gevolg dat a tot een recurrente klasse behoort en b tot een andere.

1. Stel dat we in s_i de actie die naar t_i leidt nemen. Aangezien we vanuit toestand a in toestand s_i zijn gekomen en we toestand b dus niet kunnen bereiken, leidt de actie in t_i terug naar a. Toestand s_i^* kan alleen vanuit b bereikt worden, dus is a niet te bereiken vanuit s_i^* en is de volgende toestand niet t_i, maar f_i.

2. Stel dat we in s_i de actie die naar f_i leidt nemen. Aangezien we vanuit toestand a in toestand s_i zijn gekomen en we toestand b dus niet kunnen bereiken, leidt de actie in f_i terug naar a. Toestand s_i^* kan alleen vanuit b bereikt worden, dus is a niet te bereiken vanuit s_i^* en is de volgende toestand niet f_i, maar t_i.

We nemen de volgende toekenning: laat x_i waar zijn als we onder de strategie die tot meerdere recurrente klassen leidt in toestand s_i actie 1 kiezen en laat x_i onwaar zijn als we onder deze strategie in s_i actie 2 kiezen.

We bekijken nu toestand c_j.

1. Stel dat de actie in c_j naar t_i leidt. Dat betekent dat x_i niet-ontkend in het j-de woord voorkomt en dat t_i bereikbaar is vanuit a. Daarom leidt de actie in t_i naar a. Dit betekent dat de actie s_i naar t_i leidt en dus dat x_i waar is. Nu volgt dat c_j waar is.

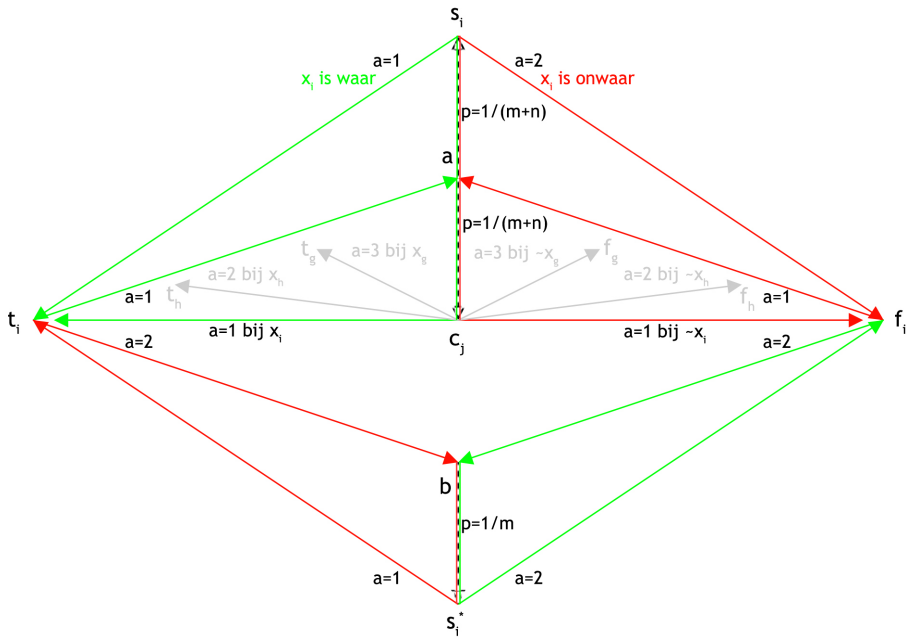
2. Stel dat de actie in c_j naar f_i leidt. Dat betekent dat x_i ontkend in het j-de woord voorkomt en dat f_i bereikbaar is vanuit a. Daarom leidt de actie in f_i naar a. Dit betekent dat de actie s_i naar f_i leidt en dus dat x_i onwaar is. Nu volgt dat $\neg x_i$, en daarmee ook c_j, waar is. (Zie figuur 3.5.)

De recurrente klassen zijn dus: $\{a, s_i, t_k, f_l, c_j;\ f(s_k) = 1, f(s_l) = 2, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\}$ en $\{b, s_i, t_k, f_l;\ f(s_k) = 2, f(s_l) = 1, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\}$.

In elk woord is minstens een letter waar en dus voldoet de toewijzing. We hebben hiermee laten zien dat een ja-instantie van MCP alleen kan ontstaan als de corresponderende instantie van 3SAT een ja-instantie was. Dit is equivalent met de bewering dat een nee-instantie van 3SAT door onze omzetting overgaat in een nee-instantie van MCP.

3.5 Conclusie

In de drie voorgaande paragrafen hebben we allereerst kunnen zien dat een instantie van het 3-satisfiability-probleem – dat NP-volledig is – in polynomiale tijd omgezet kan worden in een instantie van het multichainclassificatieprobleem.



Figuur 3.5 Schema met de toewijzing die alle woorden waar maakt en de recurrente klassen als $f(s_i) = 1$ (groen) en $f(s_i) = 2$ (rood). C_j bevat respectievelijk de variabelen x_i, x_h en x_g. We nemen aan dat $f(c_j) = 1$ met f een strategie die zorgt voor meerdere recurrente klassen.

Vervolgens hebben we aangetoond dat hierbij een ja-instantie van het eerste overgaat in een ja-instantie van de tweede, dat wil zeggen dat een zin die waar is voor een bepaalde toekenning aan de variabelen, wordt omgezet in een multichaine Markovbeslissingsketen.

Ten slotte hebben we laten zien dat ook geldt dat nee-instanties overgaan in nee-instanties, dus dat uit een zin waarvoor geen toekenning bestaat zodanig dat de zin waar is, een unichaine Markovbeslissingsketen ontstaat.

Als er nu een polynomiaal algoritme voor MCP zou zijn, zou ook het 3SAT polynomiaal oplosbaar zijn. Daaruit volgt dat MCP minstens zo moeilijk is als 3SAT, en dus dat MCP NP-volledig is. Hiermee is stelling 3.2 bewezen. □

4 Polynomiaal geval 1: Recurrente of absorberende toestanden

4.1 Inleiding

In het vorige hoofdstuk hebben we gezien dat het bepalen of een Markovbeslissingsketen unichain is, een NP-volledig probleem is. In dit hoofdstuk en de volgende twee hoofdstukken zullen we zien dat er in een aantal gevallen wél een polynomiaal algoritme is voor de classificatie.

We zullen aantonen dat er een polynomiaal algoritme voor de classificatie bestaat als de Markovbeslissingsketen een *recurrente* toestand bevat, dat wil zeggen een toestand die recurrent is onder elke strategie. Markovbeslissingsketens met recurrente toestanden komen in veel toepassingen voor. In het voorbeeld uit hoofdstuk 1 (voorbeeld 1.1) waarin de vervanging van een vrachtwagen wordt gemodelleerd is de eerste toestand recurrent.

Het probleem is ook polynomiaal als er een toestand is die absorberend is onder een bepaalde strategie. Een dergelijke toestand noemen we *absorberend*. Ook Markovbeslissingsketens met absorberende toestanden komen in de praktijk voor. Toestand 0 en n van voorbeeld 2.1 uit hoofdstuk 2 zijn absorberend. Beide polynomiale gevallen worden beschreven door Feinberg en Yang in [1].

Bij de classificatie van Markovbeslissingsketens met recurrente toestanden of absorberende toestanden, maken we gebruik van de begrippen bereikbare en vermijdbare verzameling. Deze begrippen worden gedefinieerd in paragraaf 4.2. Vervolgens geven we in paragraaf 4.3 een algoritme om de recurrente klassen te vinden en vervolgens de Markovbeslissingsketen te classificeren en lichten dit toe. Tot slot behandelen we in paragraaf 4.4 een polynomiaal algoritme dat een Markovbeslissingsketen met absorberende toestanden classificeert.

4.2 Bereikbare en vermijdbare verzamelingen

We definiëren nu eerst de begrippen bereikbare en vermijdbare verzameling. Vervolgens zullen we twee algoritmes geven die onderdeel zijn van de algoritmes in de volgende paragrafen.

Definitie 4.1 Een verzameling $Z \subseteq S$ heet *bereikbaar* vanuit $i \in S$ als geldt $i \notin Z$ en er een strategie f is, waarvoor er een $j \in Z$ is zodanig dat j bereikbaar is vanuit i in de Markovketen bij f, dat wil zeggen als er een $t \in \mathbf{N}_{>0}$ is met $p_{ij}^{(t)}(f) > 0$.

Definitie 4.2 Een verzameling $Z \subseteq S$ heet *vermijdbaar* vanuit $i \in S$ als er een strategie f is zodanig dat alle $j \in Z$ niet bereikbaar zijn vanuit i in de Markovketen bij f, dat wil zeggen als voor alle $t \in \mathbf{N}_{\geq 0}$ geldt $p_{ij}^{(t)}(f) = 0$.

We merken op dat toestand i zelf nooit tot de vanuit i bereikbare of vermijdbare verzameling behoort.

In de volgende paragrafen zullen we gebruik maken van de volgende verzamelingen:

Definitie 4.3 $B^{-1}(Z)$ is de verzameling van alle $i \in S$ zodanig dat Z bereikbaar is vanuit i.

Definitie 4.4 $V^{-1}(Z)$ is de verzameling van alle $i \in S$ zodanig dat Z vermijdbaar is vanuit i.

Deze verzamelingen kunnen opgespoord worden met behulp van algoritme 4.1 en 4.2. In het eerste algoritme is bevat de verzameling Y in de k-de iteratie van stap 2 de toestanden van waaruit Z in k stappen en niet in minder stappen bereikbaar is. Met behulp van de verzameling Y' wordt er bijgehouden welke toestanden in de volgende iteratie in Y komen.

Algoritme 4.1 (Opsporen van $B^{-1}(Z)$ voor een gegeven $Z \subset S$)

1. (a) Laat $B^{-1}(Z) = \emptyset$.
   (b) Laat $Y = Z$.

(c) Laat $Y' = \emptyset$.

2. Zolang $Y \neq \emptyset$ doe: Voor alle $i \in S/(Z \cup B^{-1}(Z))$ doe:

   (a) Als er een $a \in A(i)$ is waarvoor geldt $\sum_{j \in Y} p_{ij}(a) > 0$ doe:

      i. Voeg i toe aan Y'.

      ii. Voeg i toe aan $B^{-1}(Z)$.

   (b) Laat $Y = Y'$ en $Y' = \emptyset$.

3. (a) Output: $B^{-1}(Z)$.

   (b) Stop.

In het tweede algoritme bevat de verzameling Y in de k-de iteratie van stap 2 de toestanden van waaruit Z voor elke strategie bereikbaar is in maximaal k stappen en niet voor alle strategieën in minder stappen. De verzameling Y' houdt weer bij welke toestanden in de volgende iteratie in Y moeten komen.

Algoritme 4.2 (Opsporen van $V^{-1}(Z)$ voor een gegeven $Z \subset S$)

1. (a) Laat $V^{-1}(Z) = S/Z$.

   (b) Laat $Y = Z$.

   (c) Laat $Y' = \emptyset$.

2. Zolang $Y \neq \emptyset$ doe: Voor alle $i \in V^{-1}(Z)$ doe:

   (a) Voor alle $a \in A(i)$ doe: Als geldt $\sum_{j \in Y} p_{ij}(a) > 0$: Verwijder a uit $A(i)$.

   (b) Als geldt $A(i) \neq \emptyset$ doe:

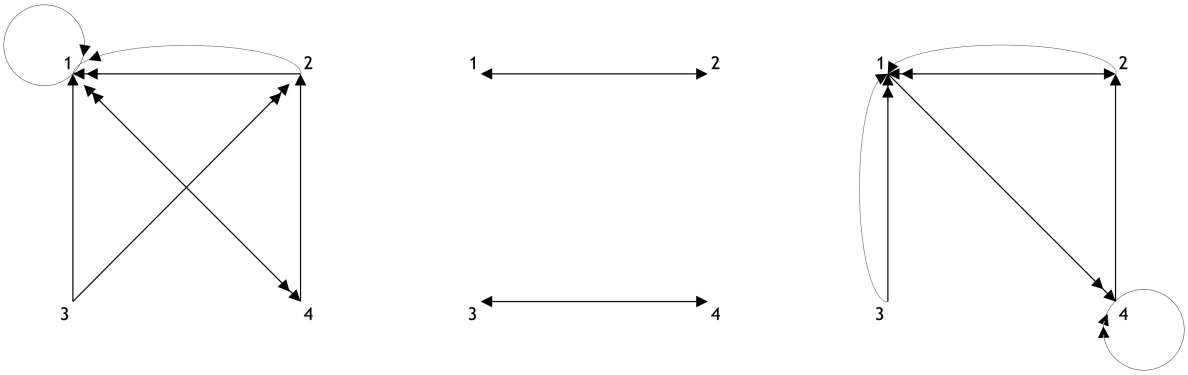
      i. Voeg i toe aan Y'.

      ii. Verwijder i uit $V^{-1}(Z)$.

   (c) Laat $Y = Y'$ en $Y' = \emptyset$.

3. (a) Output: $V^{-1}(Z)$.

   (b) Stop.



Figuur 4.1 Drie Markovbeslissingsketens met elke vier toestanden. De overgangskansen zijn overal nul of één. Een positieve overgangskans is aangegeven met één pijl als zij correspondeert met de eerste actie en met twee pijlen als zij hoort bij de tweede actie. De eerste en de derde Markovbeslissingsketen hebben in elke toestand twee acties en de tweede heeft in elke toestand één actie.

We lichten deze algoritmes toe met een aantal voorbeelden. (Zie figuur 4.1.) In deze voorbeelden is Y_k de verzameling Y na de k-de iteratie. De verzamelingen Z zijn zo gekozen dat de resultaten voor $B^{-1}(Z)$ of $V^{-1}(Z)$ gebruikt kunnen worden in de volgende paragrafen.

Voorbeeld 4.1 Zie figuur 4.1, links.
We bepalen $V^{-1}(Z)$ voor $Z = \{1\}$. Dit geeft $Y_1 = \{2\}$, $Y_2 = \{3, 4\}$ en $V^{-1}(Z) = \emptyset$.

Voorbeeld 4.2 Zie figuur 4.1, midden.
We bepalen $B^{-1}(Z)$ voor $Z = \{3, 4\}$. Dit geeft $Y_1 = \emptyset$ en $B^{-1}(Z) = \emptyset$.
We bepalen $V^{-1}(Z)$ voor $Z = \{1\}$. Dit geeft $Y_1 = \{2\}$, $Y_2 = \emptyset$ en $V^{-1}(Z) = \{3, 4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{2\}$. Dit geeft $Y_1 = \{1\}$, $Y_2 = \emptyset$ en $V^{-1}(Z) = \{3, 4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{3\}$. Dit geeft $Y_1 = \{4\}$, $Y_2 = \emptyset$ en $V^{-1}(Z) = \{1, 2\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{4\}$. Dit geeft $Y_1 = \{3\}$, $Y_2 = \emptyset$ en $V^{-1}(Z) = \{1, 2\}$.

Voorbeeld 4.3 Zie figuur 4.1, rechts.
We bepalen $B^{-1}(Z)$ voor $Z = \{4\}$. Dit geeft $Y_1 = \{1\}$, $Y_2 = \{2, 3\}$ en $B^{-1}(Z) = \{1, 2, 3\}$.
We bepalen $B^{-1}(Z)$ voor $Z = \{1, 3, 4\}$. Dit geeft $Y_1 = \{2\}$, $Y_2 = \emptyset$ en $B^{-1}(Z) = \{2\}$.
We bepalen $B^{-1}(Z)$ voor $Z = \{1, 2, 4\}$. Dit geeft $Y_1 = \{3\}$, $Y_2 = \emptyset$ en $B^{-1}(Z) = \{3\}$.
We bepalen $B^{-1}(Z)$ voor $Z = \{1, 2, 3\}$. Dit geeft $Y_1 = \{4\}$, $Y_2 = \emptyset$ en $B^{-1}(Z) = \{4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{1\}$. Dit geeft $Y_1 = \{2, 3\}$, $Y_2 = \emptyset$ en $V^{-1}(Z) = \{3, 4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{2\}$. Dit geeft $Y_1 = \emptyset$ en $V^{-1}(Z) = \{1, 3, 4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{3\}$. Dit geeft $Y_1 = \emptyset$ en $V^{-1}(Z) = \{1, 2, 4\}$.
We bepalen $V^{-1}(Z)$ voor $Z = \{4\}$. Dit geeft $Y_1 = \emptyset$ en $V^{-1}(Z) = \{1, 2, 3\}$.

De juistheid en de polynomialiteit van de algoritmes worden toegelicht in stelling 4.1 en stelling 4.2.

Stelling 4.1

1. *Algoritme 4.1 geeft de juiste output.*
2. *Algoritme 4.1 is polynomiaal.*

Bewijs

1. We voegen aan $B^{-1}(Z)$ de toestanden toe van waaruit Z bereikbaar is. In iteratie k voegen we de toestanden i toe waarvoor er een strategie en een $j \in Z$ bestaan, zodat j in k stappen (en niet in minder stappen) bereikbaar is vanuit i. Eerst voegen we de toestanden i toe die niet in Z zitten en waarvoor er een actie is zodanig Z in één stap bereikbaar is vanuit i, daarna de toestanden die voor een bepaalde strategie een positieve overgangskans hebben naar de toestanden die de vorige keer toegevoegd zijn, enz.

   Als er geen toestanden zijn van waaruit Z voor het eerst in n stappen bereikbaar is voor een zekere strategie, zijn er ook geen toestanden van waaruit Z voor het eerst in meer dan n stappen bereikbaar is voor een zekere strategie. Het algoritme stopt in dit geval (of als $B^{-1}(Z) = S/Z$). Nu zijn alle toestanden aan $B^{-1}(Z)$ toegevoegd van waaruit Z bereikbaar is.

2. De orde van een iteratie van stap 2 is $O(\sum_{i \in S/(Z \cup B^{-1}(Z)} (\#A(i)) \cdot \#Y)$. De eerste factor (de som) is $O(m)$, waarbij m het totale aantal acties $\sum_{i \in S} \#A(i)$ is. De verzamelingen Y tijdens verschillende iteraties zijn disjunct, dus de tweede factor maal het aantal iteraties is $O(n)$, met $n = \#S$. De complexiteit van algoritme 4.1 is $O(m \cdot n)$, want de andere stappen zijn van lagere orde. □

Stelling 4.2

1. *Algoritme 4.2 geeft de juiste output.*
2. *Algoritme 4.2 is polynomiaal.*

Bewijs

1. We weten dat $Z \cap V^{-1}(Z) = \emptyset$, daarom beginnen we met $V^{-1}(Z) = S/Z$. Vervolgens verwijderen we uit $V^{-1}(Z)$ alle toestanden i van waaruit Z niet vermijdbaar is, dat wil zeggen dat Z voor elke strategie bereikbaar is vanuit i, oftewel voor elke strategie is er is een $j \in Z$ die bereikbaar is vanuit i. In iteratie k verwijderen we de toestanden i van waaruit Z voor alle strategieën in maximaal k stappen (en niet voor alle strategieën in minder stappen) bereikbaar is vanuit i. Daartoe verwijderen we eerst de toestanden van waaruit Z in één stap bereikbaar is voor alle strategieën, daarna de toestanden die voor alle strategieën óf in één stap Z kunnen bereiken óf een positieve overgangskans hebben naar de toestanden die de vorige keer verwijderd zijn, enz.

   Als er geen toestanden zijn van waaruit Z in maximaal in n stappen bereikbaar is voor alle strategieën (en niet voor alle strategieën in minder dan n stappen), zijn er ook geen toestanden van waaruit Z voor alle strategieën in meer dan n stappen bereikbaar is (en niet voor alle strategieën in maximaal n stappen). Het algoritme stopt in dit geval (of als $V^{-1}(Z)$ leeg is). Nu zijn alle toestanden uit $V^{-1}(Z)$ verwijderd van waaruit Z niet vermijdbaar is, dus houden we de toestanden van waaruit Z vermijdbaar is over.

2. De orde van een iteratie van stap 2 is $O(\sum_{i \in V^{-1}(Z)}(\#A(i)) \cdot \#Y)$. De eerste factor (de som) is $O(m)$, waarbij m het totale aantal acties $\sum_{i \in S} \#A(i)$ is. De verzamelingen Y tijdens verschillende iteraties zijn disjunct, dus de tweede factor maal het aantal iteraties is $O(n)$, met $n = \#S$. De complexiteit van algoritme 4.2 is $O(m \cdot n)$, want de andere stappen zijn van lagere orde. □

Nu we over polynomiale algoritmes beschikken om $B^{-1}(Z)$ en $V^{-1}(Z)$ te bepalen, kunnen we de recurrente toestanden in Markovbeslissingsketen vinden.

4.3 Recurrente toestanden

In deze paragraaf wordt uitgelegd hoe de recurrente toestanden in een Markovbeslissingsketen gevonden kunnen worden en hoe vervolgens bepaald kan worden of de Markovbeslissingsketen unichain of multichain is. Ten slotte geven we een polynomiaal algoritme dat dit proces uitvoert.

Om te bepalen of $j \in S$ recurrent is, beschouwen we de verzameling $V^{-1}(\{j\})$. We kunnen nu drie gevallen onderscheiden.

1. Als de verzameling $V^{-1}(\{j\})$ leeg is, wil dat zeggen dat voor alle andere $i \in S$ en alle strategieën geldt dat j bereikbaar is. In dit geval is j recurrent en is de Markovbeslissingsketen unichain.

2. Voor $V^{-1}(\{j\}) \neq \emptyset$ bekijken we $B^{-1}(V^{-1}(\{j\}))$.

   (a) Als $j \in B^{-1}(V^{-1}(\{j\}))$, dan is $V^{-1}(\{j\})$ bereikbaar vanuit j, maar is j vermijdbaar vanuit $V^{-1}(\{j\})$. In dit geval is er strategie en een $i \in V^{-1}(\{j\})$, zodat i bereikbaar is vanuit j, maar j niet bereikbaar vanuit i. Toestand j is dus niet recurrent voor deze strategie. Daarom is j niet recurrent.

   (b) Als $j \notin B^{-1}(V^{-1}(\{j\}))$, dan is j alleen vermijdbaar vanuit $V^{-1}(\{j\})$, maar is $V^{-1}(\{j\})$ voor geen enkele strategie bereikbaar vanuit j. Dit betekent dat het systeem, startend in j alleen in toestanden komt van waaruit j altijd bereikbaar is, dus is j recurrent en zit j in een recurrente klasse disjunct aan $V^{-1}(\{j\})$.

   Nu is er ook een deelverzameling van $V^{-1}(\{j\})$ die een recurrente klasse vormt voor een bepaalde strategie. We bewijzen dit in lemma 4.3. Er is dus een strategie zo dat er twee verschillende recurrente klassen zijn. We kunnen concluderen dat de Markovbeslissingsketen multichain is als $j \notin B^{-1}(V^{-1}(\{j\}))$.

Lemma 4.3 *Als geldt $V^{-1}(\{j\}) \neq \emptyset$, dan is er een strategie waarvoor een deelverzameling van $V^{-1}(\{j\})$ een recurrente klasse vormt.*

Bewijs Voor elke $i \in V^{-1}(\{j\})$ geldt dat er een strategie f_i is zodanig dat j niet bereikbaar is. Neem nu bij elke toestand $i \in V^{-1}(\{j\})$ actie $f_i(i)$ en definieer zo een nieuwe strategie. Voor deze strategie geldt $p_{ik} = 0$ voor $i \in V^{-1}(\{j\})$ en $k \notin V^{-1}(\{j\})$, want vanuit k is j niet vermijdbaar. Aangezien het proces, startend in $V^{-1}(\{j\})$, altijd in $V^{-1}(\{j\})$ zal blijven, bevat $V^{-1}(\{j\})$ een recurrente klasse. □

We hebben gezien dat de Markovbeslissingsketen een recurrente klasse bevat en unichain is als voor een $j \in S$ geldt $V^{-1}(\{j\}) = \emptyset$ (geval 1), dat de Markovbeslissingsketen geen recurrente klassen bevat als geldt $V^{-1}(\{j\}) \neq \emptyset$ en voor alle $j \in S$ geldt $j \in B^{-1}(V^{-1}(\{j\}))$ (geval 2a) en dat de Markovbeslissingsketen een recurrente klasse bevat en multichain is als geldt $V^{-1}(\{j\}) \neq \emptyset$ en voor een $j \in S$ geldt $j \notin B^{-1}(V^{-1}(\{j\}))$ (geval 2b).

Algoritme 4.3 is gebaseerd op deze observaties.

Algoritme 4.3 (Classificatie van MBK's met recurrente toestanden)

1. Voor alle $j \in S$ doe:
   (a) i. Pas algoritme 4.2 toe op $Z = \{j\}$.
   ii. Als $V^{-1}(\{j\}) = \emptyset$: stop. *De MBK is unichain.*
   (b) i. Pas algoritme 4.1 toe op $Z = V^{-1}(\{j\})$.
   ii. Als $j \notin B^{-1}(V^{-1}(\{j\}))$: stop. *De MBK is multichain.*
2. Stop. *De MBK bevat geen recurrente toestanden.*

Voordat we de juistheid en de polynomialiteit van het algoritme zullen bewijzen, passen we algoritme 4.3 toe op voorbeeld 4.1 tot 4.3.

Voorbeeld 4.1 (vervolg) De linker Markovbeslissingsketen in 4.1 is unichain want $V^{-1}(\{1\}) = \emptyset$. Toestand 1 is dus recurrent.

Voorbeeld 4.2 (vervolg) De Markovbeslissingsketen in het midden van 4.1 is multichain want voor alle toestanden j geldt $V^{-1}(\{j\}) \neq \emptyset$ en voor toestand 1 geldt $1 \notin B^{-1}(V^{-1}(\{1\})) = B^{-1}(\{3,4\})) = \emptyset$. Toestand 1 is dus recurrent.

Voorbeeld 4.3 (vervolg) De rechter Markovbeslissingsketen in 4.1 kan niet geclassificeerd worden, want voor alle toestanden j geldt $V^{-1}(\{j\}) \neq \emptyset$ en $j \in B^{-1}(V^{-1}(\{j\}))$. Er zijn dus geen recurrente toestanden.

Stelling 4.4

1. *Algoritme 4.3 geeft de juiste output.*
2. *Algoritme 4.3 is polynomiaal.*

Bewijs

1. De juistheid volgt direct uit het voor algoritme 4.3 besprokene.
2. Algoritmes 4.1 en 4.2 hebben een complexiteit van $O(m \cdot n)$. Het aantal iteraties is $O(n)$. Daarom is de complexiteit algoritme 4.3 $O(m \cdot n^2)$. □

In deze paragraaf hebben we een polynomiaal geval van het multichainclassificatieprobleem voor Markovbeslissingsketens besproken. In de volgende paragraaf bespreken we nog een polynomiaal geval.

4.4 Absorberende toestanden

We zullen zien dat een Markovbeslissingsketen met absorberende toestanden eenvoudig te classificeren is. Daartoe geven we een polynomiaal algoritme om voor een Markovbeslissingsketen met een dergelijke toestand te bepalen of hij unichain of multichain is.

Een absorberende toestand j vormt een recurrente klasse voor een bepaalde strategie f. Om te bepalen of er nog een recurrente klasse is bekijken we weer de verzameling $V^{-1}(\{j\})$.

1. Als $V^{-1}(\{j\}) = \emptyset$ is j voor elke strategie bereikbaar vanuit elke andere toestand, daarom is er voor elke strategie precies één recurrente klasse en is de Markovbeslissingsketen unichain.
2. Als $V^{-1}(\{j\}) \neq \emptyset$ bevat $V^{-1}(\{j\})$ een recurrente klasse (lemma 4.3). De Markovbeslissingsketen is dan multichain, want voor strategie f zijn er twee verschillende recurrente klassen.

Het volgende algoritme berust op deze eigenschap van $V^{-1}(\{j\})$.

Algoritme 4.4 (Classificatie van MBK's met absorberende toestanden)

1. Voor alle $j \in S$ doe: Als er een $j \in S$ en een $a \in A(j)$ zijn met $p_{jj}(a) = 1$ doe:
   (a) Pas algoritme 4.2 toe op $Z = \{j\}$.
   (b) Als $V^{-1}(\{j\}) = \emptyset$: stop. *De MBK is unichain.*
   (c) Als $V^{-1}(\{j\}) \neq \emptyset$: stop. *De MBK is multichain.*
2. Stop. *De MBK bevat geen absorberende toestanden.*

We passen het algoritme toe op de voorbeelden 4.1, 4.2 en 4.3.

Voorbeeld 4.1 (vervolg) De linker Markovbeslissingsketen in figuur 4.1 is unichain want toestand 1 is absorberend en $V^{-1}(\{1\}) = \emptyset$.

Voorbeeld 4.2 (vervolg) De Markovbeslissingsketen in het midden van figuur 4.1 kan niet geclassificeerd worden, want er zijn geen absorberende toestanden.

Voorbeeld 4.3 (vervolg) De rechter Markovbeslissingsketen in figuur 4.1 is multichain, want toestand 4 is absorberend en $V^{-1}(\{4\}) = \{1, 2, 3\} \neq \emptyset$.

We kunnen vooraan algoritme 4.4 een stap toevoegen die bepaalt of er twee absorberende toestanden zijn. Op dat moment kunnen we direct concluderen dat de Markovbeslissingsketen multichain is. Dit maakt het algoritme sneller als er meerdere absorberende toestanden zijn, maar heeft geen invloed op de complexiteit van het algemene geval.

De juistheid en de polynomialiteit van het algoritme zijn eenvoudig aan te tonen.

Stelling 4.5

1. *Algoritme 4.4 geeft de juiste output.*
2. *Algoritme 4.4 is polynomiaal.*

Bewijs

1. De juistheid volgt direct uit het voor algoritme 4.4 besprokene.
2. Stap 1, het vinden van een absorberende toestand, heeft complexiteit $O(m)$. Algoritme 4.2 heeft een complexiteit van $O(m \cdot n)$, maar hoeft maximaal één keer uitgevoerd te worden, omdat één absorberende toestand voldoende is om te beslissen of de Markovbeslissingsketen unichain of multichain is. De complexiteit van algoritme 4.4 is daarom $O(m \cdot n)$. □

Uit stelling 4.5 volgt dat een Markovbeslissingsketen met absorberende toestanden polynomiaal te classificeren is. In paragraaf 4.3 hebben we gezien dat het voldoende is om te weten dat er een j is met $V^{-1}(\{j\}) = \emptyset$ om te kunnen concluderen dat de Markovbeslissingsketen unichain is, ook als j niet absorberend is. Daarom behandelen we in de laatste paragraaf een algoritme dat de twee polynomiale gevallen op een efficiënte manier in één keer classificeert.

4.5 Conclusie

In dit hoofdstuk hebben we een tweetal polynomiale gevallen van het multichainclassificatieprobleem behandeld.

De algoritmes voor de classificatie van Markovbeslissingsketens met recurrente of absorberende toestanden maken beide gebruik van de verzameling $V^{-1}(\{j\})$ en lijken nogal op elkaar. Toch is het onvoldoende om maar één van beide algoritmes te gebruiken. Voorbeeld 4.2 wordt namelijk wel geclassificeerd door algoritme 4.3, maar niet door algoritme 4.4 en voor voorbeeld 4.3 geldt het omgekeerde.

We geven daarom één algoritme met complexiteit $O(m \cdot n^2)$ dat Markovbeslissingsketens met recurrente toestanden of absorberende toestanden classificeert.

Algoritme 4.5 (Classificatie van MBK's met recurrente of absorberende toestanden)

1. Voor alle $j \in S$ doe:
   - (a) i. Pas algoritme 4.2 toe op $Z = \{j\}$.
     ii. Als $V^{-1}(\{j\}) = \emptyset$: stop. *De MBK is unichain.*
   - (b) i. Als er een $a \in A(j)$ is met $p_{jj}(a) = 1$: stop. *De MBK is multichain.*
   - (c) i. Pas algoritme 4.1 toe op $Z = V^{-1}(\{j\})$.
     ii. Als $j \notin B^{-1}(V^{-1}(\{j\}))$: stop. *De MBK is multichain.*
2. Stop. *De MBK bevat geen recurrente of absorberende toestanden.*

De juistheid en de polynomialiteit volgen direct uit de juistheid en de polynomialiteit van algoritme 4.3 en 4.4.

5 Polynomiaal geval 2: Deterministische Markovbeslissingsketens

5.1 Inleiding

Over het algemeen is het multichainclassificatieprobleem NP-volledig. In dit hoofdstuk beschouwen we *deterministische* Markovbeslissingsketens, dat wil zeggen dat alle overgangskansen 0 of 1 zijn. Een voorbeeld van een deterministische Markovbeslissingsketen hebben we in hoofdstuk 1 besproken (zie voorbeeld 1.1). McCuaig bewees in [5] dat het multichainclassificatieprobleem in het deterministische geval polynomiaal is.

We zetten het multichainclassificatieprobleem om in een probleem uit de grafentheorie. Daartoe definiëren we de aan een deterministische Markovbeslissingsketen *geassocieerde graaf* G als de gerichte graaf met als knooppunten de toestanden $i \in S$. Een pijl (i, j) is aanwezig als er een $a \in A(i)$ is zodanig dat $p_{ij}(a) = 1$. Zonder beperking van de algemeenheid nemen we aan dat G geen evenwijdige pijlen bevat. De geassocieerde graaf kan wel lussen hebben.

De graaf voldoet aan de volgende stelling, waarbij een graaf G *intercyclisch* genoemd wordt als G geen twee of meer (knooppunts)disjuncte rondes bevat.

Stelling 5.1 *Een Markovbeslissingsketen is multichain dan en slechts dan als de geassocieerde graaf niet intercyclisch is.*

Bewijs Stel de MBK is multichain, dan is er een strategie zodanig dat de bijbehorende Markovketen minstens twee recurrente klassen heeft. Laat de (mogelijkerwijs identieke) toestanden i, j in de eerste klasse zitten en de (mogelijkerwijs identieke) toestanden k, l in de tweede, dan communiceren i en j en ook k en l. Dit betekent dat G een ronde bevat met i en j en een met k en l en dat deze rondes disjunct zijn.

Andersom stel dat G twee disjuncte rondes heeft dan definiëren we een strategie f door voor elk knooppunt i op een ronde actie $a \in A(i)$ te kiezen zodanig dat $p_{ij}(a) = 1$ voor de opvolger j van i. Nu vormen de knooppunten van de disjuncte rondes recurrente klassen in de Markovketen horende bij f, dus is de Markovbeslissingsketen multichain. □

De meest voor de hand liggende methode om te bepalen of een deterministische Markovbeslissingsketen unichain of multichain is, lijkt nu om alle rondes in de geassocieerde graaf te zoeken en vervolgens na te gaan of er twee disjuncte rondes zijn. Dit levert echter geen polynomiaal algoritme op, omdat het aantal rondes exponentieel kan zijn.

Het maximum aantal rondes met k knooppunten is gelijk aan

$$\left(\begin{array}{c} n \\ k \end{array} \right) (k-1)! = \frac{n!}{k(n-k)!},$$

want we kunnen op $\left(\begin{array}{c} n \\ k \end{array} \right)$ manieren k-tallen kiezen en voor elk k-tal zijn er $(k-1)!$ verschillende volgordes, omdat het niet uitmaakt in welk knooppunt we de ronde beginnen. Het maximum aantal rondes is

$$\sum_{k=1}^{n} \frac{n!}{k(n-k)!} \geq \sum_{k=1}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) - 1 = 2^n - 1$$

en dus exponentieel.

Daarom bespreken we in dit hoofdstuk een ander algoritme, dat van McCuaig [5], dat in polynomiale tijd nagaat of een graaf intercyclisch is. In de volgende paragraaf bespreken we allereerst een algoritme dat de geassocieerde graaf reduceert, terwijl de multichainclassificatie-eigenschappen bewaard blijven. Dit levert ons een vereenvoudiging van het probleem op. Vervolgens hoeven we

nog slechts de classificatie van een gereduceerde graaf te onderzoeken. Dit is dan ook het onderwerp van paragraaf 5.4. In paragraaf 5.3 karakteriseren we de verzameling van gereduceerde en intercyclische grafen.

5.2 Reductie van de geassocieerde graaf

In deze paragraaf zullen we het multichainclassificatieprobleem voor deterministische Markovbeslissingsketens vereenvoudigen, door de geassocieerde graaf te reduceren.

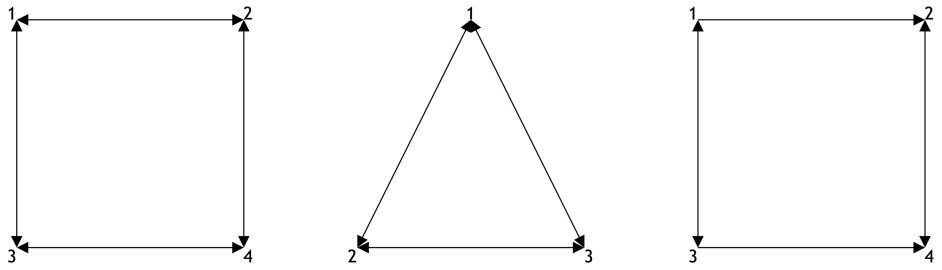
We geven een algoritme om uit een geassocieerde graaf G een *gereduceerde graaf* G' te construeren. Dit is een enkelvoudige strengsamenhangende graaf zodanig dat voor elk knooppunt i de in-graad $\delta^-(i)$ en de uit-graad $\delta^+(i)$ minstens twee is.

Algoritme 5.1 (Constructie van de gereduceerde graaf)

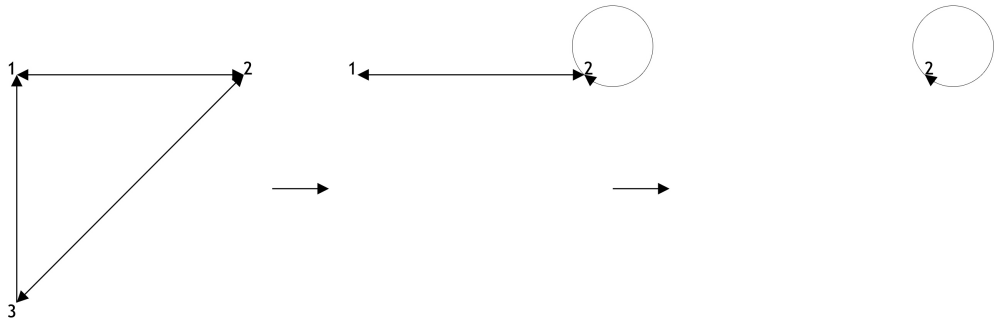
1. Construeer de aan de Markovbeslissingsketen geassocieerde graaf $G = (V(G), A(G))$.
2. (a) Bepaal de strengsamenhangende componenten van G.
   (b) Verwijder de componenten bestaande uit één knooppunt zonder pijlen.
   (c) Als het aantal componenten groter is dan één: stop. *De MBK is multichain.*
3. (a) Laat $T = \{i \in V(G) \mid \min\{\delta^-(i), \delta^+(i)\} = 1\}$.
   (b) Voor elk knooppunt $i \in T$ doe:
      i. Als $\delta^-(i) = 1$: trek pijl (j, i) samen (mits $i \neq j$), dat wil zeggen:
         A. Verwijder knooppunt i uit $V(G)$ en pijl (j, i) uit $A(G)$.
         B. Verwijder de pijlen (i, k) en voeg (j, k) toe als deze er niet waren.
         C. Als $j \notin T$ en $\min\{\delta^-(j), \delta^+(j)\} = 1$: Voeg knooppunt j toe aan T.
      ii. Anders als $\delta^+(i) = 1$: trek pijl (i, j) samen (mits $i \neq j$), dat wil zeggen:
         A. Verwijder knooppunt i uit $V(G)$ en pijl (i, j) uit $A(G)$.
         B. Verwijder de pijlen (k, i) en voeg (k, j) toe als deze er niet waren.
         C. Als $j \notin T$ en $\min\{\delta^-(j), \delta^+(j)\} = 1$: Voeg knooppunt j toe aan T.
      iii. Verwijder knooppunt i uit T.
   (c) Noem de graaf die ontstaan is G'.
4. Als het aantal knooppunten in G' één is: stop. *De MBK is unichain.*
5. Als G' een lus bevat: stop. *De MBK is multichain.*
6. (a) Output: G'.
   (b) Stop.

We laten nu eerst met een aantal voorbeelden zien dat algoritme 5.1 sommige unichaine Markovbeslissingsketens wel classificeert en andere niet en dat hetzelfde geldt voor multichaine Markovbeslissingsketens. (Zie figuur 5.1 tot 5.3.)

Voorbeeld 5.1 Zie figuur 5.1, links.
Deze graaf kan door 5.1 niet worden geclassificeerd, want er is maar één strengsamenhangende component, alle knooppunten hebben een in- en uitgraad van minstens twee, er is meer dan één knooppunt en er is geen lus. De graaf is niet intercyclisch en dus is de corresponderende Markovbeslissingsketen multichain.



Figuur 5.1 Drie geassocieerde grafen. De eerste twee worden niet geclassificeerd door algoritme 5.1, de derde wordt in stap 2c geclassificeerd.



Figuur 5.2 Een geassocieerde graaf en de reductie ervan met behulp van algoritme 5.1. In stap 4 wordt de graaf geclassificeerd.

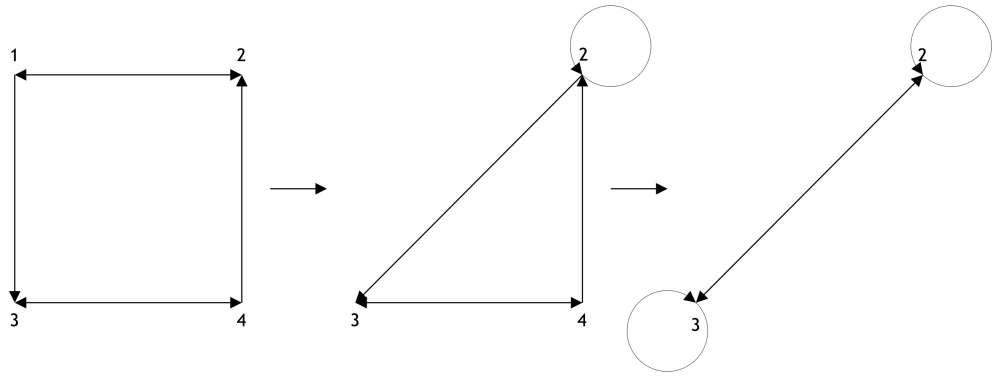
Voorbeeld 5.2 Zie figuur 5.1, midden.
Deze graaf kan door 5.1 niet worden geclassificeerd, want er is maar één strengsamenhangende component, alle knooppunten hebben een in- en uitgraad van minstens twee, er is meer dan één knooppunt en er is geen lus. De graaf is intercyclisch en dus is de corresponderende Markovbeslissinsketen unichain.

Voorbeeld 5.3 Zie figuur 5.1, rechts.
Deze graaf heeft twee strengsamenhangende componenten die beide uit meerdere knooppunten bestaan, dus wordt de graaf in stap 2c van algoritme 5.1 geclassificeerd als niet intercyclisch. De corresponderende Markovbeslissingsketen is multichain.

Voorbeeld 5.4 Zie figuur 5.2.
Deze graaf heeft één strengsamenhangende component. De graaf kan in stap 3 worden gereduceerd door pijl $(2, 3)$, de enige binnenkomende pijl in toestand 3, samen te trekken. Vervolgens wordt ook pijl $(1, 2)$ samen getrokken en blijft er één knooppunt over. De graaf wordt in stap 4 geclassificeerd als intercyclisch. De corresponderende Markovbeslissingsketen is unichain.

Voorbeeld 5.5 Zie figuur 5.3.
Deze graaf heeft één strengsamenhangende component. De graaf kan in stap 3 worden gereduceerd door pijl $(2, 1)$, de enige binnenkomende pijl in toestand 1, samen te trekken. Vervolgens wordt ook pijl $(3, 4)$ samen getrokken en blijft er meer dan één knooppunt over. Aangezien de graaf een lus heeft wordt hij in stap 5 geclassificeerd als niet intercyclisch. De corresponderende Markovbeslissingsketen is multichain.

Voordat we de werking van algoritme 5.1 toelichten, bewijzen we eerst twee lemma's die in de toelichting aangehaald zullen worden.



Figuur 5.3 Een geassocieerde graaf en de reductie ervan met behulp van algoritme 5.1. In stap 5 wordt de graaf geclassificeerd.

Lemma 5.2 *De graaf G' die ontstaan is na stap 3b is intercyclisch dan en slechts dan als G intercyclisch is.*

Bewijs Na een iteratie van stap 3b waarin knooppunt i verwijderd wordt ontstaat uit een ronde C een ronde C/i en ontstaan er geen nieuwe rondes, want een pijl (j,k) wordt dan en slechts dan toegevoegd als er een pad (j,i,k) aanwezig was. Er kunnen ook geen rondes verdwijnen, want rondes worden met maximaal één pijl ingekort en de laatste pijl, een lus, wordt niet verwijderd. De na stap 3b uit ronde C in G ontstane ronde in G' noemen we C'.

Het is nu voldoende om te bewijzen dat twee rondes C' en D' disjunct zijn dan en slechts dan als C en D disjunct waren.

Stel C, D zijn disjuncte rondes in G, dan bevatten C' en D' evenveel of minder knooppunten dan C respectievelijk D, dus zijn C', D' disjunct.

Stel C, D hebben knooppunt i gemeen. Als i wordt verwijderd wordt (j,i) (of (i,j)) met $i \neq j$ samengetrokken. Verder geldt dat (j,i) (of (i,j)) de enige pijl naar (vanuit) i is, dus geldt $j \in C, D$ en ook $j \in C', D'$, dus zijn ook C', D' niet disjunct. □

Dit lemma was van toepassing op stap 3b, het volgende lemma refereert naar stap 5.

Lemma 5.3 *Als voor alle knooppunten in een graaf zonder evenwijdige pijlen geldt dat de in- en uit-graad minstens twee is, is er voor elk knooppunt i een ronde waarin i niet bevat is.*

Bewijs Laat G een graaf zonder evenwijdige pijlen zijn waarin voor elk knooppunt j geldt: $\delta^+(j) \geq 2$ en $\delta^-(j) \geq 2$. Deelgraaf G^* ontstaat door een willekeurig knooppunt i en alle aangrenzende pijlen te verwijderen. In G^* geldt: $\delta^+(j) \geq 1$ en $\delta^-(j) \geq 1$ voor alle $j \in V(G)$. Nu vinden we een ronde door in een willekeurig knooppunt te beginnen en over pijlen van G^* te lopen tot we in een knooppunt komen die we al eerder bezocht hadden. We kunnen steeds blijven lopen, omdat de uit-graad in elk knooppunt minstens één is. Stel G' heeft n knooppunten, dan zullen we uiterlijk in n stappen op een eerder bezocht knooppunt komen. De bezochte knooppunten vormen een ronde, dezelfde ronde is ook aanwezig in G en bevat i niet. □

We lichten nu een aantal stappen van het algoritme toe.

Toelichting op 5.1

Stap 2b: Als een component uit precies één knooppunt i zonder lus bestaat, ligt i op geen enkele ronde en dus is i een onder elke strategie transiënte toestand van de Markovbeslissingsketen.

Stap 2c: Als er meer dan één component is, zijn er twee disjuncte rondes, want elke component (met pijlen) bevat een ronde. Uit 5.1 volgt dat de Markovbeslissingsketen multichain is.

Stap 3: Lemma 5.2 geeft aan dat na deze stap geldt dat G' intercyclisch is dan en slechts dan als G dat was. Na de vorige stap gold voor elk knooppunt i: $\delta^+(i) \geq 1$ en $\delta^-(i) \geq 1$, nu geldt: $\delta^+(i) \geq 2$ en $\delta^-(i) \geq 2$, of er is een lus (i, i).

Stap 4: Als er één knooppunt is, zijn er geen disjuncte rondes in G' en dus ook niet in G en is de Markovbeslissingsketen unichain (stelling 5.1).

Stap 5: G' heeft nu minstens twee knooppunten en er kunnen geen pijlen meer samengetrokken worden. Stel er is een lus (i, i) dan zijn er twee gevallen:

1. Er is nog een lus.
2. Voor elk knooppunt $j \neq i$ geldt: $\delta^+(j) \geq 2$ en $\delta^-(j) \geq 2$. Dan is er volgens lemma 5.3 een ronde die knooppunt i niet bevat.

In beide gevallen bevat G' en dus G twee disjuncte rondes. Nu volgt (stelling 5.1) dat de Markovbeslissingsketen multichain is.

We hebben gezien dat de uitkomsten juist zijn als het algoritme voortijdig stopt. De volgende stelling geeft aan dat het algoritme de gewenste output geeft, in polynomiale tijd.

Stelling 5.4

1. *Algoritme 5.1 construeert uit G een gereduceerde graaf G'.*
2. *G' is intercyclisch dan en slechts dan als G dat was.*
3. *Algoritme 5.1 is polynomiaal.*

Bewijs

1. G' is enkelvoudig, want G bevat geen evenwijdige takken en het algoritme vormt er geen en daarnaast bevat G' na stap 5 geen lussen meer.

   G' is strengsamenhangend, want na stap 2c blijven slechts de grafen met één component over.

   G' heeft na stap 4 minstens twee knooppunten en na stap 5 geen lussen meer. Nu heeft elk knooppunt een in- en uit-graad van minstens twee (want knooppunten met in- of uit-graad één zouden tijdens stap 3b verwijderd zijn).

2. Zie lemma 5.2.

3. Laat n het aantal knooppunten zijn. Het aantal pijlen m in G komt overeen met het totaal aantal acties $\sum_{i \in S} \#A(i)$ in de Markovbeslissingsketen. In onze graaf geldt $n \leq m$, want in de Markovbeslissingsketen is er in elke toestand minstens één actie. De complexiteit van de stappen is:

   Stap 1: $O(m)$,

   Stap 2a: $O(max\{n, m\} = m)$[3],

   Stap 2b: $O(n)$,

   Stap 2c: $O(1)$,

   Stap 3a: $O(n)$,

   Stap 3b: $O(n^2)$, want $O(n)$ iteraties van $O(n)$,

   Stap 3c: $O(1)$,

Stap 4: $O(1)$,

Stap 5: $O(n)$,

Stap 6: $O(m)$.

De complexiteit is $O(n^2)$, aangezien voor het aantal pijlen geldt $m \leq n(n-1) < n^2$. □

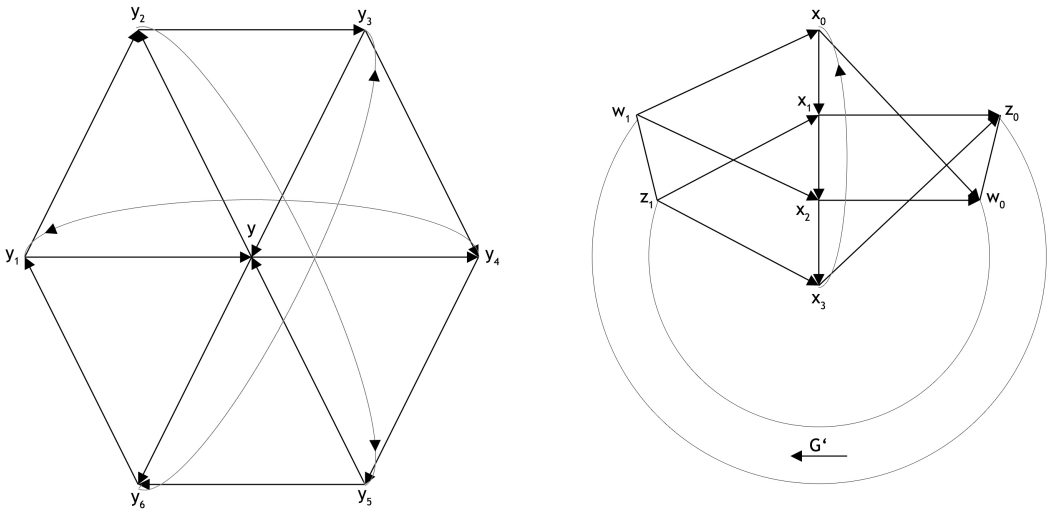
Stelling 5.1 geeft aan dat het multichainclassificatieprobleem equivalent is met de vraag of een graaf intercyclisch is. Stelling 5.4 laat zien dat dit probleem in polynomiale tijd te reduceren is tot de vraag of een gereduceerde graaf intercyclisch is. Hiermee is het multichainclassificatieprobleem van een deterministische Markovbeslissingsketen enigszins vereenvoudigd.

5.3 Karakterisering van de gereduceerde intercyclische grafen

In de vorige paragraaf hebben we de reductie van de geassocieerde graaf besproken. McCuaig beschrijft de omzetting van het multichainclassificatieprobleem naar een graaftheoretisch probleem en de graafreductie op pagina 209 van [5]. Vervolgens definieert hij $\mathcal{I}$ als de verzameling van gereduceerde en intercyclische grafen en poneert de volgende stelling.

Stelling 5.5 (Hoofdstelling) *Voor de verzameling van gereduceerde en intercyclische grafen $\mathcal{I}$ geldt $\mathcal{I} = \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.*

Hierbij is $\mathcal{T}$ de verzameling van enkelvoudige grafen G met voor elk knooppunt een in- en uitgraad van minstens twee, waarvoor er twee verschillende knooppunt x en y zijn, zodat de graaf G' acyclisch is en er geen disjuncte paden (x^+, x^-) en (y^+, y^-) bestaan, als G' verkregen wordt uit G door x op te splitsen in x^+ en x^- (waarbij alle in x binnenkomende takken (z, x) in G vervangen worden door (z, x^-) in G' en de uitgaande takken (x, z) door (x^+, z)) en y op dezelfde manier op te splitsen in y^+ en y^-.



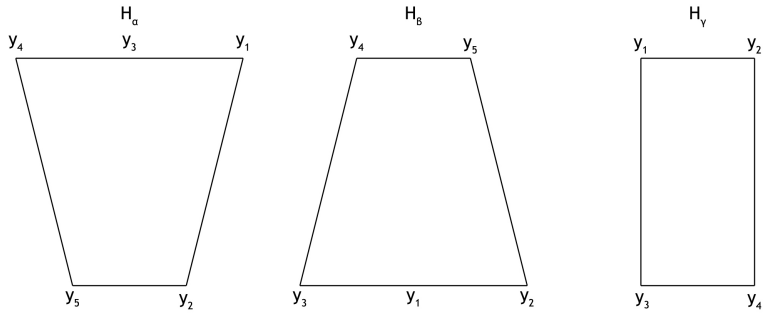
Figuur 5.4 Gereduceerde intercyclische grafen, links de graaf D_7 en rechts een graaf in $\mathcal{K}$ schematisch weergegeven.

Graaf D_7 is de graaf van figuur 5.4. De pijl (y_1, y_4) in het artikel van McCuaig is vervangen door de pijl (y_4, y_1), anders is de graaf niet gereduceerd.

Graaf G heeft een *2-transversaal* als uit G twee knooppunten verwijderd kunnen worden zodanig dat de resterende graaf acyclisch is.

De verzameling $\mathcal{K}$ bevat alle grafen G met voor elk knooppunt een in- en uitgraad van minstens twee en geen 2-transversaal, die als volgt verkregen kunnen worden: Laat G' een acyclische

graaf zijn, met knooppunten w_0 en z_0 die alleen uitgaande pijlen hebben en w_1 en z_1 met alleen binnenkomende pijlen, zo dat er geen disjuncte paden (w_0, w_1) en (z_0, z_1) bestaan. We voegen nu mogelijkerwijs één pijl uit $\{(w_0, z_0), (z_0, w_0)\}$ toe en mogelijkerwijs één pijl uit $\{(w_1, z_1), (z_1, w_1)\}$. Vervolgens voegen we vier knooppunten x_0, x_1, x_2 en x_3 en de pijlen (x_0, x_1), (x_1, x_2), (x_2, x_3), (x_3, x_0), (w_1, x_0), (w_1, x_2), (z_1, x_1), (z_1, x_3), (x_0, w_0), (x_2, w_0), (x_1, z_0) en (x_3, z_0) toe. De verkregen graaf noemen we G (zie figuur 5.4).



Figuur 5.5 Een gereduceerde en intercyclische graaf in de verzameling $\mathcal{H}$ schematisch weergegeven.

De verzameling $\mathcal{H}$ is gedefinieerd als de verzameling van alle grafen G met voor elk knooppunt een in- en uitgraad van minstens twee en geen 2-transversaal, zodanig dat G de vereniging is van de pijldisjuncte, acyclische grafen $H_\alpha, H_\beta, H_\gamma$ en G vijf knooppunten y_1, y_2, y_3, y_4, y_5 heeft waarvoor geldt dat het de enige knooppunten zijn die in meer dan één van de grafen $H_\alpha, H_\beta, H_\gamma$ zitten. Verder geldt voor H_α dat er geen disjuncte paden (y_4, y_2) en (y_3, y_5) zijn, geen disjuncte paden (y_3, y_2) en (y_1, y_5) en geen disjuncte paden (y_4, y_2) en (y_1, y_5). Voor H_β geldt dat er geen disjuncte paden (y_4, y_1) en (y_5, y_3) zijn, geen disjuncte paden (y_4, y_2) en (y_5, y_1) en geen disjuncte paden (y_4, y_2) en (y_5, y_3). En voor H_γ geldt dat er geen disjuncte paden (y_1, y_4) en (y_2, y_3) zijn (zie figuur 5.5).
De volgende dertig pagina's [5, pp. 211–240] zijn gewijd aan het bewijs van deze stelling. Het bewijs is lang en niet erg overzichtelijk. We geven hier een schets van het bewijs. In de volgende paragraaf geven we het algoritme dat bepaalt of een gereduceerde graaf intercyclisch is of niet.

We reproduceren nu eerst een aantal lemma's zonder de bewijzen te geven.

Lemma 5.6 *Er geldt $G \in \mathcal{T}$ d.e.s.d.a. $G \in \mathcal{I}$ een 2-transversaal heeft [5, Lemma 4.1].*

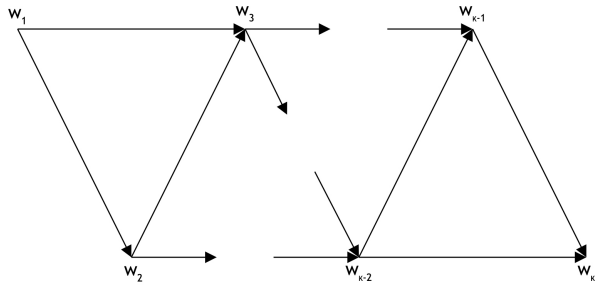
Laat $\mathcal{I}_t$ de verzameling van gereduceerde intercyclische grafen zonder *2-transversaal* zijn. Stel x, y, z zijn verschillende knooppunten van G en $(x, y), (x, z), (y, z)$ zijn pijlen in G, als $d^+(x) = 2$ noemen we (x, y) samentrekbaar en als $d^-(z) = 2$ noemen we (y, z) samentrekbaar.

Lemma 5.7 *Als $H \in \mathcal{I}$ met minder pijlen dan G in $\mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ zit en G in $\mathcal{I}_t$ dan heeft G een samentrekbare pijl [5, Stelling 6.1].*

De grafen $R_k, k \geq 2$ zijn recursief gedefinieerd: R_2 heeft knooppunten w_1, w_2 en pijl (w_1, w_2), R_k wordt gevormd uit R_{k-1} door het knooppunt w_k en de pijlen (w_{k-2}, w_k) en (w_{k-1}, w_k) toe te voegen. Zie figuur 5.6.

Een deelgraaf R_k van G heet *goed* als R_k geïnduceerd is ($(w_i, w_j) \in A(G)$ met $w_i, w_j \in V(G) \Rightarrow (w_i, w_j) \in A(R_k)$), er alleen pijlen van een knooppunt in $V(G)/V(R_k)$ naar een knooppunt in $w_i \in V(R_k)$ zijn als $i = 1$ of $i = 2$, er alleen pijlen van w_i naar een knooppunt in $V(G)/V(R_k)$ zijn als $i = k$ of $i = k - 1$ en als elk knooppunt in $V(G)/V(R_k)$ minstens één pijl heeft naar een ander knooppunt in $V(G)/V(R_k)$ en minstens één pijl vanuit een ander knooppunt in $V(G)/V(R_k)$.

De graaf H wordt verkregen uit G door middel van een *R_k-reductie* als G een goede R_k-deelgraaf heeft, deze vervangen wordt door één knooppunt ρ, de pijlen x, w_1 en/of x, w_2 vervangen worden



Figuur 5.6 De graaf R_k (met k even).

door een pijl (x, ρ) en de pijlen (w_{k-1}, x) en/of (w_k, x) door een pijl (ρ, x). Als G gereduceerd en intercyclisch is en H is verkregen uit G door R_k-reductie, is ook H gereduceerd en intercyclisch.

De verzameling van alle grafen R_k met k oneven waaraan de pijlen $(w_{k-1}, w_1), (w_k, w_1)$ en (w_k, w_2) zijn toegevoegd noemen we $\mathcal{M}$.

Lemma 5.8 *Als $G \in \mathcal{I}$ een samentrekbare pijl heeft dan heeft G een R_k-reductie of zit G in de verzameling $\mathcal{M}$ [5, Lemma 5.2].*

Deze verzameling $\mathcal{M}$ is een deelverzameling van $\mathcal{T}$, omdat elke graaf in M een 2-transversaal heeft. Daarom geldt het volgende.

Gevolg 5.9 *Als $G \in \mathcal{I}_t$ een samentrekbare pijl heeft dan heeft G een R_k-reductie.*

Als H ontstaat door R_k-reductie toe te passen op $G \in \mathcal{I}$ heeft H minder pijlen dan G en is ook H gereduceerd en intercyclisch.

Lemma 5.10 *Als $H \in \mathcal{T}$ is verkregen door R_k-reductie van $G \in \mathcal{I}_t$ geldt $G \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ [5, paragraaf 7].*

Lemma 5.11 *Als $H \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ is verkregen door R_k-reductie van $G \in \mathcal{I}_t$ geldt $G \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ [5, paragraaf 8].*

We geven nu een schets van het bewijs van de hoofdstelling (stelling 5.5).

Bewijsschets

1. Om te bewijzen dat uit $G \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ volgt dat $G \in \mathcal{I}$ moeten we nagaan of de grafen in $\mathcal{T}$,$\{D_7\}, \mathcal{K}$ en $\mathcal{H}$ gereduceerd en intercyclisch zijn. Dit is arbeidsintensief, maar niet erg ingewikkeld. We zullen de gewenste eigenschappen voor D_7 nalopen.

   De graaf D_7 is enkelvoudig, strengsamenhangend en elk knooppunt heeft een in- en uitgraad van minstens twee, dus D_7 is gereduceerd. Er zijn slechts vier rondes die het knooppunt y niet bevatten: $(y_1, y_2, y_3, y_4, y_5, y_6), (y_2, y_5, y_6, y_1), (y_4, y_1, y_2, y_3)$ en (y_6, y_3, y_4, y_5). Er is geen enkele ronde disjunct aan de eerste ronde, omdat deze zes van de zeven knooppunten bevat. Er zijn ook geen rondes disjunct aan de andere drie rondes, omdat er geen rondes door de knooppunten y, y_3, y_4 zijn, noch door y, y_5, y_6 of door y, y_1, y_2. De graaf D_7 is dus ook intercyclisch.

2. Het bewijs in de andere richting ($G \in \mathcal{I} \Rightarrow \mathcal{G} \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$) wordt gedaan met inductie naar het aantal pijlen.

   (a) Als beginstap nemen we de volledige gerichte graaf C_3 met drie knooppunten en een twee pijlen (i, j) en (j, i) tussen elke twee knooppunten i en j. Deze graaf is gereduceerd en intercyclisch en er geldt $C_3 \in \mathcal{T}$ volgens lemma 5.6.

(b) In de inductiestap willen we voor een willekeurige graaf $G \in \mathcal{I}$ bewijzen dat $G \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ als voor elke graaf $H \in \mathcal{I}$ met minder pijlen dan G geldt dat $H \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.

Vanwege lemma 5.6 is het voldoende te bewijzen dat voor een willekeurige graaf $G \in \mathcal{I}_t$ geldt dat $G \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ als voor elke graaf $H \in \mathcal{I}$ met minder pijlen dan G geldt dat $H \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.

Laat G nu een willekeurige graaf in $\mathcal{I}_t$ zijn en stel dat de inductievooronderstelling geldt. Dan volgt uit lemma 5.7 dat G een samentrekbare pijl heeft.

Met behulp van gevolg 5.9 kunnen we concluderen dat G een R_k-reductie heeft. Als H ontstaat door R_k-reductie toe te passen op G geldt volgens de inductievooronderstelling $H \in \mathcal{T} \cup \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.

We onderscheiden nu twee gevallen.

i. Stel dat $H \in \mathcal{T}$ dan volgt uit lemma 5.10 dat $G \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.

ii. Stel dat $H \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$ dan volgt uit lemma 5.11 dat $G \in \{D_7\} \cup \mathcal{K} \cup \mathcal{H}$.

(c) Uit de beginstap en de inductiestap volgt dat $G \in \mathcal{T} \cup \{\mathcal{D}_l\} \cup \mathcal{K} \cup \mathcal{H}$ als $G \in \mathcal{I}$. □

In de volgende paragraaf geven we een algoritme om te bepalen of een gereduceerde graaf intercyclisch is. Daarbij maken we gebruik van de in deze paragraaf gegeven lemma's.

5.4 Classificatie van een gereduceerde graaf

In paragraaf 5.2 hebben we de geassocieerde graaf gereduceerd. In deze paragraaf bespreken we een algoritme om voor de gereduceerde graaf vervolgens te bepalen of hij intercyclisch is.

In [5, paragraaf 9] geeft McCuaig een polynomiaal algoritme dat bepaalt of een gereduceerde graaf intercyclisch is. Als dit niet het geval is geeft het algoritme twee disjuncte rondes. We verwachtten dat dit algoritme gebruik zou maken van de hoofdstelling (stelling 5.5) en zou nagaan of een gegeven graaf in $T, \{D_7\}, K$ of H voorkomt.

Dit is echter niet het geval. Het algoritme maakt gebruik van door McCuaig niet nader genoemde resultaten uit het bewijs van de hoofdstelling: "Then the proof of Lemma 4.2 can be used to find an R_k-reduction of G to digraph H, find two disjoint dicycles of G, or show that G is in $\mathcal{M}$... If H is obtained from G using an R_k-reduction, then the results and proofs of paragraph 6 and paragraph 7 can be used to either show that G is intercyclic and give its structure, or find two disjoint dicycles of G."[5, p.241]. In dit citaat wordt lemma 5.2 (ons lemma 5.8 bedoeld in plaats van lemma 4.2 en worden paragraaf 7 en 8 bedoeld in plaats van paragraaf 6 en 7.

Het algoritme is gegeven in de vorm van een toelichting en zonder bewijs van de polynomialiteit. We hebben het algoritme hieronder schematisch gereproduceerd. Aangezien we alleen geïnteresseerd zijn in de vraag of de graaf intercyclisch is, laten we de stappen voor de bepaling van de disjuncte rondes in een niet-intercyclische graaf achterwege.

Algoritme 5.2 (Classificatie van een gereduceerde graaf G')

1. (a) Laat $i = 0$.
   (b) Laat $G(i) = G'$.
   (c) Laat $G(i-1) = \emptyset$.
2. Zolang $G(i) \neq G(i-1)$ doe:
   (a) Als $G(i)$ geen samentrekbare pijl heeft: Stop. *G' is niet intercyclisch.*
   (b) i. Pas het bewijs van lemma 5.2 toe op $G = G(i)$.
   ii. Als $G(i)$ disjuncte rondes heeft: Stop. *G' is niet intercyclisch.*
   iii. Als $G(i) \notin \mathcal{M}$: Voer de R_k-reductie uit en noem de verkregen graaf $G(i+1)$.

(c) Laat $i = i + 1$.

3. Zolang $i \neq 0$ doe:

   (a) i. Pas de resultaten en het bewijs uit paragraaf 7 en 8 toe op $H = G(i)$ en $G = G(i-1)$.
   ii. Als H disjuncte rondes heeft: Stop. *G' is niet intercyclisch.*

   (b) Laat $i = i - 1$.

4. Stop. *G' is intercyclisch.*

Het volgende lemma wordt gebruikt in de toelichting.

Lemma 5.12 *Elke intercyclische gereduceerde graaf heeft een samentrekbare pijl.*

Volgens McCuaig volgt lemma 5.12 direct uit de hoofdstelling (stelling 5.5), maar dit lijkt onwaarschijnlijk, want in dit geval zou lemma 5.7 ook direct uit de hoofdstelling volgen. Toch is het bewijs van lemma 5.7 zes pagina's lang.

Toelichting op algoritme 5.2

Stap 1: Het algoritme maakt een rij $\{G(i)\}_{i=0}^{n}$ van gereduceerde grafen, beginnend bij $G(0) = G'$.

Stap 2: Daarbij ontstaat $G(i+1)$ door een R_k-reductie toe te passen op $G(i)$. We stoppen na $k+1$ iteraties als $G(k)$ niet intercyclisch is of als $G(k) \in \mathcal{M}$ (in dat geval vindt er geen R_k-reductie plaats, dus geldt $G(k+1) = G(k)$ en stopt de loop). Als $G(k)$ niet intercyclisch is, was de oorspronkelijke graaf G' dat namelijk ook niet en als $G(k) \in \mathcal{M}$ is $G(k)$ intercyclisch en gaan we verder met stap 3.

We kunnen er op twee manieren achterkomen dat $G(k)$ niet intercyclisch is: als er geen samentrekbare pijlen zijn (zie lemma 5.12), of als we met behulp van het bewijs van lemma 5.2 (ons lemma 5.8) twee disjuncte rondes vinden.

Als we met behulp van het bewijs van lemma 5.2 niet kunnen concluderen dat $G(k)$ niet intercyclisch is en ook niet dat $G(k) \in M$, weten we niet of de graaf intercyclisch is of niet, maar kunnen we wel een R_k-reductie vinden.

Stap 3: Als $G(k) \in \mathcal{M}$ gaat het algoritme de rij $\{G(i)\}_{i=0}^{n}$ achterstevoren af om te bepalen of $G(i)$ intercyclisch is of niet. Dit gebeurt met behulp van de bewijzen en resultaten uit paragraaf 7 en 8. (Paragraaf 7 en 8 bestaan uit de bewijzen van onze lemma's 5.10 en 5.11.) Zodra er een i gevonden wordt zodanig dat $G(i)$ niet intercyclisch is, stoppen we en concluderen we dat G' niet intercyclisch was.

Stap 4: Alleen als een dergelijke i niet gevonden wordt, was G' intercyclisch.

In deze paragraaf hebben we een schets gegeven van het algoritme dat bepaalt of een gereduceerde graaf intercyclisch is. Samen kunnen algoritme 5.1 en 5.2 een deterministische Markovbeslissingsketen classificeren. Als algoritme 5.2 aangeeft dat de gereduceerde graaf intercyclisch is, was de Markovbeslissingsketen unichain en anders was hij multichain.

5.5 Conclusie

We hebben gezien dat het multichainclassificatieprobleem voor deterministische Markovbeslissingsketens in polynomiale tijd kan omgezet worden in een grafentheoretisch probleem. De Markovbeslissingsketen wordt omgezet in zijn geassocieerde graaf, waarvan vervolgens bepaald moet worden of hij intercyclisch is.

In dit hoofdstuk hebben we twee polynomiale algoritmes besproken. Algoritme 5.1 (zie paragraaf 5.2 maakt van de geassocieerde graaf een gereduceerde graaf en algoritme 5.2 (zie paragraaf 5.4 bepaalt of een gereduceerde graaf intercyclisch is.

Van het eerste algoritme hebben we de juistheid en polynomialiteit bewezen. Het tweede algoritme is impliciet gedefinieerd. Het expliciet definiëren van dit algoritme zou een onderwerp voor verder onderzoek kunnen zijn. Om het algoritme te kunnen programmeren, zouden we de bewijzen van ons lemma 5.8 en vooral de bewijzen van onze lemma's 5.10 en 5.11 nader moeten bekijken. Het gaat hier echter om zeventien pagina's aan bewijzen die niet erg leesbaar zijn.

We hebben voor deterministische Markovbeslissingsketens dus nog niet aangetoond dat het multichainclassificatieprobleem polynomiaal is, maar we hebben voor deterministische Markovbeslissingsketens wel een polynomiale reductie van het probleem gevonden.

6 Verdere polynomiale gevallen en reducties

6.1 Inleiding

In de vorige hoofdstukken hebben we een aantal polynomiale deelproblemen van het multichainclassificatieprobleem beschouwd. Met behulp van de besproken algoritmes kon voor Markovbeslissingsketens met specifieke eigenschappen worden bepaald of hij unichain of multichain is. Het ging hier om Markovbeslissingsketens met een recurrente of absorberende toestand en deterministische Markovbeslissingsketens. In dit hoofdstuk zullen we nog een aantal polynomiale algoritmes bekijken.

Allereerst bespreken we een algoritme uit Feinberg en Yang [1] dat (in polynomiale tijd) de toestanden verwijdert die voor elke strategie transiënt zijn.

In paragraaf 6.3 zullen we dit algoritme gebruiken voor de reductie van het probleem tot een multichainclassificatieprobleem voor communicerende Markovbeslissingsketens. Het algoritme (ook uit [1]) dat deze reductie uitvoert lost tegelijkertijd een deelprobleem van het multichainclassificatieprobleem op, namelijk het multichainclassificatieprobleem voor niet zwak communicerende Markovbeslissingsketens.

Tot slot beschouwen we een algoritme uit Kallenberg [2] dat de Markovbeslissingsketen in een aantal gevallen classificeert. Dit algoritme maakt gebruik van de gerichte graaf G^1 met toestandsruimte S en pijlen (i, j) met $i \neq j$ als voor alle $a \in A(i)$ geldt $p_{ij} > 0$.

Dit hoofdstuk is bedoeld om het overzicht van wat er op dit moment is gedaan om het multichainclassificatieprobleem te vereenvoudigen en deelproblemen ervan op te lossen, te completeren. Niet alle algoritmes in dit hoofdstuk zullen geconcretiseerd worden en van andere algoritmes zullen we de bewijzen voor de correctheid en de polynomialiteit achterwege laten.

6.2 Transiënte toestanden

We noemen een toestand in een Markovbeslissingsketen *transiënt* als die toestand voor elke strategie f transiënt is in $P(f)$. We zullen zien dat deze toestanden verwijderd kunnen worden zonder de multichainclassificatie-eigenschappen te veranderen.

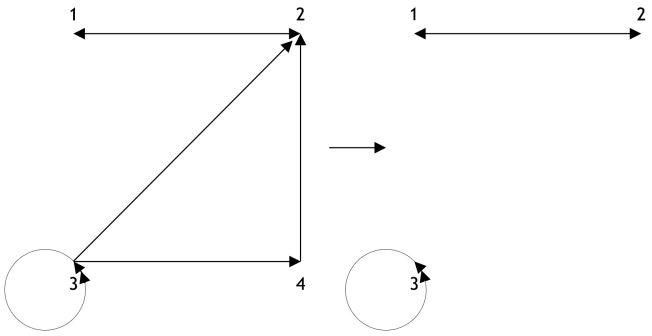
In Kallenberg [2] vinden we een polynomiaal algoritme ('Bather's decomposition algorithm', algoritme 7) dat de verzameling T van transiënte toestanden in een Markovbeslissingsketen detecteert.

We verwijderen nu alle transiënte toestanden $i \in T$ uit de toestandsruimte. Ook verwijderen we alle acties waarvoor er een positieve overgangskans is naar een transiënte toestand. Zo krijgen we een nieuwe Markovbeslissingsketen met toestandsruimte S' en actieverzamelingen $A'(i)$ voor $i \in S'$.

Algoritme 6.1 (Transiënte toestanden verwijderen)

1. (a) Pas algoritme 7 uit [2] toe.
   (b) Laat $T = T_1 \cup \cdots \cup T_m$.
2. (a) Laat $S' = S/T$
   (b) Laat $A'(i) = A(i)$ voor alle $i \in S'$.
   (c) Voor alle $i \in S'$ en alle $a \in A'(i)$ doe: Als $\sum_{j \in T} p_{ij}(a) > 0$: Verwijder a uit $A'(i)$.
3. Stop. *De MBK bevat geen transiënte toestanden meer.*

We passen het algoritme toe op de Markovbeslissingsketen van figuur 6.1.



Figuur 6.1 Een Markovbeslissingsketen met vier toestanden, met behulp van algoritme 6.1 wordt toestand 4, die transiënt is, verwijderd.

Voorbeeld 6.1 Toestand 4 is transiënt voor elke strategie en wordt dus verwijderd. Toestand 2 heeft voor actie 1 een positieve overgangskans naar de transiënte toestand, daarom wordt de pijl $(2, 1)$ bij actie 1 verwijderd.

In het algoritme verwijderen we de acties met een positieve overgangskans naar een transiënte toestand, want als voor een bepaalde strategie in toestand i een dergelijke actie wordt gekozen, is i transiënt voor deze strategie. Het kan niet gebeuren dat er een toestand $i \in S'$ is waarvoor $A'(i) = \emptyset$, want in dit geval zou i voor elke strategie transiënt geweest zijn.

De complexiteit is, net als het aangeroepen algoritme, $O(m \cdot n^2)$.

Stelling 6.1 *De multichainclassificatie-eigenschappen blijven bewaard onder algoritme 6.1.*

Bewijs Stel dat de nieuwe Markovbeslissingsketen multichain is. Dan is er een strategie f' op S' zodanig dat er meerdere recurrente klassen zijn. Elke strategie f in de oorspronkelijke Markovbeslissingsketen met $f(i) = f'(i)$ voor $i \in S'$ levert dan ook meerdere recurrente klassen op, want een gesloten communicerende klasse blijft op deze manier gesloten en communicerend. Dus was ook de oorspronkelijke Markovbeslissingsketen multichain.

Stel nu dat de oorspronkelijke Markovbeslissingsketen multichain was. Dan was er een strategie f zodanig dat er meerdere recurrente klassen waren. Elke strategie f' in de nieuwe Markovbeslissingsketen waarvoor geldt $f'(i) = f(i)$ als $f(i) \in A'(i)$ geeft meerdere recurrente klassen. Er geldt namelijk dat i transiënt en geen onderdeel van een gesloten communicerende klasse was als $f(i) \notin A'(i)$. Daarom blijft een gesloten communicerende klasse gesloten en communicerend. Nu volgt dat ook de nieuwe Markovbeslissingsketen multichain is. □

In deze paragraaf hebben we een eenvoudige reductie van het probleem besproken. In de volgende paragraaf bespreken we een reductie, die gebruik maakt van algoritme 6.1.

6.3 Communicerende Markovbeslissingsketens

In deze paragraaf maken we gebruik van een andere classificatie van Markovbeslissingsketens, namelijk van de communicerende en zwak communicerende eigenschappen. We zullen ons probleem reduceren tot het multichainclassificatieprobleem voor communicerende Markovbeslissingsketens en tegelijkertijd de Markovbeslissingsketens die niet zwak communicerend zijn classificeren.

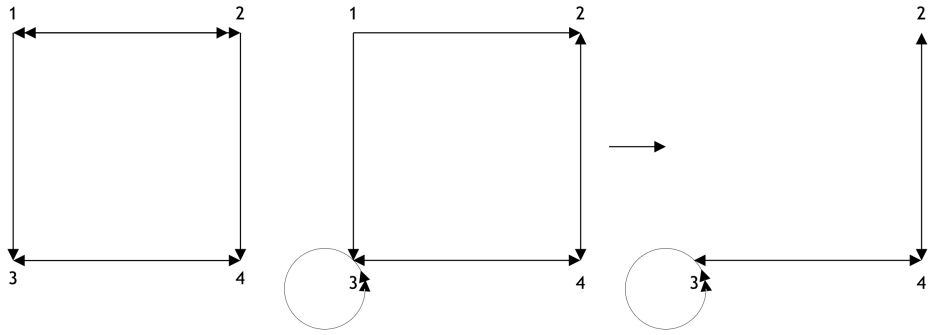
De volgende stelling volgt direct uit de definities.

Stelling 6.2 *Als een Markovbeslissingsketen unichain is, is hij zwak communicerend.*

Algoritme 4 in Kallenberg [2] bepaalt in polynomiale tijd of een Markovbeslissingsketen zwak communicerend is of niet. Als dit niet het geval is kunnen we vanwege stelling 6.2 concluderen dat de Markovketen multichain is. Met behulp van algoritme 6.1 kunnen we een zwak communicerende Markovbeslissingsketen omzetten in een communicerende Markovbeslissingsketen. Uit de definities volgt direct dat een zwak communicerende Markovbeslissingsketen overgaat in een communicerende Markovbeslissingsketen wanneer de transiënte toestanden verwijderd worden. Algoritme 6.2 voert deze acties uit.

Algoritme 6.2 (Reductie tot een communicerende Markovbeslissingsketen)

1. Pas algoritme 4 uit [2] toe.
2. Als de Markovbeslissingsketen niet zwak communicerend is: Stop. *De MBK is multichain.*
3. Pas algoritme 6.1 toe.
4. Stop. *De MBK is nu communicerend.*



Figuur 6.2 Links: een Markovbeslissingsketen met vier toestanden die met behulp van algoritme 6.1 wordt geclassificeerd als multichain. Rechts: een zwak-communicerende Markovbeslissingsketen met vier toestanden die met behulp van algoritme 6.1 communicerend wordt gemaakt.

De complexiteit van beide aangeroepen algoritmes is $O(m \cdot n^2)$, dit is dus ook de complexiteit van algoritme 6.2.

We geven twee voorbeelden. Het eerste voorbeeld wordt door algoritme 6.2 geclassificeerd als multichain. De zwak-communicerende Markovbeslissingsketen van het tweede voorbeeld wordt communicerend gemaakt.

Voorbeeld 6.2 De linker Markovbeslissingsketen van figuur 6.2 is niet zwak-communicerend aangezien toestand 1 en 3 beiden niet transiënt zijn, maar toestand 1 voor geen enkele strategie bereikbaar is vanuit toestand 3. De Markovbeslissingsketen is dus multichain.

Voorbeeld 6.3 De middelste Markovbeslissingsketen van figuur 6.2 is zwak-communicerend, want voor elk geordend tweetal uit de verzameling van niet-transiënte toestanden ($\{2, 3, 4\}$) is er een strategie zo de eerste bereikbaar is vanuit de tweede. Toestand 1 is transiënt voor elke strategie en wordt dus verwijderd (rechter afbeelding). Er zijn geen toestanden met positieve overgangskansen naar de transiënte toestand, dus de acties en overgangskansen voor toestand 2, 3 en 4 blijven gelijk. De ontstane Markovbeslissingsketen is communicerend.

We hebben nu het multichainclassificatieprobleem voor algemene Markovbeslissingsketens polynomiaal gereduceerd tot een multichainclassificatieprobleem voor communicerende Markovbeslissingsketens en het probleem opgelost voor niet zwak communicerende Markovbeslissingsketens.

6.4 Gecondenseerde graaf

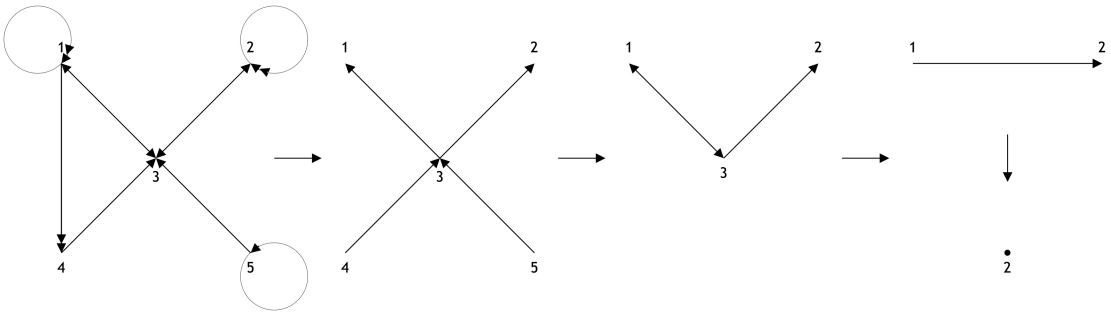
We geven nu een algoritme dat uit G^1 een zogenaamde gecondenseerde graaf construeert en de Markovbeslissingsketen in een aantal gevallen classificeert. Vervolgens zullen we het algoritme en het begrip gecondenseerde graaf kort bespreken.

In het algoritme worden telkens knooppunten samengevoegd, hierbij is V_i de verzameling van knooppunten die samengevoegd is tot knooppunt i.

Algoritme 6.3 (Classificatie van een aantal Markovbeslissingsketens)

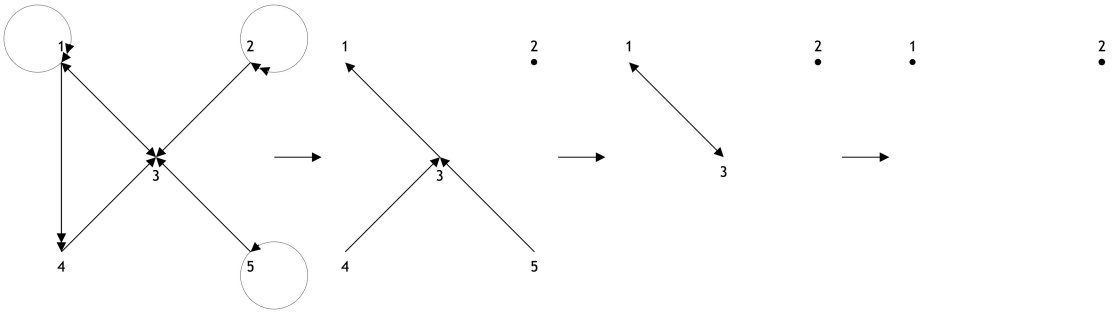
1. Construeer de graaf G^1.
2. Zolang $(G^1)^+ \neq G^1$ doe:
   (a) Laat $G^1 = (G^1)^+$.
   (b) Bepaal de strengsamenhangende componenten in G^1.
   (c) Zolang er een strengsamenhangende component is met meer dan één knooppunt doe:
      i. Vervang elke component C_i door één knooppunt i.
      ii. Voeg pijl (i, j) toe als er een $k \in V_i$ is, zodanig dat er voor alle acties $a \in A(k)$ een $l \in V_j$ is met $p_{kl}(a) > 0$.
      iii. Bepaal de strengsamenhangende componenten.
   (d) Zolang er knooppunten i, j zijn waarvoor (i, j) de enige pijl vanuit i is doe:
      i. Verwijder knooppunt i en pijl (i, j).
      ii. Verwijder alle pijlen (h, i) en (h, j).
      iii. Voeg pijl (h, j) toe als er een $k \in V_h$ is, zodanig dat er voor alle acties $a \in A(k)$ een $l \in V_j$ is met $p_{kl}(a) > 0$.
   (e) Laat $(G^1)^+$ de graaf zijn die verkregen is in stap 2c en 2d.
3. (a) Als het aantal knooppunten één is: Stop. *De MBK is unichain.*
   (b) Als het aantal knooppunten twee is: Stop. *De MBK is multichain.*
   (c) Stop. *Het algoritme kan de MBK niet classificeren.*

De complexiteit van het algoritme is $O(m \cdot n^2)$. Voordat we het algoritme verder toelichten, geven we eerst een aantal voorbeelden.



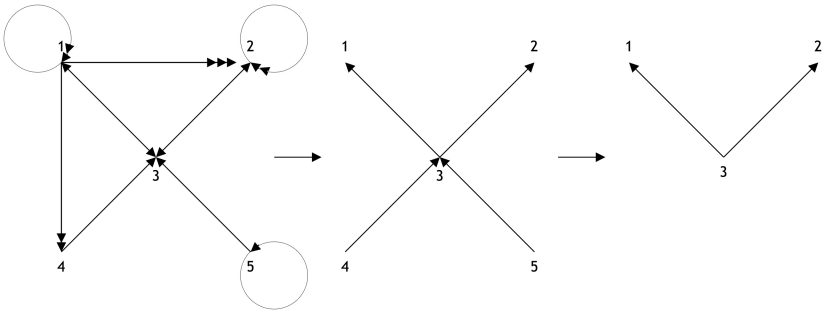
Figuur 6.3 Een Markovbeslissingsketen die behulp van algoritme 6.3 wordt geclassificeerd als unichain.

Voorbeeld 6.4 We passen het algoritme allereerst toe op de Markovbeslissingsketen van figuur 6.3. Na stap 2a ontstaat zo de graaf G^1 (zie tweede afbeelding in het figuur). In G^1 zijn er geen strengsamenhangende componenten met meer dan één knooppunt, daarom gaan we verder met stap 2d. Door knooppunt 4 en 5, die elk maar één uitgaande pijl hebben, samen te voegen met knooppunt 3 ontstaat de graaf van de derde afbeelding. Nu vormen knooppunt 1 en 3 een strengsamenhangende component (stap 2c) en ten slotte kunnen we 1 en 2 samenvoegen (stap 2d), omdat het nieuwe knooppunt 1 slechts één uitgaande pijl heeft. Er is nu één knooppunt over en daarmee kunnen we concluderen dat de Markovbeslissingsketen unichain was.



Figuur 6.4 Een Markovbeslissingsketen die behulp van algoritme 6.3 wordt geclassificeerd als multichain.

Voorbeeld 6.5 Nu passen we het algoritme toe op de Markovbeslissingsketen van figuur 6.4. Na stap 2a ontstaat weer de graaf G^1 (zie tweede afbeelding in het figuur). In G^1 zijn er weer geen strengsamenhangende componenten met meer dan één knooppunt, daarom gaan we verder met stap 2d. Door knooppunt 4 en 5, die elk maar één uitgaande pijl hebben, samen te voegen met knooppunt 3 ontstaat de graaf van de derde afbeelding. Nu vormen knooppunt 1 en 3 weer een strengsamenhangende component (stap 2c), maar nu kunnen we deze knooppunten niet samenvoegen, omdat er geen pijl van 1 naar 2 of terug is. Er zijn twee knooppunten over, dus was de Markovbeslissingsketen multichain.



Figuur 6.5 Een Markovbeslissingsketen die behulp van algoritme 6.3 niet kan worden geclassificeerd.

Voorbeeld 6.6 Als laatste passen we het algoritme toe op de Markovbeslissingsketen van figuur 6.5. Op de tweede afbeelding zien we de graaf G^1 en op de derde afbeelding de graaf die ontstaan is na stap 2d. We kunnen deze graaf niet verder vereenvoudigen, omdat er geen strengsamenhangende componenten met meerdere knooppunten zijn en geen knooppunten zijn met één uitgaande pijl. Er zijn drie knooppunten over, dus kan de Markovbeslissingsketen niet worden geclassificeerd met dit algoritme.

Toelichting op algoritme 6.3

Stap 2b: Toestanden die in dezelfde component van de graaf G^1 zitten, communiceren voor elke strategie met elkaar en zitten dus in dezelfde klasse.

Stap 2c: We vervangen vervolgens de component door één knooppunt. En brengen een pijl (i, j) aan als er een knooppunt k in V_i is zodanig dat er voor elke strategie een knooppunt l in V_j is die bereikbaar is vanuit k.

Vervolgens bepalen we opnieuw de strengsamenhangende componenten. Nu geldt nog steeds dat toestanden die in dezelfde component zitten voor elke strategie met elkaar communiceren. We herhalen deze stap zolang er geen verandering meer optreedt. De graaf die nu is ontstaan noemen we de *gecondenseerde* graaf.

Stap 2d: Als er een knooppunt i is met één uitgaande pijl (i, j), kunnen de knooppunten in V_i en V_j niet tot verschillende recurrente klassen behoren. Daarom voegen we in dit geval i en j samen.

We herhalen stap 2c en 2d tot er in $(G^1)^+$ geen wijzigingen meer zijn.

Stap 3a: We voegen nooit twee knooppunten samen die voor een bepaalde strategie tot een verschillende recurrente klasse behoren, dus als er slechts één knooppunt over is, was er voor elke strategie één recurrente klasse en is de Markovbeslissingsketen unichain.

Stap 3b: Als er twee knooppunten over zijn, is er geen pijl van knooppunt 1 naar 2 en andersom. Dit betekent dat er een strategie f_1 is zodanig dat de knooppunten in V_1 niet bereikbaar zijn vanuit de knooppunten in V_2 en een strategie f_2 waarvoor het omgekeerde geldt. Nu is er een strategie die f_1 en f_2 combineert waarvoor er twee recurrente klassen zijn. Daarom is de Markovbeslissingsketen multichain.

In algoritme 6.3 hebben we met behulp van de gecondenseerde graaf een aantal instanties van het multichainclassificatieprobleem kunnen oplossen.

6.5 Conclusie

In dit hoofdstuk hebben we een aantal polynomiale algoritmes besproken om het multichainclassificatieprobleem te vereenvoudigen of het op te lossen voor een bepaald deelprobleem.

In algoritme 6.2, dat gebaseerd is op algoritme 7 uit [2] maken we gebruik van algoritme 6.1 dat gebaseerd is op algoritme 4 uit [2]. Als we algoritme 7 vergelijken met algoritme 4, merken we op dat de algoritmes van Kallenberg erg op elkaar lijken. Het is dan ook overbodig om beide algoritmes aan te roepen. Het volgende algoritme maakt gebruik van deze observatie.

Algoritme 6.4 (Reductie tot een communicerende Markovbeslissingsketen)

1. Pas algoritme 4 uit [2] toe.
2. Als de Markovbeslissingsketen niet zwak communicerend is: Stop. *De MBK is multichain.*
3. (a) Laat $T = S/C_1$ de verzameling van transiënte toestanden zijn.
   (b) Laat $S' = S/T$.
   (c) Laat $A'(i) = A(i)$ voor alle i.
   (d) Voor alle $i \in S'$ en alle $a \in A'(i)$ doe: Als $\sum_{j \in T} p_{ij}(a) > 0$: Verwijder a uit $A'(i)$.
4. Stop. *De MBK is nu communicerend.*

De verzameling van transiënte toestanden is in dit geval eenvoudiger te bepalen dan in algoritme 7 uit [2], omdat we weten dat de Markovbeslissingsketen zwak communicerend is. Algoritme 6.4 is efficiënter dan algoritme 6.2, maar de complexiteit blijft $O(m \cdot n^2)$.

7 Conclusie

Deze scriptie behandelt de classificatie van Markovbeslissingsketens. We hebben ons daarbij toegespitst op de classificatie in unichain en multichain. Deze eigenschap is interessant, omdat de gemiddelde verwachte kosten bij unichaine Markovbeslissingsketens niet afhangen van de begintoestand. Dit betekent dat voor unichaine Markovbeslissingsketens eenvoudigere algoritmes zijn om de optimale strategie te bepalen.

We hebben allereerst gezien dat het zo goed als zinloos is om te zoeken naar een polynomiaal algoritme voor de classificatie van Markovbeslissingsketens in unichain en multichain. Het algemene geval van het multichainclassificatieprobleem is namelijk NP-volledig. (Zie hoofdstuk 3 voor het bewijs.)

Vervolgens hebben we voor een aantal deelproblemen een polynomiaal algoritme besproken. In hoofdstuk 4 hebben we een algoritme voor de classificatie van Markovbeslissingsketens met een voor elke strategie recurrente of voor een bepaalde strategie absorberende toestand gezien (algoritme 4.5).

Hoofdstuk 5 behandelt deterministische Markovbeslissingsketens (waarbij de overgangskansen altijd nul of één zijn). Uit McCuaig [5] volgt dat dit geval polynomiaal is. Helaas is dit artikel slecht leesbaar en hebben we het algoritme slechts gedeeltelijk kunnen reconstrueren. Algoritme 5.1 classificeert deterministische Markovbeslissingsketens in een aantal gevallen. Het aanvullen van algoritme 5.2, dat de overige gevallen classificeert, zou een onderwerp van nader onderzoek kunnen zijn.

In hoofdstuk 6 hebben we een polynomiaal algoritme gegeven dat tegelijkertijd niet zwak communicerende Markovbeslissingsketens classificeert en andere reduceert tot communicerende Markovbeslissingsketens (algoritme 6.4). In hetzelfde hoofdstuk hebben we ook een algoritme van Kallenberg (algoritme 6.3) besproken dat Markovbeslissingsketens in een aantal gevallen kan classificeren.

We hebben nu dus vier algoritmes voor de classificatie van Markovbeslissingsketens ter beschikking. We zullen laten zien in hoeverre deze algoritmes complementair zijn. Twee algoritmes zijn complementair als er instanties zijn die het eerste algoritme wel classificeert en het tweede niet en als er instanties zijn waarvoor het omgekeerde geldt.

Het is eenvoudig om voorafgaand aan de classificatie in unichain of multichain na te gaan of een Markovbeslissingsketen deterministisch is of niet. Daarom vergelijken we de algoritmes eerst voor deterministische en daarna voor niet deterministische Markovbeslissingsketens.

1. Het heeft geen zin om het algoritme voor Markovbeslissingsketens met recurrente of absorberende toestanden (algoritme 4.5) toe te passen op deterministische Markovbeslissingsketens. Het algoritme dat de gereduceerde graaf construeert heeft in deze gevallen al bepaald of de Markovbeslissingsketen unichain of multichain is.

   In paragraaf 4.3 hebben we laten zien dat een toestand j altijd recurrent is als geldt $V^{-1}(\{j\}) = \emptyset$ of als geldt $V^{-1}(\{j\}) \neq \emptyset$ en $j \notin B^{-1}(V^{-1}(\{j\}))$.

   (a) Als in een deterministische Markovbeslissingsketen geldt $V^{-1}(\{j\}) = \emptyset$, dan is er in de geassocieerde graaf een $i \neq j$ met alleen één uitgaande pijl naar j. Deze pijl wordt door algoritme 5.1 samengetrokken. Vervolgens is er een $k \in S/\{i, j\}$ met alleen pijlen naar i en/of j. Toestand k wordt samengetrokken met i en j. Dit proces zet zich voort tot er één knooppunt over is. Algoritme 5.1 concludeert nu dat de Markovbeslissingsketen unichain is.

   (b) Als in een deterministische Markovbeslissingsketen geldt $V^{-1}(\{j\}) \neq \emptyset$ en $j \notin B^{-1}(V^{-1}(\{j\}))$ is er geen pad van $i \in V^{-1}(\{j\})$ naar j in de geassocieerde graaf. In dit geval zijn er meerdere strengsamenhangende componenten en classificeert algoritme 5.1 de Markovbeslissingsketen als multichain.

Als een deterministische Markovbeslissingsketen een toestand bevat die voor een bepaalde strategie absorberend is, heeft de geassocieerde graaf een lus. Alle Markovbeslissingsketens waarvan de geassocieerde graaf een lus heeft, worden door algoritme 5.1 geclassificeerd, want lussen worden niet verwijderd en uiterlijk in stap 5 worden geassocieerde grafen met een lus geclassificeerd.

2. We zullen nu laten zien dat het graafreductiealgoritme (algoritme 5.1) elke instantie classificeert die door het algoritme van Kallenberg (algoritme 6.3) wordt geclassificeerd. Er zijn in algoritme 6.3 twee manieren om knooppunten samen te voegen.

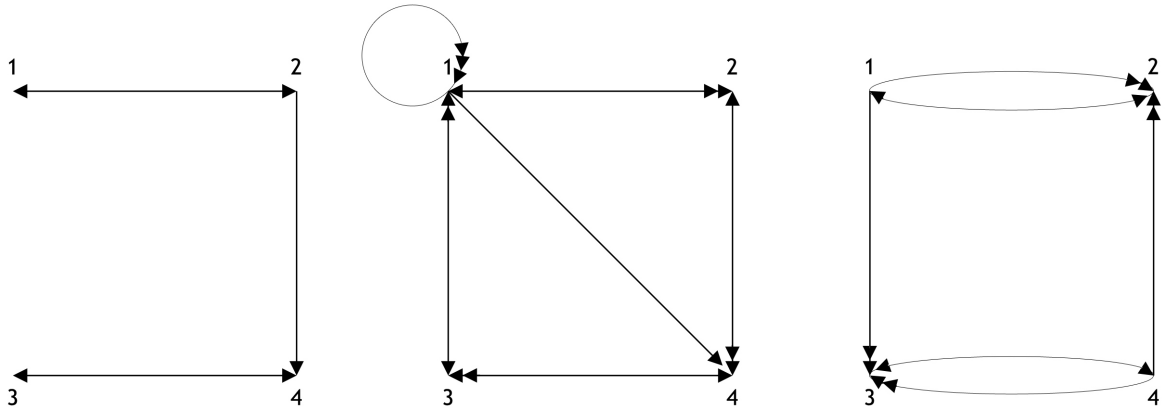
   (a) In stap 2c voegt algoritme 6.3 de knooppunten in een strengsamenhangende component C van een graaf samen. In deze graaf is er een pijl (i, j) als i voor elke actie een positieve overgangskans naar j heeft. In het deterministische geval kunnen we dan aannemen dat er in i maar één actie is. De geassocieerde graaf heeft dan in i maar één uitgaande pijl die door algoritme 5.1 wordt samengetrokken. Ook algoritme 5.1 voegt de knooppunten in C dus samen.

   (b) Vervolgens neemt algoritme 6.3 in stap 2d knooppunten i met één uitgaande pijl (i, j) samen met knooppunt j. Ook in dit geval worden i en j ook door algoritme 5.1 samengevoegd.

   Telkens als algoritme 6.3 twee knooppunten samenvoegt, doet algoritme 5.1 dat ook. Algoritme 6.3 classificeert de Markovbeslissingsketen als er in de graaf na het herhaald uitvoeren van stap 2c en stap 2d één of twee toestanden over blijven. Bij toepassen van algoritme 5.1 zouden er dan ook maximaal twee toestanden over blijven. Deze Markovbeslissingsketens worden dan ook door dit algoritme geclassificeerd, want een gereduceerde graaf heeft minstens drie knooppunten, aangezien in een gereduceerde graaf de in- en uit-graad voor elk knooppunt minstens twee is en er geen lussen en parallelle takken zijn.

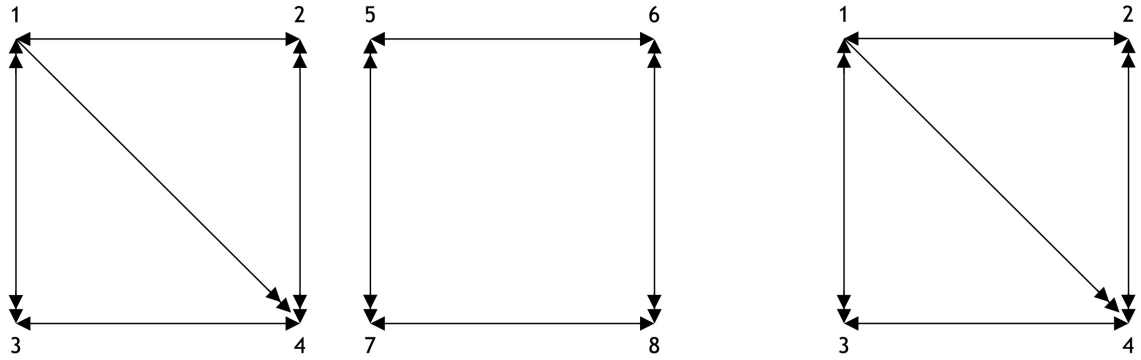
3. Ook het algoritme dat niet zwak communicerende Markovbeslissingsketens classificeert (algoritme 6.4) hoeft niet toegepast te worden op deterministische Markovbeslissingsketens. Als een deterministische Markovbeslissingsketen niet communicerend is, is de geassocieerde graaf niet samenhangend en wordt hij dus altijd door algoritme 5.1 geclassificeerd.

We kunnen concluderen dat het voor deterministische Markovbeslissingsketens voldoende is om alleen algoritme 5.1 te gebruiken.



Figuur 7.1 Drie niet deterministische Markovbeslissingsketens met elke vier toestanden. Een positieve overgangskans is aangegeven met één pijl als zij correspondeert met de eerste actie, met twee pijlen als zij hoort bij de tweede actie en met drie als zij correspondeert met de derde. De overgangskansen zijn één als er voor de actie één uitgaande pijl is en een half als er voor een actie twee uitgaande pijlen zijn.

Nu vergelijken we de algoritmes voor niet deterministische Markovbeslissingsketens.



Figuur 7.2 Twee niet deterministische Markovbeslissingsketens. De eerste heeft acht toestanden, de tweede heeft er vier. Een positieve overgangskans is aangegeven met één pijl als zij correspondeert met de eerste actie en met twee pijlen als zij hoort bij de tweede actie. De overgangskansen zijn één als er voor de actie één uitgaande pijl is en een half als er voor een actie twee uitgaande pijlen zijn.

1. Het graafreductiealgoritme voor deterministische Markovbeslissingsketens (algoritme 5.1) kan niet worden gebruikt om niet deterministische Markovbeslissingsketens te classificeren, want een pijl in de geassocieerde graaf kan bij verschillende acties horen en ook als dit niet het geval is geldt stelling 5.1 niet. De eerste Markovbeslissingsketen in figuur 7.1 laat dit zien. De Markovbeslissingsketen is unichain, maar in de geassocieerde graaf zijn er twee disjuncte rondes.

2. Algoritme 4.5 classificeert de tweede Markovbeslissingsketen in figuur 7.1 wel (want toestand 1 is absorberend), maar algoritme 6.3 en 6.4 doen dat niet (want G^1 heeft geen pijlen en de Markovbeslissingsketen is zwak communicerend).

3. Het algoritme van Kallenberg (algoritme 6.3) classificeert de derde Markovbeslissingsketen in figuur 7.1 wel (want toestand 1 en 2 vormen een strengsamenhangende component in G^1, dus het uiteindelijke aantal knooppunten is maximaal twee), maar algoritme 4.5 en 6.4 doen dat niet (want de Markovbeslissingsketen heeft geen recurrente of absorberende toestanden en is zwak communicerend).

4. Algoritme 6.4 classificeert de eerste Markovbeslissingsketen in figuur 7.2 wel (want de Markovbeslissingsketen is niet zwak communicerend), maar algoritme 6.3 en 4.5 doen dat niet (want G^1 heeft geen pijlen en de Markovbeslissingsketen heeft geen recurrente of absorberende toestanden).

5. De tweede Markovbeslissingsketen in figuur 7.2 wordt door geen van de algoritmes geclassificeerd (want G^1 heeft geen pijlen, de Markovbeslissingsketen heeft geen recurrente of absorberende toestanden, is niet deterministisch en is zwak communicerend).

Voor deterministische Markovbeslissingsketens hoeft alleen algoritme 5.1 gebruikt te worden. De andere algoritmes kunnen alleen instanties classificeren die ook door dit algoritme geclassificeerd worden. Algoritme 5.1 ook een kleinere complexiteit dan de andere algoritmes (zie tabel 7.1). De andere algoritmes zijn wel eenvoudiger te programmeren. Voor niet deterministische Markovbeslissingsketens kan algoritme 5.1 niet gebruikt worden. Van de andere drie algoritmes kunnen we in dat geval geen van de algoritmes weglaten, omdat er altijd meer instanties geclassificeerd kunnen worden met drie dan met twee algoritmes, welke we ook weg zouden laten. De complexiteit van de drie algoritmes is gelijk (zie tabel 7.1). Zoals verwacht zijn er ook Markovbeslissingsketens die door geen van deze algoritmes geclassificeerd kunnen worden, het multichainclassificatieprobleem is immers NP-volledig.

Algoritme	**Complexiteit**	**Werking**
4.5	$0(m \cdot n^2)$	Classificeert Markovbeslissingsketens met recurrente of absorberende toestanden.
5.1	$0(n^2)$	Vereenvoudigt deterministische Markovbeslissingsketens en classificeert een aantal gevallen.
6.3	$0(m \cdot n^2)$	Classificeert Markovbeslissingsketens die na een aantal vereenvoudigingen één of twee knooppunten over hebben.
6.4	$0(m \cdot n^2)$	Classificeert niet zwak communicerende Markovbeslissingsketens en vereenvoudigt zwak communicerende Markovbeslissingsketens tot communicerende.

Tabel 7.1 De vier besproken algoritmes voor de classificatie van Markovbeslissingsketens in unichain en multichain

Bibliografie

Hieronder staat een lijst van de geraadpleegde boeken, artikelen en dictaten. De werken in deze literatuurlijst zijn geordend op alfabatische volgorde van achternaam van de eerste auteur en vervolgens op verschijningsjaar.

1. E.A. Feinberg en F. Yang, 'On Polynomial Cases of the Unichain Classification Problem for Markov Decision Processes', *Proceedings of the 2008 NSF Engineering Research and Innovation Conference*, 7–10 januari 2008, Knoxville, VS.

2. L.C.M. Kallenberg, 'Classification problems in MDPs', in: Z. Hou et al (eds.), *Markov Processes and Controlled Markov Chains*, Kluwer Academic Publishers, 2002, pp. 151 – 165.

3. L.C.M. Kallenberg, *Besliskunde 1*, dictaat tweedejaarsvak wiskunde Universiteit Leiden, najaar 2007, pp. 1 – 47 en pp. 173 – 213.

4. L.C.M. Kallenberg, *Markov decision processes*, dictaat PhD-course Landelijk Netwerk Mathematische Besliskunde, fall 2007, pp. 1 – 24 en pp. 82 – 85.

5. W. McCuaig, 'Intercyclic Digraphs', in: N. Robertson en P. Seymor (eds.), 'Graph Structure Theory', *Contemporary Mathematics* **147** (1993), American Mathematical Society, pp. 203 – 245.

6. J.N. Tsitsiklis, 'NP-hardness of checking the unichain condition in average cost MDPs', *Operation Research Letters* **35** (2007), pp. 319 – 323.