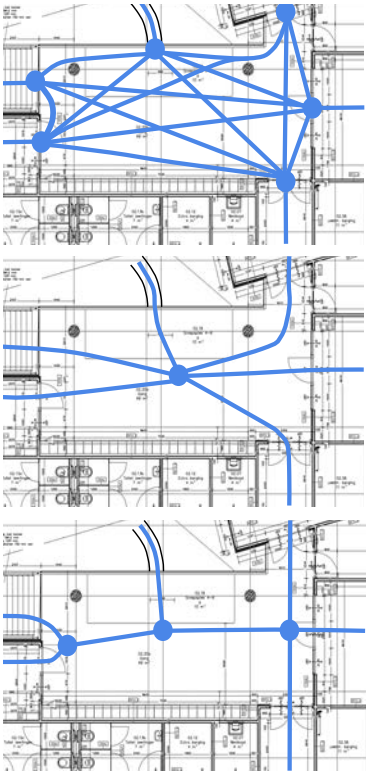
Bijlage A: Alternatieve aanpak van het dilemma tussen granulariteit en simpliciteit

Het omzetten van een plattegrond naar een graaf vereist abstrahering. Er moet een balans gevonden worden tussen granulariteit en simpliciteit. Aan de ene kant is het belangrijk dat de graaf waarheidsgetrouw is. Het systeem kan dan een goede schatting maken van de snelste route en de lengte van deze route. Hier komt bij kijken dat er meer vertices en zijden nodig zijn voor waarheidsgetrouwheid. Aan de andere kant moet de simpliciteit van de graaf gewaarborgd worden. De graaf moet handmatig getekend en ingevoerd worden. Daarnaast moeten alle zijden belopen worden. Een overvloed aan vertices en zijden zal deze processen enorm vertragen. Ook is de snelheid van het systeem belangrijk. De snelheid van Dijkstra's algoritme hangt sterk af van het aantal vertices en zijden.

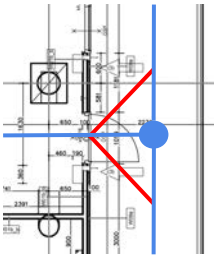


Figuur A1: verschillende maten van simplificatie in een hal

In figuur A1 is de plattegrond van een hal op de tweede verdieping te zien. Deze heeft 6 (relevante) aansluitingen: Drie lokalen, een gang, een trap en de glijbaan. Bovenaan is een compleet waarheidsgetrouwe graaf te zien waarbij elke kortste route tussen twee aansluitingen mogelijk is gemaakt met een zijde, resulterend in een groot aantal vertices en zijden. Daaronder is de hal versimpeld tot één vertex met één zijde voor elke aansluiting. In een route die over de trap, het lokaal direct ernaast in zou gaan, zou het systeem de afstand overschatten. Onderaan is een graaf weergeven met een balans tussen simpliciteit en waarheidsgetrouwheid.

In figuur A1 is een enkel geval van het dilemma tussen simpliciteit en waarheidsgetrouwheid weergeven. In werkelijkheid zijn alle vertices dergelijke *dilemmaplekken*. Elke vertex heeft namelijk kortere routes, zo kan bijvoorbeeld een kortere route gevonden worden op de voordehandliggende T-splitsing in figuur A2. Naar schatting zal de graaf uit ongeveer 100

vertices bestaan, dus zullen er ook 100 dilemmaplekken zijn. Er moet dus een duidelijke eis verzonnen worden voor het versimpelen van een dilemmaplek.

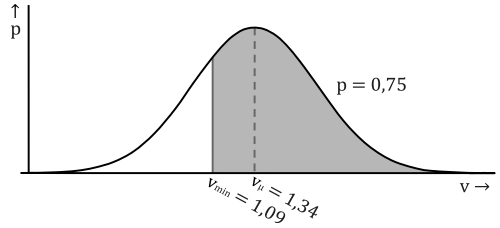


Figuur A2: dilemmaplek

Afstand d_{dp} (in m) is het verschil in afstand tussen de waarheidsgetrouwe route en de versimpelde route op een dilemmaplek, dus de overschatting van het systeem op één dilemmaplek. Er moet een maximaal verschil d_{max} (in m) worden bepaald waarbij voor alle dilemmaplekken geldt d_{dp} kleiner of gelijk moet zijn aan d_{max}.

$$d_{dp} \leq d_{max} \tag{A1}$$

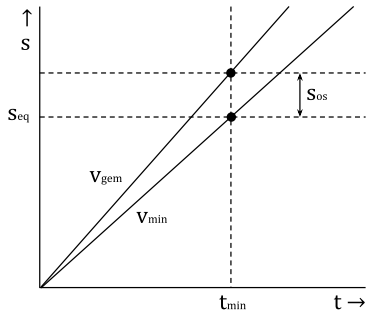
Dit maximale verschil kan gebaseerd worden op een verschil in loopsnelheden van de mens.



Figuur A3: normaalverdeling loopsnelheid

De loopsnelheid v van een mens (in m/s) is normaal verdeeld met een gemiddelde $\mu = 1,34\ m/s$ en een standaarddeviatie $\sigma = 0,37\ m/s$ (Weidmann, 1993). Wij willen dat ons navigatiesysteem een effectiviteit van 75% heeft, dit betekent dat 75% van de gebruikers op tijd komt wanneer er precies wordt vertrokken op de tijd die het systeem aangeeft. Dit resulteert in een minimale snelheid v_{min} van $1,09\ m/s$ om op tijd te komen (zie figuur A3). Echter worden de gewichten van de graaf gebaseerd op een gemiddelde snelheid v_{μ} van $1,34\ m/s$. Om nog steeds 75% effectiviteit te halen moet de equivalente afstand van de route s_{eq} (in m) overschat worden. Het systeem zal dan een langere looptijd t_{min} (in s) aangeven waardoor eerder vertrokken moet worden en een persoon met v_{min} alsnog op tijd komt.

Deze overschatting s_{os} (in m) staat gelijk aan het verschil tussen v_{min} en v_{μ}, vermenigvuldigd met t_{min}, waarbij t_{min} de tijd is die iemand met v_{min} doet over s_{eq} van een route. (zie figuur A4)



Figuur A4: visualisatie overschatting

$$v_{min} = 1,09\ m/s \tag{A2}$$

$$v_{\mu} = 1,34\ m/s \tag{A3}$$

$$t_{min} = \frac{s_{eq}}{v_{min}} \tag{A4}$$

$$s_{os} = t_{min}(v_{\mu} - v_{min}) \tag{A5}$$

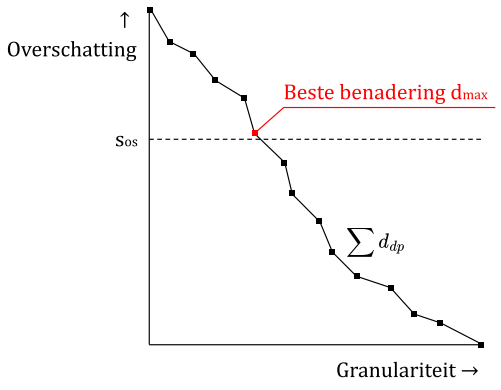
De eerder genoemde overschatting op dilemmaplekken, ontstaan door simplificatie, kan gelijkgesteld worden aan de overschatting, nodig om een effectiviteit van 75% te behalen.

Dit kan gedaan worden door de waarde van d_{max} te veranderen. Hiervoor worden formules A1, A4 en A5 samengevoegd tot de volgende formule:

$$\sum_{d_{dp} \leq d_{max}} d_{dp} = \frac{s_{eq}}{v_{min}}(v_{\mu} - v_{min}) \tag{A6}$$

Hieruit is echter nog niet direct een waarde uit d_{max} af te leiden. Daarvoor wordt het bijbehorende algoritme voorgesteld:

Neem een route r, bereken hierbij d_{os}. Simplificeer alle dilemmapunten op de route tot individuele vertices en bereken $\sum d_{dp}$. Als deze groter is dan s_{os} moet d_{max} aangescherpt worden: De bijbehorende dilemmaplek van de grootste waarde van d_{dp} moet een hogere granulariteit krijgen. Er moeten meer vertices worden toegevoegd om deze d_{dp} te verlagen. Bereken vervolgens opnieuw $\sum d_{dp}$. Herhaal dit totdat $\sum d_{dp} \leq s_{os}$. De kans dat $\sum d_{dp} = s_{os}$ is klein, dus de grootste waarde d_{dp} van de $\sum d_{dp}$ die het dichtst in de buurt komt van s_{os} is de beste benadering van d_{max}.



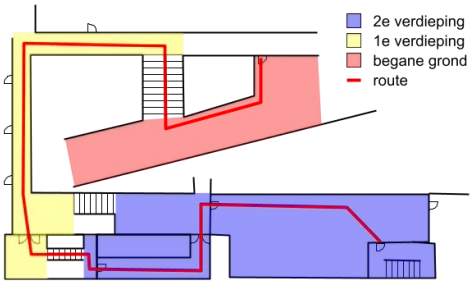
Figuur A5: visualisatie algoritme

Deze beste benadering kan fluctueren, daarom moet voor een nauwkeurige benadering van d_{max} een lange route met veel dilemmaplekken en een breed spectrum aan groottes van d_{dp}. s_{eq} zullen wij deze zelf moeten berekenen, dus moeten eerst onze eigen snelheden berekend worden op een traject van 17,1 meter:

Persoon	Tobias	Dylan
Tijd ronde 1 (in s)	14,18	12,33
Tijd ronde 2 (in s)	14,48	12,96
Tijd ronde 3 (in s)	13,53	11,60
Tijd gemiddeld (in s)	14,06	12,30
Afstand (in m)	17,1	17,1
Snelheid (in m/s)	1,216	1,391

Tabel A1: berekening eigen snelheden

De gekozen route loopt van de deur naast de conciërge op de begane grond naar de deur van het zuidelijke trappenhuis op de tweede verdieping (zie figuur A6).



Figuur A6: belopen route

Hierbij zijn de volgende gegevens verzameld:

	Tobias	Dylan
Tijd heen (in s)	113,19	115,08

Tijd terug (in s)	124,40	118,04
Tijd gemiddeld (in s)	118,975	116,56
Snelheid (in m/s)	1,216	1,391
Persoonlijke equivalente afstand (in m)	144,67	162,13
s_{eq} (in m)	153,40	

Tabel A2: Gegevens belopen route

Met $s_{eq} = 153,40\ m$ en formules 8, 9, 10 en 11 valt te berekenen dat de overschatting $s_{os} = 35,18\ m$ moet zijn.

In de gekozen route komen 9 dilemmaplekken voor, het eerder genoemde algoritme wordt hierop toegepast:

Versie	d_{dp1} (in m)	d_{dp2} (in m)	d_{dp3} (in m)	d_{dp4} (in m)	d_{dp5} (in m)	d_{dp6} (in m)	d_{dp7} (in m)	d_{dp8} (in m)	d_{dp9} (in m)	$\sum d_{dp}$ (in m)
1	9,0	8,1	6,6	5,1	4,2	3,3	0,6	7,5	14,1	58,5
2	9,0	8,1	6,6	5,1	4,2	3,3	0,6	7,5	1,8	46,2
3	2,1	8,1	6,6	5,1	4,2	3,3	0,6	7,5	1,8	39,3
4	2,1	1,5	6,6	5,1	4,2	3,3	0,6	7,5	1,8	32,7

Tabel A3: Toepassing algoritme

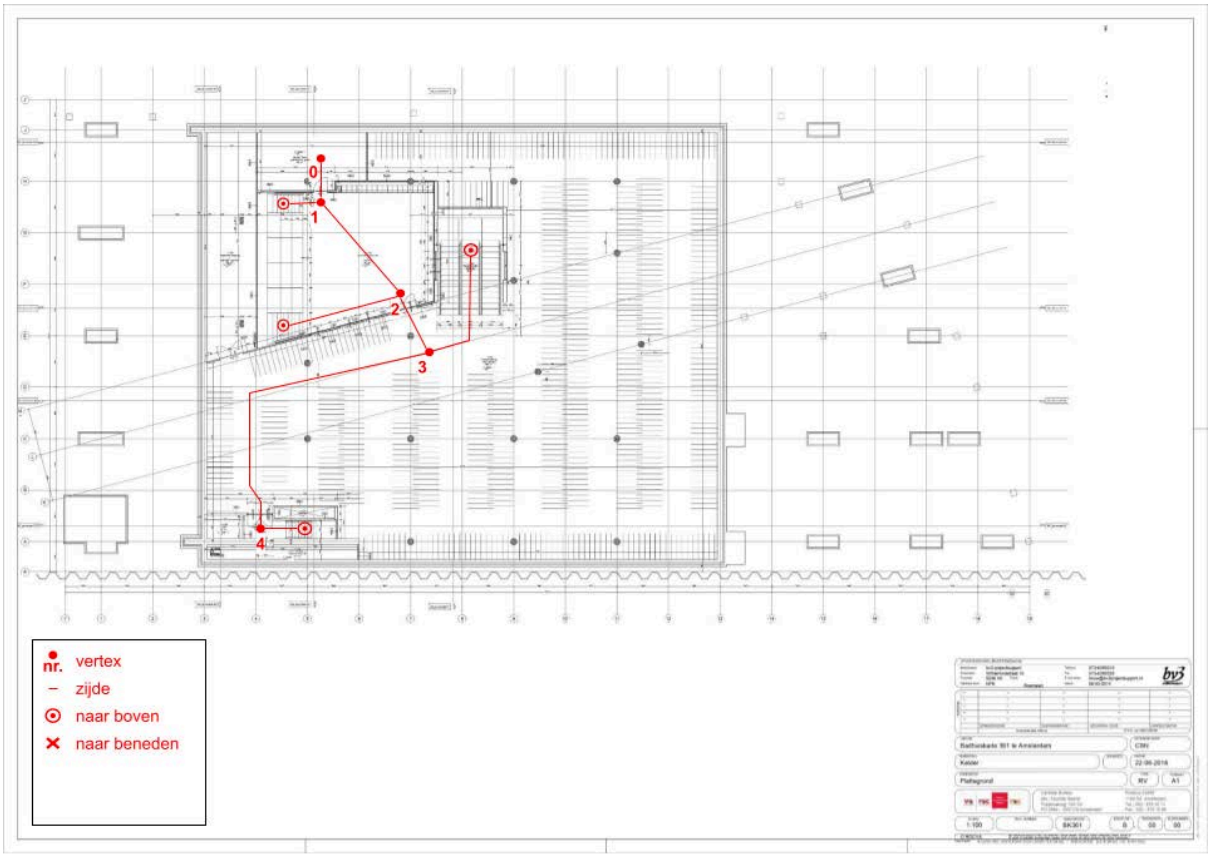
Na 4 versies te hebben gemaakt met oplopende granulariteit is de totale overschatting op dilemmaplekken kleiner geworden dan de benodigde overschatting. Men ziet dat versie 4 het dichtst in de buurt komt van deze te behalen overschatting. Dit betekent dat de grootste waarde uit versie vier kan worden aangewezen als d_{max}.

$$d_{max} = 7,5\ m \tag{A7}$$

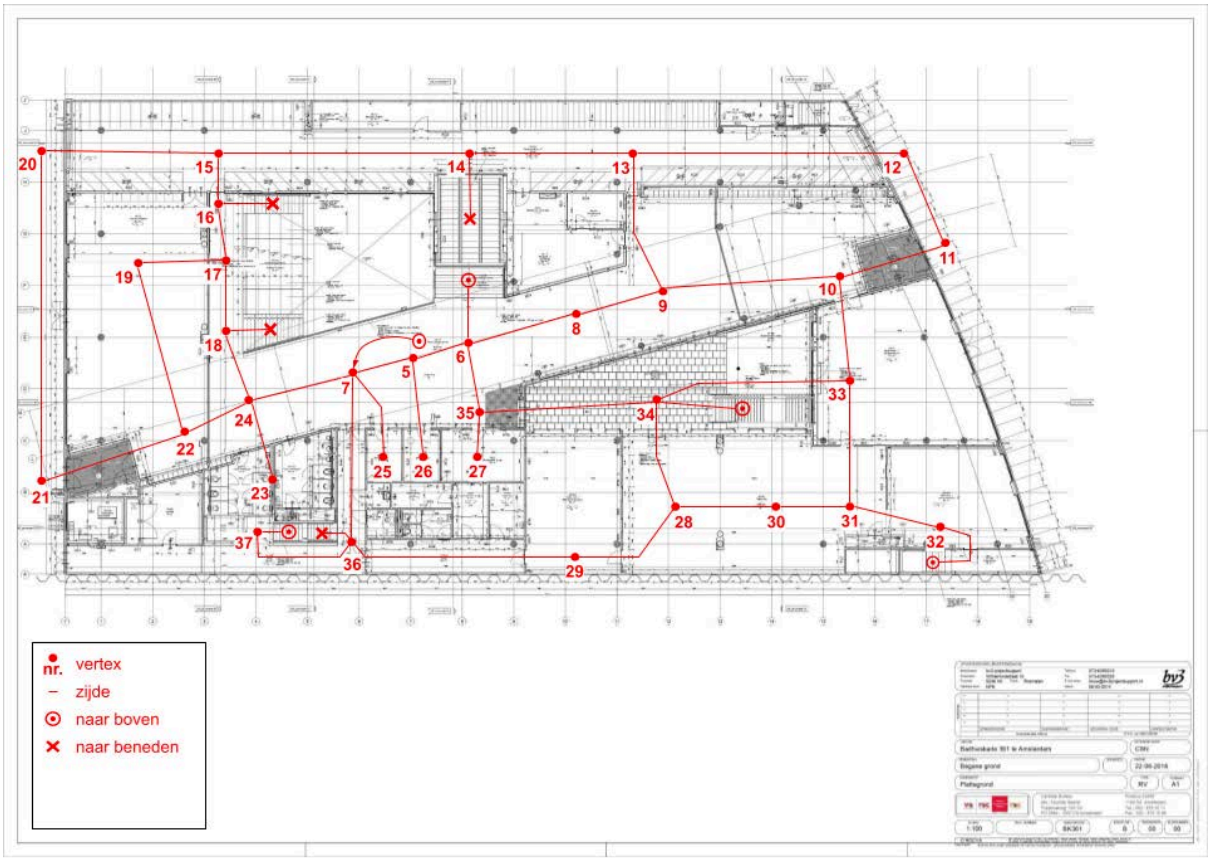
Het verschil tussen de werkelijke route en de versimpelde route mag per dilemmaplek maximaal 7,5 meter zijn om ervoor te zorgen dat (gemiddeld genomen) 75% van de gebruikers op tijd komt met de aangegeven tijden van het systeem. Deze 7,5 meter vormt dus de eis voor het versimpelen van de vertices op een dilemmaplek

Bijlage B: De (ongewogen) graaf van het Hyperion Lyceum

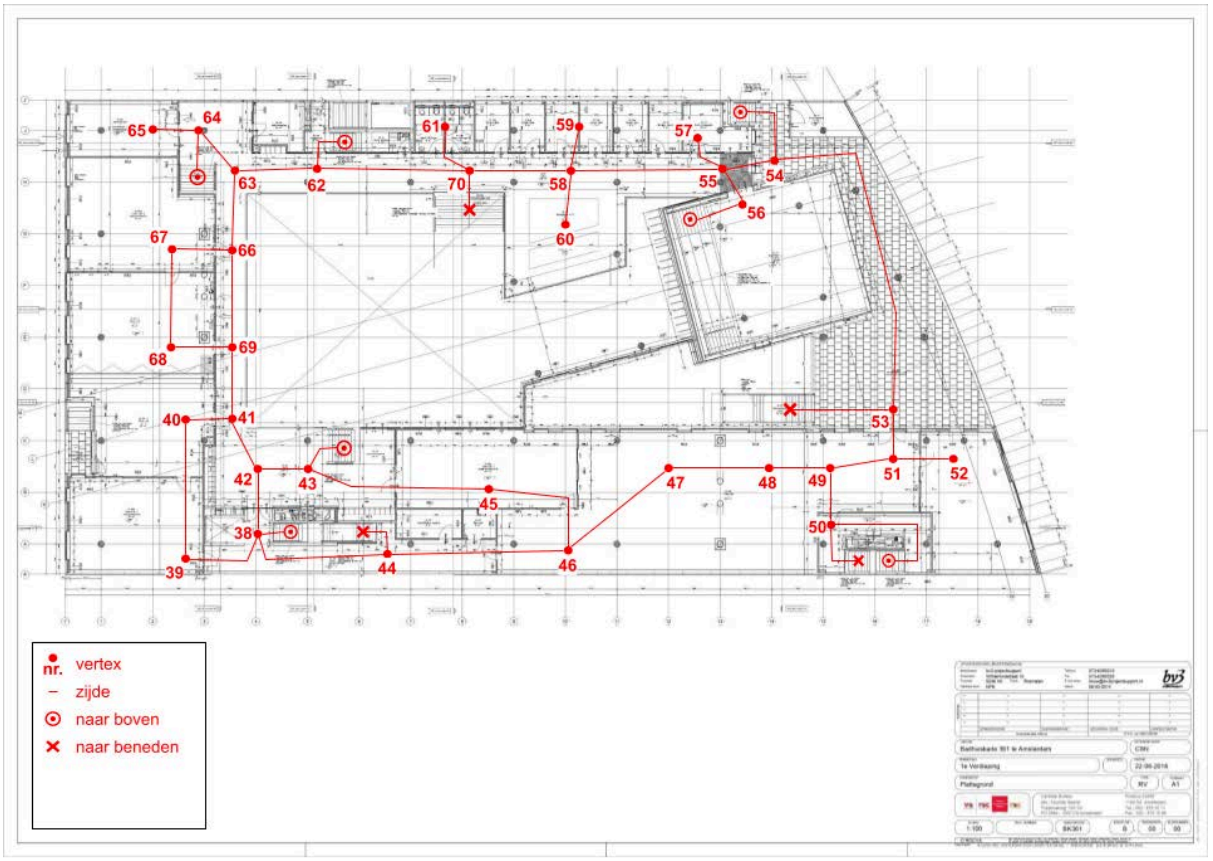
Verdieping -1:



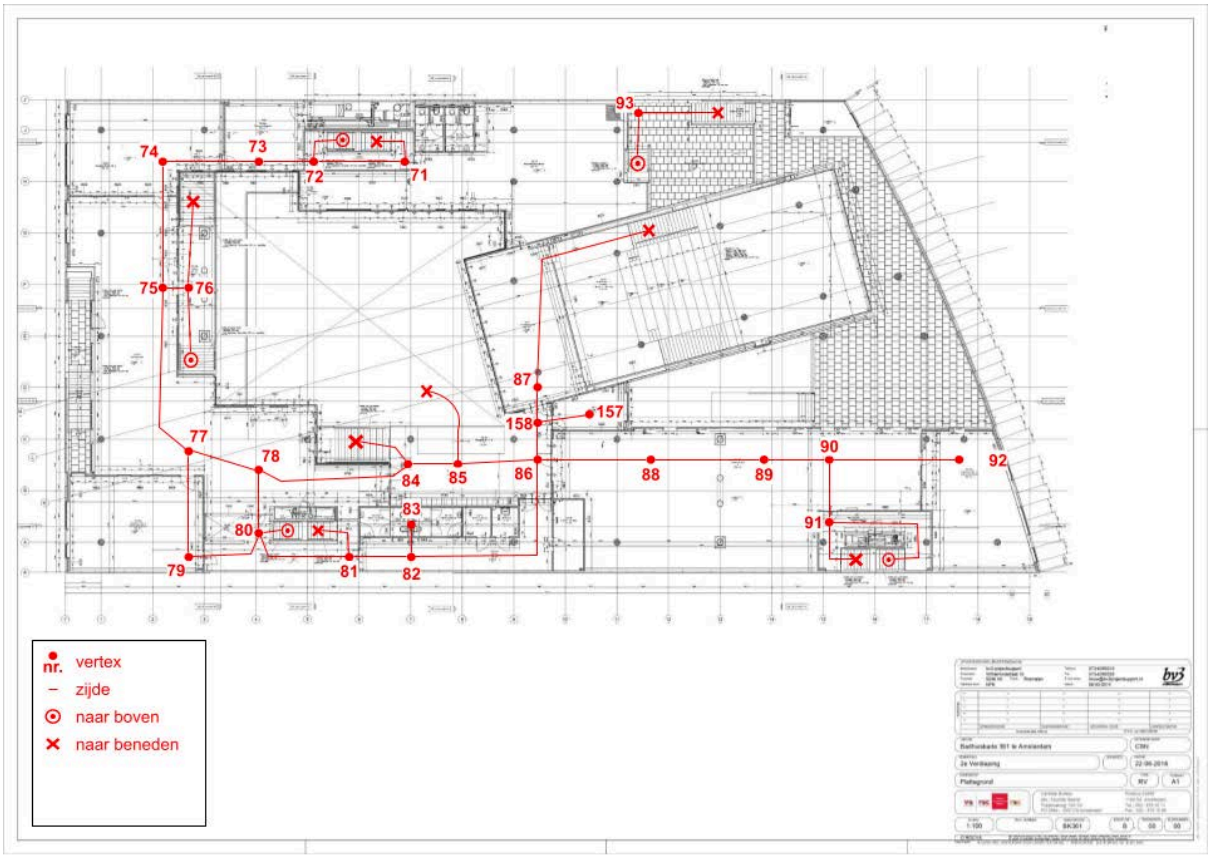
Verdieping 0:



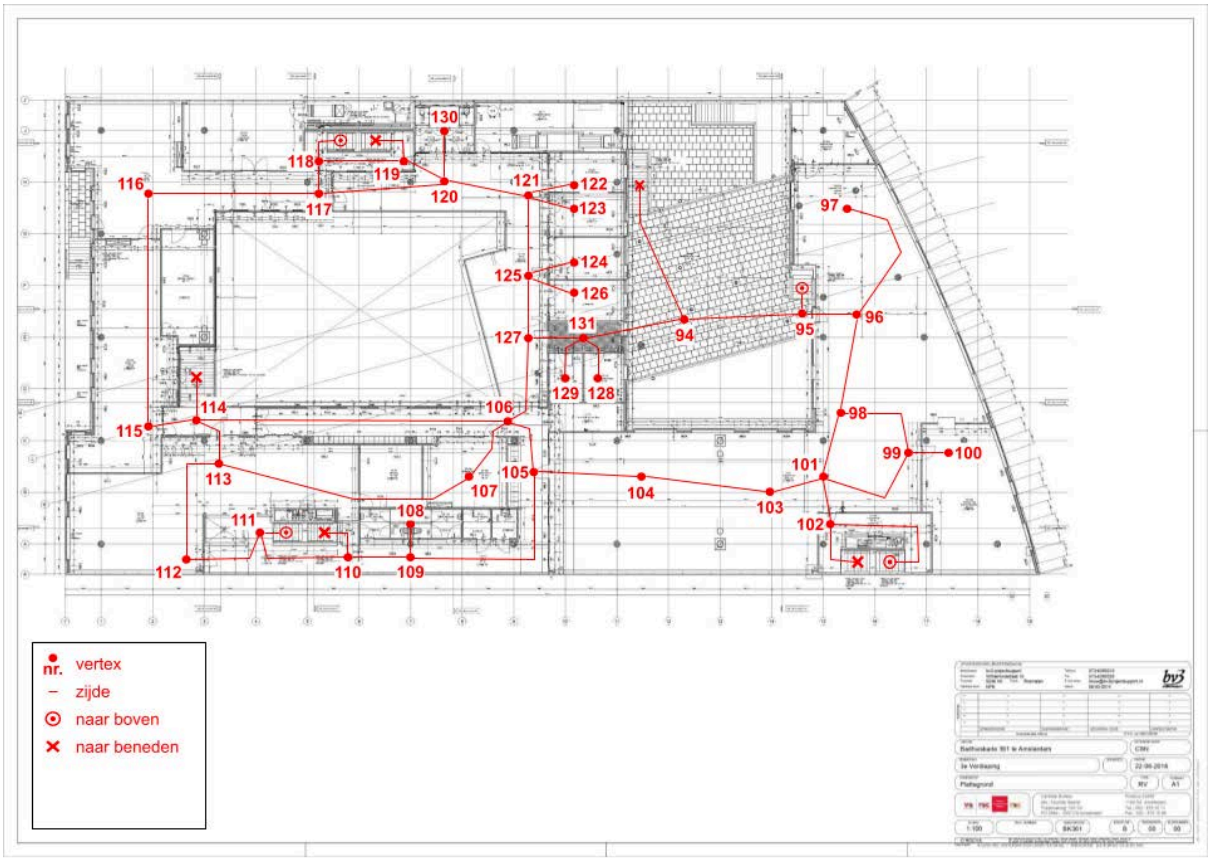
Verdieping 1:



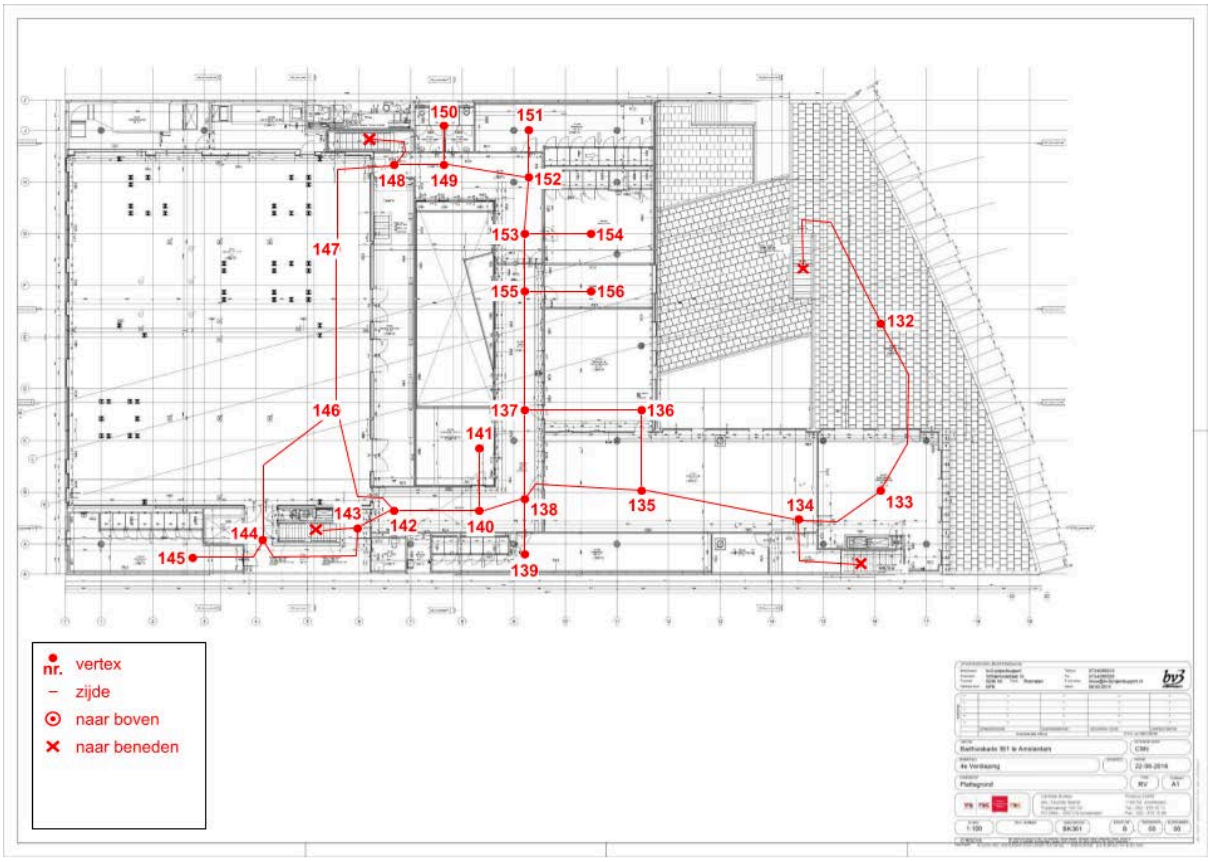
Verdieping 2:



Verdieping 3:

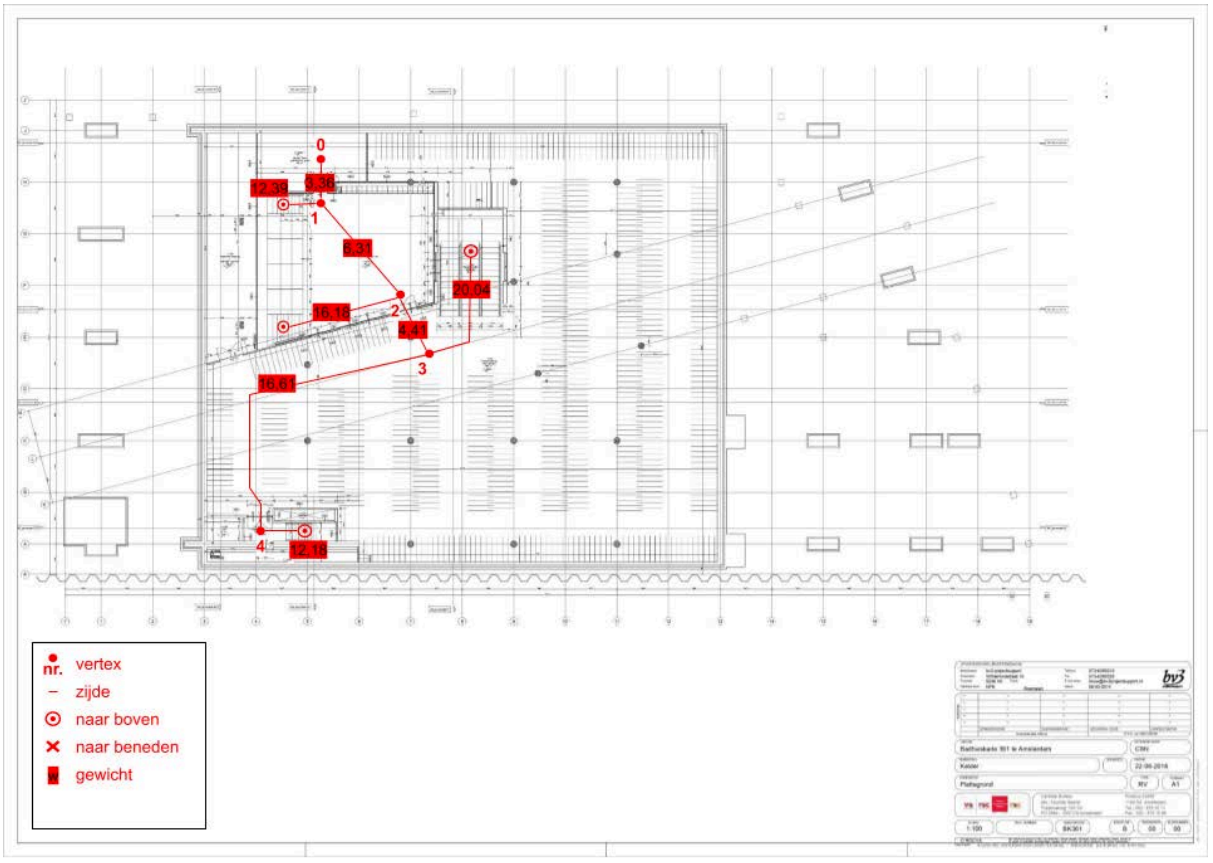


Verdieping 4:

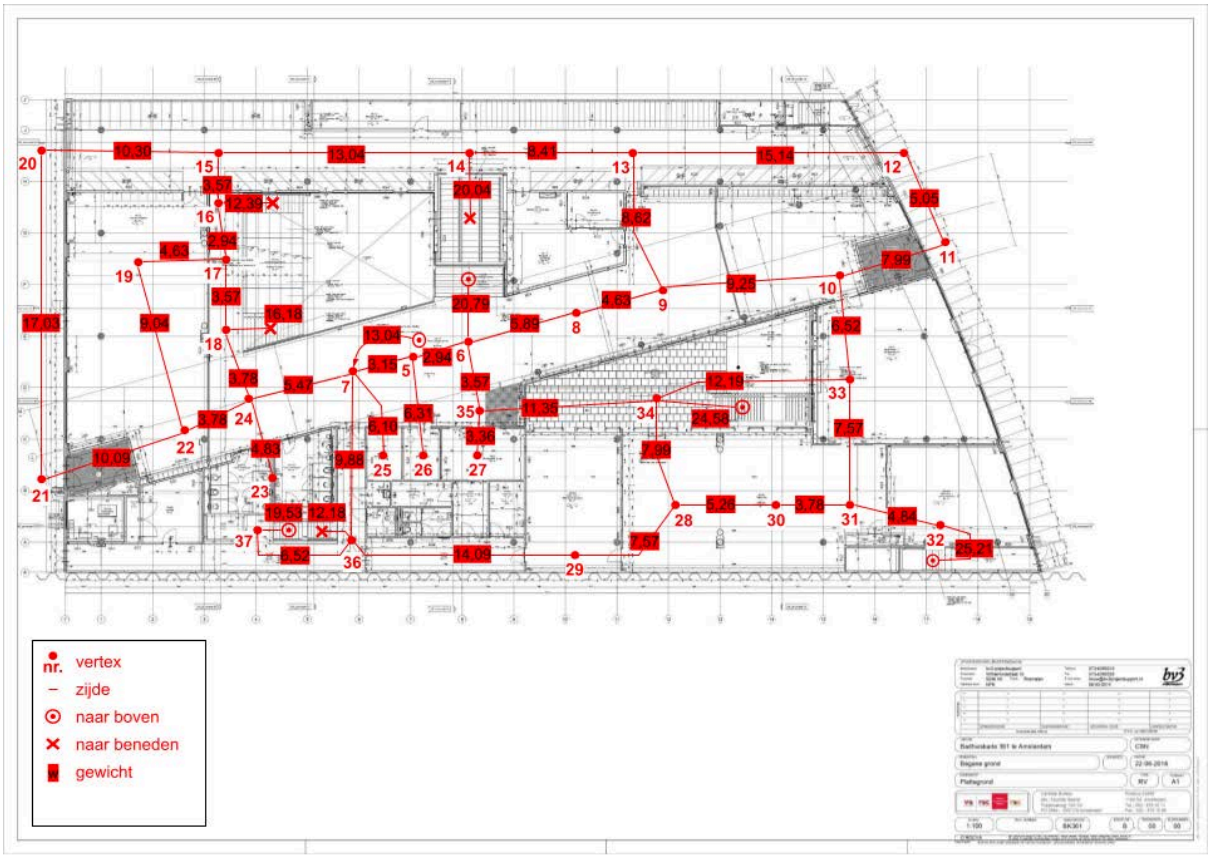


Bijlage C: De gewogen graaf van het Hyperion Lyceum

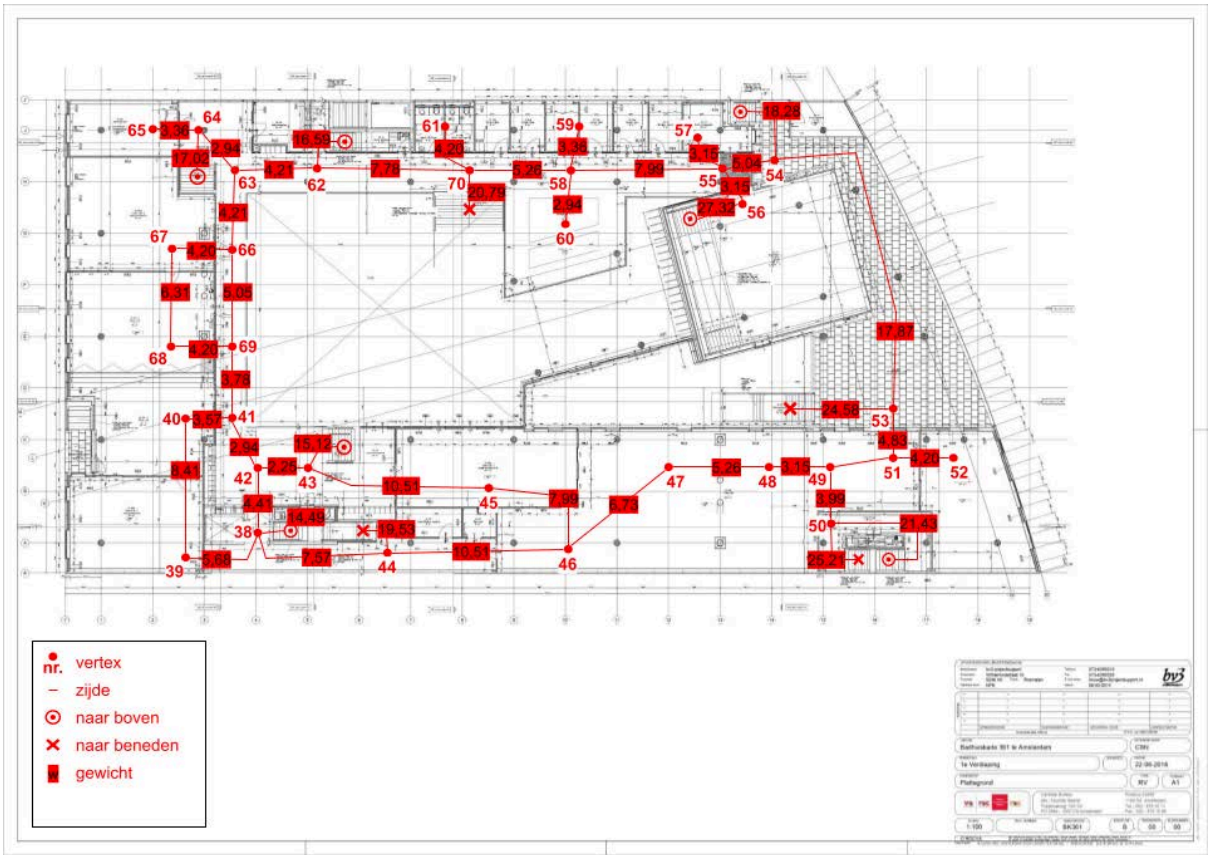
Verdieping -1:



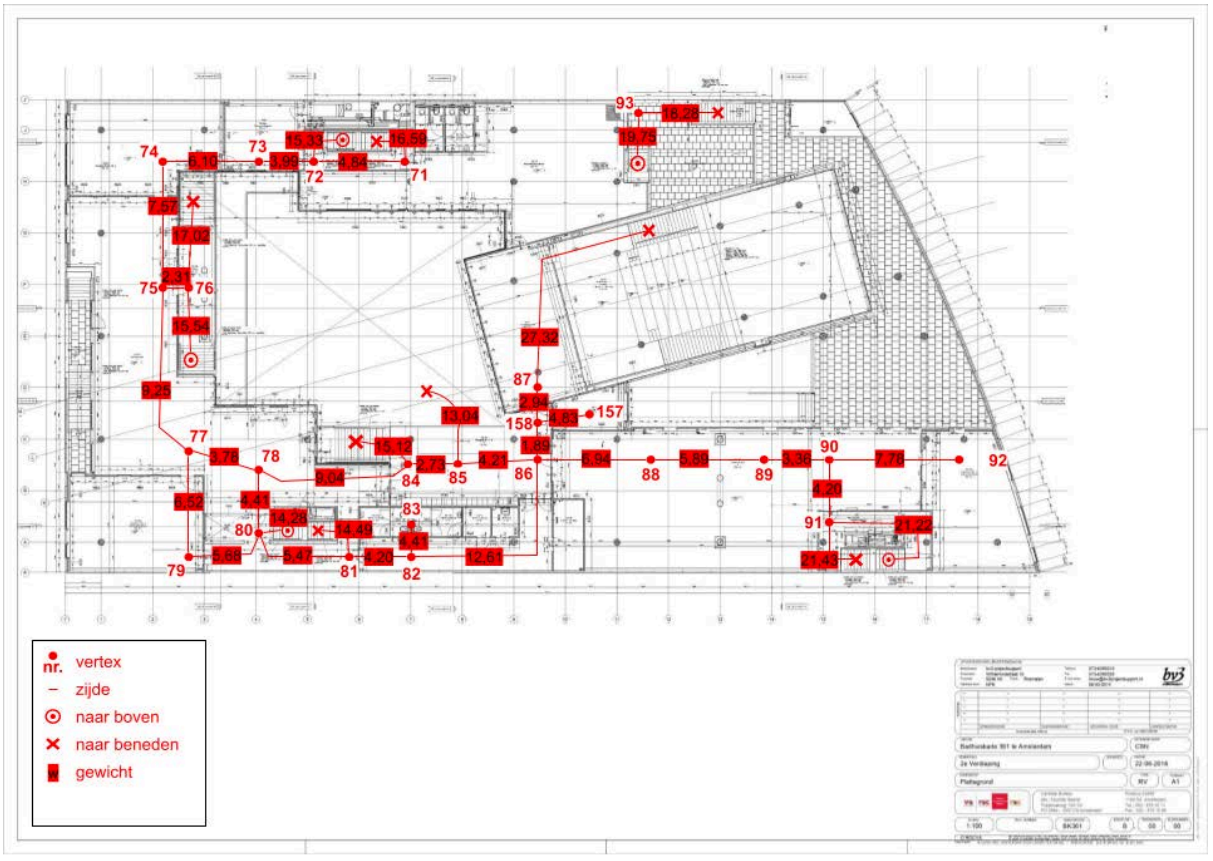
Verdieping 0:



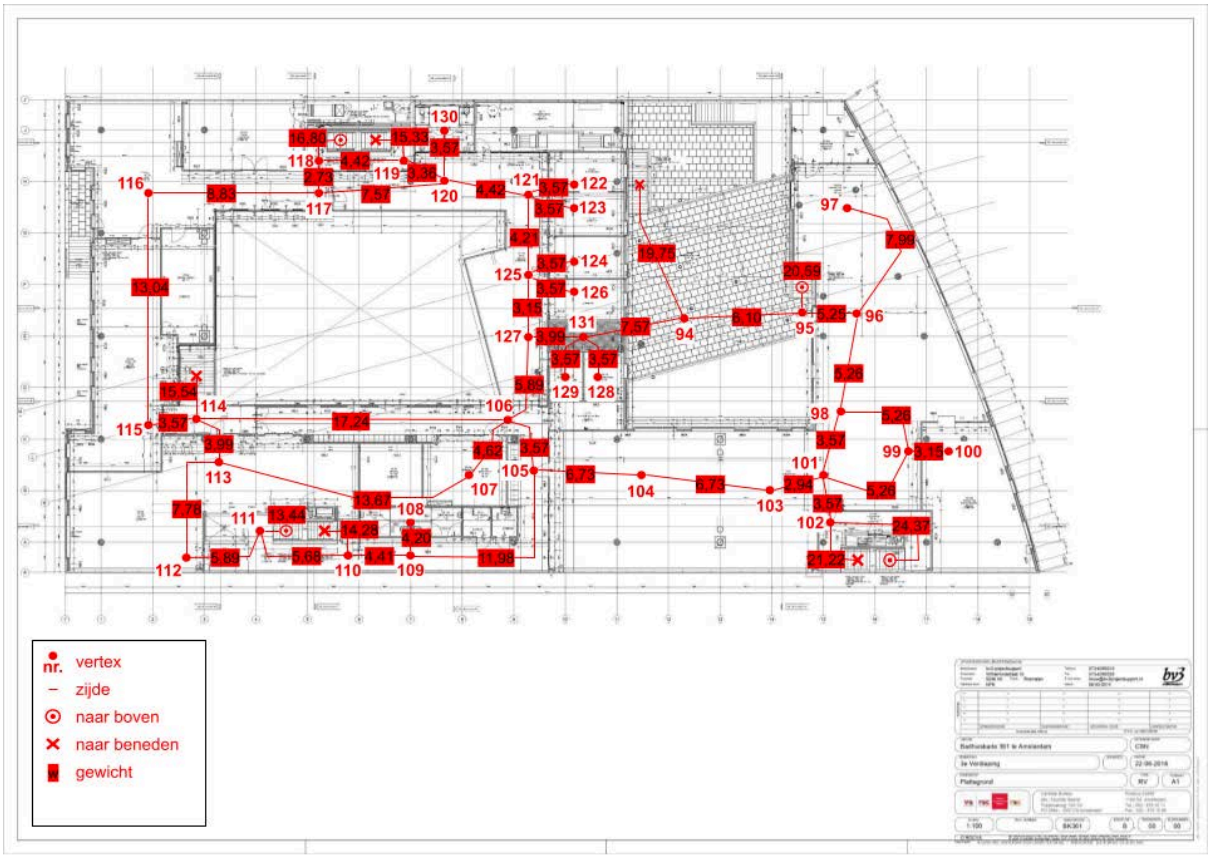
Verdieping 1:



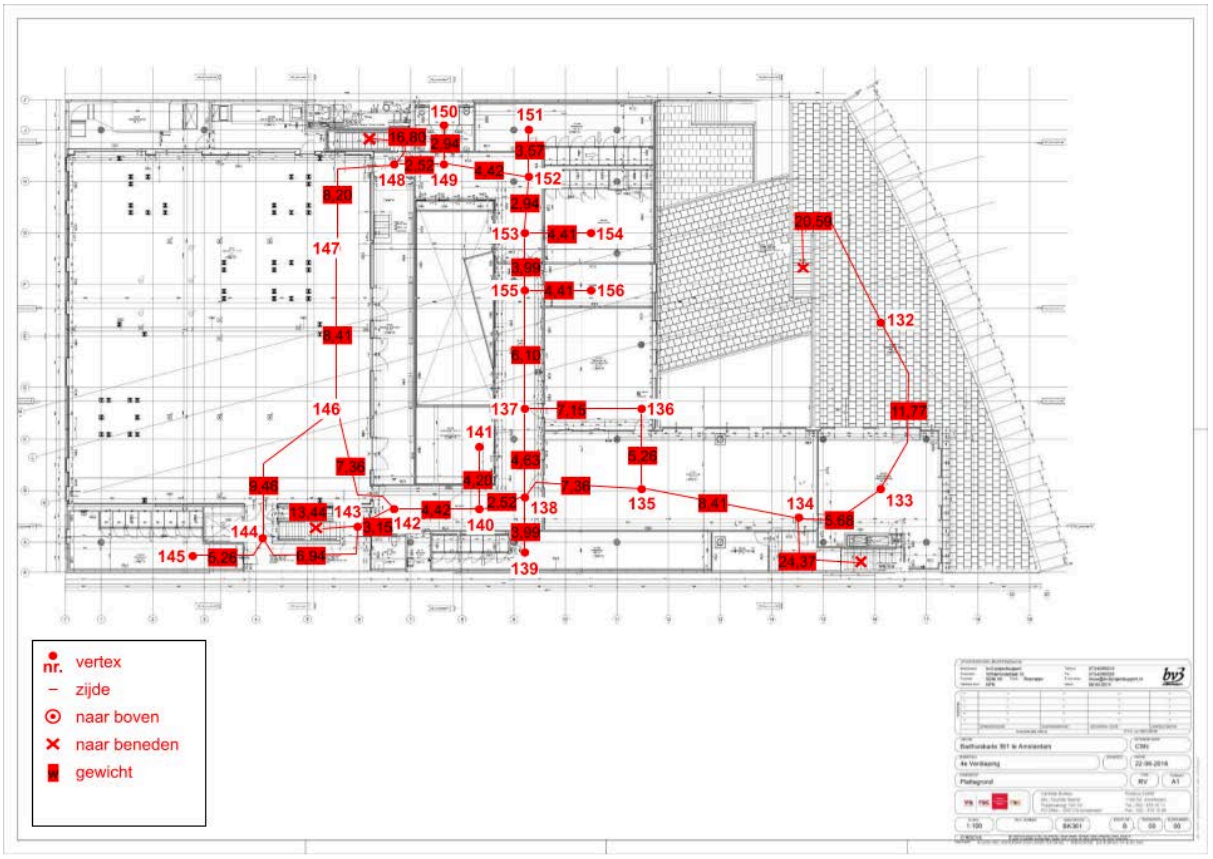
Verdieping 2:



Verdieping 3:

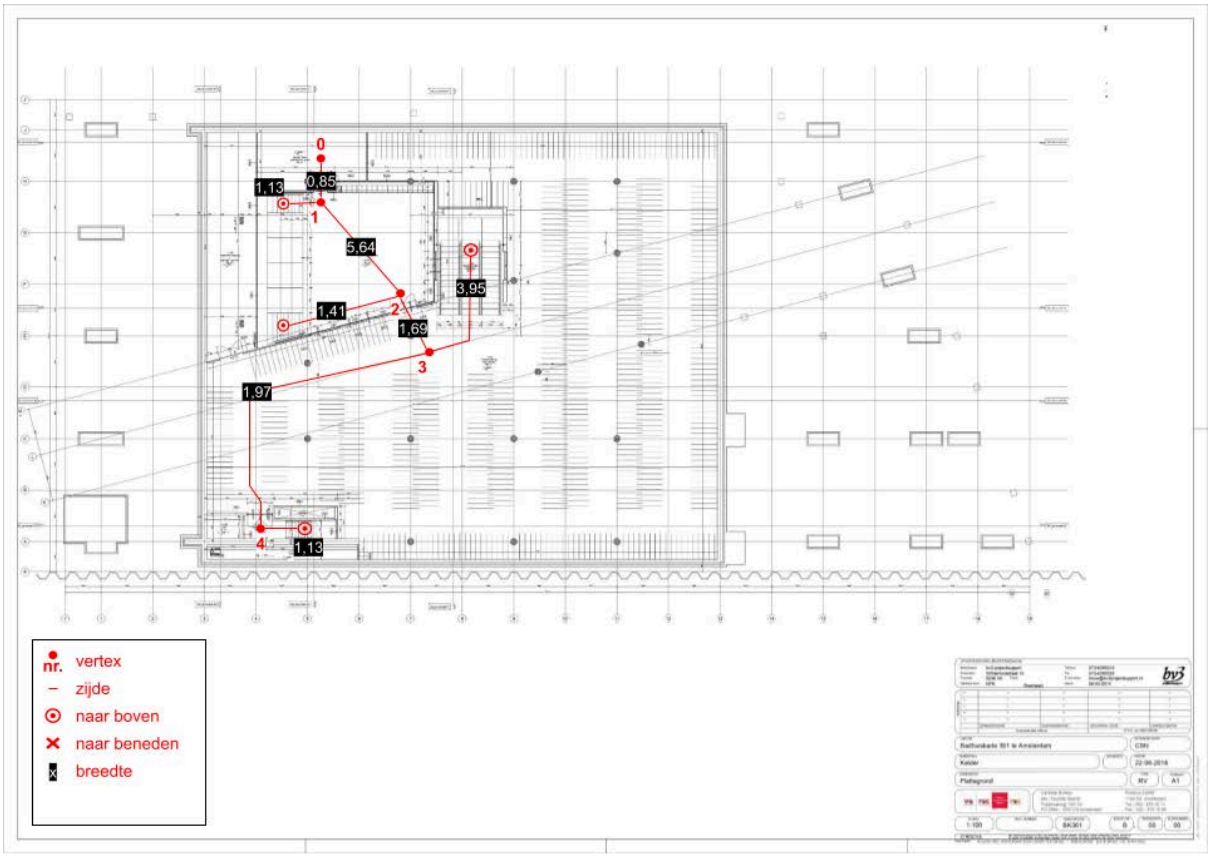


Verdieping 4:

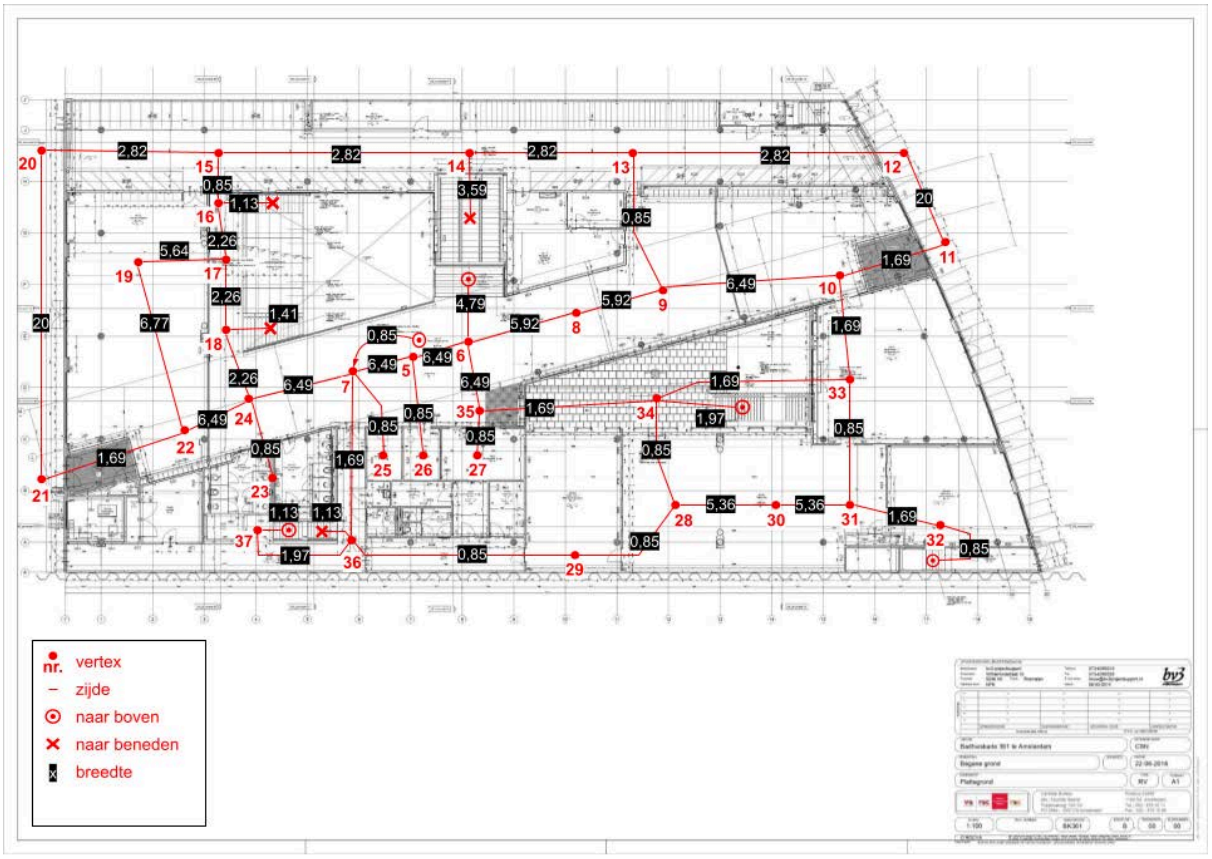


Bijlage D: De graaf met breedtes van het Hyperion Lyceum

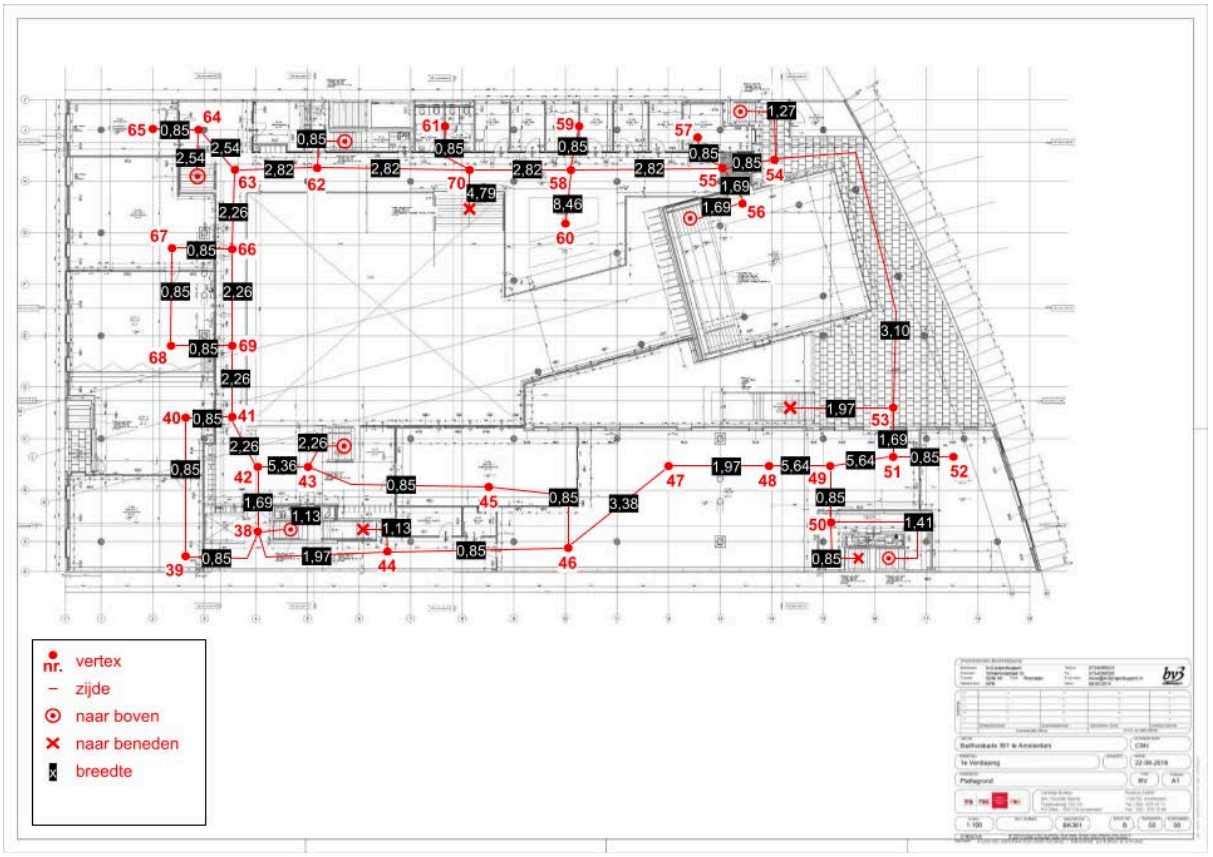
Verdieping -1:



Verdieping 0:

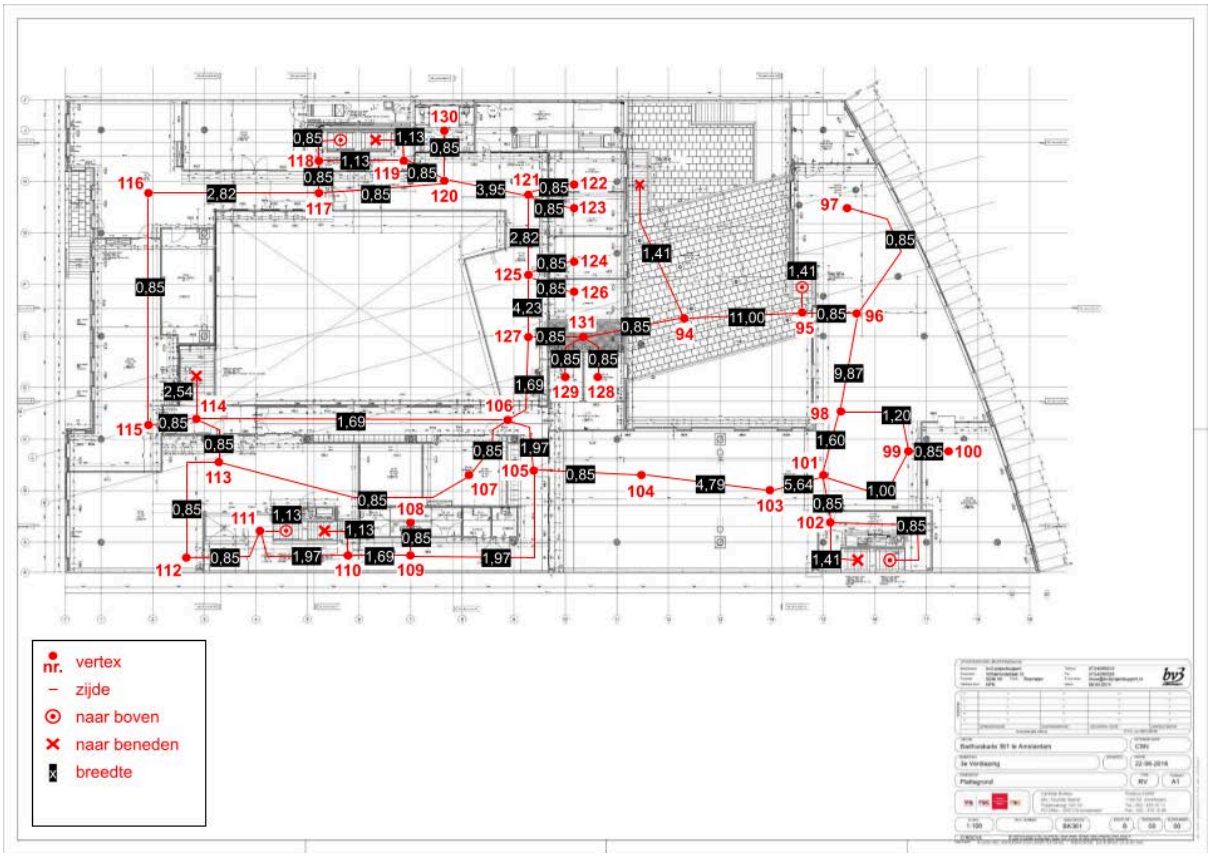


Verdieping 1:

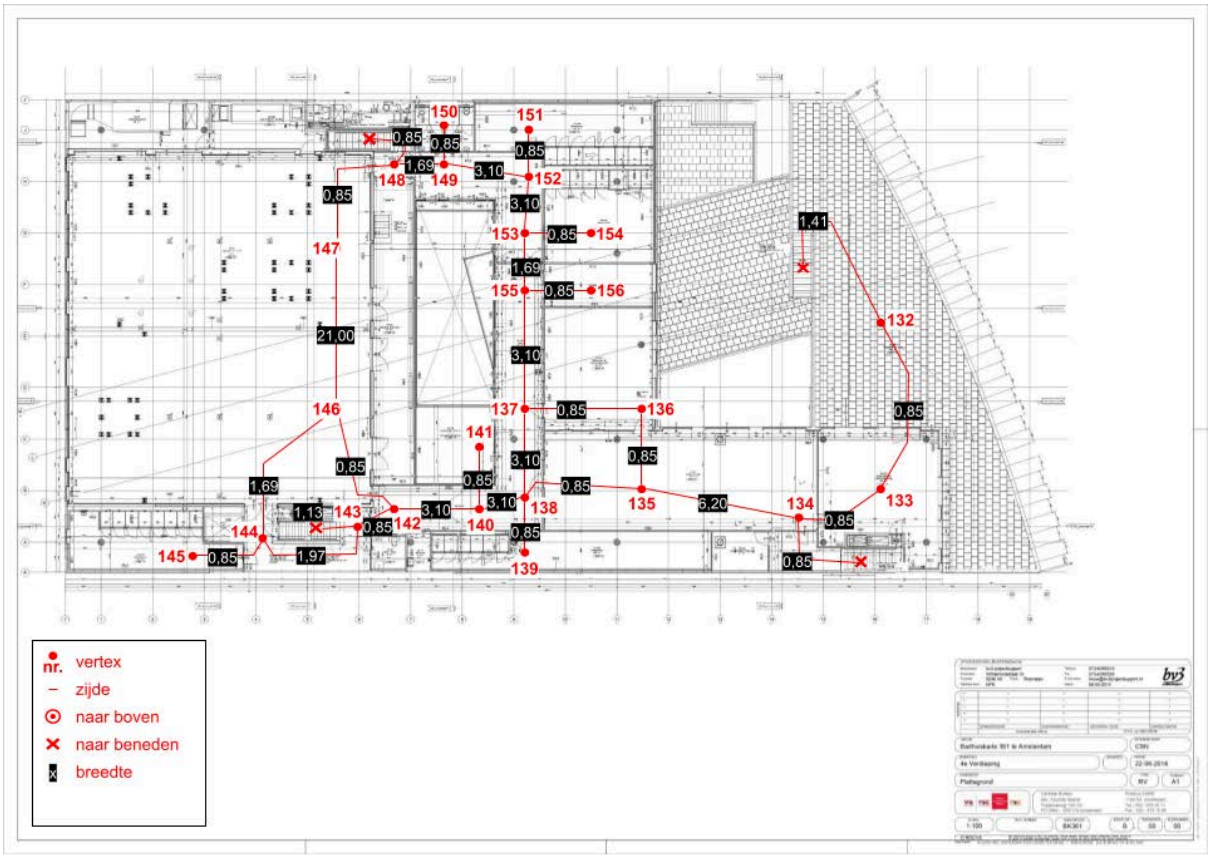


Verdieping 2:

Verdieping 3:



Verdieping 4:



Bijlage E: Externe links

Link naar de GitHub-pagina met alle rauwe Python code:
https://github.com/DylanTimothySpence/Profielwerkstuk/

Link naar de GitHub-pagina met alle code voor de GUI:
https://github.com/DylanTimothySpence/GUI-voor-PWS/

Link naar de site van de navigatie applicatie (de site is een gratis versie en kan tot drie minuten duren om te laden): https://hyperion-navigator.onrender.com/

Google Spreadsheets met daarin de adjacency lists van de drie gebruikte grafen:

- Link naar de gewogen graaf van het Hyperion Lyceum (uit paragraaf 2.4):
  https://docs.google.com/spreadsheets/d/1IbD1gFiM7MJZru5b0W2PBikFZVUaFouQrLUlvCtBBxI/edit?usp=sharing

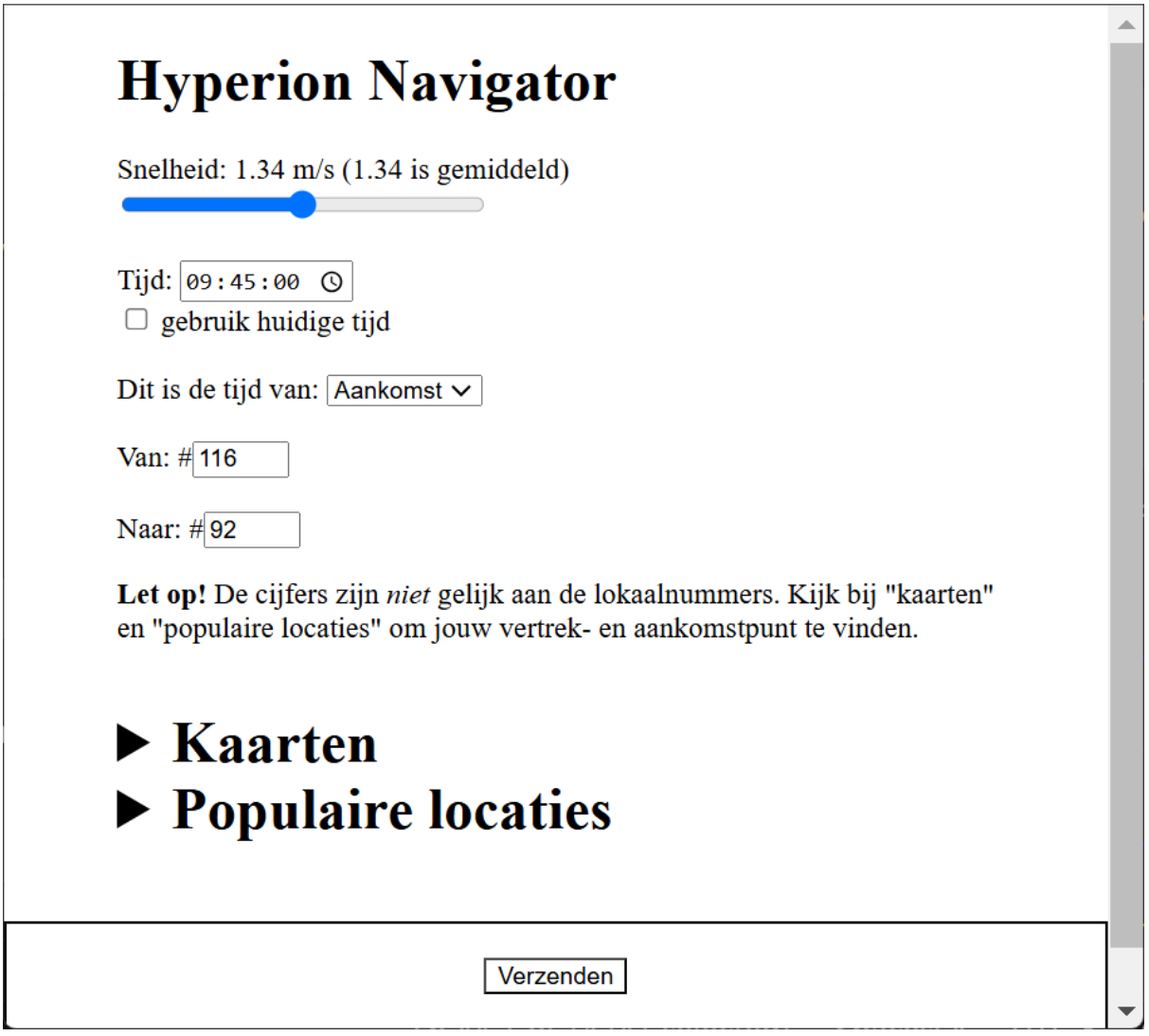
- Link naar de dubbel gewogen graaf, gebruikt in de casus (uit subparagraaf 4.2.1):
  https://docs.google.com/spreadsheets/d/1jWDS4wVVdAPbRB7g-qIORdWCBD7Uvk02Ei9FrvxfUk8/edit?usp=sharing

- Link naar de dubbel gewogen graaf van het Hyperion Lyceum (uit paragraaf 5.1):
  https://docs.google.com/spreadsheets/d/1f16xn3YNTdan2iQeKbuAsIXmbrJlNOa2SlfaI8X8zGc/edit?usp=sharing

Bijlage F: impressie gebruikerservaring

Deze bijlage toont de gebruikerservaring. De GUI is te bezoeken als website. De link naar deze website is te vinden in bijlage D. In deze bijlage staan screenshots, die gemaakt zijn tijdens het gebruik van de website (link op bijlage E).

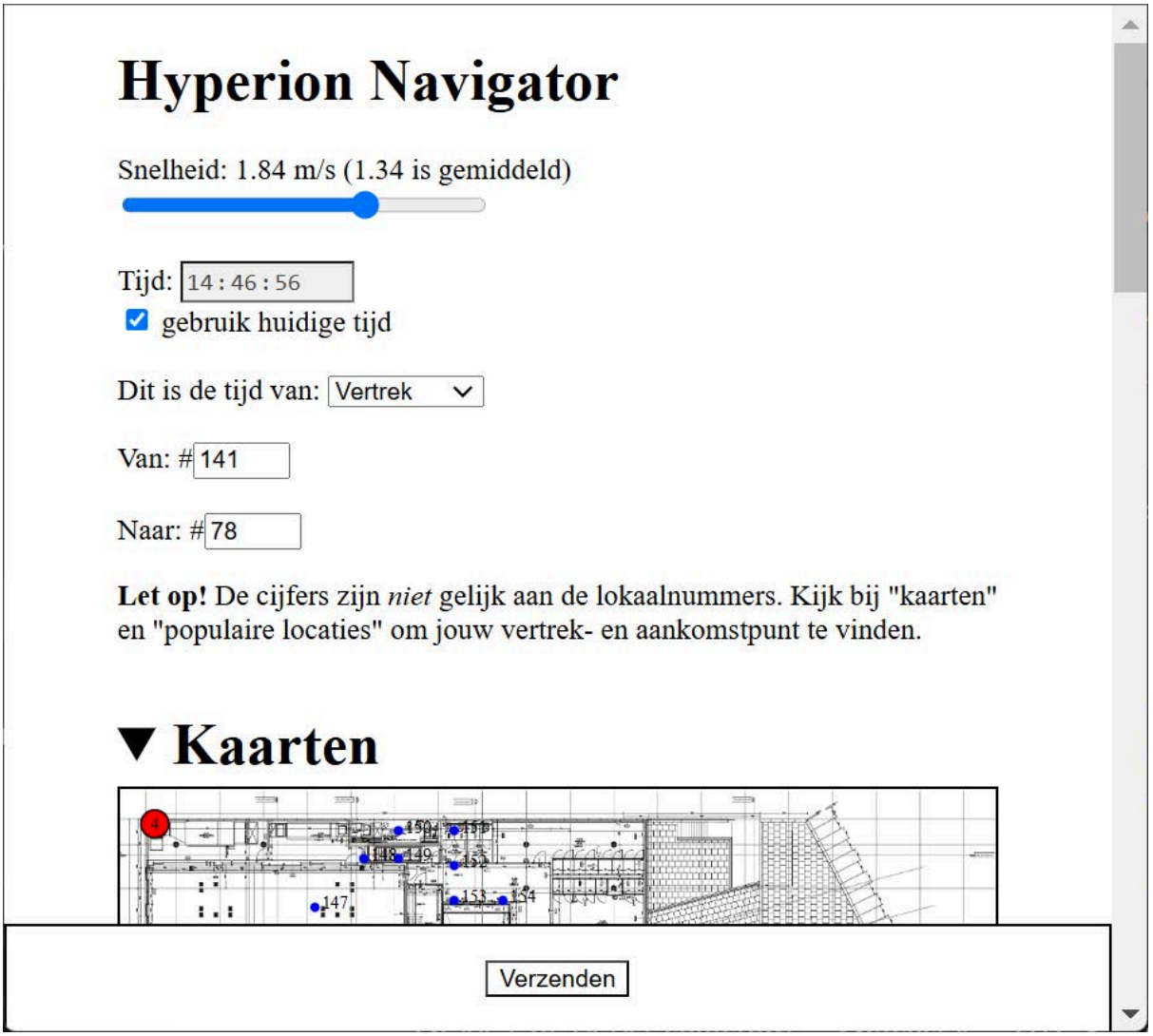
Wanneer de gebruiker de website bezoekt, wordt als eerste het beginscherm weergegeven. Op deze pagina worden de volgende gegevens ingevoerd door de gebruiker: de snelheid, de tijd, de soort tijd, het beginpunt en het eindpunt. Wanneer de gebruiker de website bezoekt, zijn de snelheid en soort tijd al ingesteld. Dit zijn "default"-waarden. De snelheid staat standaard op 1,34 m/s, de gemiddelde loopsnelheid van een mens. Deze kan aangepast worden door de gebruiker door een punt te slepen over een *slider*. Tijdens het slepen verandert de weergegeven snelheid. De soort tijd kan ook aangepast worden, door *aankomst* te kiezen uit het *dropdown menu* in plaats van *vertrek*. De volgende screenshot geeft een beeld bij het startscherm en hoe deze ingevuld kan worden:



Figuur F1: Voorbeeld van ingevuld beginscherm.

De tijd wordt op de seconde gekozen. Er is ook een *checkbox* "gebruik huidige tijd". Als deze is aangevinkt, wordt het tijd-invoer-gebied grijs en onaanpasbaar voor de gebruiker. De

huidige tijd wordt dan weergegeven, zoals in figuur F2. Deze "huidige tijd" wordt elke seconde aangepast totdat op de verzendknop wordt gedrukt. In Chrome staat op de verzendknop "Verzenden", maar dit verschilt per browser en taalinstelling, bijvoorbeeld "Submit" in het Engels.



Figuur F2. voorbeeld van ingevuld beginscherm, met gebruik huidige tijd en Vertrek geselecteerd en Kaarten uitgeklapt (maar deels zichtbaar).

Op het beginscherm staan ook de uitklapbare onderdelen: *Kaarten* en *Populaire locaties*. *Kaarten* laat de plattegronden met alle vertices zien, *Populaire locaties* is een tabel dat de vertexnummers laat zien van belangrijke locaties (zoals lokalen en kantoren).
Wanneer op de verzendknop wordt gedrukt, wordt de gebruiker doorgestuurd naar het *routescherm*, zie figuren F3 en F4:

Route

Van #116 naar #92 met snelheid 1.34 m/s.

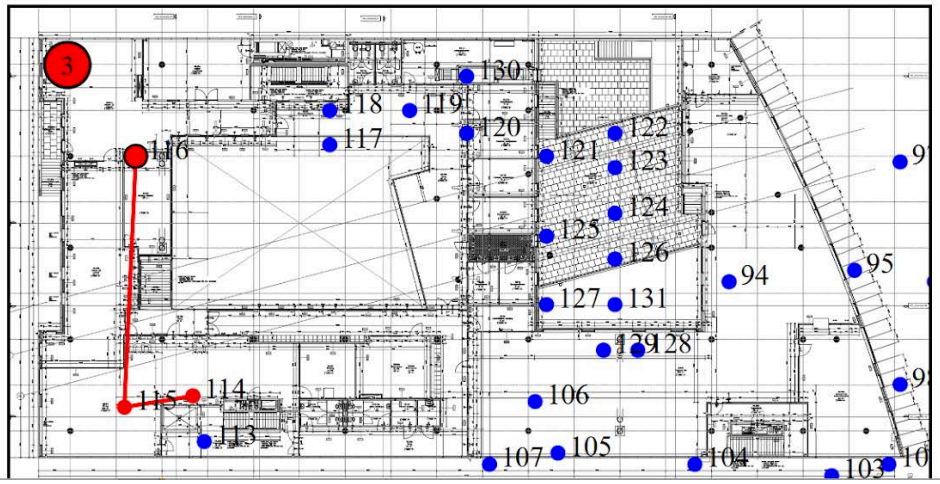
Route: [116, 115, 114, 76, 75, 77, 78, 84, 85, 86, 88, 89, 90, 92]

Tijd: 1 minuten en 16 seconden

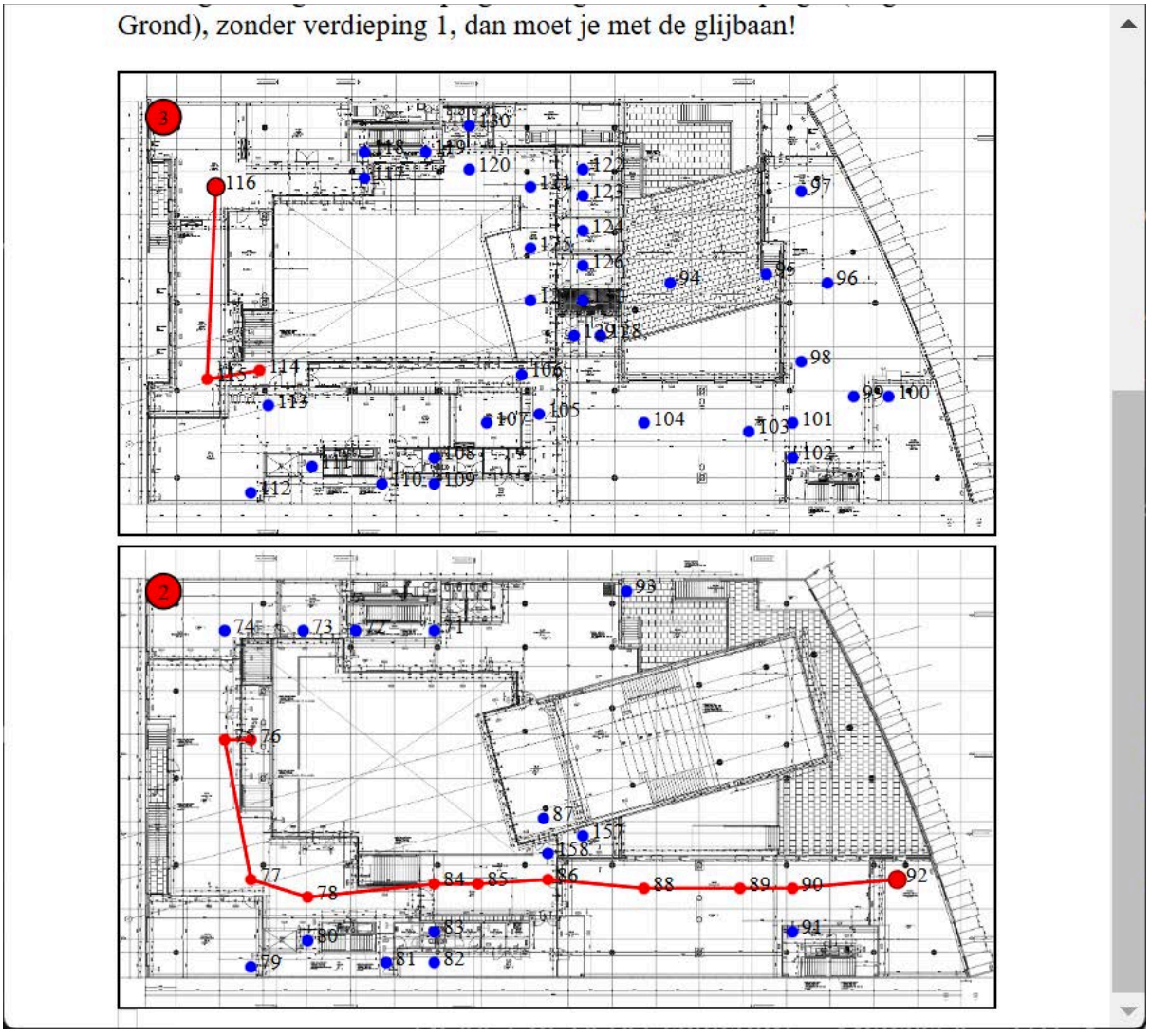
Vertrek om: 09:43:44

Als u aan wilt komen op: 09:45:00

Deze plattegronden staan in volgorde van hoogte (hoogste eerst), niet in chronologische volgorde voor de route. Vertrekpunt en aankomstpunt zijn een grote stip. Waar de route stopt op een plattegrond, moet je omhoog/omlaag. Als verdieping 2 overgaat naar verdieping 0 (Begane Grond), zonder verdieping 1, dan moet je met de glijbaan!



Figuur F3. het routescherm met de informatie ingevuld van figuur F1. Het onderste deel van de pagina, die niet op deze screenshot past, is zichtbaar in figuur F4.



Figuur F4. het routescherm met de aangegeven route. Vervolg van figuur F3.

Bijlage G: logboek

Deze bijlage toont de werkzaamheden die zijn verricht voor dit onderzoek.

Totale werktijd Tobias Crommelin: 151,5 uur

Totale werktijd Dylan Spence: 131 uur

Totale werktijd opgeteld: 282,5 uur

Logboek Tobias Crommelin:

Datum (dd-mm-jjjj)	**Tijd (uur)**	**Activiteit**
5-9-2024	1	Overleggen met Hugo
6-9-2024	1,5	Inlezen grafentheorie met boek Lex Schrijver
11-9-2024	1	Inlezen grafentheorie met boek Networks goes to school
17-9-2024	1	Uitlegfilmpjes algoritmes grafentheorie
19-9-2024	2	Inlezen en bespreken grafentheorie met Dylan, overleg met PWS begeleider
26-9-2024	1,5	Dijkstra's algoritme begin coderen met Dylan
27-9-2024	2	Dijkstra's algoritme coderen
3-10-2024	0,5	Onderzoeksplan bespreken en overleggen met PWS begeleider
7-10-2024	5,5	Dijkstra's algoritme coderen en onderzoeksplan uitbreiden met Dylan
10-10-2024	1,5	Opnieuw Lex' boek lezen, een vergelijkbaar PWS lezen, inhoud van hoofdstuk 1 plannen en overlegpunten opschrijven
11-10-2024	3	Beginnen aan schrijven van hoofdstuk 1
12-10-2024	2	Schrijven en plaatjes maken hoofdstuk 1
14-10-2024	1,5	Overleggen met Dylan en herschrijven Dijkstra code
17-10-2024	1,5	Tekenen graaf (Nutteloos omdat we ononderbouwd hebben versimpeld)
21-10-2024	1	Overleggen met Dylan over versimpelen graaf
22-10-2024	1	Overleggen met Dylan over voorwaarde versimpelen
23-10-2024	3	Definiëren versimpel-voorwaarde en bijbehorende metingen doen (met Dylan), schrijven over versimpelen
3-11-2024	3	Schrijven over versimpelen
4-11-2024	2	Testen en herschrijven Dijkstra's algoritme
7-11-2024	0,5	Overleggen met PWS begeleider
9-11-2024	3	Tekenen graaf van het Hyperion Lyceum
11-11-2024	1,5	Overleggen met Dylan over naamgeving en tekenen graaf Hyperion Lyceum
12-11-2024	3	Tekenen graaf van het Hyperion Lyceum op computer en nummeren vertices
13-11-2024	5	Excel adjacency juist indelen, constanten bepalen (deur en trap), schrijven over gewichten
14-11-2024	5	NETWORKS Masterclass (eerste van de twee)

15-11-2024	1	Trappen tellen met Dylan
17-11-2024	4	Alle gewichten berekenen
18-11-2024	5	Gewichten van papier naar een adjacency list (in een spreadsheet) zetten (met Dylan) en converteercode maken
19-11-2024	5	NETWORKS Masterclass (tweede van de twee)
21-11-2024	3	Schrijven over verkeerstheorie van de masterclass
4-12-2024	1	Overleg vertragingsfunctie met Dylan
6-12-2024	5	Zoeken en vinden vertragingsfunctie, en schrijven verkeerstheorie
7-12-2024	5	Coderen dynamische flow
8-12-2024	7	Gewogen graaf overtekenen online, en gezamenlijk aan latency functie, verslag en GUI werken
9-12-2024	4	Schrijven over (praktische) verkeerstheorie, runnen van casus en in grafiek zetten
10-12-2024	4	Schrijven over de code (van de casus) en de casus zelf
11-12-2024	5	Schrijven afmaken, grootheden en symbolen consistent maken, formules en figuren nummeren, opmaak fixen, inleveren versie 1 verslag.
13-12-2024	3	samen opzetten github en plannen volgende stap voor onderzoek, debuggen flowfunctie
14-12-2024	4	Debuggen flowfunctie: omschrijven naar asyncio, in de cloud runnen, op een andere laptop runnen
16-12-2024	3	Visualiseren en afmaken flowfunctie.
2-1-2025 t/m 24-1-2025	15	Bouwen en testen van het model van de totale verkeersstroom
27-1-2025	3	Schrijven hoofdstuk totale verkeersstroom
1-2-2025	4	In orde brengen van alle code, doorlezen volledige verslag en nummering in orde brengen
2-2-2025	5	Schrijven discussie, groter maken van de graaf-tekeningen en nieuwe maken voor de breedtes, alle code in orde brengen, nummeren formules ed.
3-2-2025	3	Schrijven discussie (stroommodel gedeelte) en schrijven conclusie
4-2-2025	4	Doorlezen verslag en feedback van mijn vader verwerken
5-2-2025	5	Kritisch doorlezen verslag met Dylan, verbeteren van enkele formules, nummering en verwijzingen in orde brengen

Logboek Dylan Spence:

Datum (dd-mm-jjjj)	**Tijd (uur)**	**Activiteit**
5-9-2024	1	Overleggen met Hugo
7-9-2024	2	Inlezen grafentheorie met boek Networks goes to school
18-9-2024	1	Uitlegfilmpjes algoritmes grafentheorie
19-9-2024	1,5	Inlezen grafentheorie met boek Lex Schrijver, overleg met PWS begeleider
26-9-2024	1,5	Dijkstra's algoritme begin coderen met Tobias
29-9-2024	2	Dijkstra's algoritme coderen
3-10-2024	0,5	Onderzoeksplan bespreken en overleggen met PWS begeleider
7-9-2024	2	Python verder inlezen en oefenen
7-10-2024	5,5	Dijkstra's algoritme coderen en onderzoeksplan uitbreiden met Tobias
14-10-2024	1,5	Overleggen met Tobias en herschrijven Dijkstra code
17-10-2024	1,5	Tekenen graaf (Nutteloos omdat we ononderbouwd hebben versimpeld)
21-10-2024	1	Overleggen met Tobias over versimpelen graaf
22-10-2024	1	Overleggen met Tobias over voorwaarde versimpelen
23-10-2024	1,5	Definiëren versimpel-voorwaarde en bijbehorende metingen doen (met Tobias)
1-11-2024 t/m 22-11-2024	15	Coderen GUI in html en javascript
14-11-2024	5	NETWORKS Masterclass (eerste van de twee)
15-11-2025	1	Trappen tellen met Tobias
18-11-2024	3	Gewichten van papier naar een adjacency list (in een spreadsheet) zetten met Tobias
4-12-2024	1	Overleg vertragingsfunctie met Tobias
5-12-2024	1,5	Begin leren Javascript (met w3schools) en D3 (youtube tutorials)
6-12-2024	2	Begin leren Flask (met uitlegvideo's op YouTube)
7-12-2024	4,5	Connectie tussen HTML en Python maken (met Flask)
8-12-2024	5	Gezamenlijk aan latency functie, verslag en GUI werken
9-12-2024	6	Begin GUI maken met D3 en Flask
10-12-2024	6	Eerste versie GUI afmaken, schrijven introductie en discussie, begin schrijven GUI uitleg
11-12-2024	2	Schrijven GUI uitleg en inleveren versie 1 verslag
13-12-2024	1	samen opzetten github en plannen volgende stap voor onderzoek
18-12-2024	1,5	Verbeteren GUI (trappen andere kleur, sticky header, aanklikbare startnode)
20-12-2024	1	Nieuwe Github aanmaken voor de GUI, begin maken nieuwe Flask app
21-12-2024	2	Verbeteren connectie tussen Python en HTML met behulp van Flask
22-12-2024	1	Opzetten GUI met render.com
23-12-2024	4	maken voorbeeldgraaf voor GUI, verbeteren dijkstra.py en converter.py,

		afmaken connectie tussen Flask en HTML, slider toevoegen voor snelheid
24-12-2024	3	Toevoegen SVG's aan GUI, debuggen GUI, begin responsive design, programmeren van het verwijderen verdiepingen uit GUI die niet in de route voorkomen, tooltip toegevoegd, mappenstructuur schoner maken
25-12-2024	3	Toepassen D3 op graaf (begin met versimpelde voorbeeldgraaf), toevoegen opmaak voor begin en eindpunt, toevoegen opmaak voor trappen (later weggehaald)
26-12-2024	4	Toevoegen relatieve tooltip voor het responsive maken van D3, verbeteren opmaak met CSS, overgaan op plattegronden van het Hyperion Lyceum
27-12-2024	3	Invoeren relatieve coördinaten van nodes, responsive maken D3 (lijndikte, puntgrootte, grootte startnode en endnode, labels)
9-1-2025	2	beginnen definitieve versie van GUI (na gesprek met Tobias) (echter: krijg vaak 404 error)
12-1-2025	3	Afmaken basis voor definitieve GUI
17-1-2025	1	"Schoonmaken" code (comments en onnodige code verwijderen, uitleg toevoegen voor gebruiker)
20-1-2025	2	Toevoegen tijd input en ETA functie (maar nog niet Dynamisch)
21-1-2025	2	Tijd afhankelijk maken selectie grafen in Flask app
22-1-2025	4	Verder programmeren tijd afhankelijk maken selectie grafen in Flask app, toevoegen "vertrek nu" knop, debuggen Flask
27-1-2025 t/m 31-1-2025	4	Toevoegen en werkend maken 'vertrek' en 'aankomst' knoppen
2-1-2025 t/m 5-2-2025	9	Schrijven hoofdstuk GUI en bijlage 'impressie gebruikerservaring' in verslag, doorlezen verslag en feedback schrijven, verbeteren formulering
5-2-2025	5	Laatste keren kritisch elke zin doorlezen met Tobias, en inleveren