*‘Space Freight’*

**1 Introductie**

Het optimaal willen inladen van transportvoertuigen met lading van variërende massa en volume levert complexe problemen op die uitvoerig bestudeerd zijn om hun evidente praktische toepasbaarheid en theoretische diepgang. In deze paper wordt een onderzoek beschreven waarin wij getracht hebben een dergelijk probleem op te lossen.

De behandelde case, *Space Freight*, is in haar algemene vorm een variant van de *bin packing problem.* In een bin packing problem wordt een lijst objecten van bijvoorbeeld variërende volume en massa verdeeld over een eindig aantal bakken van verschillende capaciteiten zodanig dat de hoeveelheid gebruikte bakken of overgebleven ruimte geminimaliseerd wordt. De complexiteitsgraad van dit probleem is NP-hard, wat inhoudt dat het minstens zo moeilijk is als ieder probleem in de complexiteitsklasse NP.

Een simpel algoritme plaatst de objecten (zonder de lijst te ordenen) in de eerste bak waarin hij past, dit is het zogeheten first-fit algoritme. Dit algoritme levert vaak snelle doch suboptimale resultaten op. Door de lijst van objecten te sorteren op basis van een doordachte heuristiek wordt het first-fit algoritme vele malen effectiever. Het *first-fit decreasing algoritme* ordent de lijst van grootste massa of volume naar de kleinste en plaatst vervolgens elk achtereenvolgend element in de eerste bak waarin hij past. D. Johnson bewees in 1973 dat deze heuristiek nooit meer dan 22% afwijkt van de optimale oplossing, bovendien bewees hij ook dat geen efficiënt bin packing algoritme gegarandeerd minder dan 22% kan afwijken van het optimum (Hoffman 1998, p.172)

Verder is ook bekend dat er altijd minstens één ordening van de objectenlijst bestaat waarmee het first-fit algoritme een optimale oplossing vindt.

Echter is het vinden van een optimale rangschikking van elementen voor het first-fit algoritme door de grote omvang van de state-space van een probleem nagenoeg onmogelijk. Hetzelfde geldt voor onze case, Space Freight.  
De probleemstelling luidt als volgt: ruimtevaartorganisaties uit vier landen willen een cargolijst anonimiseren en zodanig verdelen over hun ruimteschepen van variërende capaciteiten (kilogram en kubieke meter) dat de hoeveelheid onbenutte ruimte aan boord geminimaliseerd wordt. \*\*\* moet ik ook gedetailleerd vertellen wat onderdelen a t/m e inhouden? \*\*\*

De moeilijkheidsgraad van dit probleem ligt in de grote omvang van de state-space. \*\*\*wat is deze exact? Dus vooral voor de latere onderdelen \*\*\* Hierdoor is het vinden van de optimale ordening voor een first-fit algoritme ondoenlijk of zelfs onmogelijk voor de veel computers. De omvang van de state-space van dit probleem maakt het gebruik van constructieve algoritmes eveneens af te raden.

De hoofdvraag voor dit onderzoek luidt: \*\*\*hoe komt men zo dicht mogelijk bij een optimale allocatie van de cargolist over de ruimtevaartuigen zodanig dat de hoeveelheid onbenutte ruimte- en gewichtscapaciteit geminimaliseerd worden? \*\*\*

\*\*\*Wij hebben gepoogd de case en haar verschillende onderdelen op te lossen met random sampling, iteratieve en greedy algoritmes. \*\*\* Kort gezegd hebben we de resultaten vergeleken van verschillende startconfiguraties van de cargolijst waarop vervolgens verschillende soorten iteratieve algoritmes zijn losgelaten. De startconfiguraties werden opgesteld door een first-fit decreasing algoritme (greedy) of random sampling. De iteratieve algoritmes waren een hillclimber en een simulated annealing algoritme met meerderen variaties in de specificaties van deze algoritmes. In het onderdeel Methodiek van deze paper wordt onze aanpak gedetailleerd omschreven.

**2 Materiaal en methodiek**

**2.1 Materiaal**

*Data*

Alle data die voor dit onderzoek is gebruikt is afkomstig van het CargoLists.zip bestand dat ons is toegereikt door onze opdrachtgever. \*\*\* Zie appendix? \*\*\* Na het unzippen van dit bestand bevat het een folder met daarin drie cargolijsten. Elke cargolijst heeft dezelfde structuur, ze verschillen enkel in omvang. Elke lijst is een .txt-bestand met daarin drie kolommen. Kolom één is de unieke code van een element in het formaat ‘CLi#j’ (waarbij i = 1,2,3 en j ∈ [1,1250]. De tweede kolom is de bijbehorende massa van het element in kilogram. En de derde kolom is het bijbehorend volume in kubieke meter. De lengtes van cargolijsten één, twee en drie zijn respectievelijk 99, 98 en 1250 elementen.

*Software*

Lennert:

* **Python version** 2.7.13
* **Libraries**????
* **OS** X El Capitan (version 10.11.6)

\*\*\* Joosje misschien kan je hier iets korts zeggen over die plotly library enzo \*\*\*

*Hardware*

Lennert:

* **Apple MacBook Pro** (Retina, 13-inch, 2015)
* **Processor** 2,7 GHz Intel Core i5
* **Geheugen** 8 GB 1867 MHz DDR3

**2.2 Methodiek**

A+B+C:

Joosje: statespace, scorefunctie, greedy + random, hillclimbing 1 + 2

Lennert: annealing 1 + 2

D+E:

Nik

**3 Resultaten**

Grafieken voor cargolijst 1 + 2 en barcharts voor A + B + C: Joosje

A+B+C:

Greedy, random, hillclimbing 1+2: Joosje

Annealing 1+2: lennert

D+E:

Nik

**4 Discussie**

Samen?

**Referenties**

**[1]** Hoffman, P. [*The Man Who Loved Only Numbers: The Story of Paul Erdős and the Search for Mathematical Truth.*](http://www.amazon.com/exec/obidos/ASIN/0786863625/ref=nosim/ericstreasuretro) New York: Hyperion, 1998.