Gonno's vakkenvuller: een stap-voor-stap Sudoku Solver

Methode

De hier gebruikte stap-voor-stap oplosmethode is geïmplementeerd in de scripttaal PHP[1] en is een vertaling van een eerder door mij gemaakte Matlab[2]-versie. Het maken van die versie gaat terug naar 2009. In dat jaar heb ik een aantal oplosmethoden voor de Sudoku-puzzel uitgewerkt in Matlab. De eerste methode maakte gebruik van het feit dat een Sudoku-puzzel (wiskundig) gezien kan worden als een bijzondere vorm van een zogenoemd binair integer lineair programmeringsprobleem. Voor dit soort problemen bevat Matlab, althans de in het verlengde van de standaardversie verkrijgbare Optimization Toolbox, een adequate oplosmodule, de functie *bint*.
Toepassing van *bint* maakt duidelijk of de Sudoku een oplossing heeft. Als dat het geval is dan wordt de oplossing, of een van de mogelijke oplossingen, ook daadwerkelijk gevonden door bint. Het op deze wijze vinden van een complete oplossing in één klap is voor de puzzelaar uiteraard geen interessante optie. Als deze al gebruik zou willen maken van de hulp van de computer dan is het hoogstens voor het vinden van één nieuw vakje en dan natuurlijk alleen voor het geval hem dit zelf te veel tijd gaat kosten. Zo'n hulpje (een 1-vakjevuller), vraagt dus om een rekenmethode waarmee de oplossing van de Sudoku stap voor stap, vakje voor vakje, wordt bepaald.
Dergelijke methoden zijn in diverse uitvoeringen en smaken te vinden op internet, ook enkele implementaties in Matlab. Als startpunt voor mijn eigen implementatie heb ik gekozen voor de methode zoals die is beschreven door Edward Markovich[3]. Zijn aanpak volgt in grote lijnen de strategie van de gemiddelde puzzelaar.

In mijn Matlab-versie heb ik geen gebruik gemaakt van de door Markovich beschikbaar gestelde Matlab-functies, maar heb ik een eigen vertaling gemaakt van zijn ideeën. Volstaan is met het toepassen van de stap-voor-stap techniek, die alleen voor de niet al te moeilijke Sudoku's geschikt is: de lastiger Sudoku's kunnen hiermee NIET opgelost worden.

> Markovich geeft aan hoe met behulp van backtracking (gekoppeld aan de stap voor stap techniek) de lastigere Sudoku's zijn op te lossen. Dat procédé wordt toegepast wanneer op zeker moment er van de resterende vakjes geen is waarvoor het in te vullen cijfer valt vast te stellen. Voor het vakje waarvoor het minste aantal mogelijkheden openstaat wordt dan een keuze (een van de mogelijke cijfers) gemaakt. Daarna wordt de Sudoku weer stap voor stap ingevuld. Als zich vervolgens, na het invullen van een of meer nieuwe vakjes, problemen voordoen (de Sudoku-regels voor rijen, kolommen of 3*3-vierkantjes worden geschonden) dan moet de eerder gemaakte keuze worden herzien. Met een nieuwe keuze wordt dan het bovengenoemde proces opnieuw uitgevoerd. Deze opzet – waarbij de stap-voor-stap methode als onderdeel wordt gebruikt - kan eventueel ook 'met de hand' worden uitgevoerd.

---

[1] PHP (PHP: Hypertext Preprocessor) is een scripttaal, die bedoeld is om op webservers dynamische webpagina's te creëren. PHP is in 1994 ontworpen door Rasmus Lerdorf, een senior softwareontwikkelaar bij IBM. Lerdorf gebruikte Perl als inspiratie. Bron: Wikipedia

[2] MATLAB (oorspronkelijk *MATrix LABoratory*) is een technische softwareomgeving uitgegeven door The Mathworks en wordt gebruikt in zowel de industrie als de academische wereld voor allerhande wiskundige toepassingen zoals het berekenen van functies, bewerken van matrices, statistiek, tekenen van grafieken, schrijven en implementeren van algoritmen en het maken van grafische gebruikersinterfaces. De basis van het programma is de programmeertaal 'M-code' of 'M'. Deze wordt gebruikt voor het invoeren, bewerken en uitvoeren van gegevens. Bron: Wikipedia

[3] zie https://sites.google.com/site/edmarkovich2/matlab_doku

In de volgende globale beschrijving duiken de volgende termen op:
- rij : reeks van 9 opeenvolgende horizontale vakjes waarin de cijfers 1 t/m 9 moeten voorkomen
- kolom: reeks van 9 opeenvolgende verticale vakjes waarin de cijfers 1 t/m 9 moeten voorkomen
- quad : reeks van vakjes in de 3*3 vierkantjes waarin de cijfers 1 t/m 9 moeten voorkomen

De stap-voor-stap methode bestaat uit een aantal basistechnieken, door Markovich scans genoemd. In de zo genoemde **quad_scan** wordt voor elk van de rijen, kolommen en quads een lijstje gemaakt van de in deze rij, kolom of quad ontbrekende cijfers en de vakjes waarin deze geplaatst moeten worden (zij het dat op dit moment nog niet duidelijk is welk ontbrekend cijfer in welk leeg vakje). Neem als voorbeeld een quad met de vulling:

1 - 3
2- -
5 6 7

Hier ontbreken dus de cijfers 4, 8 en 9. De 4, bijvoorbeeld, moet in een van de drie lege vakjes komen. Als nu in een onderliggende quad een 4 voorkomt in kolom 2 dan moet de 4 in de onderzochte quad dus geplaatst worden in kolom 3. Omdat daar maar één vakje leeg is betekent dit dat de vier in dat ene vakje geplaatst moet worden. Het recept is dus: als we voor een ontbrekend cijfer in een bepaalde quad het aantal mogelijke vakjes – door het inspecteren van rijen, kolommen en andere quads – kunnen terugbrengen tot één dan moet dat ene vakje het ontbrekende cijfer bevatten. Het invullen van het cijfer in het vakje kan tot gevolg hebben dat er nu voor een ander vakje het noodzakelijk te plaatsen cijfer kan worden vastgesteld. Door systematisch deze techniek toe te passen op alle lege vakjes kan een serie van vullingen worden bepaald
De vakjes die aldus door de quad_scan successievelijk worden gevonden worden in de resultaten met de kleur groen voor het gevonden cijfer weergegeven; boven het Sudoku-diagram wordt de tekst: "Ingevulde vakje (groen) gevonden met methode QuadScan". Tevens wordt aangegeven hoeveel van de hierna te plaatsen vakjes gevonden zijn in dezelfde scan.

De volgende basistechniek is de **matrix_scan1** (bij Markovich matrix_scan genoemd). Hier wordt bij alle lege vakjes (van de in totaal 81 vakjes) gekeken welke cijfers niet voorkomen in de rij, de kolom en de quad waar het lege vakje deel van uitmaakt. Als dit maar 1 cijfer blijkt te zijn, dan moet dit cijfer dus geplaatst worden in het beschouwde vakje. Het invullen van het cijfer in het vakje kan tot gevolg hebben dat er nu voor een ander vakje het noodzakelijk te plaatsen cijfer kan worden vastgesteld. Door weer systematisch deze techniek toe te passen op alle lege vakjes kan een serie van vullingen worden bepaald. De vakjes die door de matrix_scan zijn gevonden worden in de resultaten met het gevonden cijfer in de kleur rood weergegeven; boven het Sudoku-diagram wordt de tekst: "Ingevulde vakje (rood) gevonden met methode MatrixScan". Ook wordt aangegeven hoeveel van de hierna te plaatsen vakjes gevonden zijn in dezelfde scan.

In de totale oplosmethode worden de genoemde basistechnieken achtereenvolgens toegepast. De serie van vakjes die door een basistechniek in een keer worden gevonden worden 'in de wachtkamer' geplaatst, waarna ze een voor een, op grond van de opdracht "Los Sudoku op: vul een vakje in", uit de wachtkamer worden gehaald en in het diagram ingevuld. Als alle vakjes uit een serie op zijn wordt een volgende basistechniek toegepast en worden de resultaten daarvan in de wachtkamer gezet. Dit proces wordt voortgezet tot alle vakjes zijn ingevuld.

Voor het oplossen van de traditionele Sudoku zijn de besproken technieken afdoende. Omdat ik in mijn Sudoku Solver ook de mogelijkheid wilde bieden de in NRC-Handelsblad gepubliceerde Sudoku-varianten op te lossen heb ik twee extra scantechnieken (**redquad_scan** en **matrix_scan2**) geïmplementeerd waarin rekening wordt gehouden met de extra gemarkeerde 3 bij 3 vierkantjes die voorkomen in de NRC-variant. Ze zijn opgezet in analogie met quad_scan en matrix-sccan. Cijfers die gevonden worden met redquad_scan worden in magenta weergegeven. Resultaten van matrix_scan2 worden in blauw aangegeven.

Wat als de methode niet werkt?

Als de hier gepresenteerde Sudoku-solver geen oplossing vindt dan kan de op blz. 1 beschreven backtracking handmatig worden uitgevoerd. Een methode die altijd werkt, dan wel vaststelt dat een gegeven Sudoku geen oplossing heeft, is de eerder genoemde binair integer programmeringmethode (zoals ik die in Matlab heb). Deze methode kan via internet gebruikt worden, maar daar komt wel heel wat bij kijken. Zie " Solving Sudoku using the NEOS optimization Solver" op http://neos-server.org/casestudies/sudoku/sudoku.html.

Gonno