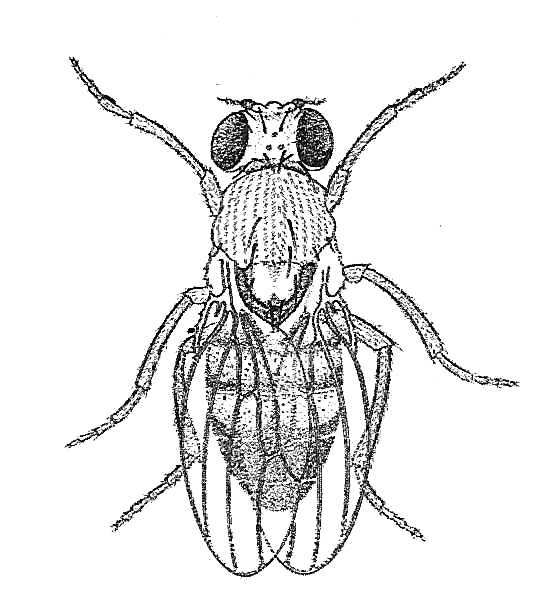
**Fruitvliegen**

Algoritmes om het ene genoom om te zetten in het andere middels inversie

Met illustratie voor Remco



Remco Blom

Hogeschool van Amsterdam

Niels Pannekeet

Hogeschool van Amsterdam

Renske Talsma

Vrije Universiteit

**1. Inleiding**

Fruitvliegen zijn kleine, tweevleugelige insecten die zich met name met suikers voeden. Het probleem in deze case heeft betrekking op twee soorten fruitvliegen, namelijk de Drosophila Melanogaster en Drosophila Miranda. Van deze fruitvliegen is veel over het genoom bekend. De twee soorten hebben identieke genen in het genoom, maar deze staan op een andere volgorde. Mutaties in het genoom van de vliegen vinden alleen plaats doordat series genen in hun geheel omkeren: inversie. In de onderstaande figuur is te zien hoe de genomen van respectievelijk Drosophila Melanogaster en Drosophila Miranda zijn opgebouwd. Voor deze case is het doel om uit te zoeken op welke manier de Drosophila Melanogaster in de Drosophila Miranda kan zijn veranderd middels mutaties, en wel in zo min mogelijk stappen en/of zo min mogelijk verplaatste genen totaal.



Figuur 1: De genomen van Drosophila Melanogaster en Drosophila Miranda

Doordat het genoom van de fruitvliegen 25 genen bevat, is het aantal mogelijke manieren waarop deze gesorteerd kunnen staan 25! ofwel 1.5×1025. Het maximale aantal inversies dat nodig is om een elke mogelijke reeks van 25 genen goed te zetten is 24.

Het minimale aantal mutaties dat nodig is voor de vlieg is 9. Dit getal is gebaseerd op de elementscore gedeeld door 2. De elementscore is een waarde die wordt gebaseerd op het aantal elementen dat zich in een genoom bevindt. Een element is ofwel een groepje aaneensluitende getallen op volgorde of precies op omgekeerde volgorde, ofwel een los getal. Hoe lager de elementscore, hoe meer getallen er aaneengesloten staan. Daarnaast is er ook gebruik gemaakt van een methode om te berekenen wat in de meest optimale situatie het mimimum aantal swaps zou zijn dat nodig is om het genoom op de goede volgorde te zetten.

Bij het omzetten van het genoom moet echter wel telkens rekening gehouden worden met het feit dat er alleen middels inversie veranderingen plaatsvinden, waardoor de minimale mutatiegrootte 2 genen bedraagt, omdat er bij een inversie altijd minstens 2 genen betrokken zijn. Op elk gegeven moment kan een inversie op 300 verschillende manieren worden uitgevoerd. Dit brengt het aantal mogelijke manieren om een genoom op de goede volgorde te zetten op 30024 ofwel 2.82×1059, wat uiteraard veel te veel mogelijkheden zijn om door te rekenen.

Hoewel er in dit kader van genomen en mutaties wordt gesproken, is er uiteindelijk natuurlijk sprake van een sorteringsalgoritme. Mogelijk kan die hieruit gevonden kennis ook worden gebruikt om andere sequenties van data te sorteren middels inversie o

**2. Methodes**

**Elk algoritme beschrijven zodat het repliceerbaar is? Uiteindelijk ook tabel met stats toevoegen denk ik maar pas bij resultaten?!!**

Er zijn voor dit onderzoek in totaal 8 verschillende algoritmes gebruikt. Omdat pas bij de laatste algoritmes echt sprake was van de gewenste resultaten worden de eerste versies slechts kort behandeld. De gebruikte algoritmes zijn Find and Swap, Find and Swap Reverse, Find and Swap Iterative, ChunkSwap, B-Star, BeamSearch en ElementShrink, Daarnaast is er middels statistisch onderzoek geprobeerd om te achterhalen welke typen genomen zich makkelijk of juist moeilijk op volgorde laten zetten. Met deze kennis zijn de algoritmes vervolgens telkens verbeterd.

**2.1 Find and Swap en variaties**

Het Find and Swap algoritme zoekt in de rij met genen naar het eerste nummer in de reeks, in dit geval 1, en zet dit op de juiste plaats door de genen tot en met dit getal in hun geheel om te draaien. Variaties op dit algoritme, Find and Swap Reverse en Find and Swap iterative voeren in de basis dezelfde handling uit, alleen begint de Reversed versie met het goedzetten van 25 en werkt vanuit daar naar 1, van rechts naar links. De Iterative versie zet ofwel eerst 1 of 25 goed, afhankelijk van wat de minste verplaatste genen vereist, en gaat van daaruit verder met de swap die vervolgens de minste genen kost om een gen aan één van de uiteinden goed te zetten. Zo itereert hij dus de hele tijd tussen het ene uiteinde van het genoom en het andere uiteinde van het genoom en werkt het algoritme van buiten naar binnen.

**2.2 ChunkSwap**

ChunkSwap is gebaseerd op hetzelfde principe als FindAndSwap, maar met een toegevoegde functionaliteit die ervoor zorgt dat “chunks” met aaneensluitende getallen die al op volgorde staan niet uit elkaar werden gehaald, maar behouden bleven. Dit algoritme was de eerste stap naar een echte heuristiek die later de basis zou vormen voor onze volgende algoritmes.

**2.3 B-Star en B-Star Elementshrink**

Het B-Star algoritme en de ElementShrinkversie maken beide gebruik van de elementscore. De elementscore is een waarde die wordt gebaseerd op het aantal elementen dat zich in een genoom bevindt. Een element is ofwel een groepje aaneensluitende getallen op volgorde of precies op omgekeerde volgorde, ofwel een los getal. Hoe lager de elementscore, hoe meer getallen er aaneengesloten staan. Daarnaast is er ook gebruik gemaakt van een methode om te berekenen wat in de meest optimale situatie het mimimum aantal swaps zou zijn dat nodig is om het genoom op de goede volgorde te zetten.

**2.4 BeamSearch**

BeamSearch is een algoritme dat vooral depth-first te werk gaat. Het algoritme kijkt drie lagen diep “vooruit” en kiest dan de best mogelijke inversies uit op basis van een scorefunctie die eveneens gebaseerd is op de elementscore.

**3. Resultaten**

Ons allereerste algoritme uit de serie van Find and Swap algoritmes leverde ons voor de vlieg een minimum van 18 inversies op. Verbeteringen en aanpassingen aan dit algoritme brachten dat voor de vlieg tot XX inversies, maar over de gehele lijn werd met random genomen geen significant verschil in performance waargenomen tussen de verschillende algoritmes. Ook de totale hoeveelheid verplaatste genen en de gemiddelde grootte van de inversies verschilde voor de random gegenereerde genomen niet significant. (BEWIJS)

Met ChunkSwap werd voor het eerst echte vooruitgang geboekt, omdat dit algoritme in tegenstelling tot zijn voorgangers tracht “chunks” van aaneensluitende getallen intact te houden, waardoor er uiteindelijk minder mutaties nodig waren om zowel het genoom van de vlieg om te zetten (klopt dit weet het niet zeker meer).

Op het totaal aantal verplaatste genen en de gemiddelde grootte van swaps bleek dit algoritme echter niet significant beter te scoren dan zijn voorgangers. (BEWIJS) Bij het 2000 maal draaien van het algoritme op random gegenereerde genomen bleek het over de gehele lijn ook niet beter te scoren dan de eerdere Find and Swap versies. Toch leek kijken naar het behouden van getallen “chunks” een goed pad om in te slaan, omdat dit bij de vlieg wel tot een verlaging van het aantal mutaties had geleid.

De mogelijke relevantie van chunks getallen werd ook door de statistiek ondersteund, na het meegeven van een score aan genomen op basis van het aantal getallen dat zich in een chunk bevond, de elementscore, bleek dat er een positieve correlatie aanwezig was tussen de elementscore van een een genoom en het aantal mutaties dat nodig zou zijn om dat genoom met bepaalde algoritmes goed te zetten. (*r* = ,467\*\*) Omdat de elementscore lager is voor genomen met meer chunks aaneensluitende getallen, betekent dat dus dat genomen met minder elementen, en dus meer chunks, over het algemeen sneller goed te zetten zouden moeten zijn.

Met deze kennis in pacht is vervolgens getracht een variantie op een A\* algoritme te bouwen. Dit algoritme draagt de naam B\* en maakt eveneens gebruik van de elementscore.

*Hier komt een hele mooie tabel met correlaties en gemiddelden en shit*

**4. Conclusie**