Verslag Lokale Zoekalgoritmen Practicum 2

Inleiding

Chronological Backtracking

In dit experiment gaan we het constraint satisfaction algoritme chronological backtracking gebruiken om een sudoku puzzel op te lossen. Chronological backtracking is een vorm van backtracking: de zoekboom wordt dynamisch gegenereerd door de nodes depth-first te expanderen. Het verschil met andere backtracking-methodes is dat Chronological backtracking alleen de nodes expandeert die leiden tot een geldige partiele oplossing van het probleem. Dat betekent dus dat het niet in strijd is met de regels van de Sudoku. Het chronological backtracking algoritme is dus efficiënt en alleen geschikt voor constraint satisfaction problems.

Forward Checking

Forward checking gaat nog een stap verder dan chronological backtracking. Stel je hebt een sudokuveld met een paar ingevulde en voor de rest lege vakken. Als er een waarde (getallen 1 t/m9) aan een variabele (leeg vak) wordt toegekend, gaat het Forward Checking algoritme na wat de constraints zijn als dit getal in dat vak wordt geplaatst. Als er bijvoorbeeld een 1 linksboven in de sudoku wordt geplaatst zal er in de gehele horizontale en verticale rij geen 1 mogen komen, ook zal er in het 3x3 vak waar de 1 zich bevindt geen 1 kunnen komen. Het forward checking algoritme verwijdert deze dus uit het domein van de zoekruimte voor de variabelen. Hierdoor wordt de zoekruimte kleiner, dus bij na backtracking zijn er minder opties om uit te kiezen waardoor het efficiënter is.

Forward Checking met MCV Heuristiek

De derde variant die we in dit experiment gaan behandelen is forward checking met most-constrained-variable (MCV) heuristiek. Deze variant bouwt verder op de vorige variant waar de zoekruimte verkleind werd. Nadat FC is toegepast ordent MCV de variabelen naar toenemende domeingrootte. Hierdoor zie je welke waardes de meeste kans van slagen hebben.

Onze verwachtingen

Onze verwachtingen zijn dat Chronological backtracking geschikter zal zijn voor minder complexe sudoku puzzels. Wanneer de sudoku complexer wordt zal de zoekruimte ook groter worden. Wanneer de zoekruimte groot is zal het relatief veel tijd kosten bij deze variant omdat er nog steeds veel geldige opties blijven die onderzocht worden.

Wij verwachten dat de Forward Checking aanpak nog efficiënter zal zijn dan enkel Chronological backtracking. Dit zul je vooral terugzien bij de complexe sudoku's waarbij het belangrijk is de zoekruimte te verkleinen.

Vanzelfsprekend verwachten wij dat de Forward Checking aanpak met MCV Heuristiek nog efficiënter zal zijn. Bij complexe sudoku's zul je dit ook merken, het zal sneller de zoekruimte verkleinen en zal daardoor efficiënter zijn en een nog snellere oplossingstijd hebben dan forward checking.

Resultaten

Het snelste algoritme was Forward Checking, nauw gevolgd door Chronological Backtracking, de langzaamste was Forward checking met Most Constrained Variable heuristiek. Chronological Backtracking en Forward Checking zijn bijna exact hetzelfde, het enige verschil tussen de twee is dat Backtracking het domein van een variabele bij expansie berekent, terwijl Forward Checking een lijst van alle domeinen bijhoudt en deze lijst update bij expansie. Bij het updaten van deze lijst hoeven minder berekeningen gemaakt te worden dan bij het berekenen van een domein. Forward Checking met Most Constrained Variable heuristiek was in ons geval heel erg langzaam, dit komt omdat het programma soms de hele sudoku voor elk mogelijke getal lijkt te checken of in een cykel lijkt te zitten (dit zou in onze implementatie niet mogelijk moeten zijn). Alleen sudoku's 1 en 5 worden opgelost door Forward Checking met MCV-heuristiek, dit zijn ook meteen de twee makkelijkste. Het kan zijn dat FC met MCV zo langzaam is omdat we het niet op de juiste manier hebben geprogrammeerd maar we zijn er na lang testen en debuggen niet achter gekomen wat we hier fout hebben gedaan. In theorie zou de MCV-heuristiek het oplossen van de sudoku gemiddeld sneller moeten laten verlopen omdat het, in tegenstelling tot de andere twee methodes, elke keer een vakje zal invullen als er maar één getal mogelijk is in dat vakje.

	Sudoku 1	Sudoku 2	Sudoku 3	Sudoku 4	Sudoku 5
CBT	1 ms	0 ms	0 ms	18 ms	1 ms
FC	0 ms	1 ms	2 ms	8 ms	0 ms
FC+MCV	0 ms	Niet opgelost	Niet opgelost	Niet opgelost	2 ms

Conclusie

De resultaten sluiten uit eindelijk redelijk goed aan op onze resultaten. Zoals we verachten werkt het Chronological backtracking algoritme zeer efficiënt bij de sudoku's die makkelijk op te lossen waren voor de algoritmes. Wel is er een verschil te zien tussen het Chronological Backtracking en het forward checking algoritme bij de sudoku die wat lastiger voor de algoritmes werd opgelost. Forward Checking is dus naar verwachting efficiënter voor het oplossen van complexere sudoku's doordat de zoekruimte effectief verkleint door het domein van variabelen bij te houden. In ons onderzoek werkt de Forward Checking met Most Constrained Variable heuristiek niet naar verwachting, maar het is zeker mogelijk dat met wat aanpassingen het algoritme nog efficiënter is dan het forward checking algoritme in het oplossen van sudoku's. Ten slotte zijn wij zeker tevreden over de resultaten van de algoritmes die als gewenst werken, de sudoku's zijn met deze algoritmes aanzienlijk sneller opgelost dan door de algoritmes van het vorige assignment, alleen is het jammer dat het Forward Checking met Most Constrained Variable heuristiek niet volledig naar verwachting werkt.