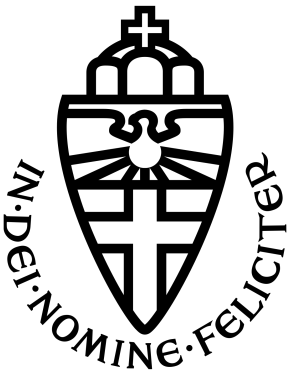
Radboud Universiteit Nijmegen



Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Het verkennen van grafen door een eindige automaat

Bachelorscriptie Wiskunde

Student:
Donja Vos
s1027630

Begeleider:
Wieb Bosma

Tweede lezer:
Henk Don

Februari 2023

Inhoudsopgave

1 Inleiding

1.1 Het doel

In deze scriptie gaat het over het verkennen van een graaf door een eindige automaat, een robot zonder geheugen. Het doel is om één eindige robot zonder geheugen te maken die elke eindige enkelvoudige, ongerichte en samenhangende graaf kan verkennen. Dit houdt in dat de robot alle punten van de graaf kan verkennen en daarna stopt. In artikel [1] wordt bewezen dat er zo een robot bestaat.

1.2 Achtergrond informatie

De grafen die hier verkend worden zijn dus samenhangende, enkelvoudige en ongerichte grafen. Wat inhoudt dat er geen lussen en pijlen in voorkomen, er maximaal maar één kant tussen twee punten is en dat er tussen elk tweetal punten in de graaf een wandeling te maken is. De graaf $G = (V, E)$ die verkend wordt is eindig en voldoet aan de genoemde eisen, met $E =$\{lijnen\} en $V =$\{punten\}. De eindige automaat $\mathcal{R}$, oftewel de robot, die de graaf G verkent, bestaat uit $(Q, \sum, f, q_0, F)$, met

Q een eindige verzameling toestanden
$\sum$ een eindige verzamelingen van symbolen, het alfabet van de automaat
f een functie $Q \times \sum \to Q \times \sum$, de transitie functie
$q_0 \in Q$ de begintoestand
$F \subseteq Q$ de eindtoestanden

Zo een robot kan je zien als een gerichte graaf, waarbij de punten de toestanden zijn en de pijlen die er zijn komen door de transitie functie. De begintoestand q_0 wordt hierin aangegeven door een toestand met een uit het niets inkomende pijl. En F is in ons geval één toestand waar $\mathcal{R}$ in komt als de verkenning klaar is. Bij het bereiken van deze toestand stopt de verkenning ook.

Er worden hier twee verkenningsalgoritmes beschreven. Beide algoritmes hebben als input de graaf G met bijbehorende poortnummers. Elke graaf kan ingebed worden in $\mathbb{R}^2$, niet noodzakelijk als een planaire graaf. Elk punt kan je nu bekijken alsof je op pleintje staat met allemaal uitgangen rondom en deze uitgangen krijgen een poortnummer. Neem een punt u dan wordt elke kant vanuit dit punt opéénvolgend met de klok mee genummerd. De nummering start bij 0 en eindigt bij $d-1$, met d de graad van het punt u. Deze nummering is dus een deel van de input die de robot krijgt. Neem nu een kant, $e = \{u, v\}$ dan leidt de kant met poortnummer i vanuit punt u naar punt v en de kant met poortnummer j vanuit punt v naar punt u. Er geldt hier niet per se dat i gelijk is aan j, het kan natuurlijk toevallig wel zo zijn. Kant e wordt vanuit het punt u aangeven met $i(u)$ en vanuit het punt v met $j(v)$. Deze poortnummers zorgen er voor dat de robot onderscheid kan maken tussen de kanten van het punt. Bij het eerste verkenningsalgoritme krijgt de robot als extra input dat de punten gekleurd zijn. Er wordt hier gebruik gemaakt van drie kleuren, een 2-bits schema. Bij het andere algoritme kleurt de robot de punten tijdens de verkenning. De robot heeft veder geen voorkennis over de totale input die hij krijgt.

Om alle grafen te kunnen verkennen met één alfabet moet in ons geval het alfabet oneindig zijn. Wel is er voor elke graaf slechts een eindige deel alfabet, $\sum_m$, nodig. De transitie functie heeft als informatie nodig: de kleur van het punt, het poortnummer van de kant waardoor de robot het punt binnen gekomen is en de graad van het punt. Deze worden dan ook als combinaties, (k, i, d), weergegeven in het alfabet. Het eindige deel alfabet hangt dus af van de maximale graad in een

graaf. Neem een graaf G met m kanten. In het uiterste geval heeft zo een graaf een punt met graad $d = m$ en is het hoogste poortnummer $d - 1$ gelijk aan $m - 1$. Aan het begin van de verkenning is er een uitzondering voor het poortnummer dat ingevuld wordt. De robot wordt aan het begin van de verkenning op een willekeurig punt u neergezet. In dit geval komt de robot het punt u dus niet binnen via een kant met een poortnummer. In dit geval wordt er voor het poortnummer, $\emptyset$ ingevuld.

Voor het eerste verkenningsalgoritme zijn er drie mogelijkheden voor de kleur, hoogstens $m + 1$ mogelijkheden voor het poortnummer, namelijk $\{\emptyset, 0, 1, .., m - 1\}$, en hoogstens m mogelijkheden voor de graad, namelijk $\{1, 2, .., m\}$. Voor dit algoritme zitten alle mogelijke combinaties van deze drie verzamelingen in het eindige deel alfabet. In totaal zijn dit $3m^2 + 3m$ combinaties. Omdat de robot aan het begin op een willekeurig punt wordt neergezet, kan dat punt elke kleur en elke graad hebben. Daarom moet $\emptyset$ in dit geval in elke mogelijke combinatie zitten.

In het tweede verkenningsalgoritme wordt een ongekleurd punt gezien als een vierde kleur. Er is hier dus een mogelijkheid meer voor de kleur. Een verschil met het eerste algoritme is dat $\emptyset$ niet in elke mogelijke combinatie hoeft te zitten. Het punt waar de robot neergezet wordt, is altijd leeg en wordt daarna met de juiste kleur gekleurd. Voor de kleur van dit punt zijn dus 2 mogelijkheden, alleen de graad kan nog verschillen. De mogelijkheden voor de graad is hier ook hoogstens gelijk aan m. Dit zorgt voor $2m$ mogelijke combinaties met $\emptyset$ erin. Voor de andere combinaties zijn er nog hoogstens m mogelijkheden voor het poortnummer, namelijk $\{0, 1, .., m - 1\}$, hoogstens m mogelijkheden voor de graad, namelijk $\{1, 2, .., m\}$, en 4 mogelijkheden voor de kleur. Ook voor dit algoritme zitten alle mogelijke combinaties van deze drie verzamelingen plus mogelijke combinaties met $\emptyset$ in het eindige deel alfabet. In totaal zijn dit $4m^2 + 2m$ combinaties.

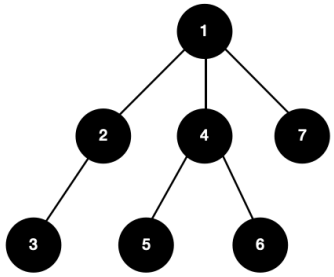
Naast deze combinaties bevat elk alfabet ook een lijst van cijfers van 0 tot en met $m - 1$. Deze cijfers corresponderen met het poortnummer waardoor de robot een punt door moet verlaten. Het oneindige alfabet zijn dus alle mogelijke combinaties uit $k \times \mathbb{N} \times \mathbb{N}$ plus $\mathbb{N}$, met $k = \{3, 4\}$ afhankelijk van het algoritme. Het eindige deel alfabet ziet er dus als volgt uit: $\sum_m = \{\{k \times (m + 1) \times m\}, 0, ..., m - 1\}$.

De transitiefunctie heeft als input de toestand waar de robot zich bevindt en een element van het alfabet. Het element van het alfabet is in dit geval één van de combinaties. Als output heeft deze functie de toestand waar de robot nu naar toe moet en een poortnummer waardoor de robot het punt u door moet verlaten. De functie ziet er dus als volgt uit $f(t, k, i, d) = (t', i')$. Voor i' zijn hier drie mogelijkheden: $i' = i - 1 \bmod d$, $i' = i$ of $i' = i + 1 \bmod d$. Dit kun je zien alsof de robot een stap tegen de klok in maakt, terug gaat langs de kant waarbij hij vandaan kwam of een stap met de klok mee maakt.

1.3 Zoekalgoritmes

Tijdens het verkennen van de graaf en het bewijzen van de stelling wordt gebruik gemaakt van twee zoekalgoritmes. De depth first search (DFS) en de breadth first search (BFS). Het zijn beide algoritmes voor een boom. Om deze zoekalgoritmes inzichtelijker te maken, neem ik aan dat elke boom wordt weergegeven als een planaire boom in $\mathbb{R}^2$ met 1 punt bovenaan, de wortel. Voor elke boom is dit mogelijk. De breadth first search wordt hier alleen niet gebruikt voor een boom, maar voor de graaf die verkend moet worden. Door een kleine aanpassing in het zoekalgoritme is dit mogelijk.

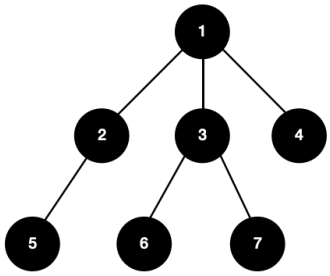
- Bij de depth first search kan op elk punt gestart worden. Neem aan dat de wortel, het startpunt is. Dit punt kan door een willekeurige kant verlaten worden. In het vervolg wordt elk punt nu verlaten door de kant rechts van de kant waardoor je naar binnen kwam. Door dit bij elk punt uit te voeren bezoek je uiteindelijk elk punt. Het idee achter dit zoekalgoritme is dat je eerst zo ver mogelijk naar beneden wilt, voordat je weer een keer omhoog gaat en vanuit daar weer zo ver mogelijk naar beneden wilt. Een voorbeeld van een looproute die je aflegt in een boom met de DFS is te zien in Figuur 1. De route die in dit voorbeeld afgelegd wordt, komt tot stand door het oplopend volgen van de punten in een zo kort mogelijke pad.



Figuur 1: Voorbeeld van een depth first search op een boom met de volgorde waarin de punten doorlopen worden.

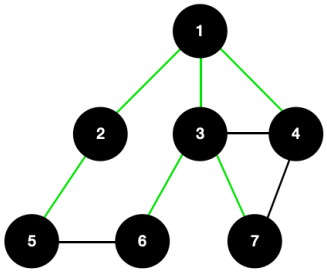
- De breadth first search is in basis ook een zoekalgoritme voor het doorlopen van een boom. Neem ook hier aan dat de wortel, het startpunt is. Eerst definieer ik de verzameling punten $\mathcal{N}_i(r)$ die op afstand i van r liggen. Daarvoor hebben we de definitie van de verzameling $\mathcal{N}(K)$ nodig, met K een verzameling punten. $\mathcal{N}(K)$ zijn alle punten die met een kant verbonden zijn aan een punt $u \in K$. Deze punten zijn dan de buren van de verzameling K. Er geldt dat $\mathcal{N}_i(r) = \mathcal{N}(\mathcal{N}_{i-1}(r)) \backslash \mathcal{N}_{i-2}(r)$ voor $i \geq 0$. Dit houdt in dat er i kanten doorlopen moeten worden vanuit r om bij één van die punten uit te komen. Bij de breadth first search wordt in fase i alle punten uit de verzameling $\mathcal{N}_i(r)$ bezocht. Neem d de diepte van de graaf, dan worden de fases 0 tot en met d achtereenvolgend aan elkaar uitgevoerd, met $\mathcal{N}_0(r) = r$. Elke fase start en eindigt in r. In fase 0 bezoek je r, in fase 1 bezoek je alle buren van r, $\mathcal{N}_1(r)$, in fase 2 bezoek je alle buren van $\mathcal{N}_1(r)$ min de punten uit $\mathcal{N}_0(r)$, $\mathcal{N}_2(r)$, enzovoort. Zie Figuur 2 voor een voorbeeld looproute van de BFS.

  Voor een breadth first search in een graaf die wij hier gaan verkennen is een kleine aanpassing nodig in het algoritme. De graaf rangschik ik op dezelfde manier zoals bij de boom, alleen is die nu niet noodzakelijk planair. Ook hier starten we in het beginpunt r. Het probleem waar je nu alleen tegenaan loopt als je hetzelfde algoritme gebruikt, is dat je een punt meerdere keren bezoekt. Dit kan omdat punten in $\mathcal{N}_i(r)$ nu ook met elkaar verbonden kunnen zijn. Er moet daarom een extra voorwaarde aan de verzameling $\mathcal{N}_i(r)$ komen. Je wilt namelijk dat elk punt maar één keer bezocht wordt. Dus als er voor een punt v geldt dat $v \in \mathcal{N}_n(r)$, met n het kleinste getal waar dit voor geldt, dan moet v niet meer bezocht worden als een punt in $\mathcal{N}_m(r)$ met $m > n$. Er geldt nu dus $\mathcal{N}_n(r) = \mathcal{N}(\mathcal{N}_{n-1}(r)) \backslash (\mathcal{N}_{n-2}(r) \cup \mathcal{N}_{n-1}(r))$. Deze extra voorwaarde zorgt er voor dat de punten in $\mathcal{N}_i(r)$ die met elkaar verbonden zijn niet worden



Figuur 2: Voorbeeld van een breadth first search op een boom met de volgorde waarin de punten doorlopen worden.

bezocht bij fase $i + 1$. Zie Figuur 1 voor een voorbeeld van een BFS in een graaf. De route die in dit voorbeeld afgelegd wordt, komt tot stand door via de groene kanten de punten in een zo kort mogelijk pad in oplopende volgorde te doorlopen.



Figuur 3: Voorbeeld van een breadth first search op een graaf met de volgorde waarin de punten doorlopen worden.

1.4 Resultaten

Door gebruik te maken van drie kleuren hebben we een robot kunnen maken, bestaande uit 41 toestanden, die elke eindige enkelvoudige, ongerichte en samenhangende graaf kan verkennen. Ook herkent deze robot wanneer de verkenning klaar is en stopt dan. Zeg dat deze graaf m kanten heeft. Dan wordt de verkenning in $O(m)$ tijd gedaan. Met andere woorden worden er $O(m)$ kanten doorlopen. De verkenningstijd groeit dus lineair met het aantal kanten. Wordt de kleuring tijdens de verkenning gedaan, dan bestaat er ook een eindige robot die dit kan. Wel is het aantal toestanden in deze robot ongeveer zes keer zo groot. Ook duurt de verkenning iets langer, dit gebeurt namelijk in $O(Dm)$ tijd. Hier groet de verkenningstijd dus lineair met het aantal kanten keer de diameter.

2 Verkenning met een vooraf gelabelde graaf

Hier beschrijf ik een verkenningsalgoritme met een labeling schema dat alleen maar gebruik maakt van 2-bit labels (het geven van een kleur aan een punt). De grafen zijn voorafgaand aan de verkenning al gelabeld, met drie kleuren.

Stelling 2.1. *Er bestaat een robot $\mathcal{R}$ van 41 toestanden die alle punten van een graaf G kan verkennen, beginnend bij een willekeurig punt en die stopt zodra de hele graaf is doorlopen. Hierbij wordt gebruik gemaakt van een gegeven labeling van de punten met 3 kleuren. Bovendien doorloopt de robot daarbij hoogstens $20m$ kanten, waar m het aantal kanten van G is.*

Om Stelling 2.1 te bewijzen heb je eerst het label schema en het verkenningsalgoritme nodig.

2.1 Label schema

Het labeling schema $\mathcal{L}$ maakt gebruik van drie kleuren, wit, zwart en rood. Neem een willekeurig punt r als beginpunt. De afstand tot een ander punt u is het aantal kanten dat doorlopen moet worden in een kortste pad tussen r en u, met $h(u) = \#$kanten. Het punt u kleur je wit als $h(u) \equiv 0 \mod 3$, zwart als $h(u) \equiv 1 \mod 3$ en rood als $h(u) \equiv 2 \mod 3$. Deze kleuring kan in $O(m^2)$ tijd gedaan worden met de BFS.
Alle punten verbonden met een kant aan een punt u zijn de buren van dit punt, $\mathcal{N}(u)$. De punten in $\mathcal{N}(u)$ kunnen verdeeld worden in drie verschillende klassen:

1. De *voorgangers* van u, Voor(u). Elk punt $v \in \mathcal{N}(u)$ waarvoor geldt dat die dichterbij r is dan u, dus $h(v) = h(u) - 1$.

2. De *naasten* van u, Na(u). Elk punt $v \in \mathcal{N}(u)$ waarvoor geldt dat die even ver weg van r is als u, dus $h(v) = h(u)$.

3. De *opvolgers* van u, Op(u). Elk punt $v \in \mathcal{N}(u)$ waarvoor geldt dat die verder weg van r is dan u, dus $h(v) = h(u) + 1$.

Omdat alle punten van G door $\mathcal{L}$ op een unieke manier gekleurd worden, kan deze verdeling gemaakt worden aan de hand van de kleuren van de punten. Neem bijvoorbeeld een punt u dat zwart is. Dan zijn alle punten $v \in \mathcal{N}(u)$ die wit zijn een voorganger van u, de punten $v \in \mathcal{N}(u)$ die zwart zijn een naasten van u en de punten $v \in \mathcal{N}(u)$ die rood zijn een opvolger van u. En op dezelfde manier is het voor een wit en rood punt ook te verdelen. Merk op dat het beginpunt r alleen maar opvolgers heeft. Er geldt dus Voor(r)= $\emptyset$, Na(r)= $\emptyset$ en Op(r)= $\mathcal{N}(r)$.

Aan de hand van deze verdeling van buren definiëren we twee verzamelingen van punten die gebruikt gaan worden bij de verkenning.

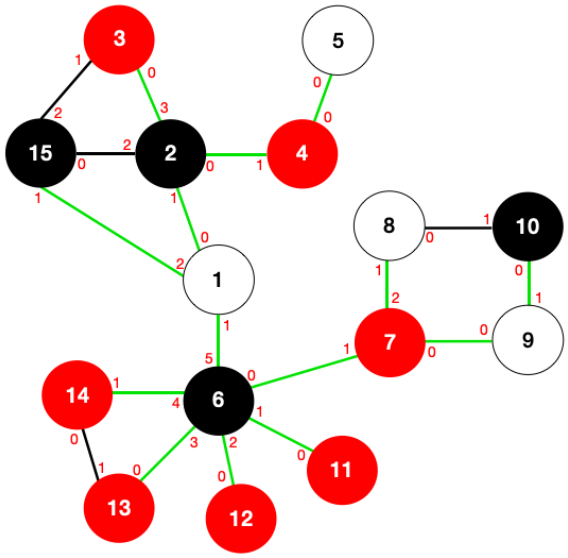
- Ouder(u) is het punt $v \in$Voor(u) waarvoor geldt dat het poortnummer van de kant $\{u,v\}$ vanuit punt u de kleinste is van alle poortnummers van de kanten vanuit u die verbonden zijn met een punt $v' \in$Voor(u).

- Kind(u) zijn de punten $v \in$Op(u) waarvoor geldt u=Ouder(v)

Een punt kan dus meerderen kinderen hebben, maar kan maar maximaal één ouder hebben. Merk op dat het beginpunt r het enige punt is dat alleen maar kinderen heeft en geen ouder. Er geldt dus $\mathcal{N}(r)$=Op(r)=Kind(r) en Ouder(r)=$\emptyset$.

2.2 Verkenningsalgoritme

Het verkenningsalgoritme van $\mathcal{R}$ is eigenlijk een depth first search door gebruik te maken van de kind en ouder functies. In het geval van een graaf die gekleurd is door $\mathcal{L}$ is het punt r het begin punt. Als je alleen gebruik maakt van de kanten die naar ouders of kinderen leiden dan creëer je in elke graaf een boom. Daarom kan je dit algoritme hierbij gebruiken. Je gaat dan via de kinderen "omlaag" en via de ouders "omhoog". In Figuur 4 zie je een voorbeeldgraaf met de vooraf aangebrachte nummering van de poorten per punt, met kleine rode cijfers. Ook is de kleuring van $\mathcal{L}$ al toegepast. Als je bijvoorbeeld naar punt 2 kijkt dan zie je dat punt 3 en 4 kinderen zijn volgens de definitie. Als je dit voor elk punt doet dan komen alle groene kanten in de boom die doorlopen wordt bij de verkenning. Als de punten oplopend via de groene lijnen gevolgd worden dan krijg je de looproute van de depth first search. Merk wel op dat de robot geen geheugen heeft en dus niet kan onthouden welke kanten bij de boom horen. Door het verkenningsalgoritme toe te passen kan die op het moment zelf na gaan of het een kind of ouder is. Maar aan het einde van de verkenning is er dus niet een gecreëerde boom te zien. Dit is alleen om inzicht te geven hoe de robot door een graaf heen loopt.



Figuur 4: De gecreëerde boom door de groene kanten. Met de looproute van de depth first search.

Het herkennen of een kant $i(u)$ vanuit u naar v, bij de boom hoort en dus naar een ouder of een kind leidt, is niet direct af te leiden uit de kleuren van u en v. Om hier achter te komen is het nodig om de buren van u of v te bezoeken. Er zijn drie mogelijke uitkomsten voor het punt v. Het is een kind, een ouder of geen van beide als het zowel geen kind als ouder is. Zeg dat de kant $i(u)$ dan

drie waardes aan kan nemen, {kind, ouder, niks}, afhankelijk naar wat voor punt v de kant leidt. De kanten die waarde "niks" hebben, en dus leiden naar punten die zowel geen ouder als kind zijn, worden "genegeerd" in het depth first search. In het voorbeeld zijn dat de zwarte kanten. Later bij de procedure voor het controleren van een kant $i(u)$ wordt uitgelegd hoe er achter gekomen wordt welke uitkomst bij $i(u)$ hoort.

Depth first search
Ga ervan uit dat $\mathcal{R}$ begint in het beginpunt r, dat er is door de kleuring van $\mathcal{L}$. In dit geval verlaat $\mathcal{R}$ punt r door de kant met poortnummer 0. Merk op dat we weten dat alle buren van r een kind zijn. Door naar "beneden" te gaan komen we nu in een punt u door poort $i(u)$ binnen. Merk ook op dat de kant waardoor je een punt u voor het eerst in komt door naar "beneden" te gaan, om het punt u te verkennen, de kant is die naar de ouder van u leidt. Noem deze kant $p(u)$. Het is namelijk mogelijk dat je tijdens de verkenning van een punt v het punt u al wel een keer binnen gekomen bent. Je wilt eerst zo ver mogelijk naar beneden, dus gaan we kijken of u kinderen heeft. Dit doen we door alle kanten met oplopend poortnummer één voor één te bezoeken, dus de eerst volgende kant is $p(u)+1$. Is er een kind gevonden dan ga je via die kant naar "beneden" en anders ga je weer via de ouder, dus $i(u) = p(u)$, "omhoog". Kom je een punt u binnen via poort $j(u)$, doordat je weer "omhoog" gaat dan ga je kijken of er nog kinderen zijn bij de kanten $j(u)+1$, $j(u)+2$,...., $p(u)-1$. Ben je bij $p(u)$ aangekomen dan ga je hier weer "omhoog". Alle kanten voor $j(u)$, dus $j(u)-1$, $j(u)-2$,...,$p(u)+1$ zijn bij een eerdere verkenning van dit punt al doorlopen. Dit geldt omdat de verkenning van het punt u start bij poort $p(u)$ en vervolgens voor oplopend poortnummer alle kanten langs gaat tot en met het poortnummer $j(u)$ dat leidt naar een kind. Veder in de verkenning kom je vervolgens weer via poort $j(u)$ het punt u binnen en dus zijn de kanten hiervoor al verkend. Als je dan vervolgens weer bij $p(u)$ aankomt, heb je alle kanten van het punt u verkend en kant je het punt "omhoog" verlaten door kant $p(u)$. Hoe de robot de kanten herkend staat iets verderop bij "De procedure voor de controle van kant $i(u)$ uitgelegd.

Als $\mathcal{R}$ niet bij het beginpunt start, dan kan je via de voorgangers "omhoog" lopen om r te vinden. Aan de kleuren van de punten is namelijk te zien of een punt een voorganger is. Er geldt dat een wit punt een voorganger is van een zwart punt, een zwart punt van rood punt en een rood punt van een wit punt. En als je in een wit punt bent met alleen maar zwarte buren dan ben je in r. Ook kan $\mathcal{R}$ herkennen wanneer de verkenning klaar is. De depth first search is klaar als je via de kant met de hoogste poortnummer, $d-1$, het beginpunt r weer inkomt. Als $\mathcal{R}$ een wit punt binnen komt via poort $d-1$ dan moet de robot controleren of die in het begin punt zit. Zo ja, dan ben je klaar met de verkenning en anders gaat de verkenning gewoon door zoals hier boven uitgelegd.

De procedure voor de controle van kant $i(u)$
De procedure om na te gaan welke waarde bij kant $i(u)$ hoort, beschrijft de acties die de robot moet uitvoeren om er achter te komen wat het punt v van u is. De kant die gecontroleerd wordt start bij punt u met poortnummer $i(u)$ en eindigt bij het punt v met poortnummer $j(v)$. $\mathcal{R}$ is nu in het punt v en door de kleuren van de punten kan die een onderscheid maken. De drie mogelijkheden zijn:

1. $v \in$Na(u). De punten hebben in dit geval dezelfde kleur, waardoor deze kant direct de waarde niks krijgt. $\mathcal{R}$ gaat in dit geval via poort $j(v)$ weer terug naar punt u.

2. $v \in$Voor(u). Het punt v is dan dichterbij r dan u. Het zou in dit geval kunnen dat v de ouder van u is. Hiervoor moeten de buren van u met een lager poortnummer dan $i(u)$ gecontroleerd worden. $\mathcal{R}$ gaat nu eerst terug naar het punt u. Via poortnummer $j(v)$ verlaat die punt v en komt die via $i(u)$ punt u binnen. Nu gaat $\mathcal{R}$ één voor één de buren $i(u)-1$, $i(u)-2$, ... aflopen tot en met een buur $v' \in$Voor(u) of tot met poort 0 en voor alle buren moet dan gelden $\tilde{v} \neq$Voor(u). In het eerste geval is er een buur met een lager poortnummer gevonden die in Voor(u) zit. Dit houdt in dat v geen ouder is en dat $\mathcal{R}$ de waarde "niks" aan de kant $i(u)$ kan toe schrijven. In het andere geval is er niet zo een buur en is punt v de ouder. Laat $k(u)$ de laatst bezochte kant zijn door $\mathcal{R}$, in het eerste geval is dat het poortnummer die leidt naar het punt v' en in het andere geval is het poort 0. Nu loopt $\mathcal{R}$ de kanten $k(u)+1$, $k(u)+2$, ... één voor één op tot en met de kant die naar een punt leidt in Voor(u). Dit is dan ook gelijk aan het poortnummer $i(u)$ die leidt naar punt v en waar $\mathcal{R}$ nu de waarde van weet. Als v de ouder is dan blijft $\mathcal{R}$ in dit punt anders keert die terug naar punt u.

3. $v \in$Op(u). Het punt v is nu verder weg van r dan u. Er moet nu gecontroleerd worden of v een kind is van u. Hiervoor moeten de buren van v met een lager poortnummer gecontroleerd worden. $\mathcal{R}$ blijft nu in het punt v en gaat de kanten $j(v)-1$, $j(v)-2$, ... controleren tot en met een buur $v' \in$Voor(v) of tot en met poort 0 en voor alle buren moet dan gelden $\tilde{v} \neq$Voor(v). In het eerste geval is er een buur v' met een lager poortnummer die in Voor(v) zit, wat inhoudt dat v geen kind is en dat $\mathcal{R}$ de waarde "niks" aan de kant $i(u)$ kan toe schrijven. In het andere geval is er niet zo een buur en is v een kind. Laat ook nu $k(v)$ de laatst bezochte kant zijn door $\mathcal{R}$. Nu worden de kanten $k(v)+1$, $k(v)+2$,, ... één voor één opgelopen tot en met de kant die leidt naar een punt die in Voor(v) zit. Nu is $\mathcal{R}$ via poort $i(u)$ het punt u binnen gekomen. Dit is de kant waar $\mathcal{R}$ nu een waarde voor weet. Als v een kind is dan keert $\mathcal{R}$ terug naar punt v en anders blijft die in punt u.

Merk op dat het oplopen vanaf poort $k(u)$ of $k(v)$ gebeurt, omdat de robot het poortnummer $i(u)$ en $j(v)$ niet kan opslaan. Door het oplopen vanaf poort k is de eerste kant die leidt naar een voorganger gelijk aan de kant $i(u)$ en $j(v)$, wat dezelfde kant is.

2.3 Bewijs

Label schema

Een breadth first search is een goede manier om de punten in een samenhangende graaf G met m kanten te kleuren op de manier zoals beschreven bij $\mathcal{L}$. Het is duidelijk dat op deze manier van kleuren alle punten gekleurd worden in eindige tijd. De meeste kanten worden doorlopen in een graaf waar alle punten in een rechte lijn liggen, zie Figuur 5. Elke keer na het bezoeken van een punt die nog niet gekleurd is, gaat de robot weer terug naar r. Er worden dus in totaal $\sum_{i=0}^{m-1} 2(m-i) = m(m+1)$ kanten door lopen. Dit is af te schatten met $2m^2$. Dit betekent dat de kleuring in $O(m^2)$ tijd gedaan wordt.

Figuur 5

2.3.1 Bewijs van stelling 2.1

Depth first search
Als de procedure voor het controleren van een kant $i(u)$ goed gedefinieerd is dan is het snel in te zien dat $\mathcal{R}$ een depth first search uitvoert. Hierbij wordt gebruik gemaakt van de kanten $\{u, v\}$, waarvoor moet gelden dat ofwel $u = \text{Ouder}(v)$ of $u \in \text{Kind}(v)$. In een samenhangende graaf geldt dat elk punt één ouder heeft, behalve het beginpunt, dat heeft geen ouder. Als er geldt dat $u = \text{Ouder}(v)$ dan geldt er ook dat $v \in \text{Kind}(u)$. Elk punt is dus een kind van één ander punt, behalve het beginpunt. Doordat het algoritme in elk punt op zoek, gaat naar een kind wordt elk punt bereikt met de verkenning, vanwege de samenhang. Ook heeft elk punt per definitie maximaal maar één ouder. Dit zorgt er voor dat gebruikmakend van de kanten $\{u, v\}$, zoals hierboven genoemd, er een boom gevormd wordt. Er kan namelijk geen cykel ontstaan, omdat er dan moet gelden dat een punt meerdere ouders heeft en/of naasten moeten met elkaar verbonden zijn. Alleen beide gevallen kunnen niet optreden als de genoemde kanten $\{u, v\}$ gebruikt worden. Dit maakt het samen met het verkenningsalgoritme duidelijk in te zien dat het een depth first search is.

Procedure voor kant $j(u)$
Voor de correctheid van de procedure voor het controleren van een kant $i(u)$ moeten er zoals genoemd drie mogelijkheden worden nagegaan. De kleuren van de punten u en v zorgen er voor dat $\mathcal{R}$ onderscheid kan maken tussen de drie mogelijkheden.

1. Als $v \in \text{Na}(u)$ dan is v zowel geen kind als een ouder van u. De waarde "niks" die aan de kant met poortnummer $i(u)$ wordt gegeven klopt. Nu gaat $\mathcal{R}$ via poort $j(v)$ weer terug naar u zoals beschreven in het algoritme.

2. Als $v \in \text{Voor}(u)$ dan geldt er $v = \text{Ouder}(u) \Leftrightarrow$ voor alle buren $v'_{i(u)-k} = \{v' \in \mathcal{N}(u) |$ via poort $i(u) - k$ kom je bij $v'\}$ met $k \in \{1, 2, ..., i\}$ geldt $v'_{i(u)-k} \notin \text{Voor}(u)$. De robot controleert deze eigenschap in geval 2. Dit gebeurt door terug te gaan naar punt u en alle buren $v'_{i(u)-k}$ te controleren, zolang er geldt dat $v'_{i(u)-k} \notin \text{Voor}(u)$. Aan het einde van deze controle blijft $\mathcal{R}$ in punt v als het de ouder is en anders gaat die terug naar punt u en komt die via poort $i(u)$ binnen. En dus gaat deze procedure correct.

3. Als laatste het geval dat $v \in \text{Op}(u)$ dan geldt er $v = \text{Kind}(u) \Leftrightarrow$ voor alle buren $w'_{j(v)-k} = \{w' \in \mathcal{N}(v) |$ via poort $j(v) - k$ kom je bij $w'\}$ met $k \in \{1, 2, ..., j\}$ geldt dat $w'_{j(v)-k} \notin \text{Voor}(v)$. De robot controleert deze eigenschap in geval 3, door alle buren $w'_{j(v)-k}$ te controleren zolang er geldt dat $w_{j(v)-k} \notin \text{Voor}(v)$. Aan het einde van deze controle keert $\mathcal{R}$ terug in punt u door poort $i(u)$. Hij blijft in punt u als v geen kind is en anders keert die terug in punt v. Dus ook deze procedure gaat correct.

In alle drie de gevallen gaat de procedure voor het controleren van een kant $i(u)$ correct. Er moet alleen nog bewezen worden dat de robot bij deze verkenning hoogstens $20m$ kanten doorloopt.

Maximaal aantal kanten dat doorlopen wordt
Tijdens een depth first search worden alle kanten twee keer doorlopen, een keer "omlaag" en een keer "omhoog". Tijdens de verkenning van een graaf worden de kanten die een ouder zijn dus in ieder geval twee keer doorlopen door de depth first search. Verder wordt een kant doorlopen als er gecontroleerd moet worden welke waarde bij de kant hoort of tijdens een controle van een andere

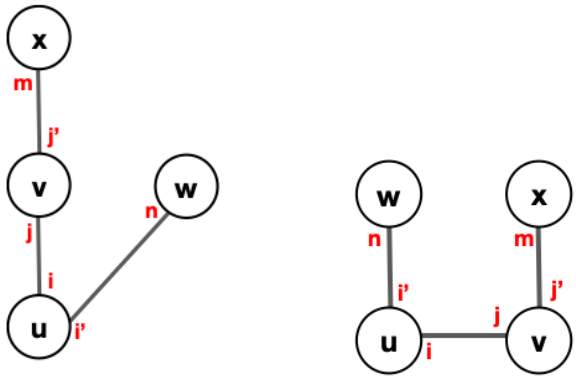
kant. Merk op dat bij de controle van de kant $i(u)$ met als eindpunt het punt v die via poort $j(v)$ binnen komt, alleen de kanten verbonden aan deze punten met een lager poortnummer worden doorlopen. Ook komen alle kanten aan de buurt om gecontroleerd te worden. Tijdens de controle van kant $i(u)$ zijn er zoals eerder beschreven drie mogelijkheden, vanwege de kleuren van de punten.

1. Als $v \in$Na(u) dan keert $\mathcal{R}$ direct terug naar u en doorloopt verder geen andere kanten. De kant $i(u)$ wordt hier twee keer doorlopen.

2. Als $v \in$Voor(u) dan gaat $\mathcal{R}$ terug naar u en loopt terug tot en met poort $k(u)$, waarvoor geldt dat $0 \leq k(u) < i(u)$ en het hoogste poortnummer is dat leidt naar een punt $v' \in$ Voor(u), als zo een punt v' bestaat. Als er niet zo een buur v' bestaat dan geldt er $k(u) = 0$. Daarna loopt $\mathcal{R}$ weer terug naar poort $i(u)$. Bij deze controle worden de kanten $i(u) - 1$, $i(u) - 2$, ..., $k(u) + 1$ vier keer doorlopen en kant $k(u)$ wordt twee keer doorlopen. Bij kant $i(u)$ zijn er twee mogelijkheden. Als punt v een ouder is dan wordt de kant $i(u)$ drie keer doorlopen, waarvan de laatste bij de DFS hoort, en anders wordt kant $i(u)$ vier keer doorlopen.

3. Als $v \in$Op(u) dan blijft $\mathcal{R}$ in het punt v en loopt tot en met poort $k'(v)$, waarvoor moet gelden dat $0 \leq k'(v) < j(v)$ en het hoogste poortnummer is dat leidt naar een punt $w \in$ Voor(v), als zo een punt w bestaat. Als er niet zo een buur w bestaat dan geldt er $k'(v) = 0$. Bij deze controle worden de kanten $j(v) - 1$, $j(v) - 2$, ..., $k'(v) + 1$ vier keer doorlopen en de kanten $k'(v)$ en $i(u)$ worden twee keer doorlopen. Merk op dat kant $i(u)$ gelijk is aan de kant $j(v)$.

Om te kijken hoe vaak een kant maximaal doorlopen kan worden, is er onderscheid nodig in de verschillende soorten kanten. Er zijn drie mogelijkheden voor een kant $e = \{u, v\}$. Geval 1) het leidt naar een ouder van u, geval 2) het leidt niet naar een ouder van u, maar er geldt wel dat $v \in$Voor(u) of geval 3) het is een verbinding met een naasten. Elke kant dat naar een ouder leidt, leidt ook naar een kind in de tegengestelde richting. Omdat het om dezelfde kant gaat, hoeft daar geen extra onderscheid voor gemaakt te worden.



(a) Geval 1 en 2 (b) Geval 3

Figuur 6: Twee verschillende situaties voor het tellen hoe vaak de kant e doorlopen wordt

Geval 1) $v =$ Ouder(u). De kant e wordt buiten alle verkenningen van de kanten al 2 keer doorlopen vanwege depth first search.

- Voor de controle van poort i zit je in geval 2 en wordt de kant e 2 keer doorlopen.
- Voor de controle van poort j zit je in geval 3 en wordt de kant e 2 keer doorlopen.
- Het geval dat er een kant $e' = \{u, w\}$ bestaat waarvoor geldt dat $i' > i$ en $w \in$Voor(u). Bij de controle van de poort n zit je in geval 3 en wordt de kant e 2 keer doorlopen. En bij de controle van de poort i' zit je in geval 2 en wordt de kant e 2 keer doorlopen.
- Het geval dat er een kant $\tilde{e} = \{v, x\}$ bestaat waarvoor geldt dat $j' > j$ en $x \in$Voor(v). Bij de controle van poort m zit je in geval 3 en wordt de kant e 4 keer doorlopen. En bij de controle van poort j' zit je in geval 2 en wordt de kant e 4 keer doorlopen.

Dit allemaal bij elkaar opgeteld wordt de kant e maximaal 18 keer doorlopen bij de verkenning van de hele graaf.

Geval 2) $v \in$Voor(u) maar $v \neq$ Ouder(u). De kant e wordt buiten de verkenning van de kanten nu niet doorlopen bij de depth first search.

- Voor de controle van poort i zit je in geval 2 en wordt de kant e 4 keer doorlopen.
- Voor de controle van poort j zit je in geval 3 en wordt de kant e 2 keer doorlopen.
- Het geval dat er een kant $e' = \{u, w\}$ bestaat waarvoor geldt dat $i' > i$ en $w \in$Voor(u). Bij de controle van de poort n zit je in geval 3 en wordt de kant e 2 keer doorlopen. En bij de controle van de poort i' zit je in geval 2 en wordt de kant e 2 keer doorlopen.
- Het geval dat er een kant $\tilde{e} = \{v, x\}$ bestaat waarvoor geldt dat $j' > j$ en $x \in$Voor(v). Bij de controle van poort m zit je in geval 3 en wordt de kant e 4 keer doorlopen. En bij de controle van poort j' zit je in geval 2 en wordt de kant e 4 keer doorlopen.

Dit allemaal bij elkaar opgeteld wordt de kant e maximaal 18 keer doorlopen bij de verkenning van de hele graaf.

Geval 3) $v \in$Na(u). Ook in dit geval wordt de kant e buiten de verkenning van de kanten niet doorlopen bij de depth first search.

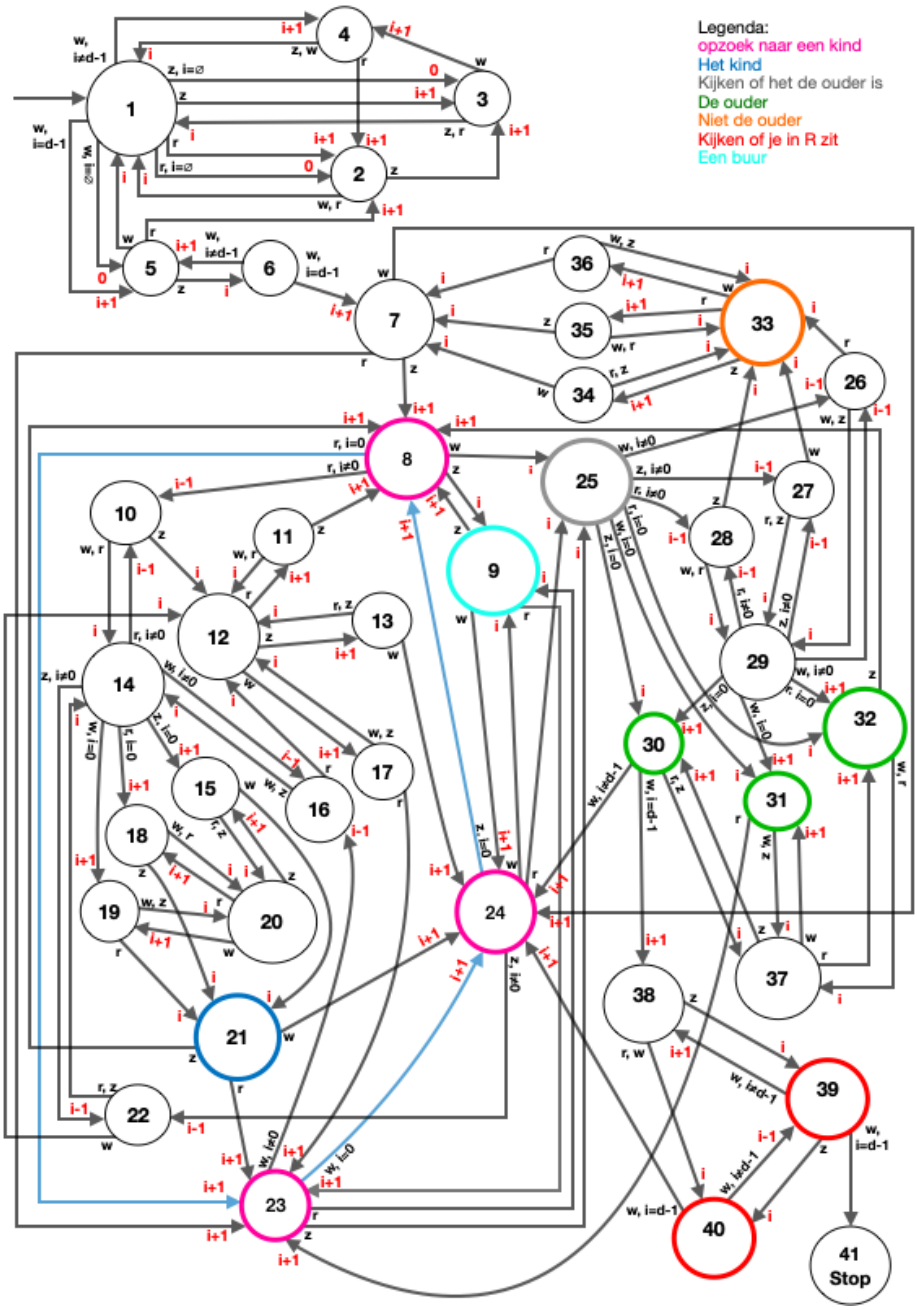
- Voor de controle van poort i zit je in geval 1 en wordt de kant e 2 keer doorlopen.
- Voor de controle van poort j zit je in geval 1 en wordt de kant e 2 keer doorlopen.
- Het geval dat er een kant $e' = \{u, w\}$ bestaat waarvoor geldt dat $i' > i$ en $w \in$Voor(u). Bij de controle van de poort n zit je in geval 3 en wordt de kant e 4 keer doorlopen. En bij de controle van de poort i' zit je in geval 2 en wordt de kant e 4 keer doorlopen.
- Het geval dat er een kant $\tilde{e} = \{v, x\}$ bestaat waarvoor geldt dat $j' > j$ en $x \in$Voor(v). Bij de controle van poort m zit je in geval 3 en wordt de kant e 4 keer doorlopen. En bij de controle van poort j' zit je in geval 2 en wordt de kant e 4 keer doorlopen.

Dit allemaal bij elkaar opgeteld wordt de kant e maximaal 20 keer doorlopen bij de verkenning van de hele graaf.

En dus wordt er tijdens de verkenning, van een graaf G met m kanten, met het gebruikte algoritme maximaal $20m$ kanten doorlopen.

□

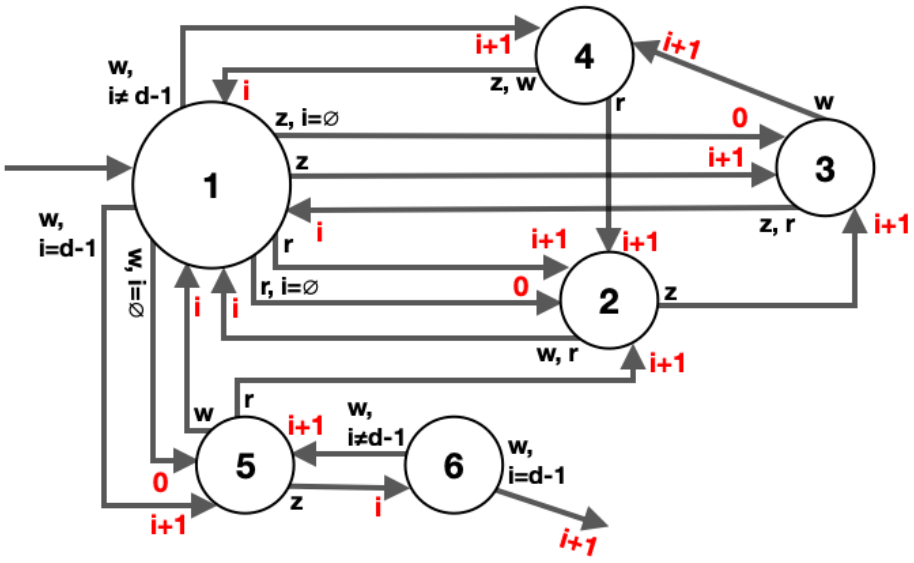
2.4 De Robot



Figuur 7: De robot

In de Robot staat aan het begin van elke pijl in het zwart aangegeven wat de input moet zijn voor de transitiefunctie. Dit is de kleur van het punt waar de robot in aankomt en het poortnummer waardoor die dit punt binnen komt, als dat van extra belang is. Bij toestand 1 zijn er drie pijlen waar $i = \emptyset$ staat. Om te beginnen wordt de robot op een punt neer gezet. De robot komt dit punt niet binnen via een poort, daarom wordt dit aan het begin aangegeven met $i = \emptyset$. Doordat de robot bij poort nul neer gezet wordt, kan die via deze poort het punt verlaten. De pijl geeft vervolgens aan naar welke toestand de robot gaat. En aan het einde van elke pijl staat in het rood door welk poortnummer de robot het punt, waar die zich bevindt, door moet verlaten.

2.4.1 Het beginpunt vinden



Figuur 8: De eerste 6 toestanden

In de eerste zes toestanden (zie Figuur 8) kan de robot naar het beginpunt toelopen, als die zich daar nog niet bevindt, en controleren of een wit punt het beginpunt is. Het idee achter dit stukje van de robot is dat die via de voorgangers van de punten "omhoog" naar r loopt. $\mathcal{R}$ bekijkt de buren van een punt u door de poortnummers oplopend langs te gaan, tot die bij een Voor(u) aankomt, als die bestaat. Dit is te herkennen aan de kleuren van de punten. Ter herinnering: er geldt Voor(wit)=rood, Voor(zwart)=wit en Voor(rood)=zwart. Tussen toestand 1 en 2 gaat de robot op zoek naar een voorganger van rood, tussen toestand 1 en 3 naar een voorganger van zwart en tussen toestand 1 en 4 naar een voorganger van wit. Vervolgens blijft $\mathcal{R}$ in de voorganger van u, als die gevonden is, en gaat die in dit punt weer op dezelfde manier op zoek naar een voorganger. In een wit punt is het mogelijk dat er geen voorgangers meer zijn, wat inhoudt dat $\mathcal{R}$ zich in het beginpunt bevindt. Dit is te herkennen door beginnend bij poort nul alle buren langs te gaan via oplopend poortnummer. Als deze allemaal zwart zijn dan is $\mathcal{R}$ in het beginpunt en kan de verkenning beginnen. Is die nog niet in r dan komt die bij deze verkenning een rood of wit punt

tegen en gaat die op dezelfde manier op zoek naar een voorganger. Dit wordt tussen toestand 5 en 6 gecontroleerd.

$\mathcal{R}$ bevindt zich in een zwart of rood punt

Het beginpunt is wit van kleur. Dus als $\mathcal{R}$ zich in een zwart of rood punt bevindt, maakt niet uit op welk moment van de verkenning, dan is het direct duidelijk dat die niet in het beginpunt zit. Stel $\mathcal{R}$ wordt op een zwart punt neergezet, dan wil je via een wit punt, een voorganger, "omhoog" richting r. Nu gaat $\mathcal{R}$ beginnend in toestand 1 met input u =zwart en $i = \emptyset$ naar toestand 3 en verlaat die het punt u door poort nul. Hierdoor komt die punt v binnen via poort $i(v)$. Als het punt v zwart of rood is, wat geen voorganger is, wil je weer terug naar punt u om de kanten verder via oplopend poortnummer af te gaan naar een voorganger. Dit gebeurt doordat $\mathcal{R}$ met de input v ={zwart, rood} weer naar toestand 1 gaat en v verlaat door poort $i(v)$, het poortnummer waardoor die binnen kwam. Het punt waar $\mathcal{R}$ nu in terecht komt, is weer het zwarte punt waar die begon. Met input v =zwart gaat $\mathcal{R}$ weer terug naar toestand 3 en verlaat die het punt u door poort $i(u) + 1$. Je bent nu bij een volgende buur v' van u aan het kijken. Is het punt v' weer zwart of rood dan gaat het op dezelfde manier als hier boven tot en met $\mathcal{R}$ een buur $\tilde{v}$ tegen komt die wit is. Is de buur v' wit dan gaat $\mathcal{R}$ met de input v =wit naar toestand 4 en verlaat die het witte punt door poort $i(v) + 1$. $\mathcal{R}$ heeft een voorganger $\tilde{v}$ gevonden van u. Hij blijft nu in het punt $\tilde{v}$ en gaat hier op zoek naar een voorganger. Dit gebeurt door bij het witte punt $\tilde{v}$ ook weer alle buren via oplopend poortnummer te bezoeken. In het geval dat $\mathcal{R}$ op een rood punt start gaat de procedure analoog, alleen gaat $\mathcal{R}$ naar andere toestanden en is de voorganger een zwart punt. De procedure verloopt ook op dezelfde manier wanneer $\mathcal{R}$ in een rood of zwart punt aankomt tijdens de zoektocht naar r.

Starten in een wit punt

Bij het starten in een wit punt is er een mogelijkheid dat dit al het beginpunt is. Dit wordt direct gecontroleerd door te kijken of alle buren zwart zijn. Nu verlaat $\mathcal{R}$ punt u door poort nul en gaat die naar toestand 5. Toestanden 5 en 6 worden gebruikt om te controleren of een wit punt het beginpunt is. Toestand 5 komt $\mathcal{R}$ ook alleen binnen als die de buren van een wit punt controleert, startend bij poort nul.

Als de buur v zwart is dan wil je terug naar het witte punt en controleren of de andere buren ook zwart zijn. $\mathcal{R}$ gaat daarom vanuit toestand 5 naar toestand 6 als de buur v van u zwart is. $\mathcal{R}$ is de buur v door poort $i(v)$ binnen gekomen en verlaat het punt v ook weer door die poort, omdat $\mathcal{R}$ naar toestand 6 is gegaan. Dan komt $\mathcal{R}$ weer in het witte punt uit waar die begon. Zolang er voor de input geldt dat $i \neq d(u) - 1$ gaat $\mathcal{R}$ weer terug naar toestand 5 en verlaat die u door poort $i(u) + 1$, om de buren die nog niet bezocht zijn te controleren. Merk op dat wanneer $\mathcal{R}$ in toestand 6 is en voor de input geldt ook dat $i = d(u) - 1$ dan heeft die alle buren van u gecontroleerd en zijn ze allemaal zwart. Dit houdt in dat $\mathcal{R}$ in het beginpunt zit en dat de verkenning kan beginnen. Daarom gaat die dan vanuit toestand 6 met input v =wit en $i = d - 1$ naar toestand 7 en verlaat die punt u door poort $i(u) + 1$. Omdat $\mathcal{R}$ door poort $d(u) - 1$ het punt u binnen kwam is poort $i(u) + 1$ gelijk aan poort nul. Dit is precies wat je wil bij de start van de verkenning.

Komt $\mathcal{R}$ tijdens het controleren van de buren van het witte punt in toestand 5 via poort $i(u)$ een witte of en rode buur v' tegen, dan weet je direct dat je niet in r zit en wil je toestand 5 verlaten. In het geval dat v' rood is heeft $\mathcal{R}$ een voorganger gevonden. In dit punt wilt die dan blijven en hier weer veder naar een voorganger zoeken. $\mathcal{R}$ gaat in dit geval van toestand 5 naar toestand 2 en verlaat het rode punt v' door poort $i(v') + 1$. Als v' wit is dan ben je bij een naasten. Het enige

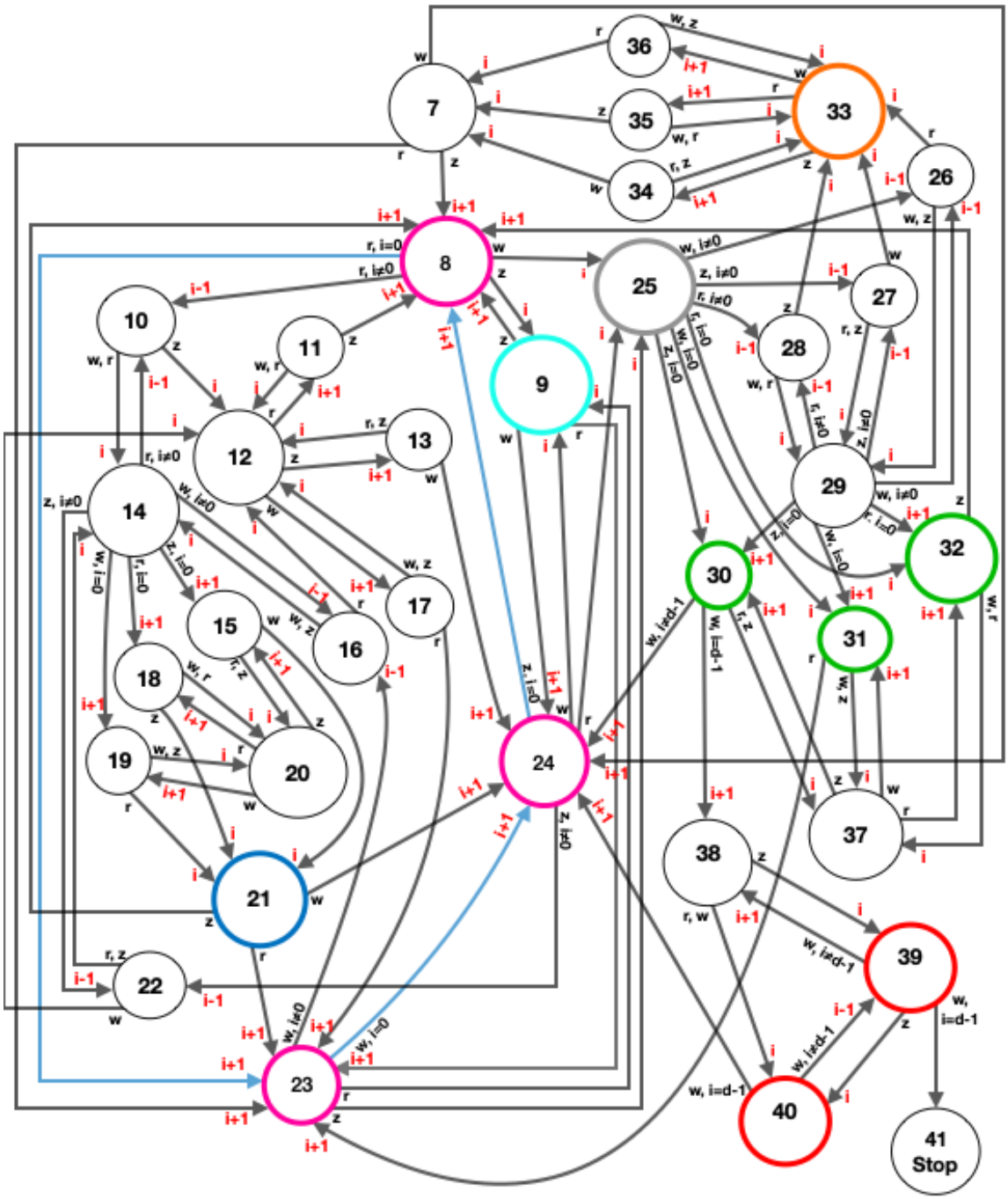
wat je dan weet is dat u niet r is. Je wilt in dit geval dus weer terug naar het witte punt u en op zoek gaan naar een voorganger. Dit doet $\mathcal{R}$ door toestand 5 te verlaten en naar toestand 1 te gaan. Hij verlaat punt v' door poort $i(v')$, waar die door binnen kwam. Hierdoor komt die weer in het witte punt u terecht waardoor $\mathcal{R}$ vervolgens naar toestand 4 gaat. Merk op dat in dit geval de graad nooit gelijk aan $d(u) - 1$ is. Als $\mathcal{R}$ namelijk voor het eerst een niet zwarte buur tegen komt, in dit geval een wit punt, dan weet je dat er een voorganger is. De voorganger van een wit punt is rood. Er is in dit geval dus zeker nog een kant met een hoger poortnummer dan $i(u)$ dat naar een rood punt moet leiden. Dus is $\mathcal{R}$ het witte punt zeker door een lager poortnummer dan $d(u) - 1$ binnen gekomen. En dit verklaar waarom je altijd naar toestand 4 gaat en niet naar toestand 5 kan gaan. In dit geval verlaat $\mathcal{R}$ het witte punt door poort $i(u) + 1$. Nu gaat $\mathcal{R}$ weer alle buren oplopend langs om een voorganger te vinden.

$\mathcal{R}$ bevindt zich in een wit punt
Bij het zoeken naar de voorgangers van een wit punt is het poortnummer van belang voor de transitie functie. Als je tijdens de zoektocht naar het beginpunt een wit punt tegen komt, wil je kunnen controleren of dit punt het beginpunt is. Merk op als je een wit punt u binnen komt, is dat niet noodzakelijk via poort 0. Als $\mathcal{R}$ de buren van u controleert en tot en met poort $d(u) - 1$ geen voorganger is tegen gekomen dan is het mogelijk dat $\mathcal{R}$ zich in het beginpunt bevindt. Er is daarom een pijl in toestand 1 naar toestand 5 met de input $u =$wit en $i = d(u) - 1$. Deze pijl zorgt ervoor dat $\mathcal{R}$ in een wit punt, beginnend bij poort nul, alle buren door via oplopend poortnummer controleert of ze zwart zijn. Is dit het geval dan bevindt die zich in r en kan de verkenning beginnen. Is dat niet het geval, dan ga je verder met het zoeken naar een voorganger, zoals hier boven beschreven. Het is mogelijk dat je tot en met poort $d(u) - 1$ geen rode buur tegen komt, maar wel een witte. Dit houdt in dat je dus niet in r zit. Toch gaat $\mathcal{R}$ dan naar toestand 5 om te controleren of dat wel het geval is. Dit maakt niet uit, want ook dan komt $\mathcal{R}$ hier gewoon achter, doordat die ofwel als eerst een rood punt tegenkomt en zijn voorganger gevonden heeft of een wit punt en dan gaat die zoals boven uitgelegd via toestand 1 naar toestand 4.

2.4.2 De verkenning

Wanneer $\mathcal{R}$ met de eerste 6 toestanden het beginpunt gevonden heeft, komt die toestand 7 binnen en verlaat die het beginpunt via poort 0. Hier begint de verkenning van de samenhangende graaf G met m kanten zoals eerder in het algoritme beschreven. Het idee van de overige 35 toestanden kan opgedeeld worden in 3 delen. Het controleren of een punt een kind is, een ouder is of dat een wit punt het beginpunt is.
Er zijn drie "hoofdpunten": toestand 8, 23 en 24. Vanuit toestand 8 ga je op zoek naar een kind van een zwart punt, vanuit toestand 23 van een rood punt en vanuit toestand 24 van een wit punt. Vanuit alle drie de toestanden gaat het proces analoog.
Neem nu aan dat we in het geval zitten dat we in toestand 8 zijn aangekomen. $\mathcal{R}$ komt dan uit toestand 7 en komt een zwart punt u via poort $i(u)$ binnen. Er hoeft nu niet gecontroleerd te worden of het punt u een kind is. $\mathcal{R}$ komt namelijk vanuit het beginpunt en er geldt $\forall v' \in \mathcal{N}(r)$ $v' = \text{Kind}(r)$. Omdat het punt zwart is, gaat die naar toestand 8 en verlaat het punt u door poort $i(u) + 1$ om op zoek te gaan naar een kind. Als $\mathcal{R}$ bij het verlaten van poort $i(u)$ een zwart punt v binnenkomt dan is het een buur en wil je terug. Dit gebeurt doordat $\mathcal{R}$ naar toestand 9 gaat en via

dezelfde poort weer terug moet. Nu kom je weer via poort $i(u)$ binnen in het zwarte punt u en ga je weer terug naar toestand 8. Hierbij wordt het punt verlaten door poort $i(u) + 1$, om verder te zoeken naar een kind. Je kan een rood punt tegen komen. In dit geval wil je controleren of dit een kind is, deel 1 van de robot. Als een kind gevonden is dan ga je via dit punt naar "beneden" en ga je hier weer op zoek naar een kind en anders ga je verder met de verkenning in het zwarte punt. Kom je een wit punt tegen dan wil je controleren of het een ouder is, deel 2 van de robot. Als het een ouder is dan wil je via dit punt weer "omhoog". Is het niet een ouder dan wil je terug naar het zwarte punt gaan en verder op zoek gaan of er een kind is. Als laatste wil je kunnen herkennen of je weer in het beginpunt bent, deel 3 van de robot. Als dat zo is moet de verkenning stoppen en anders wil je gewoon verder gaan met de verkenning en op zoek gaan naar een kind. Dit gebeurt in toestand 38 tot en met 41.



Figuur 9: Deel 1, 2 en 3 van de robot.

Deel 1 van de robot: op zoek naar een kind
Voor het zoeken naar een kind, voor alle kleuren, worden de toestanden 10 tot en met 22 gebruikt (zie Figuur 9). Een deel van deze toestanden wordt maar gebruikt in het geval dat we vanuit toestand 8 naar een kind gaan zoeken. In dit geval komen we van een zwart punt v, die via de kant met poortnummer $i(v)$ het punt verlaat, en gaan we controleren of het rode punt u een kind is, die door de kant met poortnummer $i(u)$ is binnen gekomen. Er zijn drie verschillende mogelijkheden hier. 1) het is direct duidelijk dat het een kind is, 2) het blijkt een kind te zijn na alle buren van u, met de poortnummers $i(u) - 1$ tot en met 0, te hebben gecontroleerd of 3) het is geen kind.

1. $\mathcal{R}$ komt het rode punt u binnen via poort nul. Het is nu direct duidelijk dat het rode punt u het kind is van het punt v. Er kunnen namelijk geen zwarte buren van u zijn met een lager poortnummer dan $i(u) = 0$. In dit geval gaat $\mathcal{R}$ naar toestand 23 en verlaat die het rode punt door poort $i(u) + 1$. Hier gaat die weer op zoek naar een kind vanuit een rood punt.

2. $\mathcal{R}$ komt het rode punt u binnen door poortnummer $i(u) \neq 0$. In dit geval geldt er, het rode punt u is een kind $\Leftrightarrow \forall k \in \{1, 2, ..., i\}$ geldt dat $v'_{i(u)-k} = \{v' \in \mathcal{N}(u) |$ via poort $i(u) - k$ kom je bij $v'\}$ niet zwart is. Tussen toestand 10 en 14 loopt $\mathcal{R}$ de kanten met poortnummers $i(u) - k$ van het rode punt af, voor oplopende k. $\mathcal{R}$ gaat namelijk van toestand 10 naar 14 als de buur van u rood of wit is, in dit geval waren $\forall k$ de buren $v'_{i(u)-k}$ ongelijk aan zwart, dus rood of wit. Aangekomen in een buur $v'_{i(u)-k}$ gaat $\mathcal{R}$ via dezelfde kant weer terug naar het rode punt u. Zolang er geldt dat het poortnummer $i(u) - k \neq 0$ gaat $\mathcal{R}$ weer terug naar toestand 10 om vervolgens weer naar de volgende buur $v'_{i(u)-k-1}$ te gaan. Heeft $\mathcal{R}$ alle buren gecontroleerd dan komt die in toestand 14 via de kant met poortnummer 0 het rode punt u binnen. Nu gaat $\mathcal{R}$ niet terug naar toestand 10, maar gaat die naar toestand 18 en verlaat het rode punt door poort $1 = i(u) + 1$. Nu loopt $\mathcal{R}$ tussen toestand 18 en 20 de kanten af met oplopend poortnummer. Dit gaat op dezelfde manier zoals dat eerder tussen toestand 10 en 14 gebeurde, alleen oplopend in plaats van aflopend. $\mathcal{R}$ loopt hier door tot en met de kant die leidt naar een zwart punt, dit is het punt v. Op dit moment is $\mathcal{R}$ in toestand 18 en gaat die naar toestand 21. Hierbij verlaat $\mathcal{R}$ het zwarte punt weer door de kant met poortnummer $i(v)$, waardoor die binnen kwam. Nu komt $\mathcal{R}$ weer terecht in het rode punt u, waarvan nu bekent is dat het een kind is van punt v. Je wilt nu in dit punt u weer zoeken naar een kind. Dit gebeurt doordat $\mathcal{R}$ naar toestand 23 gaat en het rode punt u door de kant met poortnummer $i(u) + 1$ verlaat.

3. Ook hier komt $\mathcal{R}$ het rode punt u binnen door poortnummer $i(u) \neq 0$. Alleen is het rode punt u niet het kind van het zwarte punt v. Er geldt nu dat $\exists k \in \{1, 2, ..., i\}$ zodat $v'_{i(u)-k} = \{v' \in \mathcal{N}(u) |$ via poort $i(u) - k$ kom je bij $v'\}$ zwart is. $\mathcal{R}$ begint hier op dezelfde manier als in geval 2. Alleen is er nu een k waarvoor geldt dat de buur $v'_{i(u)-k}$ zwart is. In het geval dat $\mathcal{R}$ bij deze buur aankomt, is die in toestand 10 en gaat die naar toestand 12. Hierbij verlaat die het zwarte punt $v'_{i(u)-k}$ door deze zelfde kant, waardoor die binnengekomen was. Dit zorgt er voor dat $\mathcal{R}$ weer in het rode punt u is. Ook nu gaat $\mathcal{R}$ weer de kanten af met oplopend poortnummer tot en met een buur die zwart is, wat direct weer het zwarte punt v is. In dit geval gebeurt dit tussen toestand 12 en 11. Wanneer $\mathcal{R}$ in het zwarte punt v zit, bevindt die zich in toestand 11 en gaat die naar toestand 8. Nu komt $\mathcal{R}$ weer via dezelfde kant, met poortnummer $i(v)$, het zwarte punt v binnen waardoor die aan het begin het punt door verliet. Nu wil je in dit punt gewoon weer verder gaan zoeken of er een kind is. Dit gebeurt doordat

het zwarte punt door de kant met poortnummer $i(v)+1$ verlaten wordt bij binnenkomst van toestand 8.

Deel 2 van de robot: controleren of het een ouder is
Voor het vinden van een ouder voor alle kleuren worden de toestanden 25 tot en met 37 gebruikt. Ook hier wordt maar een deel van deze toestanden gebruikt voor het geval dat we vanuit toestand 8 naar een kind gaan zoeken. In dit geval komen we van een zwart punt v, die via de kant met poortnummer $i(v)$ het punt verlaat, en gaan we controleren of het witte punt u een ouder is, die door de kant met poortnummer $i(u)$ is binnen gekomen. Dit gebeurt door de buren van v te controleren. Wanneer $\mathcal{R}$ een wit punt binnenkomt, gaat die naar toestand 25 en verlaat die het punt u door de poort $i(u)$. Hierdoor komt die weer terug in het zwarte punt v, door poort $i(v)$. Ook hier zijn dan weer drie mogelijkheden: 1) het is direct duidelijk dat het een ouder is, 2) het blijkt een ouder te zijn na alle buren van v, met de poortnummers $i(v)-1$ tot en met poort 0, te hebben gecontroleerd of 3) het is geen ouder.

1. $\mathcal{R}$ komt het zwarte punt v weer binnen via poort 0. Het is nu direct duidelijk dat het witte punt u de ouder is van het punt v. Er kunnen namelijk geen witte buren van v zijn met een lager poortnummer dan $i(v)=0$. In dit geval gaat $\mathcal{R}$ naar toestand 30 en verlaat die het zwarte punt door poort $i(v)$. Hierdoor gaat die "omhoog" naar het witte punt u. In principe wil je dan weer verder zoeken naar een kind in dit punt. Alleen wordt er hier in toestand 30 onderscheid gemaakt in het poortnummer waardoor $\mathcal{R}$ het witte punt binnenkomt. Is het poortnummer ongelijk aan $d(u)-1$, $i(u)\neq d(u)-1$, dan is er niks aan de hand, ook als dit misschien het beginpunt is, en ga je hier weer op zoek gaan naar een kind. Dit gebeurt doordat $\mathcal{R}$ dan naar toestand 24 gaat en het witte punt door poort $i(u)+1$ verlaat. Is het poortnummer wel gelijk aan $d(u)-1$, $i(u)=d(u)-1$, dan moet er gecontroleerd worden of het witte punt het beginpunt is. Als dat het geval is moet de verkenning namelijk stoppen. Dit gebeurt doordat $\mathcal{R}$ dan naar toestand 38 gaat. Dit wordt verderop in 2.4.2 uitgelegd bij deel 3 van de robot.

2. $\mathcal{R}$ komt het zwarte punt v weer binnen via poort $i(v)\neq 0$. In dit geval geldt er, het witte punt u is een ouder $\Leftrightarrow \forall k \in \{1, 2, ..., \text{i}\}$ geldt dat $u'_{i(v)-k} = \{u' \in \mathcal{N}(v) |$ via poort $i(v)-k$ kom je bij $u'\}$ niet wit is. Nu verlaat $\mathcal{R}$ toestand 25 en gaat naar toestand 27, waarbij die het zwarte punt v door poort $i(v)-1$ verlaat. Tussen toestand 27 en 29 loopt $\mathcal{R}$ de kanten met poortnummers $i(u)-k$ van het zwarte punt af, voor oplopende k. Dit gaat op dezelfde manier als in geval 2 bij deel 1 van de robot. Wanneer $\mathcal{R}$ alle buren gecontroleerd heeft, is die in toestand 29 en komt die het zwarte punt v via poort nul binnen. Nu gaat $\mathcal{R}$ niet terug naar toestand 27, maar gaat die naar toestand 30 en verlaat het zwarte punt door poort $1 = i(v)+1$. Nu loopt $\mathcal{R}$ tussen toestand 30 en 37 de kanten af met oplopend poortnummer. Ook dit gaat weer op dezelfde manier zoals in geval 2 bij deel 1 van de robot. $\mathcal{R}$ loopt hier door tot en met de kant die leidt naar een wit punt, dit is het punt u. Op dit moment is $\mathcal{R}$ in toestand 30 en gaat die naar toestand 24 of 38, afhankelijk van het poortnummer waardoor $\mathcal{R}$ het witte punt binnen komt.

3. Ook hier komt $\mathcal{R}$ het zwarte punt v weer binnen via poort $i(v)\neq 0$. Alleen is het witte punt u in dit geval niet de ouder van het zwarte punt v. Er geldt nu dat $\exists k \in \{1, 2, ..., i\}$ zodat $u'_{i(v)-k} = \{u' \in \mathcal{N}(v) |$ wit is. $\mathcal{R}$ begint hier op dezelfde manier als in geval 2 hierboven. Alleen is er nu een k waarvoor geldt dat de buur $u'_{i(v)-k}$ wit is. In het geval dat $\mathcal{R}$ bij deze buur

aankomt, is die in toestand 27 en gaat die naar toestand 33. Hierbij verlaat die het witte punt $u'_{i(v)-k}$ door deze zelfde kant, waardoor die binnengekomen was. Dit zorgt er voor dat $\mathcal{R}$ weer in het zwarte punt v is. Ook nu gaat $\mathcal{R}$ weer de kanten af met oplopend poortnummer tot en met een buur die wit is, wat direct weer het witte punt u is. In dit geval gebeurt dit tussen toestand 33 en 34. Wanneer $\mathcal{R}$ in het witte punt u zit, bevindt die zich in toestand 34 en gaat die naar toestand 7. $\mathcal{R}$ verlaat het witte punt door dezelfde poort en komt weer in het zwarte punt v terecht, via poort $i(v)$. Dit is de poort waar de controle voor het witte punt begon. Nu is $\mathcal{R}$ dus weer terug bij deze kant en weet die dat het niet leidt naar een ouder. Je wilt hier vervolgens weer verder gaan zoeken naar een kind. Dit gebeurt doordat $\mathcal{R}$ na de aankomst in het zwarte punt, naar toestand 8 vertrekt en het zwarte punt verlaat door poort $i(v) + 1$.

Merk op dat geval 1 en 2 iets anders verloopt als je vanuit een rood of wit punt start. In het genomen voorbeeld kan een witte ouder het beginpunt zijn. Dit is van belang om te weten als $\mathcal{R}$ een wit punt door poort d(witte ouder)-1 binnen komt. In dit geval wordt er dan gecontroleerd of het witte punt het beginpunt is. Als dat het geval is moet de verkenning namelijk stoppen. Zoals uitgelegd in geval 1 en 2 wordt er in toestand 30, door twee pijlen, onderscheid gemaakt tussen de poortnummer waardoor $\mathcal{R}$ het witte punt is binnen gekomen. Dit onderscheid is alleen niet nodig in het geval dat je vanuit een rood of wit punt start. De ouders van deze punten zijn namelijk niet wit en kunnen dus niet het beginpunt zijn. Als de ouder hier gevonden is gaat het altijd, ongeacht door welk poortnummer $\mathcal{R}$ de ouder binnen komt, analoog aan het geval dat er geldt dat $i \neq d-1$.

Deel 3 van de robot: controleren of het punt het beginpunt is
Als de robot het beginpunt via poort $d(r)-1$ "omhoog" binnenkomt, is de verkenning klaar. Daarom moet in elk geval dat $\mathcal{R}$ een wit punt u via poort $d(u) - 1$ binnen komt, gecontroleerd worden of het het beginpunt is. Dit gebeurt in de toestanden 38 tot en met 41. Toestand 38 komt $\mathcal{R}$ binnen vanuit toestand 30 als die een wit punt binnen komt via poort $d(u) - 1$. Vervolgens verlaat die het witte punt door poort $i(u) + 1$, wat gelijk is aan poort $d(u) - 1 + 1 = d(u) = 0$. Nu wil je dus controleren of alle buren zwart zijn. Deze controle start bij poort 0 van het witte punt, zodat het duidelijk is wanneer alle buren gecontroleerd zijn. Er zijn hier twee mogelijkheden, 1) $\mathcal{R}$ zit in het beginpunt en dus moet de verkenning stoppen of 2) $\mathcal{R}$ zit niet in het beginpunt en de verkenning gaat gewoon door.

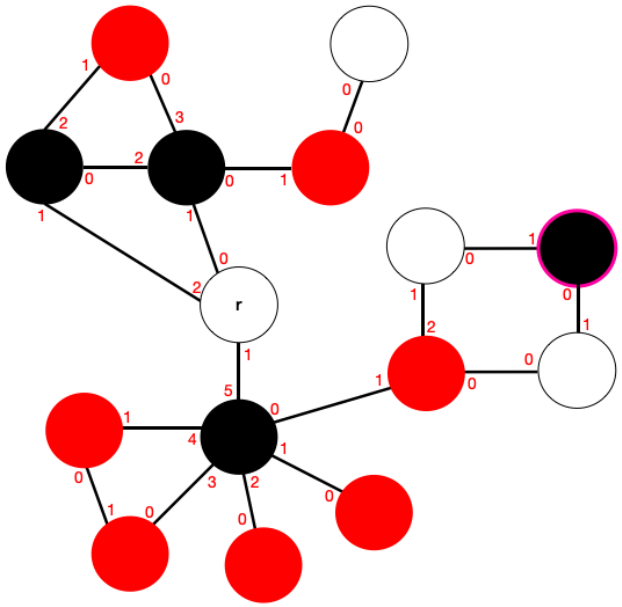
1. Er geldt dat een wit punt u het beginpunt is $\Leftrightarrow \forall k \in \{1, 2, ..., \text{i}\}$ geldt dat $v'_{i(u)+k} = \{v' \in \mathcal{N}(u) |$ via poort $i(u) + k$ kom je bij $v'\}$ zwart is. Tussen toestand 38 en 39 loopt $\mathcal{R}$ de kanten met poortnummers $i(u) + k$ van het witte punt op, voor oplopende k. Dit gaat op dezelfde manier zoals in de gevallen hierboven beschreven. Bij de pijl van 39 naar 38 is er voorwaarde voor de poort waardoor je het witte punt naar binnen komt. Is deze namelijk gelijk aan $d(u) - 1$ dan heb je alle buren gecontroleerd en wil je niet terug gaan naar toestand 38. Als $\mathcal{R}$ door poort $d(u) - 1$ het witte punt binnen komt en alle buren zijn zwart dan is het duidelijk dat die in r zin. In dit geval gaat $\mathcal{R}$ van toestand 39 naar 41. Het binnen komen van toestand 41 zorgt er voor dat de verkenning stopt. Dit is precies wat je wil in deze situatie.

2. Er geldt dat een wit punt u niet het beginpunt is $\Leftrightarrow \exists k \in \{1, 2, ..., \text{i}\}$ zodat $v'_{i(u)+k} = \{v' \in \mathcal{N}(u) |$ via poort $i(u) + k$ kom je bij $v'\}$ niet zwart is. Ook nu loopt $\mathcal{R}$ tussen toestand 38 en 39 de kanten met poortnummers $i(u) + k$ van het witte punt op, voor oplopende k. Alleen komt

$\mathcal{R}$ een buur $v'_{i(u)+k}$ tegen die rood of wit is. Als dit gebeurt, is die in toestand 38 en gaat die naar toestand 40. Hierbij verlaat die de buur $v'_{i(u)+k}$ door dezelfde kant als waardoor die naar binnen gekomen is. Hierdoor komt $\mathcal{R}$ weer in het witte punt u terecht. Nu wil je weer naar poort $d(u)-1$, zodat de verkenning verder kan gaan waar die gebleven was. Dit gebeurt tussen toestand 40 en 39. Hierbij worden de kanten met poortnummers $i(u)+k$ van het witte punt afgelopen, voor aflopende k. Zodra $\mathcal{R}$ het witte punt door poort $d(u)-1$ binnen komt, is die in toestand 40 en gaat die naar toestand 24. Hierbij verlaat die het witte punt u door poort $i(u)+1$ en gaat die weer gewoon op zoek naar een kind.

2.5 Voorbeeldgraaf

Om de robot te verduidelijken is het goed om te zien hoe die door de voorbeeldgraaf (zie Figuur 10) heen loopt. Neem aan dat de robot op het zwarte punt met de roze rand bij poort 0 wordt neergezet. Gebruikmakend van de eerste zes toestanden zal $\mathcal{R}$ naar het beginpunt lopen, dit is het witte punt met de hoofdletter R er in. Vanuit hier wordt de verkenning gestart.



Figuur 10: Voorbeeld graaf

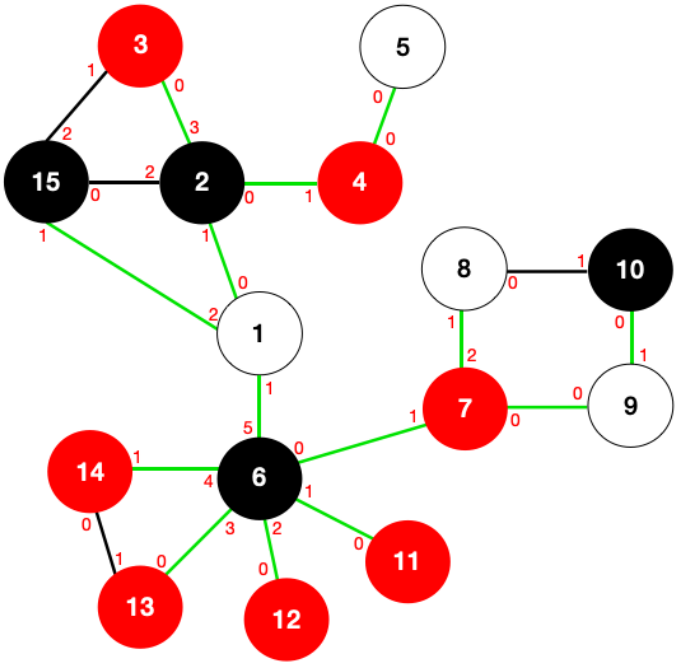
Het beginpunt van de voorbeeldgraaf vinden
$\mathcal{R}$ begint op een zwart punt en gaat daarom startend van toestand 1 naar toestand 3. Bij het verlaten van poort nul komt die een witte punt binnen. Dit is een opvolger en daar wil je in blijven. Dat doet $\mathcal{R}$ ook door vervolgens naar toestand 4 te gaan. Hierbij moet die het witte punt door $i(wit) + 1$ verlaten. Het witte punt is binnen gekomen via poort 1, de graad van het punt is 2 en dus is poort $i(wit) + 1$ gelijk aan poort 0. De robot komt daardoor in een rood punt uit wat direct weer een voorganger is. Vanuit toestand 4 gaat $\mathcal{R}$ dan naar toestand 2 en verlaat het rode punt door $i(rood) + 1$. Wat in dit geval naar een zwart punt leidt. $\mathcal{R}$ gaat dan naar toestand 3 en verlaat het zwarte punt door poort 1. Nu komt die in een rood punt terecht. Dit is geen voorganger en dus wil je terug om verder te zoeken. Dat gebeurt doordat $\mathcal{R}$ nu naar toestand 1 gaat en via poort $i(rood)$, nul, weer terug moet. Hierdoor komt $\mathcal{R}$ weer in het zwarte punt terecht en gaat vervolgens weer naar toestand 3. Nu verlaat die het zwarte punt door poort 2. Ook nu komt die weer in een rood punt terecht. Het heen-en-weer lopen tussen toestand 3 en 1 wordt nog drie keer herhaald. Als $\mathcal{R}$ door het laatste rode punt het zwarte punt binnen komt, gaat die van toestand 1 naar 3 en verlaat die het zwarte punt door poort 5. Nu komt die in een wit punt terecht wat een voorganger is, waardoor die naar toestand 4 gaat. Het witte punt moet door poort 2 verlaten worden. Hierdoor komt $\mathcal{R}$ via poort 1 uit bij een zwart punt. $\mathcal{R}$ gaat nu naar toestand 1 en verlaat het zwarte punt

weer door poort 1. Nu komt $\mathcal{R}$ het witte punt weer in door poort 2. In dit punt geldt $d(wit) = 3$ en dus gaat $\mathcal{R}$ in dit geval naar toestand 5, want $i(wit) = 2 = d(wit) - 1$. Hij verlaat het witte punt nu door poort nul en komt weer in een zwart punt terecht. $\mathcal{R}$ gaat nu naar toestand 6 en keert terug in het witte punt door poort nul. Omdat de graad gelijk is aan drie keert die weer terug in toestand 5 en verlaat het witte punt door poort 1. Dit wordt nog één keer helemaal herhaald, omdat de volgende buur weer zwart is. Bij de derde zwarte buur keert $\mathcal{R}$ niet terug naar toestand 5, maar gaat die naar toestand 7. Bij het binnen komen van toestand 7 verlaat $\mathcal{R}$ het witte punt door poort $i(wit) + 1 = 2 + 1 = 0 \bmod 3$. $\mathcal{R}$ was in het beginpunt en heeft die nu via poort nul verlaten voor de start van de verkenning.

De verkenning van de voorbeeldgraaf

Bij de start van de verkenning zit $\mathcal{R}$ in toestand 7 en heeft het beginpunt r door poort 0 verlaten. Nu komt die door poort 1 een zwart punt binnen. Omdat $\mathcal{R}$ in dit geval vanuit r komt, is het direct duidelijk dat dit een kind is en hoeft dat niet gecontroleerd te worden. In het zwarte punt kan direct verder op zoek gegaan worden naar kinderen. Bij het aankomen in het zwarte punt gaat $\mathcal{R}$ naar toestand 8 en verlaat het punt door poort 2. Hij komt nu bij een buur van het zwarte punt uit die een naaste is. Dit zorgt er voor dat die naar toestand 9 gaat en het zwarte punt, de naaste, via poort 0 verlaat. Weer aangekomen in het zwarte punt gaat die terug naar toestand 8 en verlaat die het punt door poort 3. $\mathcal{R}$ komt nu een rood punt binnen via poort 0 en dus is het direct duidelijk dat dit een kind is. Daarom gaat $\mathcal{R}$ naar toestand 23 en verlaat het rode punt door poort 1 om hier verder te zoeken naar kinderen. Hier komt $\mathcal{R}$ een zwart punt tegen en dan moet er gecontroleerd worden of dit de ouder is. $\mathcal{R}$ gaat naar toestand 25 en verlaat het zwarte punt weer door poort 2. Door poort 1 komt $\mathcal{R}$ het rode punt weer binnen en gaat dan naar toestand 28. Hierbij verlaat $\mathcal{R}$ het rode punt door poort 0 en komt die in een zwart punt terecht. Nu is duidelijk dat het zwarte punt, waarvan gecontroleerd werd of het de ouder was, geen ouder is. $\mathcal{R}$ gaat daarom naar toestand 33 en verlaat het zwarte punt door poort drie om weer in het rode punt uit te komen. Hier wil je weer oplopen tot het punt dat gecontroleerd werd. Dit gebeurt via toestand 35, $\mathcal{R}$ verlaat hierbij het rode punt door poort 1 en komt weer in het zwarte punt terecht. Omdat $\mathcal{R}$ direct in een zwarte punt terecht komt, gaat die naar toestand 7 en verlaat het zwarte punt door poort 2 het rode punt weer in. Opnieuw gaat $\mathcal{R}$ naar toestand 23 om te controleren of er een kind is van dit rode punt. Alleen bij het verlaten van het rode punt door poort nul komt die uit in een zwart punt en moet er weer gecontroleerd worden of het de ouder is. $\mathcal{R}$ gaat nu naar toestand 25 en verlaat het zwarte punt hierbij door poort 3 en komt daardoor het rode punt via poort nul weer binnen. Nu is direct duidelijk dat het zwarte punt de ouder is en gaat $\mathcal{R}$ naar toestand 32. Hierbij verlaat die het rode punt door poort 0 om "omhoog" te gaan naar de ouder. Bij het binnen komen van het zwarte punt gaat $\mathcal{R}$ naar toestand 8 en verlaat die het punt door poort 0. nu komt die een rood punt en moet er gecontroleerd worden of dit een kind is. Dit gebeurt doordat $\mathcal{R}$ naar toestand 10 gaat en het rode punt door poort $i(rood) - 1 = 0$ verlaat. $\mathcal{R}$ komt nu uit in een wit punt, wat er voor zorgt dat die naar toestand 14 gaat en het witte punt door poort 0 verlaat. $\mathcal{R}$ komt nu het rode punt via poort 0 binnen, wat inhoudt dat alle buren met een lager poortnummer dan $i(rood)$ gecontroleerd zijn en dat het rode punt het kind van het zwarte punt is. Nu wil $\mathcal{R}$ door oplopend de poorten af te gaan weer terug naar het zwarte punt waar de controle begon. Dit gebeurt doordat die naar toestand 18 gaat en daarbij verlaat die het rode punt door poort 1. Hierdoor is $\mathcal{R}$ weer terug in het zwarte punt en gaat die naar toestand 21. Hij verlaat hierbij het zwarte punt weer door poort 0 en komt daardoor aan in het rode punt waar nu van bekend, is dat het een kind is. Omdat het een rood

punt is, gaat die naar toestand 23 om hier weer te zoeken naar een kind en verlaat het punt door poort 0. Nu komt $\mathcal{R}$ een wit punt binnen via poort 0. Het punt is dus een kind en daarom gaat $\mathcal{R}$ naar toestand 24 en verlaat het witte punt door poort $i(wit) + 1 = 1 \bmod 1 = 0$. $\mathcal{R}$ komt nu een rood punt binnen via poort nul en gaat naar toestand 25, om te controleren of het de ouder is. Het rode punt wordt weer verlaten door poort 0 en komt het witte punt binnen via poort 0. Dit zorgt ervoor dat het direct duidelijk is dat het de ouder is, daarom gaat $\mathcal{R}$ naar toestand 31 en verlaat het witte punt. Bij aankomst in het rode punt gaat $\mathcal{R}$ naar toestand 23 en verlaat het punt door poort 1 om naar een mogelijk kind te zoeken. Bij het binnen komen van het zwarte punt moet weer gecontroleerd worden of het de ouder is. $\mathcal{R}$ gaat daarom naar toestand 25 en keert terug in het rode punt om buren met een lager poortnummer dan $i(rood)$ te controleren. Dit gebeurt doordat $\mathcal{R}$ naar toestand 28 gaat en door poort 0 het rode punt verlaat. Het witte punt waar die nu in terecht komt, zorgt er voor dat die naar toestand 29 gaat en terug keert naar het rode punt. Het rode punt komt $\mathcal{R}$ nu via poort 0 binnen en dat betekent dat alle voorgaande buren zijn gecontroleerd en dat het zwarte punt de ouder is. $\mathcal{R}$ gaat dan ook naar toestand 32 en verlaat het rode punt door poort 1. Hierdoor komt die uit bij het zwarte punt waarvan nu bekend is dat het de ouder is. Nu gaat $\mathcal{R}$ naar toestand 8 om verder te zoeken naar kinderen. Het zwarte punt wordt vervolgens verlaten door poort 1 en $\mathcal{R}$ komt nu aan in een wit punt. Ook hier moet dan weer gecontroleerd worden of het de ouder is. Dit gebeurt doordat $\mathcal{R}$ weer toestand 25 gaat en dan op dezelfde manier als hierboven komt $\mathcal{R}$ via toestand 27, 29 en 30 er achter dat dit de ouder is. Omdat $\mathcal{R}$ het witte punt via poort 0 binnen komt, gaat die naar toestand 24 om weer naar een kind op zoek te gaan en verlaat het punt door poort 1.
In de rest van de graaf gaat het allemaal ongeveer op dezelfde manier. Als je de hele graaf verkent dan zie je de looproute van de depth first search die de robot aflegt in Figuur 11. In de looproute volg je oplopend de punten via de groene kanten in het kortst mogelijke pad. Als je de zwarte kanten weg denkt dan zie je de gecreëerde boom die veroorzaakt wordt door de kind en ouder functie.



Figuur 11: De route van de verkenning

Wel zie je tijdens de verkenning dat je al veel eerder in het punt 15 komt, maar op dat moment nog niet als bezocht wordt gezien. Dit komt omdat je er tijdens de controle van een kant in komt die niet leidt naar een kind. In dit algoritme wordt een punt als bezocht gezien als het vanaf de ouder van dat punt komt, dus via de kant in de gecreëerde boom. En voor punt 15 gebeurt dat pas op het laatst. Deze boom is veder niet te zien aan het einde van de verkenning. Het is alleen ter verduidelijking voor hoe de robot door de graaf loopt.

3 Verkenning tijdens het labelen van een graaf

Met een kleine aanpassing in het bestaande algoritme, zie 2.2, en het toevoegen van toestanden aan de robot $\mathcal{R}$ is het mogelijk om een *ongekleurde* graaf te verkennen. Hierbij wordt er toegestaan dat de robot $\tilde{\mathcal{R}}$ de graaf kan kleuren tijdens de verkenning.

3.1 Verkenningsalgoritme

Aan het begin hebben alle punten nog geen kleur. Neem aan dat $\tilde{\mathcal{R}}$ deze ongekleurde punten kan herkennen en interpreteer dit als een vierde kleur naast de drie kleuren in $\mathcal{L}$. Deze vierde kleur wordt aan gegeven met l van leeg. Dit maakt het een volledig 2-bits schema. Voor het kleuren van de punten moet de robot twee extra dingen kunnen. Hij moet herkennen of een punt gekleurd is ja of nee en hij moet kunnen bepalen wat de volgende kleur moet zijn waarmee de punten gekleurd moeten worden. Dus *gekleurd*={ja, nee} en *volgende_kleur*={wit, zwart, rood}.

Deze verkenning wordt uitgevoerd door achtereenvolgende fases. Laat G_i een subgraaf zijn van G met alle punten erin die op afstand hoogstens i van r zitten, met $i \geq 0$. De robot heeft geen voorkennis over de vorm en grootte van de graaf G en daarom ook niet over de subgraaf G_i. Maar door de manier van kleuren, zoals later uitgelegd gaat worden, kan de robot weten welke punten daarbij horen. Het doel van fase i is om startend bij r in G_i alle punten met afstand i tot r te kleuren met de volgende kleur en weer te eindigen bij r.

Aan het begin van de verkenning, fase 0, kleurt de robot het punt waar die zich bevindt wit. Vervolgens wordt *volgende_kleur*=zwart en *gekleurd*=nee en start fase 1. Bij elke start van een nieuwe fase wordt *volgende_kleur* veranderd naar de volgende kleur in de lijst modulo drie en wordt *gekleurd* terug gezet naar nee.

Voor alle gevallen dat $i \geq 1$ voert de robot in de graaf G_i gewoon de verkenning uit die beschreven is in sectie 2.2. Alleen komt er een aanpassing bij de procedure voor het controleren van een kant $i(u)$. De procedure verloopt gewoon zoals beschreven behalve in het geval dat de kant $i(u)$ de robot naar een ongekleurd punt v leidt. In dit geval kunnen er twee dingen optreden:

1. De kleur van punt u is ongelijk aan *volgende_kleur*. In dit geval wordt het punt v gekleurd met de *volgende_kleur* en gaat $\tilde{\mathcal{R}}$ terug naar punt u. Om vervolgens in dit punt weer verder te zoeken naar een kind.

2. De kleur van punt u is gelijk aan *volgende_kleur*. In dit geval gaat $\tilde{\mathcal{R}}$ terug naar het punt u en doet alsof de kant $i(u)$ niet bestaat.

In het laatste geval kunnen er twee mogelijkheden zijn voor het punt v. Het kan een naaste zijn van het punt u of een opvolger. Maar omdat het nu nog onduidelijk is in welk geval je zit, wordt het punt niet gekleurd. Als v namelijk een naaste is, hoort die wel bij de subgraaf G_i en moet die met de kleur *volgende_kleur* gekleurd worden. Alleen als dit het geval is zal $\tilde{\mathcal{R}}$ via een ander punt u', met een kleur ongelijk aan *volgende_kleur*, het punt v binnen komen en kleuren met de kleur *volgende_kleur*. Per definitie geldt er namelijk dat elk punt op afstand i van r verbonden is met een punt dat op afstand $i-1$ van r afligt. En in het geval dat v een opvolger van u is, hoort die niet bij de subgraaf G_i en moet die nu nog niet gekleurd worden.

Wanneer je in het eerste geval zit en je voor het eerst in fase i een punt kleurt met *volgende_kleur* wordt *gekleurd* naar ja gezet. Als de robot van fase i naar fase $i+1$ wilt, dan moet die daar een paar dingen voor doen. Als $\tilde{\mathcal{R}}$ klaar is met de verkenning van de graaf G_i moet die controleren of *gekleurd* op ja staat. Is dit het geval dan moet *volgende_kleur* veranderd worden naar de volgende

kleur in de lijst en *gekleurd* moet terug gezet worden naar nee, zodat de verkenning van de graaf G_{i+1} kan beginnen. Is dit niet het geval en staat *gekleurd* nog op nee dan is de verkenning klaar. Als er namelijk tijdens de verkenning van graaf G_i geen punt meer gekleurd is, houdt dat in dat alle punten in de graaf G gekleurd zijn. Dus geldt er in dit geval dat $G_i = G$. Zoals beschreven in het algoritme stop $\tilde{\mathcal{R}}$ na de verkenning van G en dus stopt die ook in dit geval.

Stelling 3.1. *Er bestaat een eindige robot $\tilde{\mathcal{R}}$ die alle punten van een graaf G kan verkennen tijdens het kleuren. Gebruik makend van het aangepaste algoritme stopt de robot bij het beginpunt, nadat de hele graaf is gekleurd en verkend.*

3.2 Bewijs van stelling 3.1

Er moet na gegaan worden dat voor elke $i \geq 0$ de fase i aan de volgende eisen voldoet:

1. Alle punten in de graaf G_i worden met de juiste kleur gekleurd volgens $\mathcal{L}$.
2. Alleen de punten in G_i zijn gekleurd

We bewijzen dit met inductie.
Voor het geval $i = 0$. Elk punt waar de robot op start, kan gezien worden als het beginpunt r. Per constructie wordt dit punt, wit gekleurd en wordt er aan de voorwaarden voldaan.

Neem nu aan dat op het einde van fase $i-1$ aan de voorwaarden wordt voldaan. Dan moet nu bewezen worden dat aan he einde van fase i dat ook geldt. $\forall i$ geldt dat de *volgende_kleur* = wit als $i \bmod 3 = 0$, *volgende_kleur* = zwart als $i \bmod 3 = 1$ en *volgende_kleur* = rood als $i \bmod 3 = 2$. In fase $i-1$ zijn alle punten in G_{i-1} met de juiste kleur gekleurd. Dit houdt in dat de *volgende_kleur* voor fase i dan automatisch de juiste kleur is, wat het hoort te zijn volgens het schema $\mathcal{L}$. Uit dit samen met de manier hoe de punten gekleurd worden, volgt direct dat alle punten op afstand i van r met de juiste kleur gekleurd worden en dat alle punten op afstand $i+1$ van r niet gekleurd worden. En dus voldoet fase i ook aan de voorwaarden. □

Er geldt dus dat de verkenning zoals uitgelegd in 2.2 $D+1$ keer gedaan wordt, met D de diepte van de graaf. In de eerste D fases wordt de graaf volledig gekleurd en verkend. Alleen op dit moment kan de robot dit nog niet herkennen. Daarom wordt er nog een laatste fase uit gevoerd waarna de robot dit wel herkent en stopt. Ook is er al bewezen dat een gekleurde graaf in $O(m)$ tijd verkent kan worden en daarna stopt. Dit samen houdt in dat de kleuring en verkenning van een ongekleurde graaf in $O(Dm)$ tijd gedaan wordt.

3.3 De robot

De Robot $\tilde{\mathcal{R}}$ die deze verkenning kan doen is ongeveer 6 keer zo groot als de robot uit 2.4. De eerste 6 toestanden uit $\mathcal{R}$ zijn hier niet nodig. Elk punt kan namelijk gebruikt worden als het beginpunt r. De overige toestanden uit $\mathcal{R}$, met wat aanpassingen, heb je zes keer nodig in $\tilde{\mathcal{R}}$. Definieer de deel robot $\tilde{\mathcal{R}}_k$, met $1 \leq k \leq 6$, als een kopie van $\mathcal{R}$ met de benodigde aanpassingen en t_k met t de toestand in deel robot k. In $\tilde{\mathcal{R}}$ moet duidelijk zijn met welke kleur de punten op afstand i van r in de subgraaf G_i gekleurd moeten worden en of er al een punt gekleurd is. Deze opties gecombineerd geven zes mogelijkheden. Elke deel robot $\tilde{\mathcal{R}}_i$ geeft één van deze situaties aan. Wel moeten er per deel robot een paar aanpassingen gedaan worden. Een leeg punt moet namelijk gekleurd kunnen

worden als dat moet of genegeerd worden als dat niet moet. Ook moeten de deel robots naar elkaar kunnen overschakelen. Om dit duidelijk te maken leg ik dit aan de hand van twee voorbeelden uit. De deel robot waar nog geen punt zwart gekleurd is, $\tilde{\mathcal{R}}_3$, en de deel robot waar al wel een punt zwart gekleurd is, $\tilde{\mathcal{R}}_4$. Voor de andere deel robots gaan de aanpassingen op dezelfde manier.

In $\tilde{\mathcal{R}}_3$ is er één aanpassing die gedaan moeten worden. Voor de deel robot $\tilde{\mathcal{R}}_5$, waar nog geen punt rood gekleurd is, geldt dit ook, maar voor $\tilde{\mathcal{R}}_1$, waar nog geen punt wit gekleurd is, zijn er twee aanpassingen nodig. De tweede aanpassing voor $\tilde{\mathcal{R}}_1$ volgt na de uitleg over $\tilde{\mathcal{R}}_4$. De eerste aanpassing is dat $\tilde{\mathcal{R}}_3$ na het kleuren van een punt moet kunnen overschakelen naar $\tilde{\mathcal{R}}_4$. In dit geval kleurt de robot een punt zwart als die vanuit een wit punt komt. Dit kan plaats vinden wanneer $\tilde{\mathcal{R}}_3$ in toestand 24_3 is. Er moet nu in toestand 24_3 een extra uitgaande pijl komen die naar de nieuwe toestand 42_3 leidt, wanneer $\tilde{\mathcal{R}}$ in een leeg punt terecht komt. In toestand 42_3 kleurt $\tilde{\mathcal{R}}$ het punt zwart. Daarna verlaat die het punt via dezelfde kant en komt die weer in het witte punt terecht. Nu is er voor het eerst een punt gekleurd en wil je overschakelen naar de deel robot $\tilde{\mathcal{R}}_4$. Dit gebeurt doordat er een uitgaande pijl in toestand 42_3 komt die naar toestand 24_4 leidt. $\tilde{\mathcal{R}}$ verlaat toestand 42_3 via deze pijl als die in een wit punt terecht komt en bij het binnen komen van toestand 24_4 verlaat die het witte punt door poort $i+1$.

In $\tilde{\mathcal{R}}_4$ zijn er drie aanpassingen die gedaan moeten worden. Dit geldt ook voor $\tilde{\mathcal{R}}_2$, waar een punt al wit gekleurd is, en voor $\tilde{\mathcal{R}}_6$, waar een punt al rood gekleurd is. De eerste aanpassingen is voor een deel hetzelfde als die hierboven genoemd is, de tweede aanpassing is als $\tilde{\mathcal{R}}_4$ vanuit een zwart punt een leeg punt binnen komt, dan moet dit lege punt genegeerd worden en de derde aanpassing is dat als de verkenning normaal klaar zou zijn en $\mathcal{R}$ zou stoppen, dan moet de verkenning nu door gaan en wil je naar de volgende fase. De volgende fase houdt ook in dat je naar de volgende deel robot moet gaan.

1. Deze aanpassing is bijna hetzelfde als in deel robot $\tilde{\mathcal{R}}_3$. Het enige verschil nu is dat je na het kleuren van een punt in de deel robot $\tilde{\mathcal{R}}_4$ blijft en je niet over hoeft te schakelen naar een andere deel robot. Dus nu gaat $\tilde{\mathcal{R}}$ na het kleuren van het lege punt in toestand 42_4 naar toestand 24_4.

2. Als $\tilde{\mathcal{R}}_4$ vanuit een zwart punt door $i(u)$ in een leeg punt terecht komt, is het niet direct duidelijk of dit punt gekleurd moet worden. In dit geval moet $\tilde{\mathcal{R}}_4$ de kant $i(u)$ kunnen negeren. Dit kan plaats vinden als $\tilde{\mathcal{R}}_4$ zich in toestand 8_4 bevindt. Door een extra uitgaande pijl toe te voegen aan toestand 8_4 die leidt naar toestand 9_4 kan je dit voor elkaar krijgen. Als je vanuit toestand 8_4 een leeg punt tegen komt, ga je via deze pijl naar toestand 9_4 en verlaat je het punt door de kant waardoor je naar binnen kwam. Nu is $\tilde{\mathcal{R}}_4$ weer terug in het zwarte punt. In toestand 9_4 is er al een uitgaande pijl die er nu voor zorgt dat je terug naar toestand 8_4 gaat en verder gaat zoeken naar een kind door poort $i(u)+1$ te verlaten.

3. Als $\tilde{\mathcal{R}}$ in fase i is en $\mathcal{R}$ zou op dit moment gestopt zijn met de verkenning dan wil je nu in $\tilde{\mathcal{R}}$ naar fase $i+1$. Toestand 41_4 moet nu verwijderd worden en dezelfde pijl leidt nu naar toestand 7_5. Bij het binnen komen van toestand 7_5 verlaat $\tilde{\mathcal{R}}$ het witte punt door poort $i+1$. Merk op dat je nu in het beginpunt r zit en je het door poort $i+1=0$ verlaat.

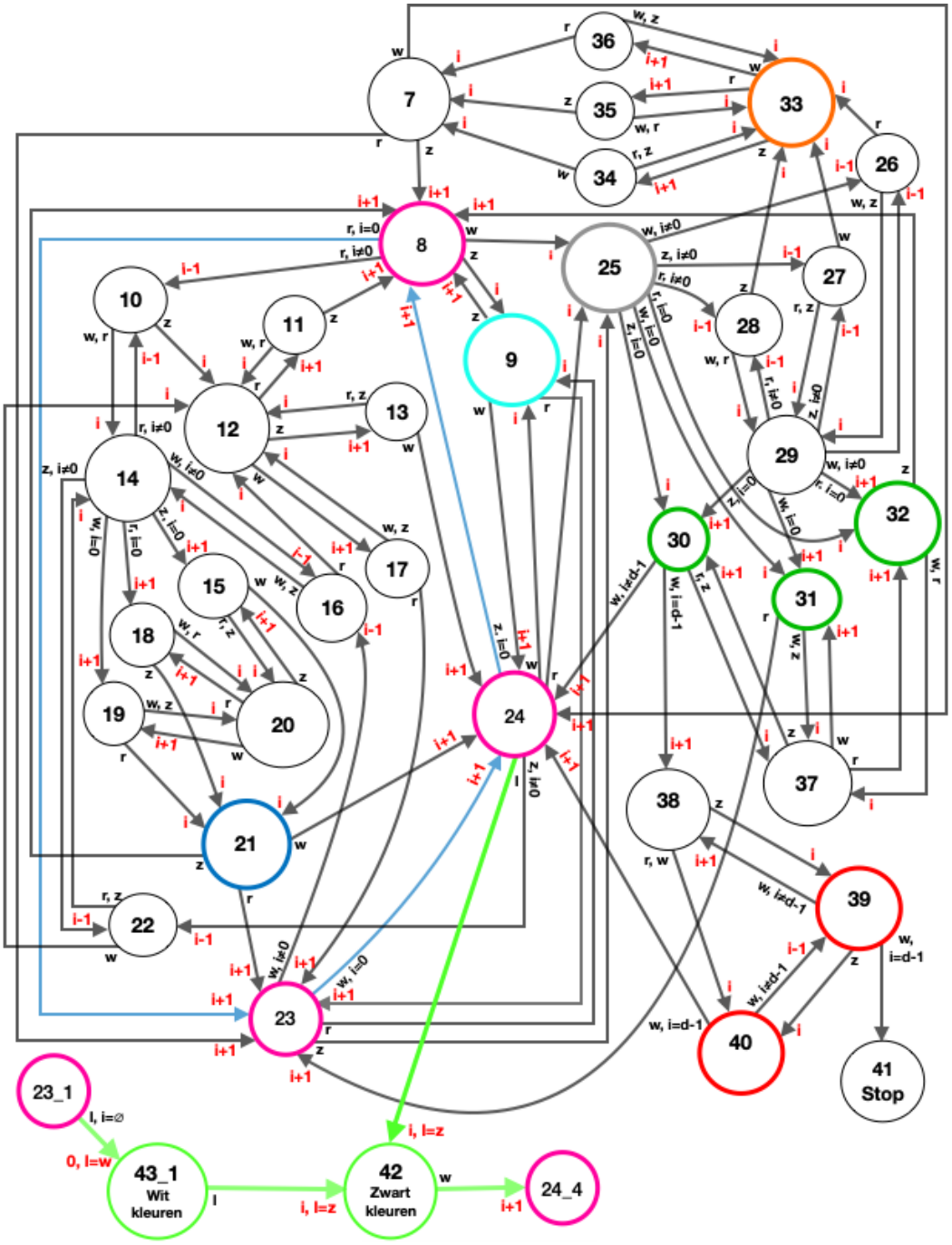
Merk ook op dat de tweede aanpassing alleen nodig is in de deel robots $\tilde{\mathcal{R}}_2$, $\tilde{\mathcal{R}}_4$ en $\tilde{\mathcal{R}}_6$. In de andere deel robots zijn nog geen punten gekleurd. Zodra dat wel het geval is zullen ze direct overschakelen naar één van deze deel robots. En dus kan het bijvoorbeeld in deel robot $\tilde{\mathcal{R}}_3$ nooit

voorkomen dat je vanuit een zwart punt in een leeg punt terecht komt. Ook kan je in deel robot $\tilde{\mathcal{R}}_4$ nooit vanuit een rood punt in een leeg punt terecht komen. Je zit in dit geval in de situatie dat alle punten op afstand i van r zwart gekleurd moet worden. Per definitie zijn al deze punten verbonden aan alle punten die op een afstand $i-1$ van r liggen en die zijn wit gekleurd en niet aan een rood punt die op afstand $i-2$ van r ligt.

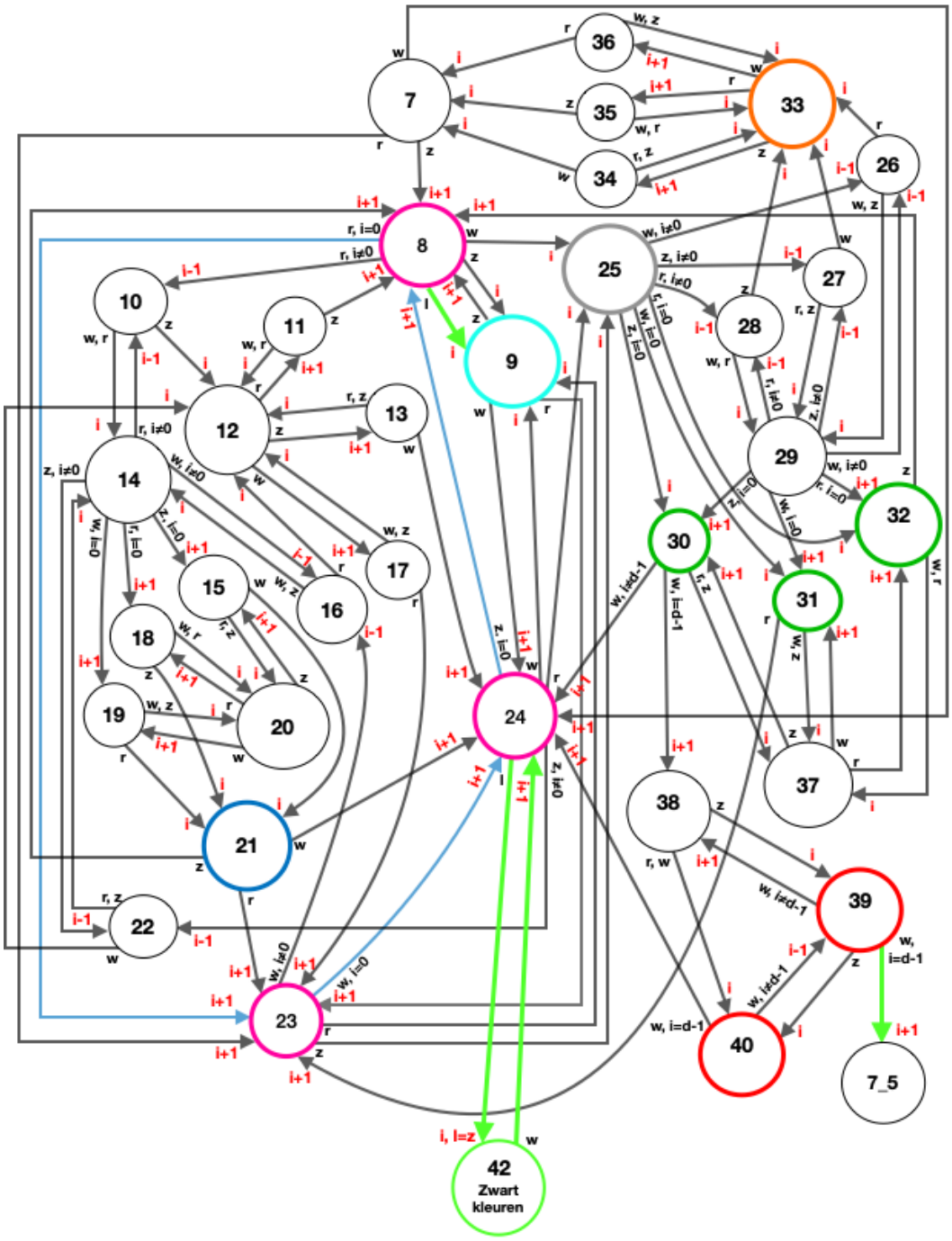
Voor de derde aanpassing geldt dat de verkenning in $\tilde{\mathcal{R}}$ alleen kan eindigen in de deel robots $\tilde{\mathcal{R}}_1$, $\tilde{\mathcal{R}}_3$ en $\tilde{\mathcal{R}}_5$. In deze gevallen zijn er namelijk nog geen punten gekleurd. Dus als je de hele verkenning in deze deel robots kan uitvoeren, betekent het dat alle punten in G gekleurd en bezocht zijn.

Na al deze aanpassingen blijft er nog één probleem over. Namelijk in welke toestand begint de robot de verkenning. Aan het begin wordt de robot op een willekeurig punt neergezet in de graaf. Dit punt is dan het beginpunt en moet wit gekleurd worden. $\tilde{\mathcal{R}}$ begint de verkenning dus in de deel robot $\tilde{\mathcal{R}}_1$. Na het kleuren van het beginpunt kan je direct door naar fase 1, omdat je weet dat de volgende punten direct zwart gekleurd kunnen worden. Er kan daarom een pijl toegevoegd worden tussen de deel robot $\tilde{\mathcal{R}}_1$ en $\tilde{\mathcal{R}}_3$. Daar is wel een extra toestand voor nodig, zeg toestand 43_1. $\tilde{\mathcal{R}}$ start in toestand 23_1 en gaat met input $k = l$ en $i = \emptyset$ naar toestand 43_1. Aangekomen in deze toestand kleurt de robot het punt wit en verlaat het punt door poort 0. Bij het verlaten van het beginpunt r komt de robot in een leeg punt terecht. Het is duidelijk dat dit punt direct zwart gekleurd kan worden. Daarom is er een pijl van toestand 43_1 naar 42_3 die hier voor zorgt. Bij het aankomen in toestand 42_3 wordt het lege punt zwart gekleurd en gaat de verkenning verder zoals het normaal zou gaan. De robot start dus in toestand 23_1.

De aanpassingen die gedaan zijn voor $\tilde{\mathcal{R}}_3$ en $\tilde{\mathcal{R}}_4$ kan je zien in Figuur 12 en 13. De aanpassingen worden aangegeven door de licht groene pijlen en toestanden.



Figuur 12: Deel robot $\tilde{\mathcal{R}}_3$



Figuur 13: Deel robot $\tilde{\mathcal{R}}_4$

4 Discussie

De eindige automaten die gebruikt worden voor beide algoritmes kunnen eigenlijk een paar extra dingen, wat een eindige automaat normaal niet kan. Deze robots zijn namelijk in staat om cijfers te lezen. Ook hebben ze als uitkomst van de transitie functie niet alleen de volgende toestand, maar ook een element uit het alfabet. En de robot van het algoritme van 3.1 moet labels achter kunnen laten op een punt. Het nadeel van dit algoritme is dat de robot de poortnummer en de graad nodig heeft voor de transitiefunctie. Dit zorgt er namelijk voor dat elk punt een extra label nodig heeft met de graad van dat punt. Het kan de robot $\lceil \log_2(m) \rceil$ bits kosten om de graad te beschrijven en net zo voor het poortnummer, met m het aantal kanten van de graaf G. Ook zorgt de graad en de poortnummer er voor dat de robot niet één eindig alfabet heeft, maar per graaf met m kanten een eindig deel alfabet moet kiezen.

Een andere mogelijkheid om deze algoritmes uit te voeren is door middel van een robot met een natuurlijke ruimtelijke oriëntatie. Als extra moet de robot nu kunnen bewegen zoals je in een fysieke omgeving kan bewegen. De robot weet nu met de outputs: 'verlaat door de poort links van je', 'verlaat door dezelfde poort' of 'verlaat door de poort rechts van je' waar die naartoe moet zonder een poortnummer te weten. De robot heeft dus geen poortnummer of graad van de punt meer nodig. De voordelen hiervan zijn dat er niet meer zo veel bits nodig zijn om het te beschrijven en dat er nu één eindig alfabet is waar de robot mee kan werken. De label van de graad op elk punt kan nu namelijk weg. Wel moet de robot voor het algoritme in staat kunnen zijn om de kanten met poortnummer 0 en $d-1$ te herkennen. Dit is op te lossen door elke kant met poortnummer 0 een extra label te geven. Hierdoor weet de robot wanneer die bij poort 0 is en door een stap terug, naar links, te nemen weet je dat die bij poort $d-1$ zit. Ook hoeft er geen deel alfabets meer gekozen te worden afhankelijk van het aantal kanten van de graaf G. Je hebt nu namelijk als input van de transitiefunctie: de huidige toestand, de kleur van het punt en of de robot zicht bij poort nul bevindt. Als de robot zich niet bij poort nul bevindt dan is de input hiervan leeg, $\emptyset$. Dus voor de kleur zijn er drie mogelijkheden, $k = \{\text{zwart, rood, wit}\}$, en voor de poort heb je twee mogelijkheden, $i' = \{0, \emptyset\}$. Er zijn in totaal 6 mogelijke combinaties van (k, i) voor het alfabet. Als output heeft de transitiefunctie nu door welke richting het punt verlaten moet worden, i'=\{Rechts, Links, Zelfde\} en de volgende toestand. Hierdoor komen er nog drie tekens bij het alfabet. In totaal bestaat het alfabet dus uit 9 tekens. In het geval dat de robot de punten nog moet kleuren, komen er vier mogelijkheden voor de kleur. In totaal heeft deze robot een alfabet van 11 tekens.

Door een kleine aanpassing in de bestaande robot, zie Figuur 2.4, krijg je een robot die werkt zoals hierboven beschreven. Er zijn hiervoor twee aanpassingen nodig in de bestaande robot. De transitiefunctie ziet er nu als volgt uit, $f(t, k, i) = i', t'$. Allereerst worden daarom de poortnummers in de outputs als volgt veranderd, $i + 1 = R$, $i = Z$ en $i - 1 = L$. Veder komen er in de bestaande robot voor de input vier verschillende soorten eisen op de poortnummer voor. Namelijk: $i = 0$, $i \neq 0$, $i = d - 1$ en $i \neq d - 1$. De eerste twee eisen $i = 0$ en $i \neq 0$ kunnen herkend worden door de label op de kant met poortnummer 0. Hier hoeft dus geen aanpassing voor gedaan te worden. Voor de laatste twee eisen, $i = d-1$ en $i \neq d-1$, is wel een aanpassing nodig. Deze robot kan deze poort namelijk niet herkennen. Deze eisen komen op meerdere plekken in de robot voor, namelijk wanneer er gecontroleer moet worden of de robot mogelijk in het beginpunt r zit. Dit gebeurt aan het begin, wanneer er gecontroleerd moet worden of de verkenning gestart kan worden, en wanneer de robot een wit punt 'omhoog' naar binnen komt, om te controleren of de verkenning gestopt kan worden. De aanpassing die gedaan moet worden op de plekken waar $i = d - 1$ en $i \neq d - 1$ staat,

is dat de robot moet controleren of de kant met poortnummer 0 recht van de kant ligt waardoor die naar binnen gekomen is, noem deze kant $i(u)$. Als dit het geval is dan is het poortnummer $i(u)$ gelijk aan $d(u) - 1$ en anders niet. Hier kan je achter komen door de robot het punt u door de kant rechts van $i(u)$ te laten verlaten. Bij aankomst in het punt v via dezelfde kant weer terug laten komen, zodat de robot weer in punt u terecht komt. Komt de robot nu via poort 0 het punt u binnen dan is $i(u)$ gelijk aan $d(u) - 1$ en anders niet. Nu moet de robot weer terug naar poort $i(u)$ en weet die in welke situatie die zit, $i = d - 1$ of $i \neq d - 1$.

Referenties

[1] Reuven Cohen, Pierre Fraigniaud, David Ilcinkas, Amos Korman, and David Peleg. Label-guided graph exploration by a finite automaton. *ACM Trans. Algorithms*, 4(4):18, 2008. Id/No 42.