Treemaps voor weergave van hiërarchische data

Bruggemans Jens
Universiteit Hasselt

ABSTRACT
In deze paper wordt beschreven hoe wij voor ons project van het vak Informatie Visualisatie aan de Universiteit Hasselt een treemap geïmplementeerd hebben. Er wordt uitgelegd wat de verschillende problemen waren en hoe we deze hebben opgelost. Er wordt zowel informatie gegeven over het uiterlijk van de treemap als over de interactie.

INTRODUCTIE
Voor ons project van het vak Informatie Visualisatie aan de Universiteit Hasselt moesten we een aantal visualisaties maken om data van de inschrijvingen aan de UHasselt te visualiseren. We implementeerden dit project in Adobe Flex[1]. Een van onze vereisten was dat alle visualisaties interactief moesten zijn voor de gebruiker.

We hadden nood aan een visualisatie om een globaal overzicht te krijgen van onze data. Deze visualisatie moest flexibel genoeg zijn om ook een detailweergave te kunnen geven en moest bovenalles interactief zijn. Standaard visualisaties zoals pie- en barcharts geven wel een goed overzicht van data, maar ze worden snel onoverzichtelijk wanneer men veel verschillende items toevoegt. Wat we ook missen bij deze visualisaties is de mogelijkheid om hiërarchische data te tonen. Onze data was immers hiërarchisch op te delen: onze data bestaat uit een lange lijst van studenten die zich inschrijven aan de UHasselt. Per element hebben we verschillende velden zoals richting, middelbare richting, middelbare school, gemeente, etc. We kunnen dus onze data per veld groeperen om zo een verdeling te bekomen.

Na wat opzoekwerk gedaan te hebben was het duidelijk dat een treemap [3] een goede oplossing zou zijn voor dit probleem. We vonden een framework met een implementatie van een treemap waarop we konden verder werken, genaamd IBM ILOG Elixir [2]. In de rest van deze paper wordt uitgelegd wat een treemap net is, welke algoritmes we gebruikt hebben, waarvoor we de kleuren gebruikt hebben en er wordt afgesloten met een besluit.

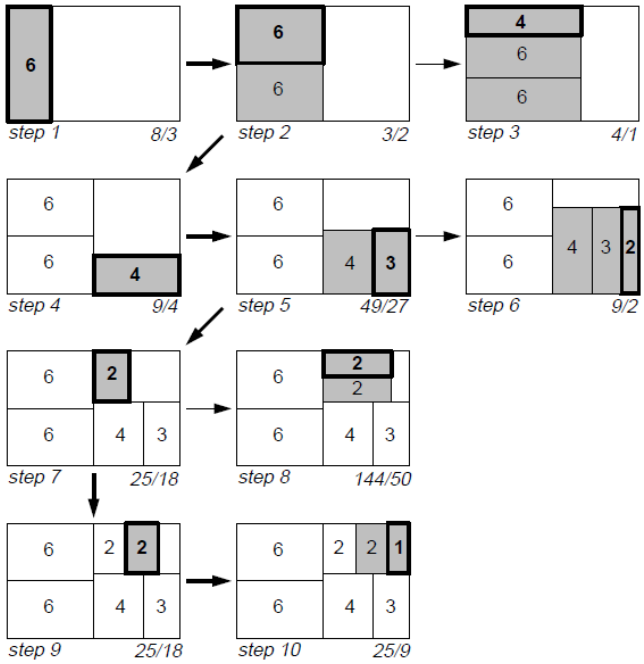
BASIS
Het basisidee van de treemap is vrij simpel. Een treemap wordt gebruikt om een boom grafisch voor te stellen. Een boom bestaat uit een wortel, knopen en bladeren. De wortel wordt voorgesteld door het venster waarin we de treemap gaan tekenen, het is dus de verzameling van alle knopen. De knopen worden voorgesteld door rechthoeken die in dit venster getekend worden. Een blad is een knoop zonder kinderen. Indien een knoop kinderen heeft wordt deze knoop op zijn beurt onderverdeeld in kleinere rechthoeken. De grootte van de rechthoeken komt overeen met het relatieve aantal elementen dat voorgesteld wordt door die rechthoek. Naast de grootte van de rechthoeken kunnen we ook de kleur gebruiken om extra informatie weer te geven.

We moesten eerst een interface voorzien die de gebruiker kon gebruiken om de treemap op te stellen. We hebben er voor gekozen om de gebruiker elk niveau waarop de gegroepeerd wordt te laten instellen. Zo kan de gebruiker bijvoorbeeld kiezen om eerst op jaartal te groeperen, daarna op gemeente, daarna op school en ten slotte op de middelbare richting. We voorzien hiervoor vier dropdownboxen. We geven de gebruiker slechts vier dropdownboxen omdat een treemap met meer dan 4 niveaus onoverzichtelijk wordt. In ons geval volstaan 3 niveaus meestal.

LAYOUT
Het spreekt voor zichzelf dat er meerdere manieren zijn om een rechthoek met kleinere rechthoeken op te vullen. In ons project hebben we 2 manieren geïmplementeerd. Als eerste hebben we het squarified algoritme [4] toegevoegd. Dit algoritme probeert de verhouding tussen de breedte en de hoogte van een rechthoek zo dicht mogelijk bij 1 te brengen (i.e. een vierkant). Dit is handig omdat de gebruiker dan beter de grootte van de verschillende rechthoeken met elkaar kan vergelijken. Het standaard algoritme genereert immers lange en dunne rechthoeken die moeilijk selecteerbaar zijn en waarvan het niet simpel is om de oppervlakte in te schatten. Figuur 1 toont de werking van dit algoritme.

Het nadeel van dit algoritme is dat alle rechthoeken gesorteerd worden op grootte. De grootste rechthoek staat immers altijd in de hoek recht tegenover de kleinste rechthoek. Dit maakt het voor de gebruikers moeilijk om een bepaalde knoop te lokaliseren en nog moeilijker om dezelfde knopen te vergelijken. Een voorbeeld van dit laatste is wanneer men eerst de treemap verdeelt in de verschillende jaren en in elk jaar de richtingen wil weergeven. Doordat sommige richtingen het ene jaar groter zijn dan het andere zullen deze van plaats wisselen waardoor de gebruiker moeite heeft om deze terug te vinden. Om er voor te zorgen dat de gebruiker de treemap ook kan gebruiken om dit soort vergelijkingen te maken, hebben we ook een tweede algoritme geïmplementeerd, het alternating algoritme.

Dit algoritme is simpeler dan het squarified algoritme. Per niveau in de treemap delen we afwisselend horizontaal en verticaal de rechthoeken op. We sorteren de rechthoeken nu niet op grootte maar alfabetisch. Figuur 2 toont duidelijk dat dit het voor de gebruikers veel simpeler maakt om de evolutie van de richtingen doorheen de jaren te vergelijken. Een leuke extra is dat dit algoritme er ook voor zorgt dat bijna alle labels getoond kunnen worden.

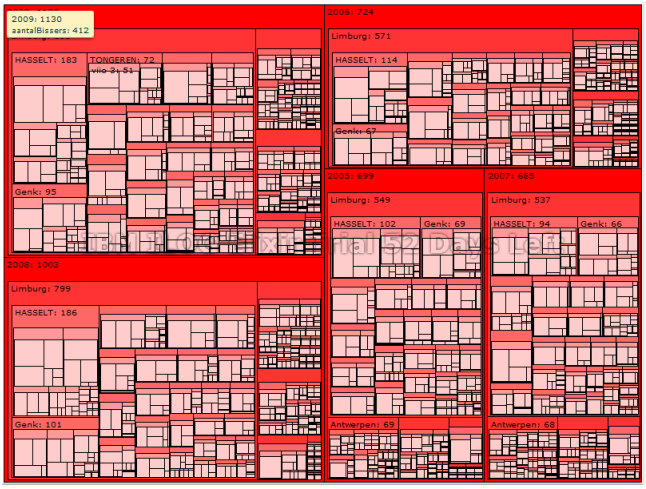


Figuur 1: Het squarified treemap algoritme.

Figuur 2: Een treemap met het alternating algoritme.

Nu we een basis versie van de treemap hebben moeten we er voor zorgen dat de gebruiker weet welke rechthoek bij welk element hoort. We plaatsten daarom de naam van de knoop in de rechthoek. Wanneer we een treemap hebben met meerdere dimensies kunnen we dit echter niet eenvoudig toepassen omdat een knoop in de boom gevuld is met zijn kinderen en de tekst dus zou overlappen. Onze oplossing hiervoor is om 20% van elke rechthoek vrij te houden voor een label. We kiezen hierbij voor een relatieve grootte omdat een vaste grootte er voor zou zorgen dat de rechthoeken vervormd zouden worden. Het spreekt voor zichzelf dat we bij de bladeren geen ruimte moeten voorzien voor een label vermits we hier over de volledige ruimte beschikken voor een label.

Door een relatieve grootte te gebruiken introduceren we een nieuw probleem: niet alle rechthoeken zullen groot genoeg zijn om een label te tonen. Om de consistentie te bewaren kiezen we er voor om in dit geval de voorziene ruimte leeg te laten, maar niet te verwijderen. Zo blijven de verhoudingen intact. We voorzien er ook voor dat de gebruiker labels kan weergeven door over een rechthoek te hoveren, zodat de treemap nog steeds bruikbaar is indien een groot deel van de labels wegvalt.



Figuur 3: Gebruik van kleuren om de diepte aan te duiden.

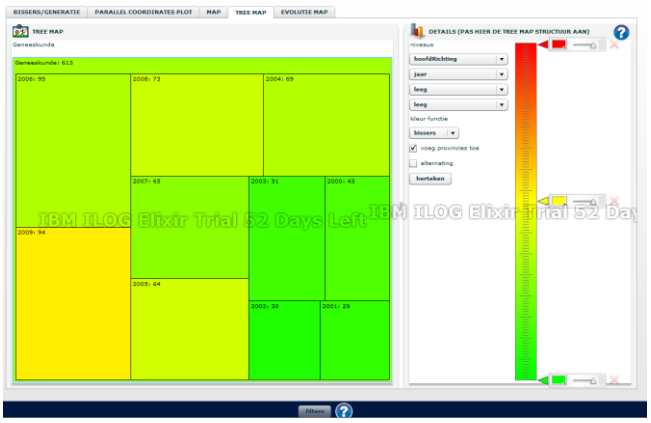
KLEUREN

Het eerste kleurenschema dat we geïmplementeerd hebben is het diepteschema, te zien in figuur 3. Dit schema geeft aan elk niveau in de treemap een steeds lichter wordende tint van rood. Rood is hiervoor een geschikte kleur omdat de zwarte tekst nog steeds goed leesbaar blijft. In combinatie met de zwarte randen en de labels krijgt de gebruiker een goed gevoel van de diepte van de treemap. Een nadeel van dit gebruik is dat men hier de kleuren al gebruikt om de diepte weer te geven, terwijl die al weergegeven wordt door de plaatsing van de rechthoeken, waardoor we dus kleur verliezen als een manier om extra informatie te tonen. Een betere manier om de diepte extra te benadrukken is het gebruik van cushion treemaps [5]. Cushion treemaps maken gebruik van een simpel shading algoritme om de hiërarchische structuur van de treemap duidelijker te maken. Het grote voordeel van cushion treemaps is dat ze de samenhang tussen nodes en dus niet alleen de diepte van nodes weergeven. Omdat het shading algoritme onafhankelijk is van de kleur, kunnen we de kleuren nog altijd gebruiken voor het weergeven van extra informatie. Vanwege de beperktheid van het door ons

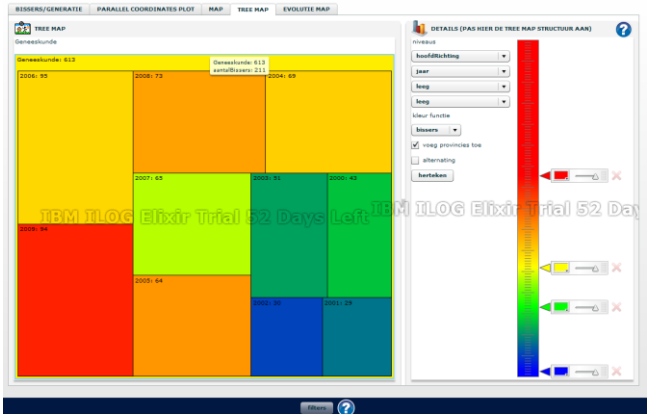
gebruikte platform hebben we dit niet kunnen implementeren. Verder hadden we twijfels over de combinatie van een cushion treemap en ons systeem om ruimte te voorzien voor labels, vermits dit ongewenste resultaten zou kunnen geven.

Het tweede kleurenschema geeft aan elke rechthoek een unieke kleur. In ons project hebben we enkel er voor gekozen om voor de geslachten een verschillende kleur te maken, zijnde blauw indien de knoop het mannelijke geslacht voorstelt en roze voor het vrouwelijke geslacht. Het is eenvoudig om dit uit te breiden naar een kleurenschema voor bijvoorbeeld de verschillende richtingen die men kan volgen aan de UHasselt. Tests hebben uitgewezen dat dit gebruik van kleuren de gebruikers enorm kan helpen bij het herkennen van de verschillende nodes.

Ons laatste kleurenschema is ook het meest geavanceerde. Aan elk van de knopen in de boom kunnen we extra data meegeven. In ons project geven wij bijvoorbeeld naast het aantal leerlingen bijvoorbeeld ook het aantal bissers in die knoop mee. We definiëren daarna een kleurfunctie die voor elke knoop wordt aangeroepen en de kleur van die knoop geeft. In deze functie gebruiken we het aantal bissers en het totaal aantal leerlingen om procentueel het aantal bissers te berekenen. Dit getal mappen we dan op een kleurmodel, een soort van gradiënt waarin we verschillende kleuren kunnen toevoegen. Ons standaard kleurmodel bestaat uit de kleuren rood, geel en groen op respectievelijk 100%, 50% en 0%. Tussen deze kleuren wordt er dan geïnterpoleerd, zo zal 75% oranje geven. We voorzien echter een mechanisme voor de gebruiker om zelf de kleuren in stellen. Hij kan een slider gebruiken om zelf kleuren toe te voegen, te verplaatsen en te verwijderen. Dit is nodig omdat ons standaard kleurmodel niet altijd even bruikbaar is. Wanneer alle waardes bijvoorbeeld tussen de 30 en de 40 procent liggen, zal er weinig verschil in kleur merkbaar zijn tussen de verschillende knopen. Het is in dit geval veel beter om groen op 30%, geel op 35% en rood op 40% te plaatsen. In figuur 4 en 5 is duidelijk te zien dat het gebruik van een kleurslider handig kan zijn om data beter te kunnen interpreteren.



Figuur 4: Een treemap met het standaard kleurmodel.



Figuur 5: De treemap met een aangepast kleurschema. Merk op dat de kleurverschillen nu veel duidelijker zijn.

INTERACTIE

Zoals er in de inleiding staat, is het belangrijk dat onze treemap interactief is. Momenteel kan de gebruiker al de structuur van de treemap en het kleurenschema aanpassen. Wanneer men echter veel niveaus instelt of een niveau instelt dat veel kleine groepen heeft (e.g. middelbare school) worden de rechthoeken te klein om nog bruikbaar te zijn. Om dit probleem op te lossen hebben we voorzien dat de gebruiker kan inzoomen op de treemap. Ons eerste plan was om het muiswiel te gebruiken om te zoomen. Naar beneden scrollen zou er voor zorgen dat men 1 niveau dieper gaat, naar waar de gebruiker zijn muis zich op dat moment bevindt. Uitzoomen zou men dan doen door naar boven te scrollen. Door bugs in het gebruikte framework konden we uitzoomen echter niet implementeren. Hierdoor hebben we er voor gekozen om de gebruiker te laten inzoomen door te klikken met de muis. Per muisklik gaat men 1 niveau dieper in de treemap. Wanneer de gebruiker nog een keer op dezelfde knoop klikt, of wanneer er volledig is ingezoomd, gaat de treemap terug naar het begin. We zorgen er ook voor dat er een animatie getoond wordt tijdens het inzoomen, zodat de gebruiker ziet wat er gebeurt en zijn oriëntatie niet verliest. Omdat de gebruiker

tijdens het inzoomen niet kan zien waar hij zich exact bevindt, voegen we boven de treemap de locatie toe.

CONCLUSIE

Al bij al waren wij heel tevreden over onze treemap. Onze mening is uiteraard niet het belangrijkste, daarom hebben wij ook usertests afgenomen. Uit onze usertests bleken een aantal gebreken. Zo gaf het verwarring dat sommige rechthoeken geen labels kregen omdat de tekst te klein was. Hierdoor had men niet meteen door dat een bepaalde rechthoek zonder label groter was dan een kleinere met label. Om dit op te lossen zouden we bij alle rechthoeken een label moeten plaatsen, of eventueel een afkorting hiervan. Men had echter de meeste problemen met het interpreteren van de kleurslider. Men wist niet wat de kleuren betekenden en men had geen idee wat de invloed was van het verslepen van de kleuren. Om dit op te lossen zouden we een markering moeten toevoegen aan de kleurslider. We hebben nu enkel streepjes, maar dit was niet genoeg. Verder vonden de gebruikers het niet handig dat men om de kleuren te wijzigen steeds op een knop moet duwen. Ook werd het inzoomen niet intuïtief gevonden door de gebruikers, hier konden we weinig aan doen door de bugs in het gebruikte framework. De gebruikers merkten de toegevoegde breadcrumbs ook niet op. Dit zouden we eventueel kunnen oplossen door deze in de bovenste rechthoek te plaatsen.

Wijzelf vinden dat onze treemap geslaagd is, en zeer bruikbaar is. Door onze usertests merkten we wel dat er wat wenning nodig is eer iedereen vlot met de treemap kan werken.

REFERENTIES

1. IBM ILOG Elixir.

   http://ftp.ilog.fr/products/ilogelixir.

2. Adobe Flex

   http://www.adobe.com/nl/products/flex.

3. B. Johnson and B. Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structures. 1991.

4. Mark Bruls, Kees Huizing, and Jarke J. van Wijk.

   Squarified Treemaps.

5. Jarke J. van Wijk and Huub van de wetering. Cushion Treemaps: Visualization of Hierarchical Information. 1999.