Computationele Intelligentie

Practicum 2: Chronological Backtracking (CBT)

Opdracht. Net als in de eerste opdracht wordt gevraagd om een zoekalgoritme te implementeren en te bespreken dat Sudoku puzzels oplost. We gebruiken nu het constraint satisfaction algoritme Chronological Backtracking, waarvan we 3 varianten vergelijken.

1. **Chronological Backtracking** (CBT, zie slide 18): de variabelen (= lege vakken) worden geëxpandeerd in de volgorde van links naar rechts en van boven naar onder (rij per rij). De waarden (= cijfers) van de variabelen worden toegekend in oplopende waarden, i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9.

2. **Forward Checking** (FC, zie slide 49): de variabelen worden geëxpandeerd in de volgorde van links naar rechts en van boven naar onder (rij per rij). De waarden (= cijfers) van de variabelen worden toegekend in oplopende waarden, i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9.

   Vooraleer je FC toepast maak je het CSP probleem eerst knoopconsistent (zie slide 30).
   In Sudoku sluiten de cijfers in de vaste, gegeven vakken op voorhand een aantal waarden uit bij de variabele vakken. Deze constraints kunnen we beschouwen als reflexieve constraints op het domein van de variabelen. Bijvoorbeeld als een gegeven, gefixeerd vak in de puzzel een vaste waarde 3 heeft, dan kan de waarde 3 niet meer voorkomen in de variabelen V_i in dezelfde rij, kolom, of 3x3 blok. De variabelen V_i hebben dus allemaal de reflexieve constraint $(3, 3) \notin C_{i,i}$.

3. **Forward Checking** (FC) waarbij de variabelen geëxpandeerd worden volgens de **most-constrained-variable (MCV) heuristiek** (zie slide 63). De MCV heuristiek wordt dynamisch toegepast, m.a.w. de variabelen worden na iedere variabele expansie gesorteerd op domeingrootte en FC wordt toegepast op de eerstvolgende variabele in deze ordening.

Reductie algoritmen maken de zoekboom die CBT moet doorzoeken kleiner. De hamvraag is of deze winst opweegt tegen de rekentijd van de reductiealgoritmen. Ga dit na voor de vijf gegeven Sudoku puzzels: m.a.w. hoe verhouden de bovenstaande 3 varianten zich wat betreft efficiëntie ?

Verslag. Het doel van het practicum is om inzicht te verkrijgen in lokaal zoeken: in het verslag moet dit inzicht zo goed mogelijk tot uiting komen. Bespreek dus wat je hebt gedaan, waarom, wat je observeert, welke factoren zijn van invloed, hoe gevoelig is het zoekalgoritme voor bepaalde parameter keuzes, enz. ... .

Rapporteer hoe efficiënt (rekentijd) je implementatie is om Sudoku puzzels op te lossen.

Reflecteer ook over hoe het programmeren is verlopen. Hoeveel tijd heeft het je bijvoorbeeld gekost om de zoekalgoritmes te implementeren.

Geef voldoende commentaar bij de code: het moet duidelijk zijn wat je code doet door enkel de commentaar te lezen.

Je programma wordt getest op andere puzzels. Geef duidelijk aan hoe je programma Sudoku puzzels inleest.

Groepen. Maak het practicum in groepen van 3 studenten. iedere groep ontwikkelt zijn eigen code. Bij plagiaat worden alle betrokkenen gesanctioneerd en de fraude wordt gemeld bij de examencommissie. Geef ook aan wat de bijdrage is van ieder groepslid bij het ontwerp, implementatie, en testen van het programma, en de bijdrage tot het verslag.

Programmertaal. C# or Python.

Deliverables. Submit je verslag en je code in Brightspace.

Beoordeling.

1. • 3 pnt: Correctheid van code
   Zijn de algoritmen correct geimplementeerd?

2. • 1 pnt: Efficientie en duidelijkheid van code
   Is de implementatie efficient? Runt het programma snel?

3. • 2 pnt: Verslag en uitleg algoritmen
   Zijn alle gevraagde items in het verslag aanwezig? Is het verslag net vormgegeven? Blijkt uit het verslag dat de student de algoritmen heeft begrepen? Worden de algoritmen en experimenten duidelijk en in eigen woorden uitgelegd?

4. • 2 pnt: Experimenten
   Zijn de experimenten goed uitgevoerd? Komen de resultaten overeen met de code? Zijn de resultaten goed weergegeven?

5. • 2 pnt: Bespreking
   Zijn de resultaten goed besproken? Blijkt uit de conclusies dat de student inzicht heeft?

6. Bijkomend: • 0,5 - 1 pnt: Bonus bij extra inspanning
   Bijv: erg efficiente code, extra algoritmes geimplementeerd, extra experimenten uitgevoerd en besproken.

Deadline. Vrijdag, 23 januari 2026 (23:59)

Laattijdig inleveren. Je kan tot 1 week na de deadline inleveren (vrijdag, 30 januari 2026, 23:59), maar je cijfer is dan maximaal 6 !