Computationele Intelligentie

Lokale zoekalgoritmen

Practicum 1

Probleem. Als probleemdomein gebruiken we een makkelijk te implementeren puzzel: sudoku. Sudoku bestaat uit $N \times N$ vakken gegroepeerd in N blokken van $\sqrt{N} \times \sqrt{N}$ vakken. In de vakken moeten de cijfers 1 tot en met N ingevuld worden zodat in elke horizontale rij en in elke verticale kolom en in elk van de N blokken de cijfers 1 tot en met N precies één keer voorkomen. In de begintoestand van de puzzel zijn een aantal vakken reeds ingevuld. Standaard Sudoku puzzels hebben $N = 9$ vakken maar om te kijken hoe de rekentijd opschaalt gebruiken we hier ook $N = 16$ puzzels.

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 $\sqrt{N} \times \sqrt{N}$ vakken alle cijfers van 1 tot N 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.

1. 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 N ($\sqrt{N} \times \sqrt{N}$)-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).

In principe ga je bij ILS het nieuwe lokale optimum vergelijken met het vorige optimum en de beste van de twee kiezen. Dit heeft bij Sudoku wellicht niet veel nut. Hier in het practicum gaan we steeds verder met zoeken vanaf het nieuwe optimum.

2. Simulated annealing Een ander veel gebruikt lokaal zoekalgoritme is simulated annealing. We kunnen dezelfde probleemrepresentatie, zoekoperator en evaluatiefunctie gebruiken als bij ILS. Kies zelf een begintemperatuur en koelingschedule. Leg in het verslag uit hoe je tot je keuze bent gekomen.

Sudoku puzzels.

1. Test allereerst op de volgende Sudoku puzzel:

   0 0 0 0 0 0 0 0 0
   0 0 0 0 0 3 0 8 5
   0 0 1 0 2 0 0 0 0
   0 0 0 5 0 7 0 0 0
   0 0 4 0 0 0 1 0 0
   0 9 0 0 0 0 0 0 0
   5 0 0 0 0 0 0 7 3
   0 0 2 0 1 0 0 0 0
   0 0 0 0 4 0 0 0 9

   (Hierbij representeert nul de blanco ruimte, en zijn de niet-nullen de vaste cijfers in de puzzel).

2. Test op de bijgeleverde 10 (9 x 9)-puzzels en 3 (16 x 16)-puzzles.

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

Verdere richtlijnen (deliverables en beoordeling) vind je in het bijgevoegd requirements document.

Groepen. Maak het practicum in groepen van 2 of 3 studenten.

Programmertaal. Je moet het practicum implementeren in C# of Python (voorkeur voor C# ivm snelheid).

Deliverables. Lever het verslag en je code in via **SUBMIT**:

`http://www.cs.uu.nl/docs/submit/`

Deadline. Woensdag, 29 mei (23:59).