Computationele Intelligentie

Lokale zoekalgoritmen

Practicum 1

Probleem. Als probleem domein gebruiken we een makkelijk te implementeren puzzel: sudoku. Standaard Sudoku bestaat uit 9×9 vakken gegroepeerd in 9 blokken van 3×3 vakken. In de vakken moeten de cijfers 1 tot en met 9 ingevuld worden zodat in elke horizontale rij en in elke verticale kolom en in elk van de 9 blokken de cijfers 1 tot en met 9 precies één keer voorkomen. In de begintoestand van de puzzel zijn een aantal vakken reeds ingevuld.

Opdracht. Bij het gebruik van lokale zoekalgoritmen is een goede keuze van de representatie van de probleemtoestanden, de lokale zoekoperator, en de evaluatiefunctie belangrijk. Een goede keuze is als volgt:

- Als probleemtoestanden beschouwen we volledig ingevulde Sudoku puzzels waarbij in ieder blok van 3×3 vakken alle cijfers van 1 tot 9 precies één keer voorkomt.
- De zoekoperator verwisselt 2 cijfers - uit twee niet gefixeerde vakken - binnen hetzelfde blok. De zoekruimte bestaat dus enkel uit toestanden die voldoen aan de eis dat de cijfers in ieder blok precies één keer voorkomen.
- Als evaluatiefunctie tellen we voor iedere horizontale rij en iedere verticale kolom het aantal cijfers dat ontbreekt in de rij/kolom. Een optimale oplossing heeft een score gelijk aan nul. Het updaten van de evaluatiefunctie vergt enkel een update van ten hoogste twee rijen en twee kolommen.

Iterated Local Search: Het hill-climbing algoritme kan uitgevoerd worden met de best-improvement of met de first-improvement variant. Best-improvement neemt de beste oplossing uit alle buuroplossingen. First-improvement genereert de successor toestanden één voor één en test of een toestand beter is dan de huidige toestand. Indien dit zo is, dan wordt deze toestand de nieuwe huidige toestand en wordt de rest van de neighborhood niet meer onderzocht. In het practicum werken we met een tussenvorm:

1. Kies willekeurig één van de 9 (3×3)-blokken,
2. Probeer alle mogelijke swaps - binnen het blok - van 2 niet-gefixeerde cijfers,
3. Kies hieruit de beste indien die een verbetering of gelijke score oplevert.

Hill-climbing stagneert in locale optima of blijft op een plateau ronddolen. Een handige manier om hier mee om te gaan is iterated local search (ILS). ILS combineert hill-climbing met een (korte) random-walk: wanneer hill-climbing een lokaal optimum heeft bereikt (of te lang op een plateau blijft ronddolen) wordt de zoekoperator S keer toegepast zonder te kijken naar de evaluatiewaarden van de successor toestanden. Merk op: je moet een stop-criterium instellen, het zoeken kan immers oneindig lang blijven doorlopen op de plateaus. Onderzoek het gedrag van ILS voor verschillende waardes van S (vertrek hierbij van een vrij lage S waarde).

Sudoku puzzels. Test je programma op de bijgeleverde 5 puzzels.

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 in je verslag duidelijk aan hoe je programma deze puzzels moet inlezen.

Groepen. Maak het practicum in groepen van 3 studenten. Schrijf je in als groep in Blackboard (en onthoudt je groepsnummer !). Iedere groep ontwikkelt zijn eigen code. Bij plagiaat worden alle betrokkenen gesanctioneerd en de fraude wordt gemeld bij de examencommissie.

Programmertaal. C# of 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, 12 december (23:59 hrs).

Laattijdig inleveren. Je kan tot 1 week na de deadline (22/12) inleveren, maar je cijfer is dan maximaal 6 !