Efficiëntie-analyse van Compressed Sensing in een booleaanse setting

Noé Boddez en **Simon Coessens**
Faculteit wetenschappen, KU Leuven
{noe.boddez, simon.coessens}@student.kuleuven.be

Abstract

Compressed sensing is een techniek binnen de signaalverwerking die het mogelijk maakt om met weinig metingen toch een goede reconstructie van een opgemeten signaal te verkrijgen. Over het algemeen wordt deze techniek als efficiënt beschouwd, maar hoe efficiënt is dit nu in de praktijk? We onderzoeken dit door tijdsmetingen uit te voeren voor het oplossen van compressed sensing problemen binnen een booleaanse setting, en dit voor verschillende parameters. De praktische toepassing waar we in deze paper op focussen is die van de groepstesten.

1 Inleiding

Stel dat een grote populatie getest moet worden op een ziekte. Er is geweten dat het aantal positieven onder de populatie schaars is. Gezien de omvang van de populatie kan het te kostelijk zijn om iedereen individueel te testen. Een mogelijke aanpak is de volgende, we nemen van elke persoon een staal. Er kan een gemengde staal gecreëerd worden met de helft van de stalen van de populatie en deze kan getest worden. Indien deze negatief is kan men de helft van de populatie reeds buiten beschouwing laten. Indien dit proces verdergezet wordt, verminderd het aantal testen drastisch. Uiteraard is deze aanpak enkel mogelijk als het aantal positieven laag is. Deze testmethode werd het eerst geïntroduceerd door Dorfmann [1943] als oplossing voor het testen van syphilis onder grote populaties. Dorfmann heeft zo de deur geopend naar het domein van de combinatorische groepstesten. Het probleem van het groepstesten kan ook geformuleerd worden als een compressed sensing probleem.

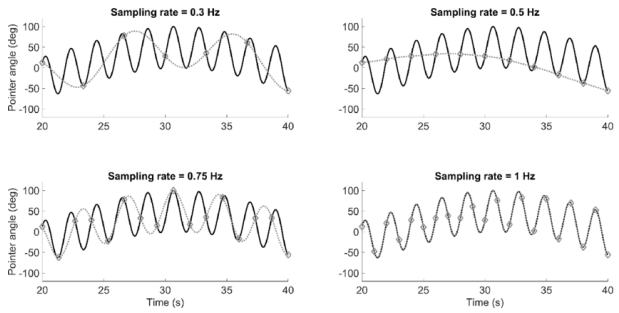
Groepstesttechnieken kunnen doorslaggevend zijn in het bestrijden van ziekten. Met het oog op het gebruik van deze technieken in de strijd tegen ziekten is het belangrijk dat men kan inschatten wat mogelijke uitvoeringstijden zijn. Ook het bepalen van nodige parameters met betrekking tot deze groepstesten is essentieel. In dit onderzoek behalen we resultaten waaruit deze parameters optimaal gekozen kunnen worden. Deze resultaten hebben uiteraard niet enkel betrekking tot het testen van ziekten maar kunnen ook voor andere toepassingen geïnterpreteerd worden.

2 Compressed sensing

Compressed sensing is een techniek binnen het domein van de signaalverwerking. Het doel is om een bepaald signaal te reconstrueren door metingen op te nemen van dit signaal.

2.1 Signaal reconstructie

Een voorbeeld van een mogelijk signaal is een geluidsgolf. Figuur 1 illustreert het concept van reconstructie.



Figuur 1: Illustratie van de reconstructie van een signaal. Het signaal bestaande uit 2 sinusgolven is weergegeven in het zwart. De grijze gestipte lijnen representeren de reconstructies.

Het signaal in deze figuur is een samenstelling van 2 sinusgolven met frequenties 0.03 Hz en 0.48 Hz. In de subfiguur in de linkerbovenhoek wordt er onderbemonsterd (eng: undersampled). Het resultaat is dan ook dat het gereconstrueerde signaal niet correct is. Dit zien we aangezien het gereconstrueerde signaal, de grijze lijn, niet perfect samenvalt met het te meten signaal, de zwarte lijn. Ook in de subfiguur rechtsboven en linksonder is de reconstructie niet correct. In de subfiguur rechtsonder wordt een perfecte reconstructie bekomen. Deze figuur toont dus het belang van het aantal testen in het reconstrueren van een signaal. Een belangrijke theoretische stelling i.v.m het aantal metingen is de volgende:

Voor een correcte reconstructie moet er bemonsterd worden aan minstens 2 maal de hoogste frequentie

Dit resultaat staat bekend als het bemonsteringstheorema van Nyquist-Shannon. Het stelt een ondergrens op het aantal metingen om een perfecte reconstructie te bekomen. In het voorbeeld in Figuur 1 is de hoogste frequentie 0.48 Hz. Een perfecte reconstructie wordt bekomen bij een bemonsteringsfrequentie van 0.96 Hz, 2 maal de hoogste frequentie.

Verrassend aan de theorie van compressed sensing is dat het tegenstrijdig is met dit theorema. Compressed sensing probeert een perfecte reconstructie te bekomen met een onderbemonstering van het te meten signaal. Hoe is dit mogelijk? Het is de bijkomende aanname van spaarsheid die dit mogelijk maakt. In volgende sectie leggen we in detail uit waarom deze aanname zo belangrijk is.

2.2 Spaarsheid

Spaarsheid (of ijlheid) is een sleutelconcept om compressed sensing mogelijk te maken [Candes and Wakin, 2008]. Zonder dit concept zouden we terugvallen op het bemonsteringstheorema van Nyquist-Shannon. We illustreren hoe we spaarsheid kunnen gebruiken aan de hand van een voorbeeld:

Gevraagd: 2 getallen die sommeren tot 4
Oplossing: oneindig veel oplossingen

1 en 3, 2 en 2, en zelfs -10 en 14 zijn allemaal mogelijke oplossingen voor dit gevraagde. We stellen dus vast dat er geen unieke oplossing is. Om het toch mogelijk te maken één oplossing hieruit te kiezen zullen we een extra voorwaarde stellen in het gevraagde:

Gevraagd: 2 getallen die sommeren tot 4, zo spaars mogelijk
Oplossing: 4 en 0

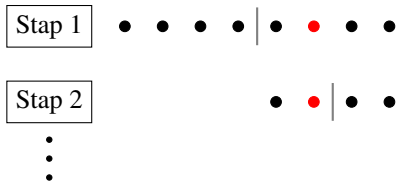
De aanname van spaarsheid heeft als gevolg dat er voor een ondergedetermineerd stelsel toch een unieke oplossing kan bekomen worden. De betekenis van spaarsheid is in elke context anders. Bijvoorbeeld, in de context van de eerder aangehaalde groepstesten heeft spaarsheid als betekenis dat het aantal positieven binnen een populatie zeer laag is.

2.3 Toepassingen

Compressed sensing is vooral nuttig voor toepassingen waarbij de kost van het meten hoog ligt. Door minder metingen te doen wordt deze kost geminimaliseerd. Een duidelijk voorbeeld hiervan is het gebruik van compressed sensing in MRI-scans. Aangezien het nemen van een MRI-scan schadelijk is voor het menselijk lichaam moet het aantal metingen geminimaliseerd worden. Om dit op te lossen kan compressed sensing gebruikt worden bij het reconstrueren van de MRI-afbeelding.

Groepstesten en compressed sensing komen samen binnen het domein van de booleaanse groepstesten. Hier zal een populatie getest worden door middel van gemengde testen. Met compressed sensing kan men, gebruikmakend van het resultaat van deze gemengde testen, de positieven personen uit de populatie selecteren. De meeste groepstesttechnieken zijn adaptief. Er wordt gebruik gemaakt van tussentijdse resultaten om een deel van de testpopulatie vroegtijdig uit te sluiten. Vervolgens kan in een volgende stap het overige deel van de populatie getest worden. Figuur 2 verduidelijkt dit concept.

Een essentieel verschil tussen compressed sensing en andere groepstesttechnieken is het niet-adaptieve karakter van deze aanpak. De resultaten van alle testen zijn beschikbaar voordat de compressed sensing techniek toegepast wordt.



Figuur 2: Illustratie van een adaptieve groepstesttechniek. Een populatie met 1 positief individu (rood) is weergegeven. De populatie wordt in Stap 1 in twee delen opgesplitst. Uit het resultaat van de groepstest in Stap 1 volgt dat elk individu in het linkerdeel van de populatie negatief is. In Stap 2 kan het linkerdeel van de populatie uitgesloten worden. Verder kan hetzelfde proces herhaald worden. In Stap 2 worden er opnieuw testen afgenomen.

3 Formulering van het probleem

Het probleem dat we onderzoeken in deze paper is de vraag of compressed sensing wel efficiënt genoeg is om in de praktijk gebruikt te worden. Anderzijds lost compressed sensing zelf ook weer een probleem op, een optimalisatieprobleem. In het vervolg van de paper verwijst het woord "probleem" dus naar het optimalisatieprobleem dat compressed sensing gaat oplossen en niet de vraag of dit wel efficiënt kan gebeuren.

Het probleem dat we met compressed sensing willen oplossen en waarvan we uiteindelijk de efficiëntie wensen te bepalen begint bij volgende vergelijking:

$$\mathrm{A}\,\mathrm{x} = \mathrm{y}\,. \tag{1}$$

Hierin is A de "metingsmatrix", een $m \times n$ matrix. Deze matrix zal de metingen uitvoeren op x. Deze x, een $n \times 1$ vector, is dus het signaal dat we willen reconstrueren en y is een $m \times 1$ vector die het resultaat van de metingen bevat. Er worden met andere woorden m metingen gedaan op een signaal dat n componenten bevat.

Om dit allemaal duidelijker te maken geven we een concreet voorbeeld. Stel

$$\mathrm{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

en

$$\mathrm{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Dan is

$$\mathrm{y} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

Hier is het aantal metingen, m, gelijk aan 3 en de grootte van het signaal, n, is 4. Het doel is om x te reconstrueren, gegeven

A en y en wetende dat

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

A stellen we op als willekeurige matrix waarbij alle $a_{i,j}$ onafhankelijke, identiek verdeelde toevalsgrootheden zijn, gekozen uit de Bernoulli-verdeling [Candes and Wakin, 2008]. Er zijn ook andere manieren om deze op te stellen, wij hebben voor deze simpele maar effectieve aanpak gekozen.

Een eigenschap van compressed sensing is dat $m \ll n$, wat het over het algemeen onmogelijk maakt om x hieruit op te lossen. We zijn dan namelijk een stelsel van meer onbekenden dan vergelijkingen aan het oplossen. Daarom is de bijkomende voorwaarde dat x spaars is van essentieel belang, zoals uitgelegd in sectie 2.2. Dit wil dus zeggen dat x weinig niet-nul elementen mag bevatten. Een manier om de spaarsheid van een vector te meten is door de ℓ_0 norm te berekenen. Wat deze norm doet is het aantal niet-nul elementen tellen. Echter, als we verder zouden werken met deze norm bekomen we een NP-hard optimalisatieprobleem, dus gaan we een relaxatie uitvoeren door de ℓ_1 norm te gebruiken. Aangezien we niet in de continue, maar wel de booleaanse setting werken, is elk niet-nul element 1 en geeft de ℓ_1 norm hetzelfde resultaat als de ℓ_0 norm. $\| \mathrm{x} \|_{\ell 1}$ is namelijk de som van alle elementen in x.

Naar analogie met de formulering van het probleem in Malioutov en Malyutov [2012] beschrijven we in het volgende deel in verschillende stappen hoe we tot de finale formulering komen. Beginnend bij de formulering in woorden: gegeven A en y, bereken x zodat $\| \mathrm{x} \|_{\ell 1}$ minimaal is en voldoet aan vergelijking (1).
In wiskundige termen:

$$min \, \| \mathrm{x} \|_{\ell 1} \quad \text{zo dat: } \mathrm{x} \in \{0,1\} \qquad (2)$$
$$\mathrm{A} \vee \mathrm{x} = \mathrm{y}\,.$$

Merk op dat we de vergelijking zoals in (1) hebben veranderd zodat nu de logische *of* ('$\vee$') wordt gebruikt in plaats van een vermenigvuldiging. Deze aanpassing is een gevolg van het feit dat we in de booleaanse setting werken. Het resultaat van deze logische *of* is als volgt: indien er een j is zodat $a_{i,j} = 1$ en $x_j = 1$, dan zal $y_i = 1$ en anders is $y_i = 0$.

We kunnen deze vergelijking ook aanpassen zodat we niet met booleaanse algebra moeten rekenen:
Stel $\mathcal{I} = \{i \mid y_i = 1\}$ en $\mathcal{J} = \{j \mid y_j = 0\}$. Dan is $\mathrm{A} \vee \mathrm{x} = \mathrm{y}$ hetzelfde als zeggen dat $\mathrm{A}_{\mathcal{I}}\,\mathrm{x} \geq \mathrm{y}_{\mathcal{I}}$ en $\mathrm{A}_{\mathcal{J}}\,\mathrm{x} = 0$.

Als we daarbovenop de ℓ_1 norm nog iets intuïtiever schrijven als de som van de elementen, dan bekomen we volgende herformulering van het probleem:

$$min \sum_j x_j \quad \text{zo dat: } \mathrm{x} \in \{0,1\} \qquad (3)$$
$$\mathrm{A}_{\mathcal{I}}\,\mathrm{x} \geq \mathrm{y}_{\mathcal{I}}, \;\; \mathrm{A}_{\mathcal{J}}\,\mathrm{x} = 0.$$

Dit is een lineair optimalisatieprobleem. Zoals we in wat volgt gaan zien is dit een belangrijke observatie. Om het ook nog oplosbaar te maken voor bestaande software moeten we een relaxatie uitvoeren op de waardes die x kan aannemen. Door te stellen dat $\mathrm{x} \in \{0,1\}$ zorgt dit ervoor dat het probleem NP-hard is, dus relaxeren we deze voorwaarde zodat x continue waardes tussen 0 en 1 kan aannemen. Deze relaxatie kan leiden tot oplossingen die getallen verschillend van 0 en 1 bevatten en als gevolg niet booleaans zijn. Dit lossen we op door kleine waarden naar 0 af te ronden en grote waarden naar 1. De formulering van het lineair optimalisatieprobleem is dan de volgende:

$$min \sum_j x_j \quad \text{zo dat: } 0 \leq \mathrm{x} \leq 1 \qquad (4)$$
$$\mathrm{A}_{\mathcal{I}}\,\mathrm{x} \geq \mathrm{y}_{\mathcal{I}}, \;\; \mathrm{A}_{\mathcal{J}}\,\mathrm{x} = 0.$$

4 Methoden

Het vrij abstracte compressed sensing probleem waar we mee zijn begonnen is nu geformuleerd als een lineair optimalisatieprobleem. Aan de hand van deze herformulering kunnen we lineair programmeersoftware gebruiken om een oplossing te vinden. In deze studie vergelijken we drie verschillende toolkits.

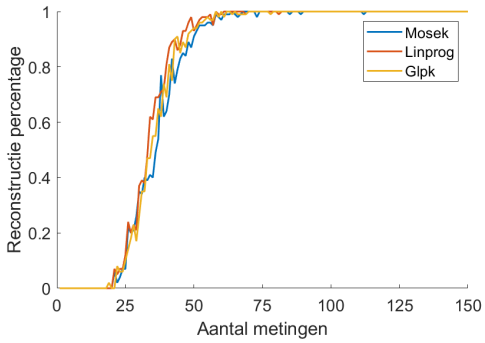
MATLAB Optimization Toolbox: een ingebouwde toolbox in MATLAB om optimalisatieproblemen op te lossen. Binnen deze toolbox hebben we de LINPROG functie gebruikt. [MathWorks, 2022]

GNU Linear Programming Kit: deze toolkit biedt een interface voor MATLAB. In deze interface hebben we de GLPK functie gebruikt. [GNU, 2022]

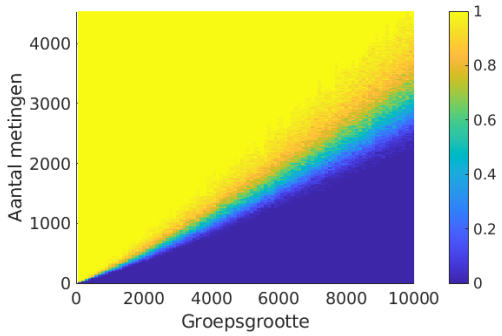
MOSEK: MOSEK biedt ook een toolbox voor MATLAB aan met verschillende optimalisatiefunctionaliteit. Wij hebben de MOSEKOPT functie gebruikt. [MOSEK ApS, 2022]

Voor elk van deze toolkits hebben we het probleem geïmplementeerd om uiteindelijk tijdsmetingen uit te voeren op de oplossingstijd. De tijdsmetingen gebeuren allemaal binnen MATLAB en meten enkel de uitvoeringstijd van elk van bovenvermelde functies. De initialisatietijd van de parameters wordt hier buiten beschouwing gelaten.

Door meerdere methodes te vergelijken kunnen we met meer zekerheid een besluit nemen. Ook zouden we interessante inzichten kunnen verwerven door deze te vergelijken.



Figuur 3: Kans op exacte reconstructie in functie van het aantal testen. We nemen een gemiddelde van 100 iteraties en stellen n gelijk aan 150 en K gelijk aan 4.



Figuur 4: Kans op perfecte reconstructie in functie van groepsgrootte en aantal metingen. We stellen k gelijk aan 0.02 en nemen een gemiddelde van 100 iteraties.

5 Voorbereidend werk

5.1 Parameters

Het absoluut aantal positieven stellen we voor als K. De groepsgrootte stellen we voor als n. Ook gebruiken we k als het percentage van het aantal positieven, m.a.w $K = kn$.

5.2 Implementatie model

Elke implementatie die het lineair optimalisatieprobleem zoals uitgeschreven in (4) oplost, zou voor hetzelfde probleem ook dezelfde resultaten moeten geven. We controleren dit aan de hand van Figuur 3. De parameters zijn als volgt gekozen: $n = 150$, $k = 4/150$. Hier zien we dat, voor willekeurig gegenereerde problemen met gegeven parameters, de resultaten van verschillende toolkits gemiddeld genomen overeenkomen.

5.3 Experimentele resultaten

Het doel van dit onderzoek is om de efficiëntie van compressed sensing te bestuderen in de booleaanse setting. Dit doen we aan de hand van tijdsmetingen. Een tijdsmeting die uitgevoerd wordt voor een bepaalde groepsgrootte en spaarsheid is enkel relevant indien het geteste programma een correcte reconstructie kan voortbrengen. Een correcte reconstructie wordt voortgebracht indien het aantal testen (m) groot genoeg gekozen wordt. Aangezien efficiëntie getest wordt mag deze (m) niet te groot gekozen worden want dan zal de gemaakte tijdsmeting niet representatief zijn. Hoe kiezen we deze m minimaal zodat er een perfecte reconstructie voortgebracht wordt?

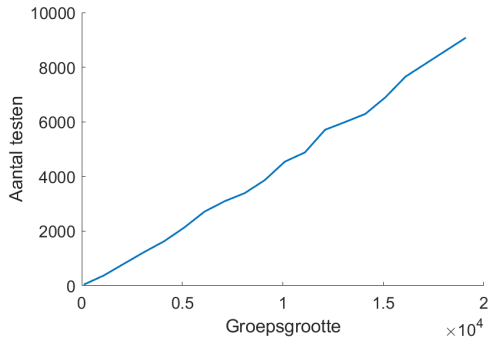
Zoals aangetoond in [Atia and Saligrama, 2012] is $m = O(K \log(n))$. In de tijdsmetingen die we zullen uitvoeren laten we K lineair schalen met n. We verwachten dus dat

$$m = O(n \log(n)). \tag{5}$$

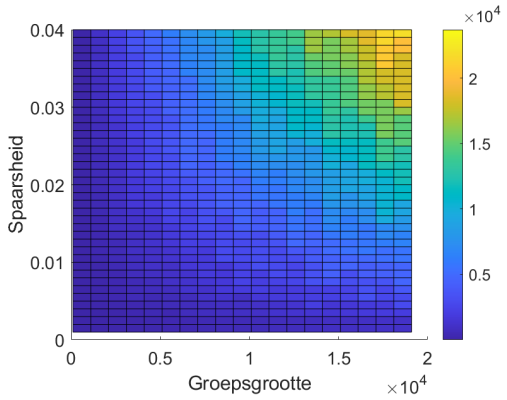
In volgend onderzoek zullen we het aantal testen (m) experimenteel bepalen. We bekijken vooreerst waar deze grens van niet-perfecte naar perfecte reconstructie ligt en hoe deze evolueert voor een stijgende groepsgrootte. In deze sectie nemen we de spaarsheid k vast op $k = 0.02$. Bij elk paar (n, m) bekijken we de behaalde reconstructiegraad. Het resultaat hiervan is te zien in Figuur 4. We zien hier het verloop van de reconstructiegraad voor stijgende n.

Figuur 5 toont hetzelfde experiment, deze keer voor een grotere probleemgrootte, en enkel de lijn van de overgang van niet-perfecte naar perfecte reconstructie wordt weergegeven. Zoals we kunnen verwachten uit stelling (5) vertoont deze reconstructielijn een linearitmisch verloop in functie van n. In de getoonde grafieken werd een vaste spaarsheid (k) genomen. Aangezien er ook tijdsmetingen moeten worden uitgevoerd voor een variabele spaarsheid moet ook hiervoor het aantal benodigde testen bepaald worden. Dit bekomen we door de drie parameters n, m, k te laten variëren. Dit bezorgt ons de grafiek te zien in figuur 6.

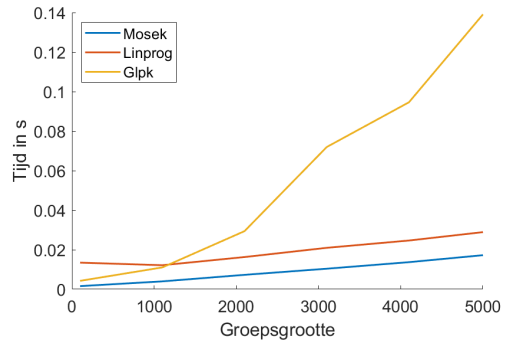
Deze figuur is al een belangrijk resultaat. We kunnen er namelijk voor elk paar parameters aflezen hoe groot we de laatste parameter moeten nemen. Wij gaan deze dus gebruiken om te weten hoe groot we onze metingsmatrix moeten nemen in onze tijdsmetingen bij een bepaalde groepsgrootte en een gegeven spaarsheid.



Figuur 5: Minimaal aantal vereiste testen in functie van groepsgrootte voor perfecte reconstructie. ($k = 0.02$)



Figuur 6: Minimaal aantal vereiste testen in functie van groepsgrootte en spaarsheid. We nemen een gemiddelde van 100 iteraties.



Figuur 7: Uitvoeringstijd van Mosek, Linprog en Glpk voor een groepsgrootte tot 5000, gegeven in seconden. We nemen een gemiddelde van 100 iteraties en stellen k gelijk aan 0.02.

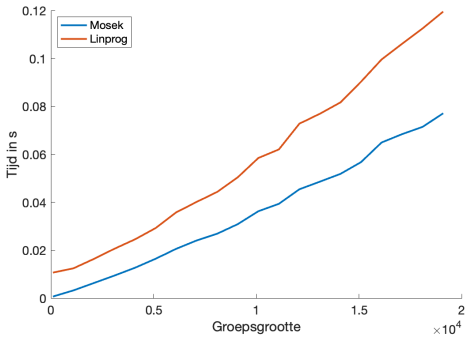
6 Tijdsmetingen

Nu de benodigde experimentele gegevens beschikbaar zijn kunnen de tijdsmetingen uitgevoerd worden.

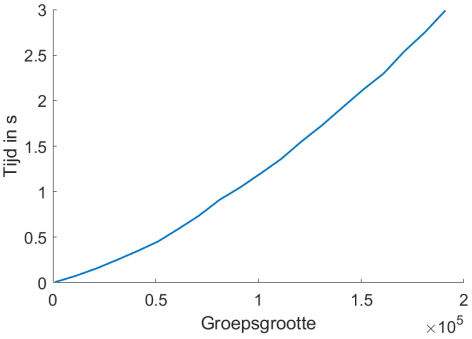
6.1 Tijdsmetingen voor een vaste spaarsheid

We bekijken eerst de resultaten voor een vaste spaarsheid $k = 0.02$. We meten de uitvoeringstijd voor de 3 geïmplementeerde toolkits voor een groepsgrootte van 0 tot 5000. Het resultaat hiervan zien we in Figuur 7. Uit deze resultaten kunnen we reeds concluderen dat Linprog en Mosek performanter zijn dan Glpk. We bekijken dan ook in komende tijdsmetingen enkel Mosek en Linprog en laten Glpk verder buiten beschouwing.

In Figuur 8 zien we de uitvoeringstijd van Linprog en Mosek tot aan een groepsgrootte van 20 000. Hierin zien we dat Mosek performanter is dan Linprog. Merkwaardig is hier dat de uitvoeringstijd zeer laag blijft. Dit is een resultaat dat we vooraf niet verwacht hadden. Gezien het snel stijgende verloop van de uitvoeringstijd van Glpk hadden we verwacht dat Mosek en Linprog ook een kwadratisch of sneller stijgend verloop zouden hebben met grotere constanten. We kunnen dus gaan kijken naar een groepsgrootte van enkele ordes groter.



Figuur 8: Uitvoeringstijd van Mosek en Linprog voor een groepsgrootte tot 2×10^4. We stellen k gelijk aan 0.02 en nemen een gemiddelde van 100 iteraties.



Figuur 9: Tijdsmetingen voor Mosek voor een groepsgrootte tot 2×10^5, gegeven in seconden. We nemen een gemiddelde van 20 iteraties en stellen k gelijk aan 0.02.

In Figuur 9 zien we het resultaat hiervan. Ook hier is het merkwaardig dat de uitvoeringstijd zeer laag blijft.

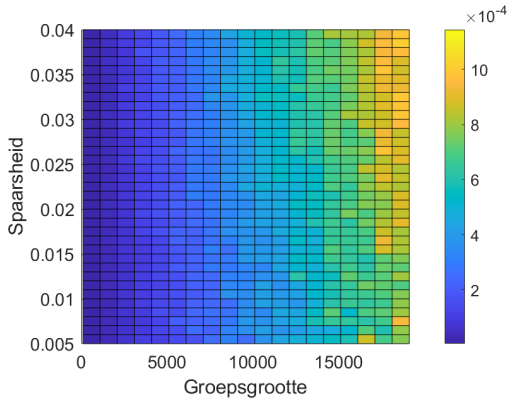
6.2 Tijdsmetingen voor een variërende spaarsheid

De tijdsmetingen kunnen we ook uitvoeren voor een variërende k. Het resultaat hiervan is te zien in Figuur 10. Zoals verwacht zien we dat de stijging van de groepsgrootte en een daling van de spaarsheid een hogere uitvoeringstijd teweeg brengt. Ook zien we dat de verandering in groepsgrootte merkwaardig meer invloed heeft op de uitvoeringstijd in vergelijking met de verandering in spaarsheid. Uit het resultaat van Figuur 6 zouden we verwacht hebben dat deze twee parameters een evenredig effect zouden hebben op de uitvoeringstijd.

7 Model met ruis

7.1 Introductie model met ruis

Vaak brengt groepstesten nog een extra factor in het spel. Elke test heeft namelijk een bepaalde betrouwbaarheid, bestaande uit de sensitiviteit en de specificiteit. Ook zal de sensitiviteit van de test dalen als we meerdere stalen mengen. Om dit effect te simuleren in ons model voegen we ruis toe. Elke meting y_i heeft een bepaalde kans om omgedraaid te



Figuur 10: Uitvoeringstijd van Mosek in functie van de groepsgrootte en de spaarsheid, gegeven in seconden. Het aantal metingen werd genomen zoals berekend in Figuur 6. We nemen het gemiddelde van 100 iteraties.

worden; een 0 wordt dan een 1 en omgekeerd. Deze kans blijft over het algemeen zeer laag. In onze modellen hebben we deze kans altijd op 5% gezet. De ruis zal het moeilijker maken om tot een correcte oplossing te komen. De gevolgen hiervan worden besproken in deel 8 en deel 9.

7.2 Formulering probleem met ruis

Om ruis op de metingen te modelleren maken we gebruik van slack variabelen $\xi_1, \ldots, \xi_m$. Als $\xi_i \geq 1$ betekent dit dat er ruis zit op meting i, zo niet is $\xi_i = 0$. Nu kunnen we het optimalisatieprobleem herformuleren, dit doen we weer naar analogie met de formulering in Malioutov en Malyutov [2012]:

$$min \sum_j x_j + \alpha \sum_i \xi_i \tag{6}$$

$$\text{zo dat: } 0 \leq \mathrm{x} \leq 1, \quad 0 \leq \xi, \quad \xi_{\mathcal{I}} \leq 1$$

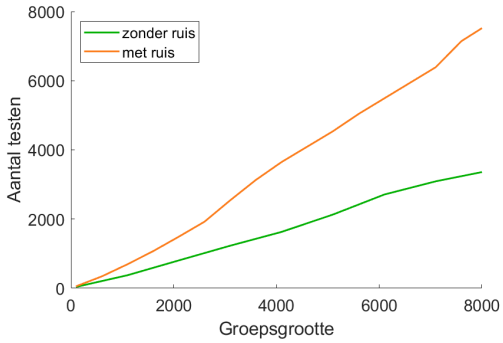
$$\mathrm{A}_{\mathcal{I}}\,\mathrm{x} + \xi_{\mathcal{I}} \geq \mathrm{y}_{\mathcal{I}}, \quad \mathrm{A}_{\mathcal{J}}\,\mathrm{x} = 0 + \xi_{\mathcal{J}}.$$

Ter herinnering: $\mathcal{I} = \{i \mid y_i = 1\}, \mathcal{J} = \{j \mid y_j = 0\}$.

In het model met ruis minimaliseren we dus niet enkel de som van x_j, maar ook die van ξ_i. Bovendien is er nu een bijkomende parameter α, deze stelt ons in staat om een bepaald gewicht toe te kennen aan het belang van de minimalisatie van x of van ξ. Zoals in Malioutov en Malyutov [2012] kiezen we $\alpha = 1$.

8 Voorbereidend werk model met ruis

Analoog aan het geval zonder ruis willen we het aantal metingen m zo klein mogelijk nemen om een maximale efficiëntie te garanderen, maar we willen ook zeker zijn dat we wel genoeg metingen doen zodat de reconstructie correct is.



Figuur 11: Het minimaal aantal metingen nodig om bij gegeven n en m een perfecte reconstructie te bekomen met ruis. We nemen k gelijk aan 0.02.

We voeren dus eerst een onderzoek uit naar het minimale aantal metingen dat we moeten doen bij een bepaalde groepsgrootte en een vaste spaarsheid. In volgende experimenten nemen we $k = 0.02$.

De resultaten van dit experiment zijn te zien in Figuur 11. We kunnen duidelijk zien dat het vereiste aantal metingen voor een perfecte reconstructie groter is wanneer er ruis aanwezig is dan wanneer er geen ruis is. Dit hadden we wel kunnen verwachten. Ook zien we dat de ruis er voor zorgt dat het aantal metingen m ongeveer even groot wordt als de groepsgrootte n.

Aan de hand van de informatie die we in de grafiek terugvinden, voeren we de tijdsmetingen uit.

9 Tijdsmetingen in setting met ruis

Zoals we voorheen gedaan hebben bij het ruisloos model kunnen we nu ook de tijdsmetingen uitvoeren voor het model met ruis. We verwachten hier hogere uitvoeringstijden aangezien het aantal testen hoger ligt.

9.1 Tijdsmetingen voor een vaste spaarsheid

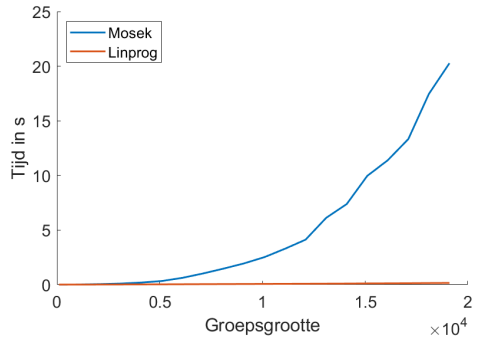
In Figuur 12 zien we de uitvoeringstijd van Linprog en Mosek in het model met ruis.

Hier zien we dat Mosek een tragere uitvoeringstijd heeft dan Linprog. Dit is opvallend aangezien Mosek voor het ruisloos model een lagere uitvoeringstijd heeft.

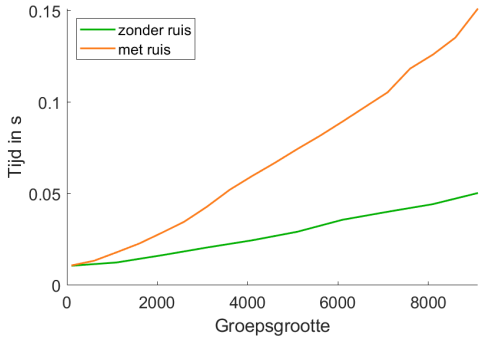
In Figuur 13 zien we de uitvoeringstijd voor Linprog met ruis vergeleken met die van Linprog zonder ruis. Deze figuur bevestigt onze verwachting dat de ruis de uitvoeringstijd vertraagt.

9.2 Tijdsmetingen voor een variërende spaarsheid

Net zoals in het ruisloos model kunnen we tijdsmetingen maken voor een variërende spaarsheid. Net zoals voorheen berekenen we eerst het benodigd aantal testen. Dit resultaat is te



Figuur 12: Tijdsmetingen voor Mosek en Linprog in het model met ruis. We nemen een gemiddelde van 100 iteraties en stellen k gelijk aan 0.02.

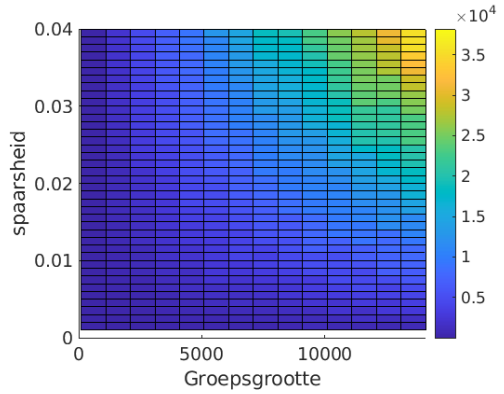


Figuur 13: Vergelijkende tijdsmetingen voor Linprog, enerzijds in het model met ruis en anderzijds in het model zonder ruis. We nemen een gemiddelde van 100 iteraties en stellen k gelijk aan 0.02.

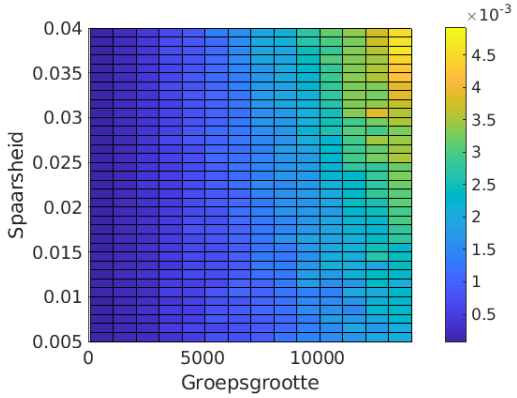
zien in Figuur 14. De tijdsmeting is te zien in Figuur 15. Hier merken we op dat we een gelijkaardig verband bekomen als in Figuur 11. De verandering in groepsgrootte heeft meer invloed op de uitvoeringstijd in vergelijking met de verandering in spaarsheid.

10 Conclusie

In deze studie hebben we de efficiëntie van compressed sensing onderzocht in de booleaanse setting. Dit resultaat zou bepalend kunnen zijn om te kiezen of men voor een gegeven toepassing deze techniek gaat gebruiken of niet. Voor bepaalde parameters hebben we aangetoond dat het lineair optimalisatieprobleem zeer snel kan opgelost worden, zelfs voor grote groepsgroottes. Dit zegt nog niets over de effectieve "efficiëntie" van de oplossingsmethode. Om te kunnen besluiten of dit nu efficiënt is of niet moeten we eerst gaan kijken wat efficiëntie precies betekent. Hier is geen eenduidig antwoord op, voor elke toepassing betekent dit namelijk iets anders. In een communicatienetwerk waar meerdere gigabytes per seconde wordt doorgestuurd zal deze aanpak niet efficiënt zijn. Anderzijds, als we het personeel van een ziekenhuis gaan testen op een bepaalde ziekte zou dit wel efficiënt kunnen zijn.



Figuur 14: Minimaal aantal vereiste testen in functie van groepsgrootte en spaarsheid in het model met ruis. We nemen een gemiddelde van 100 iteraties.



Figuur 15: Uitvoeringstijd van Linprog in functie van de groepsgrootte en de spaarsheid, gegeven in seconden in het model met ruis. Het aantal metingen werd genomen zoals berekend in Figuur 6. We nemen het gemiddelde van 100 iteraties.

In de setting van groepstesten, bijvoorbeeld voor SARS-CoV-2, kunnen we wel besluiten dat het algoritme snel genoeg is. De tijdsmetingen voor groepsgroottes en spaarsheden die overeenstemmend zijn met die binnen dit domein [Sciensano, 2021] tonen aan dat het probleem in enkele seconden opgelost kan worden. In deze setting zou een algoritme pas inefficiënt zijn als het meerdere uren of zelfs dagen duurt om het uit te voeren. Daarom stellen we vast dat compressed sensing hier theoretisch gezien wel toepasbaar is. Anderzijds is niet enkel de uitvoeringstijd van het algoritme belangrijk, het neemt praktisch gezien namelijk veel tijd en organisatie om dit proces in goede banen te leiden. [Sciensano, 2021]

We kunnen dus niet besluiten dat compressed sensing efficiënt is voor alle booleaanse problemen. Over de keuze van de parameters kunnen we wel iets zeggen. We hebben namelijk empirisch bepaald wat de grenzen zijn waarvoor het optimalisatieprobleem opgelost kan worden. Zo kan men aan de hand van de groepsgrootte en de spaarsheid bepalen hoeveel metingen men minimaal zal moeten doen.

11 Verder werk

Vele interessante onderwerpen hebben we nog niet kunnen onderzoeken. Zo zijn we geïnteresseerd in wat de maximale waarde is voor n die de software en onze computer aankan.

In verwante werken wordt al aangehaald hoe de metingsmatrix op verschillende manieren gevormd kan worden [Candes and Wakin, 2008]. In nog andere werken werd al onderzoek gedaan naar het effect hiervan op de minimaal vereiste aantal metingen [Aldridge *et al.*, 2019]. Wij zijn benieuwd naar het effect hiervan in de booleaanse setting.

Elk groepstestprobleem introduceert een verschillende hoeveelheid ruis op de metingen. Het zou dus interessant zijn om te zien hoe het aantal metingen en de uitvoeringssnelheid veranderen in functie van de hoeveelheid ruis.

Referenties

[Aldridge *et al.*, 2019] M. Aldridge, O. Johnson, and J. Scarlett. Group testing: An information theory perspective. *Foundations and Trends® in Communications and Information Theory*, 15(3-4):196–392, 2019.

[Atia and Saligrama, 2012] G. K. Atia and V. Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901, 2012.

[Candes and Wakin, 2008] E. J. Candes and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[Dorfman, 1943] R. Dorfman. The Detection of Defective Members of Large Populations. *The Annals of Mathematical Statistics*, 14(4):436 – 440, 1943.

[GNU, 2022] GNU. Gnu linear programming kit, 2022.

[Malioutov and Malyutov, 2012] D. Malioutov and M. Malyutov. Boolean compressed sensing: Lp relaxation for group testing. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3305–3308, 2012.

[MathWorks, 2022] MathWorks. Matlab optimization toolbox, 2022.

[MOSEK ApS, 2022] MOSEK ApS. Mosek, 2022.

[Sciensano, 2021] Sciensano. Advice on pooling of samples for detection of sars-cov-2 with rt-pcr. 2021.