**Simulated annealing:**

Bij onze simulated annealing zijn er grofweg twee “**problemen**” waarmee rekening gehouden moet worden met betrekking tot de case protein pow(d)er

**Probleem 1)**

Onze hillclimber en dus ook onze simulated annealing neemt grote stappen: hele stukken ketens worden omgeklapt, wat ertoe kan leiden dat de keten er na 1 iteratie meteen heel anders uit kan zien. Het gevaar hierbij is, dat wanneer er in een laat stadium een verslechtering wordt geaccepteerd, de keten meteen dusdanig kan verslechteren, dat er geen goede score meer behaald kan worden.

**Probleem 2)**

Kleine verschillen in score wegen zwaar. Bijvoorbeeld: een nieuwe score van -5 ten opzicht van een oude score van -7, betekent een cost van “maar” -2. Echter, wanneer je bedenkt dat de maximaal haalbare score -8 is, is een verslechtering van 2 punten wel degelijk veel.

Uit bovenstaande kunnen twee **onderzoeksvragen** geformuleerd worden:

**Onderzoeksvraag 1)**

Kunnen er in de protein pow(d)er case betere scores behaald worden met simulated annealing door een cooling schedule te gebruiken waarbij de temperatuur al vroeg in de iteraties richting 0 gaat?

**Hypothese 1):** Een cooling schedule waarbij de temperatuur vroeg richting 0 gaat, levert betere scores op dan een cooling schedule waarbij de temperatuur geleidelijk over de iteraties richting 0 gaat.

**Onderzoeksvraag 2)**

Kunnen er in de protein pow(d)er case betere scores behaald worden met simulated annealing door verslechteringen zwaarder te laten wegen?

**Hypothese 2):** Er kunnen betere scores behaald worden door verslechteringen zwaarder te laten wegen.

**ONDERZOEK 1:**

**Experiment:** Om onderzoeksvraag 1 te beantwoorden zal een exponentiele cooling schedule vergeleken worden met een lineaire cooling schedule. Voor dit experiment zal de 2D simulated annealing worden gerund met als start een rechte keten. Elk van de cooling schedules zal 10 keer worden gerund met 10.000 iteraties, waarover de gemiddelde score per iteratienummer zal worden bepaald. Elke run begint met een temperatuur van 100 (arbitrair) en eindigt met een temperatuur van 1 (laagst mogelijk).   
Er is gekozen voor een dimensie van 2D, omdat hiervan de state space kleiner is dan in 3D. Hierdoor kan met een relatief minder aantal iteraties hetzelfde resultaat behaald worden. Er worden 10 000 iteraties gemaakt per run, omdat dit uit een aantal pilot tests goede scores leek op te leveren, zonder dat het erg lang duurt. De kortste en de langste H/P-ketens zullen worden getest

**Voorspelling:** de exponentiele cooling schedule zal betere score opleveren dan de lineaire cooling schedule

|  |  |  |
| --- | --- | --- |
| **Plot cooling schedule** | **Files** | **Formula cooling schedule** |
|  | cooling\_exp.csv  cooling\_exp.png | T\_current = T\_begin \* math.pow((T\_end / T\_begin), ((current\_iteration \* 1 ) / total\_iterations)) |
|  | cooling\_linear.csv  cooling\_linear.png | T\_current = T\_begin - current\_iteration \* (T\_begin - T\_end) / total\_iterations |

**Resultaten:**

|  |  |  |
| --- | --- | --- |
| **Sequence** | **Plot data** | **Files** |
| 1)  HHPHHHPHPHHHPH |  | Data:  simulatedannealing\_sequence1\_2d\_expo.csv  simulatedannealing\_sequence1\_2d\_lin.csv  Plot:  simulatedannealing\_sequence1\_scoreverloop\_expo\_vs\_ linear.png |
| 4)  HHPHPHPHPHHHHPHPPPHPPPHPPPPHPPPHPPPHPHHHHPHPHPHPHH |  | Data:  simulatedannealing\_sequence4\_2d\_expo.csv  simulatedannealing\_sequence4\_2d\_lin.csv  Plot:  simulatedannealing\_sequence4\_scoreover\_expo\_vs\_linear.png |

**Conclusie**:

Bij de korte keten is er nog niet heel duidelijke een verschil te onderscheiden. Toch is te zien dat de exponentiele cooling schedule gemiddeld iets betere scores behaald dan de lineaire cooling schedule. Dit verschil wordt duidelijker te zien met de wat langere ketens. De exponentiële cooling schedule lijkt dus beter te werken. Het is daarom aannemelijk dat een cooling schedule waarbij de temperatuur vroeg richting 0 gaat betere scores oplevert dan een cooling schedule waarbij de temperatuur geleidelijk over de iteraties richting 0 gaat.

**Discussie**

Dat er bij de lange keten duidelijkere verschillen zijn tussen beide cooling schedules komt mogelijk doordat het bij de langere ketens moeilijker is om in een laat stadium vanuit een verslechtering weer terug te gaan naar een betere score. Veel vouwingen zijn dan al niet meer mogelijk doordat de ketens over elkaar heen komen te liggen. Hierdoor wordt het effect dat wordt beschreven in “probleem 1” duidelijker zichtbaar.

Ondanks dat de exponentiele cooling schedule het iets beter doet dan de lineaire cooling schedule, zijn de scores aan de lage kant. Uit eerdere tests met de breadth first was gebleken dat de hoogst haalbare score voor sequentie 1 een score van -6 is. Voor sequentie 4 is een upperbound score berekend van -24. Mogelijk loopt de temperatuur in deze exponentiële cooling schedule niet vroeg genoeg richting 0 om de beste score te behalen. Het feit dat de scores tot het einde toe heel wisselvallig zijn, duidt hier ook op: er worden (tot aan het einde toe) nog te veel verslechteringen worden geaccepteerd   
In vervolg onderzoek zullen exponentiele cooling schedules met elkaar worden vergeleken waarbij de temperatuur sneller richting 0 loopt

**VERVOLG ONDERZOEK 1**

|  |  |  |
| --- | --- | --- |
| **Plot cooling schedule** | **Files** | **Formula cooling schedule** |
|  | cooling\_expo.csv  cooling\_expo.png | T\_current = T\_begin \* math.pow((T\_end / T\_begin), ((current\_iteration \* 1 ) / total\_iterations)) |
|  | cooling\_expo1.csv  cooling\_expo1.png | T\_current = T\_begin \* math.pow((T\_end / T\_begin), ((current\_iteration \* 1.5 ) / total\_iterations)) |
|  | cooling\_expo2.csv  cooling\_expo2.png | T\_current = T\_begin \* math.pow((T\_end / T\_begin), ((current\_iteration \* 2 ) / total\_iterations)) |
|  | cooling\_expo3.csv  cooling\_expo3.png | T\_current = T\_begin \* math.pow((T\_end / T\_begin), ((current\_iteration \* 4 ) / total\_iterations)) |

**Resultaten**

|  |  |  |
| --- | --- | --- |
| **Sequence** | **Plot data** | **Files** |
| 1)  HHPHHHPHPHHHPH |  | Data:  simulatedannealing\_sequence1\_2d\_expo.csv  simulatedannealing\_sequence1\_2d\_expo1.csv  simulatedannealing\_sequence1\_2d\_expo2.csv  simulatedannealing\_sequence1\_2d\_expo3.csv  Plot:  simulatedannealing\_sequence1\_scoreverloop\_expo\_vs\_ expo.png |
| 4)  HHPHPHPHPHHHHPHPPPHPPPHPPPPHPPPHPPPHPHHHHPHPHPHPHH |  | Data:  simulatedannealing\_sequence4\_2d\_expo.csv  simulatedannealing\_sequence4\_2d\_expo1.csv  simulatedannealing\_sequence4\_2d\_expo2.csv  simulatedannealing\_sequence4\_2d\_expo3.csv  Plot:  simulatedannealing\_sequence4\_scoreover\_expo\_vs\_expo.png |

**Conclusie**

Zowel bij de korte als bij de lange sequentie is duidelijke te zien dan exponentieel 1, 2 en 3, het beter doen dan exponentieel. Bij de korte keten lijkt het er zelfs op, dat hoe sneller de temperatuur richting 0 gaat, hoe betere scores er bereikt worden. Daarentegen is dit bij de lange keten niet het geval: exponentieel 1, 2 en 3 blijven tot het einde toe vrij instabiel en verslechteren naar het einde toe flink. Exponentieel 1 eindigt op de beste score. De resultaten van dit experiment suggereren dat het niet perse betekent dat hoe sneller de score naar 0 loopt hoe beter de resultaten zijn, maar dat er een gulden middenweg is en dat de ene cooling schedule niet per se even goed werkt voor alle ketens. Exponentieel 1 lijkt het over het algemeen het beste te doen.

**Discussie**

Het is opmerkelijk dat er bij de lange ketens naar het einde toe nog zoveel verslechteringen worden geaccepteerd. Aangezien bijvoorbeeld bij exponentieel 3 de temperatuur rond 3000 iteraties al bijna op 0 zit: wat betekent dat de kans om verslechteringen te accepteren erg klein is. Echter lijkt het er op dat vanaf 3000 iteraties juist méér verslechteringen geaccepteerd worden. Dit is niet helemaal wat je zou verwachten. In vervolg onderzoek zou het experiment nog een keer gerepliceerd kunnen worden waarbij er goed gelet wordt op de acceptatiekans per iteratie om te kijken of de formule voor acceptatiekans (acceptance\_prob = math.e\*\*(cost / T\_current)) nog steeds goed werkt voor zulke kleine getallen van T\_current. Daarnaast zou het interessant zijn om probleem 1 aan te pakken, door onderzoek te doen naar een betere formule voor de acceptatiekans (die de costen zwaarder laat wegen). Helaas zijn wij hier in verband met tijd niet meer aan toe gekomen.