Tuintopia

door Malthus

Mijn Tuintopia-oplossing maakt gebruik van een paar verschillende algoritmes. Er wordt een depth-first zoekalgoritme gebruikt om snel een goede oplossing te vinden, en er worden enkele genetische algoritmes gebruikt om deze oplossing te verbeteren. Daarnaast is er nog een derde algoritme dat willekeurig de kaarten schudt en zo een goede oplossing hoopt te vinden. Deze algoritmes worden naast elkaar uitgevoerd en delen zo nu en dan de (tot dan toe) beste oplossing, om elkaar zo verder te helpen.

Algoritmes

De DepthFirstSearchBrain is een depth-first zoekalgoritme. Het legt kaarten aan bij kaarten die al in de tuin liggen en probeert daarbij steeds de maximale score te behalen. Kaarten die op een bepaalde plaats afvallen worden soms opgenomen in een lijst met later te doorzoeken mogelijkheden. Als het algoritme de hele tuin gevuld heeft, loopt het langs de zoekboom terug totdat het nieuwe, nog niet uitgezochte mogelijkheden tegen komt. Deze mogelijkheden worden dan weer depth-first doorzocht. Om het geheugengebruik en beetje binnen de perken te houden worden er heel erg veel mogelijkheden niet onderzocht. Hierdoor is er geen garantie dat dit algoritme ook de beste oplossing vindt.

De GeneticAlgorithmBrain is een genetisch algoritme. Het is een steady state algoritme, wat betekent dat nieuwe oplossingen (kinderen) in de populatie teruggeplaatst worden, er wordt niet gewerkt met generaties. Voor de selectie wordt tournament selectie gebruikt. Er worden twee oplossingen geselecteerd, en de slechtste oplossing wordt vervangen door een kopie van de beste oplossing. Deze kopie wordt mogelijk gekruisd met een willekeurige andere oplossing, en mogelijk gemuteerd en eventueel zelft helemaal geschud. De kans op kruising is 80% per oplossing, 100%/aantal posities van kaarten in de tuin voor iedere kaart. Als een oplossing niet gekruisd is en ook niet gemuteerd, dan wordt deze geschud. De grootte van de populatie is afhankelijk van de grootte van de tuin en het aantal verschillende kaarten, en bestaat minimaal uit 100 oplossingen.

De RandomBrain probeert om een goede oplossing te vinden door willekeurig wat kaarten in de tuin te verplaatsen. Eerst wordt slechts een enkele kaart verplaatst, maar als dit lange tijd geen resultaat oplevert, dan wordt de hele tuin geschud. Als er dan nog geen verbetering is, dan wordt de tot nu toe beste oplossing gekopieerd en dient deze als nieuwe uitgangssituatie.

De NetworkBrain zou door middel van een netwerk tot een goede oplossing moeten komen, maar vanwege tijdgebrek is deze niet gefinetuned of getest (en deze wordt dus ook niet gebruikt).

Performance

Alle verschillende bronnen heb ik samengevoegd, dus zon en wind worden net zo behandeld als andere bronnen, alleen dan met een richting-component. Ook de negatieve scores worden generiek opgelost. Iedere kaart heeft een bit-set, waarin de bitjes aangeven welke bronnen ze nodig hebben, welke bronnen ze leveren, en welke bronnen strafpunten opleveren. De score van een kaart kan nu snel berekend worden door de bit-sets met elkaar te vergelijken (maar helaas kost de richting-component nog wat extra tijd).

Iedere kaart krijgt een code, bestaande uit de bronnen die deze kaart oplevert aan de buren. Deze bronnen vormen, samen met vlaggetjes die aangeven of de kaart ten zuiden en/of ten westen van de andere kaart ligt, een index. Met behulp van deze index kunnen eerder berekende scores snel opgezocht worden in de hashmap. Als een score eenmaal berekend is, hoeft deze dus niet nogmaals berekend te worden.

Tuintopia

De Tuintopia-class bevat de main-methode en kan dus gerund worden. De meegeleverde jar-file kan met het commando "*java -jar Tuintopia.jar*" gestart worden. Het programma geeft enige informatie over het verloop van de optimalisatie en schrijft om de paar seconde indien nodig de beste oplossing naar het output-bestand.

Dit Tuintopia programma probeert om multi-threaded een goede oplossing voor de gegeven tuin en kaarten te vinden. Hiervoor worden een aantal van de hierboven beschreven Brains gestart. Er wordt in

ieder geval een DepthFirstSearchBrain gestart, en volgende threads worden gevuld met GeneticAlgorithmBrains. Als er ten minste vier threads beschikbaar zijn, wordt er ook een RandomBrain gebruikt, maar deze levert niet veel resultaat op. Omdat een genetisch algoritme nooit echt klaar is, gebruikt dit Tuintopia-programma de volle vijf minuten.

Met een kleine aanpassing in Tuintopia kan dit programma ook gebruikt worden om single-threaded een oplossing te zoeken, of om een oplossing in een output-file te analyseren. Het was de bedoeling om dit door middel van command-line switches mogelijk te maken, maar door tijdgebrek is deze functionaliteit vervallen.