Take home-opdracht over metaheuristieken

Nu je met de 3 frameworks gewerkt hebt, kies je één van de drie frameworks of een ander framework om volgende opdracht te implementeren. Het betreft een zg. exam timetabling benchmark van de University of Toronto. Deze benchmark is één van de oudste benchmarks voor exam timetabling en is dus ook zeer goed bestudeerd in de literatuur. Merk op dat deze benchmark geen realistische data bevat. Zo wordt er geen rekening gehouden met de capaciteit van de examenlokalen. Men gaat ervan uit dat de lokalen oneindig groot zijn.
De benchmarks kunnen teruggevonden op de Benchmark Data Sets in Exam Timetabling webpagina van prof. Rong Qu.Zie http://www.cs.nott.ac.uk/~pszrq/data.htm
Eigenlijk zijn er voor dit probleem maar 2 beperkingen waarmee je rekening dient te houden:

- 1 harde beperking die stelt dat studenten maar 1 examen per tijdslot kunnen doen
- 1 zachte beperking die studenten zoveel mogelijk tijd wilt geven tussen 2 opeenvolgende examens. Hoe minder tijd er is tussen de examens hoe hoger de kost zal zijn. De kostfunctie ziet er als volgt uit:
    - 16 als een student 2 aaneensluitende examens heeft
    - 8 als er 1 tijdslot tussen 2 opeenvolgende examens zit
    - 4 als er 2 tijdsloten tussen 2 opeenvolgende examens zit
    - 2 als er 3 tijdsloten tussen 2 opeenvolgende examens zit
    - 1 als er 4 tijdsloten tussen 2 opeenvolgende examens zit

Op de website van prof. Qu vindt je ook een applicatie terug waarmee je de kost van je eigen oplossing kan berekenen.

Implementeer in 1 van de 3 bovenstaande frameworks (of in een ander framework) het exam timetabling problem van de University of Toronto en test je code op 1 van de 13 data sets.
Experimenteer met:

- 2 verschillende zoekalgoritmes en bekijk welke beter is voor dit probleem of
- als je OpenTS gebruikt met 2 verschillende soorten moves. Je kan nl. meer dan 1 move gebruiken. Zo kan je bijv. als een bepaalde move al een tijdje geen betere oplossing meer oplevert, beslissen om een andere soort move te gaan gebruiken.

Zorg ervoor dat de oplossing die je bekomt feasible is (maw dat er geen harde beperkingen overtreden zijn).

TIP: probeer een feasible startoplossing te maken, en probeer ervoor te zorgen dat je moves geen harde beperkingen overtreden.
TIP: start eerst met de oplossing volledig te berekenen adhv de kostfunctie. Concentreer je achteraf op het correct berekenen van de aangepaste oplossing adhv de delta-evaluatie indien je Optaplanner niet gebruikt.
TIP: op Toledo vindt je een IntelliJ-project met daarin reeds de code om de benchmarks in te lezen en om te zetten naar studenten en examens.

Test je gevonden oplossing met de applicatie die je kan vinden op de exam timetabling website.

Schrijf een verslag met daarin minstens

- een beknopte literatuurstudie
    - bestudeer gelijkaardige problemen (exam en course timetabling) uit de literatuur
    - hoe zijn die opgelost?
- het framework dat je gebruikt hebt
- hoe je je oplossing gemodelleerd hebt (hoe stel je een oplossing voor in je code?)
- de zet(ten) die je geïmplementeerd hebt. Geef uitleg wat er juist uitgevoerd wordt.
- hoe je de kostfunctie geïmplementeerd hebt
- de resultaten van je testen.

Je code (gezipt) en verslag laadt je op op de voorziene plaats op Toledo **voor 31/12 om 20 uur**.

Puntenverdeling:

- take home-opdracht:
    - verslag: **50**% van de punten
        - literatuurstudie
        - welk framework?
        - modellering van de oplossing
        - zetten
        - kostfunctie
        - resultaten
    - eigenlijke code exam timetabling: **50**% van de punten
        - heb aandacht voor de juiste datastructuren
            - denk hierbij aan de snelheid (zie Algo & Data)
            - OO
        - juistheid van de oplossing