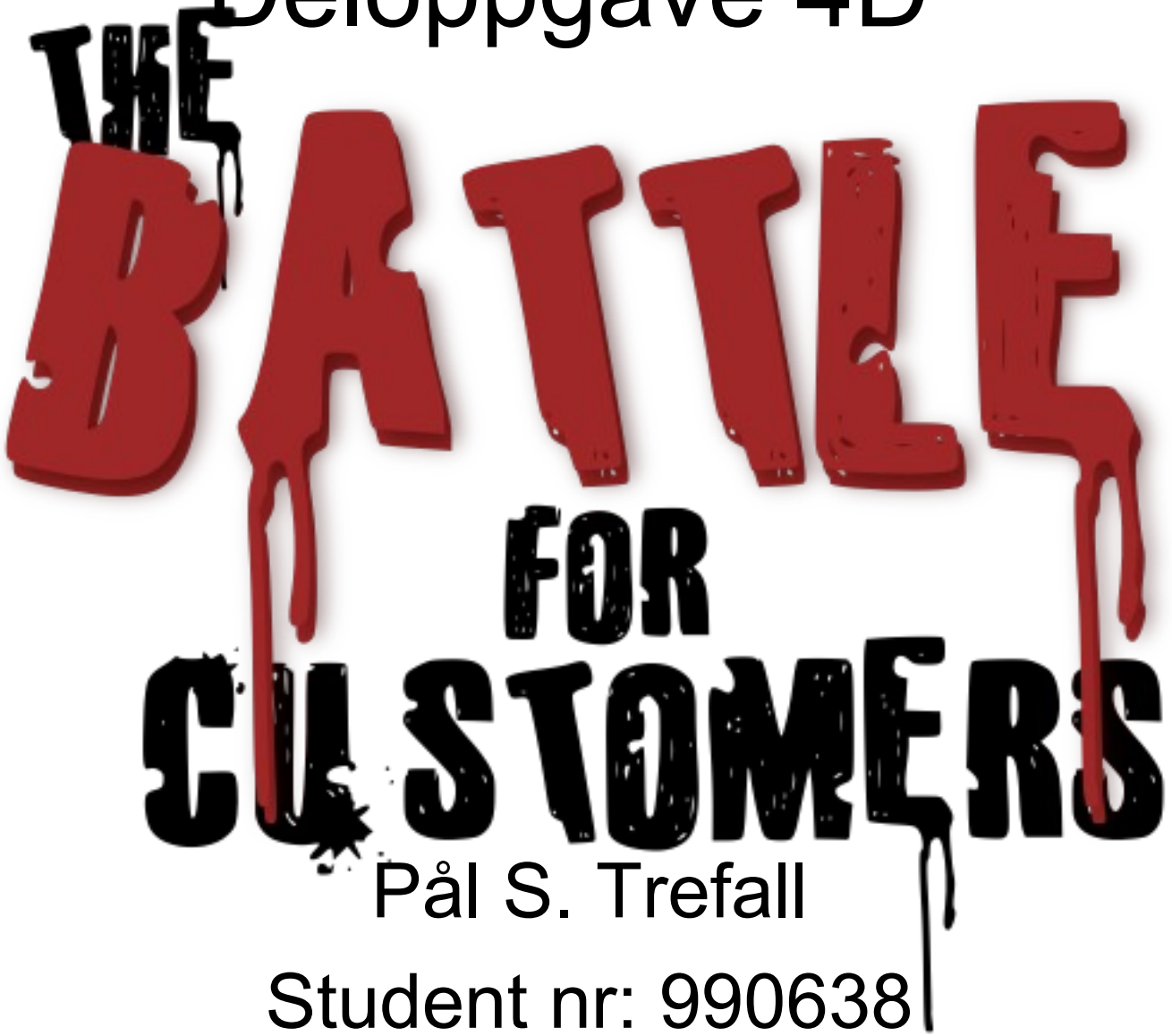
Deloppgave 4D



Pål S. Trefall

Student nr: 990638

HIN 21.12.2011

Oppgavebeskrivelse

”**The Battle for customers**”

- **Prosumentagentene** (*Prosumer*) er karakterisert som følger:
  - Økonomisk kapasitet
  - Egenproduksjonskapasitet
  - Energiforbruk
  - Brukerfleksibilitet
  - Policy/strategi

- **Leverandøragenter** (*Supplier*) finnes tre typer:
  - Fastpris
  - Spotprisbasert
  - Hybrid pris, som blander litt fastpris og litt spotpris

(har samarbeidet med medstudent Gunnar Salberg med denne oppgaven – dette er først og fremst definisjon av matematikk – ogi forutsetninger som manglet, uoppstilte ligninger)

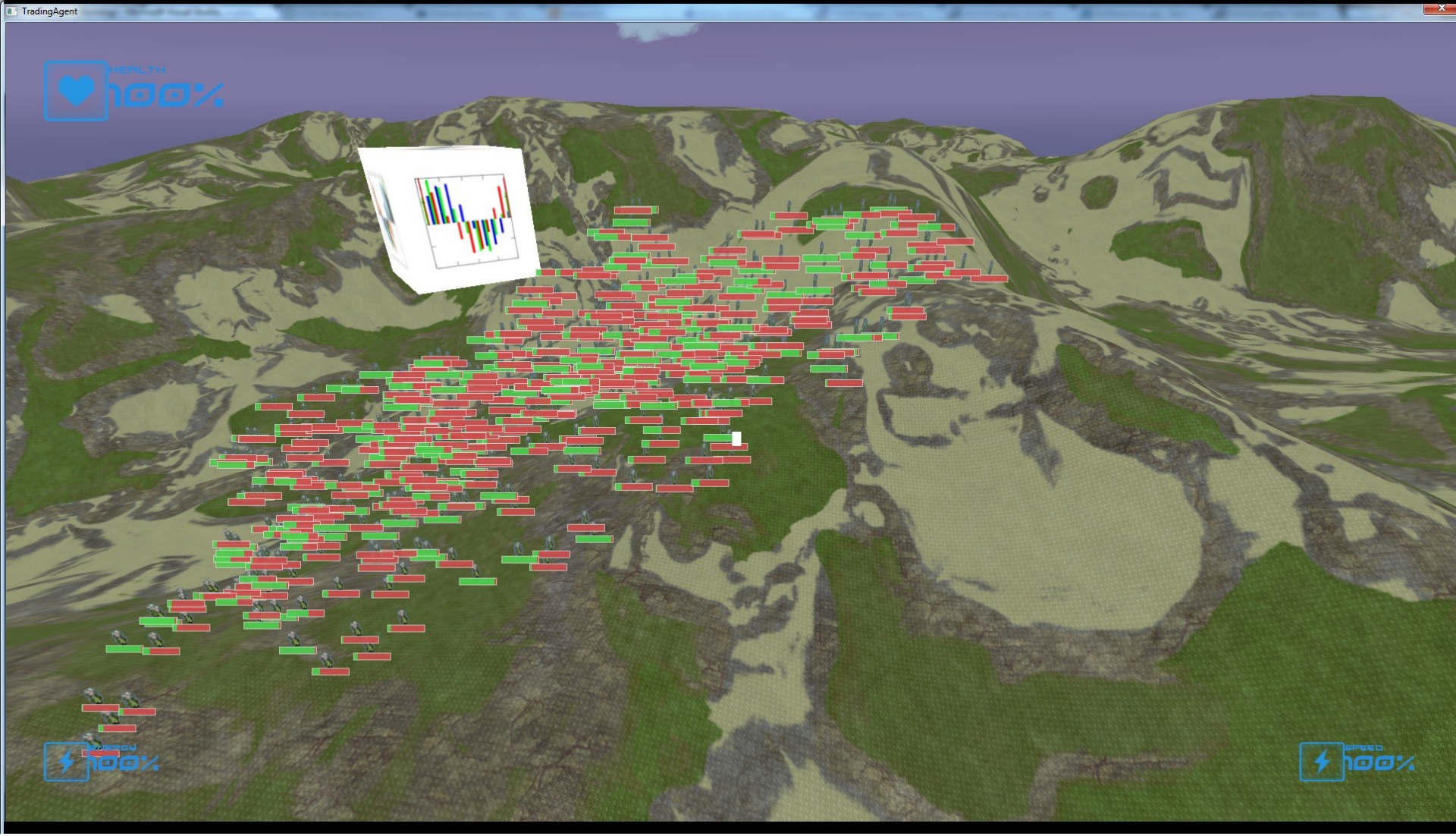
Each individual on the terrain represents a prosumer individual.
The bar under the prosumer represents it's saldo/fitness.
Green is good, red is bad.
When entire bar is red, prosumer is dead

Forutsetninger, antagelser og avgrensninger

```
ProsumerGeneticAlg(
        40,  //Population Size
        0.0,  //Fitness threshold
        0.02,  //Chance for crossover
        1,   //Max children from crossover
        0.05,  //Chance for mutation
        0.1,  //Start saldo
        12.0,  //Øk
        1.0,  //Ep
        24000.0, //Ef
        0.1,  //Flex
        0);   //Policy
```

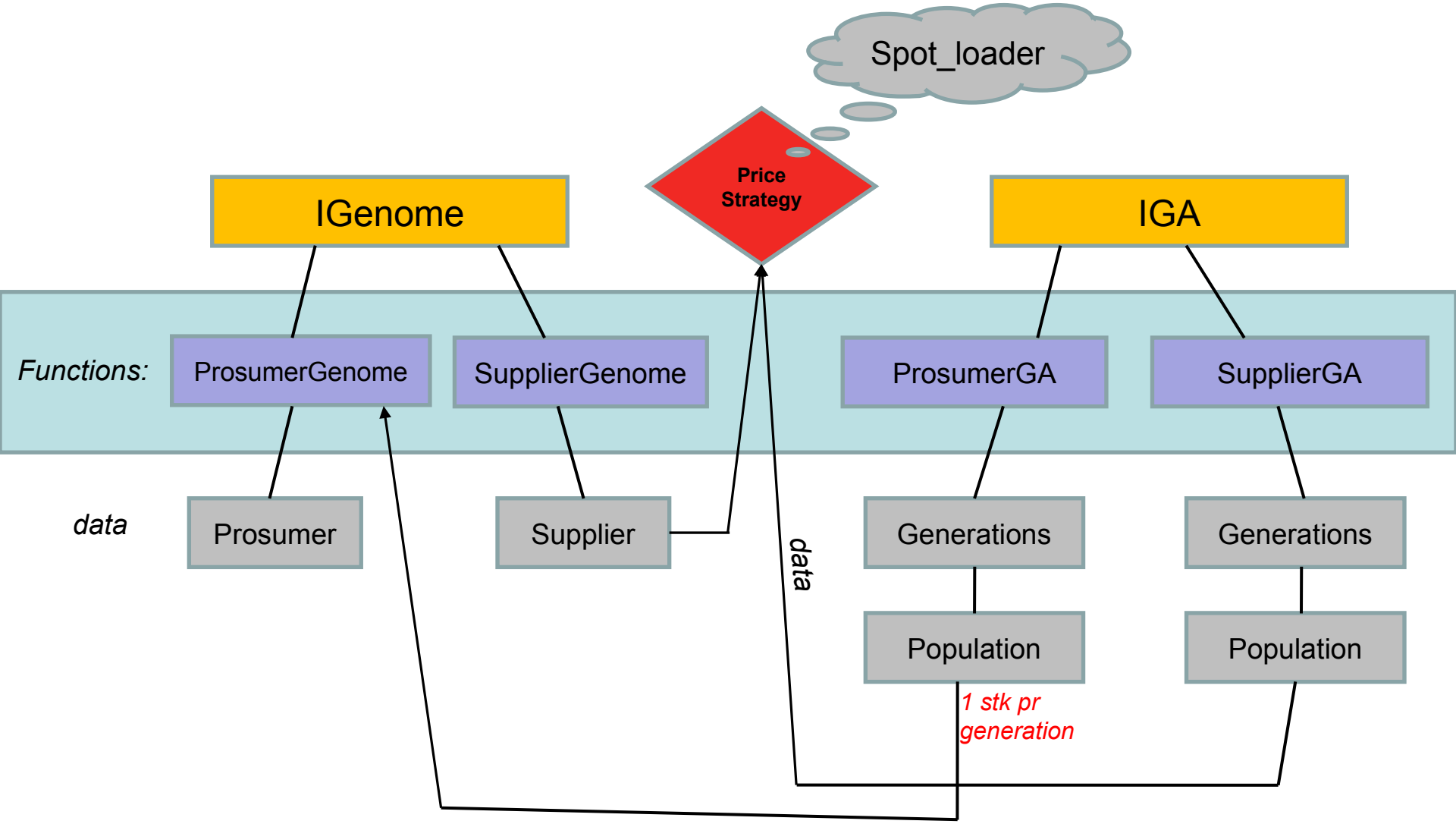
```
SupplierGeneticAlg(
        10,  //Population Size
        0.0,  //Fitness threshold
        0.05,  //Chance for crossover
        1,   //Max children from crossover
        0.1,  //Chance for mutation
        0.1,  //Start saldo
        0.15,  //Price Offer
        10.0,  //Supply capacity
        0.05,  //Participation cost
        0.4,    //Hybrid % of Spot Price
        0.75); //Hybrid % of Fixed Price
```

- `prosumer.saldo = (1.0 - (ef_lifetime_reduction_factor+bills_lifetime_reduction_factor)) * prosumer.saldo;`
- `supplier.saldo = (1.0 - price_factor) + customer_factor - participation_factor_accumulator;`



Resulting population after 100 generations

Genetisk algoritmer (GA)



Beskrivelse av løsningsstrategi og metode anvendt

- getProsumerPopulationSize()
- getProsumerEconomicCapacity()
- getProsumerEnergyConsumption()

- //Helper functions for Bargaining
  - findBestPriceOffer()
  - findWorstPriceOffer()

- //Helper functions for Supplier GA
  - getSupplierPopulationSize()
  - getSupplierPriceOffer()
  - getSupplierUnreservedSupplyCapacity()

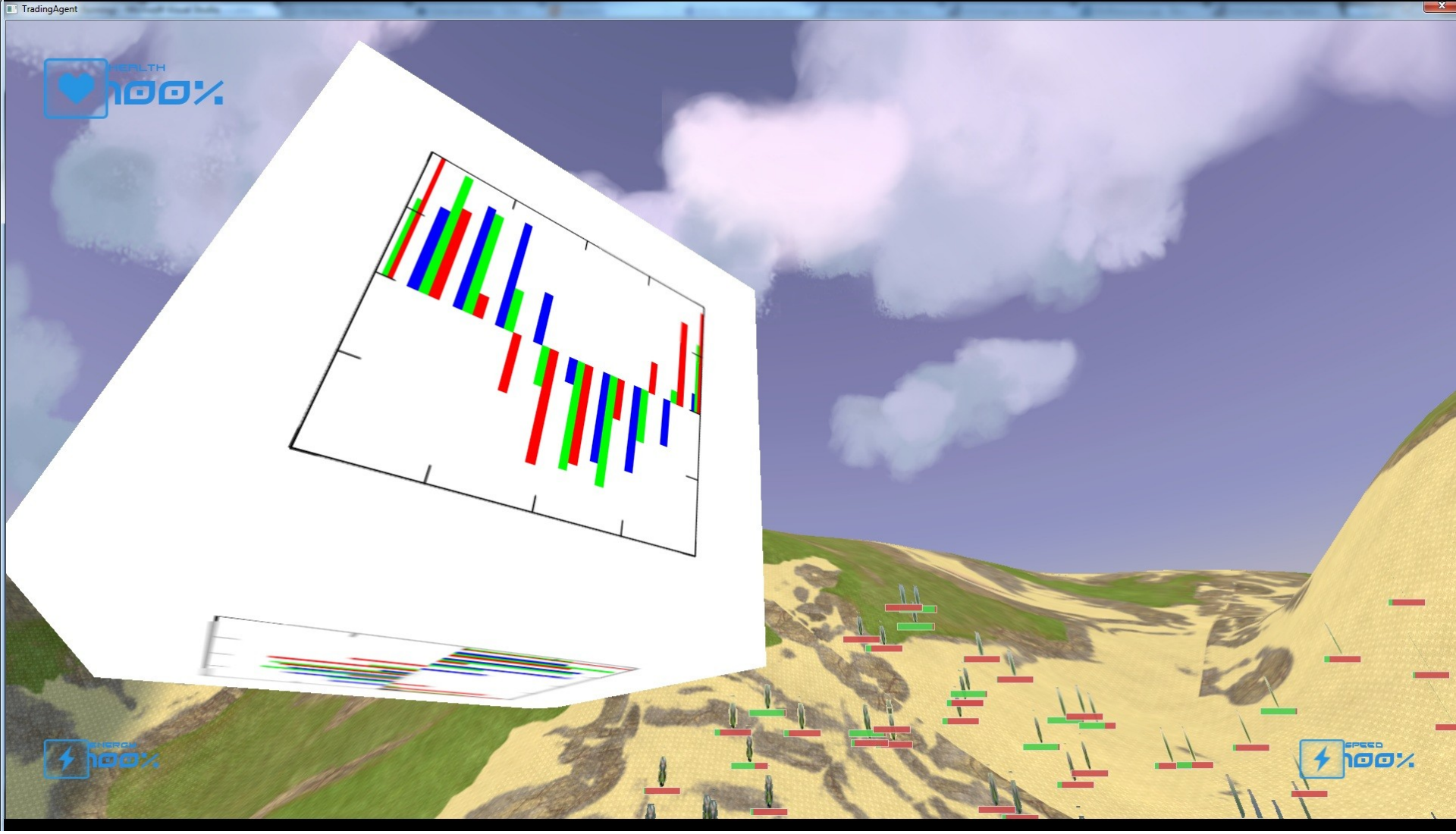
- //For each customer reserving energy
  - getSupplierSupplyCapacity()
  - getSupplierSupplyCapacitySUM()
  - getSupplierCustomerCount()

- Economic Capacity
- Energy Production Capacity
- Energy Consumption
- Flexi Rate
- Policy

The user can move about the scene freely and look at the status of individual prosumers. The idea was to be able to click on each to get more specific information, but there was never enough time to complete this feature.

En beskrivelse hva systemet gjør

- Systemet velger ut beste energileverandør til mange kunder
  - Billigste leverandør velges først
  - Det kan forhandles, +- 15%
  - Det går i epoker
- ”Kunder” (customer) formerer seg uendelig til de dør ut
  - Dør når de ikke har økonomisk kapasitet til å kjøpe energi
  - Dør når de ikke får dekket sitt energibehov
  - Forlenger levetiden når den produserer egen energi, men stor innvesteringskostnad
  - Egen fleksibilitet forlenger også livet til kundene
- ”Leverandører” (suppliers) leverer så lenge de er konkuransedyktige
  - Dør når den ikke har kunder
  - Dør når kostnader for energi ikke er konkurransedyktig, og det ikke er nok kunder
  - Jo eldre en leverandør er, jo mer har leverandør betalt i faste avgifter
  - Elitebetraktning, de med høyest saldo rangerer høyest og får krysse
- Pris:
  - Fastpris er fast, og variasjoner mellom ulike leverandører vises
  - Spotpris er variabel og baseres på timesverdier fra Troms (www.norpol.no)
  - Hybrid er litt av fast og litt av spot.



The cube of plotting was supposed to show the best genome each generation, adding another status bar on the cube's graph. There was never enough time to complete this idea either.

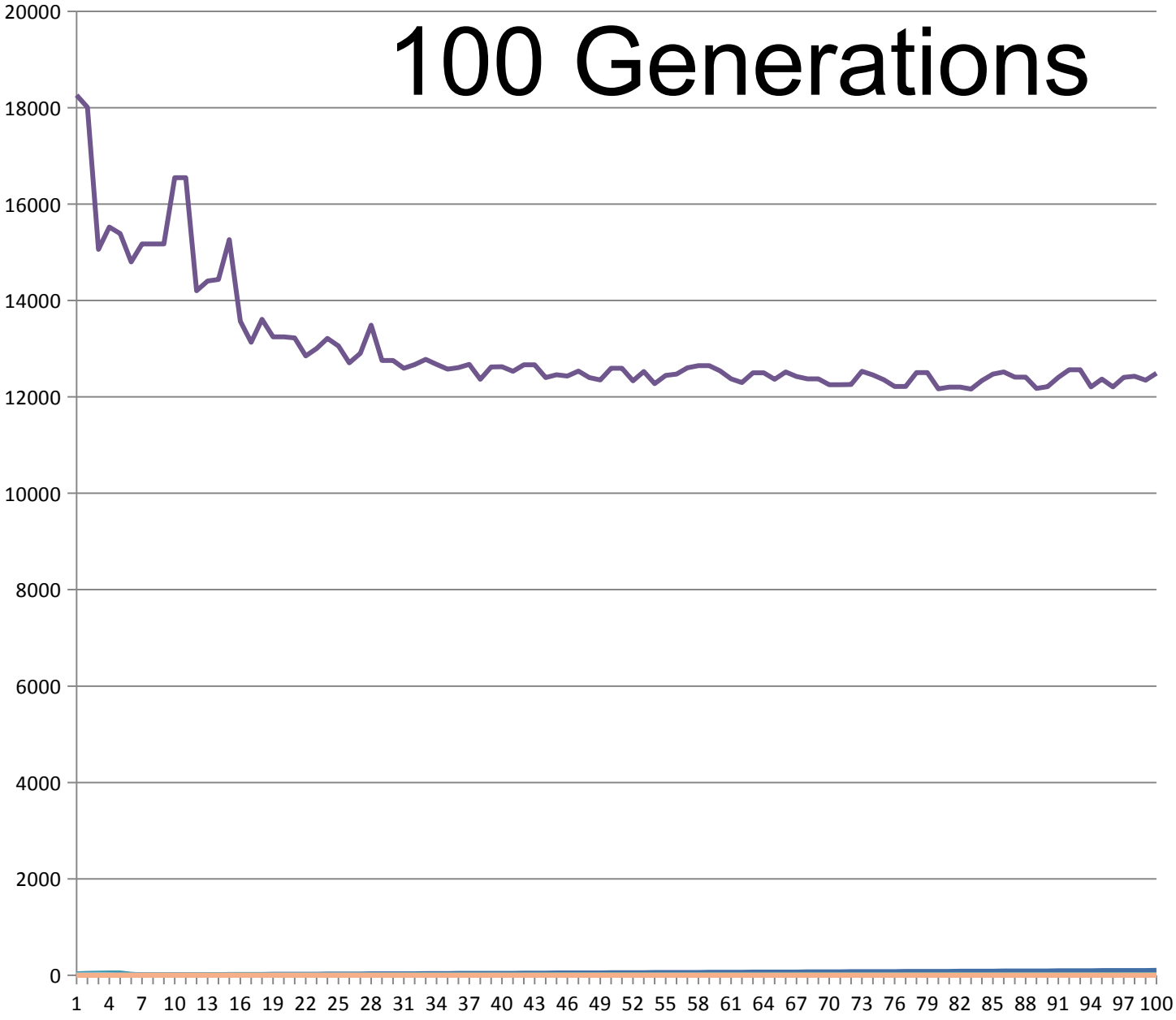
Arkitektur og hovedkomponenter

- GA arkitekturen består i bunnen av et IGenome interface og et IGeneticAlg interface. Individer overlaster IGenome, og selve implementasjonen av den spesifike genetiske algoritme med tilhørende generasjoner og populasjoner overlaster IGeneticAlg.
- Irrlicht3D brukt til å tegne scenen. Støtter både å tegne på CPU (software) og på GPU (OpenGL). Anbefalt visning er helt klart OpenGL for denne oppgaven.
- Komponent-basert entity system, Totem, skrevet av meg, er brukt for å håndtere alle de grafiske individene i verdenen på en fleksibel måte.
- MathGL er brukt for å tegne grafer (men fikk desverre ikke tid til å vise dette skikkelig).
- Sanntid dag/natt syklus (OpenGL versjon) skulle egentlig følge timene riktig og gi et inntryk av døgn variasjon. Ble ikke tid å synkronisere dette, men effekten er like vel lagt inn.
- Arkitekturen lagt til grunn gjør det enkelt å utvide denne oppgaven ytterligere, ved grafisk representasjon av leverandører, som det ikke ble tid til å gjøre.
- Det er også veldig enkelt å legge inn nye typer genetiske algoritmer, selv om denne delen av arkitekturen kunne hatt godt av et par evolusjoner, da den bærer noen preg av førstegangsimplementasjon.

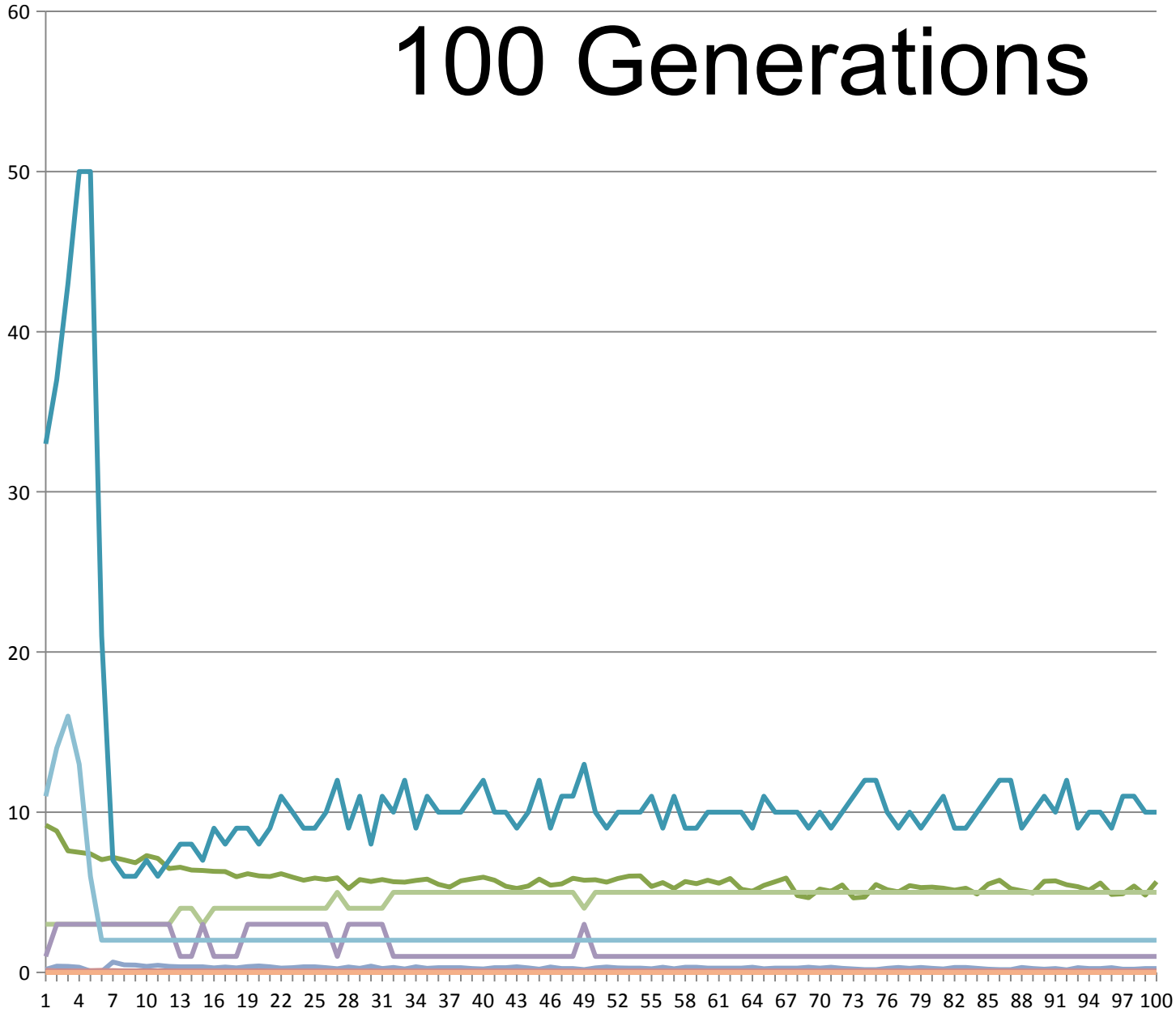
Hva slags resultater produseres

- Beskriv hva som skjer ved 100 og 500 hundre epoker.
- Hvilke kunder dominerer?
- Hva slags leverandører og leverandørtyper dominerer? Fastpris, spotpris etc.
- Er det noen av populasjonene som dør ut?
- Hvilken leverandør gjør det best ved epoke = 500

Vi ser raskt over påfølgende grafer av kjørte data for et inntrykk (se Grafer_Paal.ods for nærmere innblikk i denne dataen)

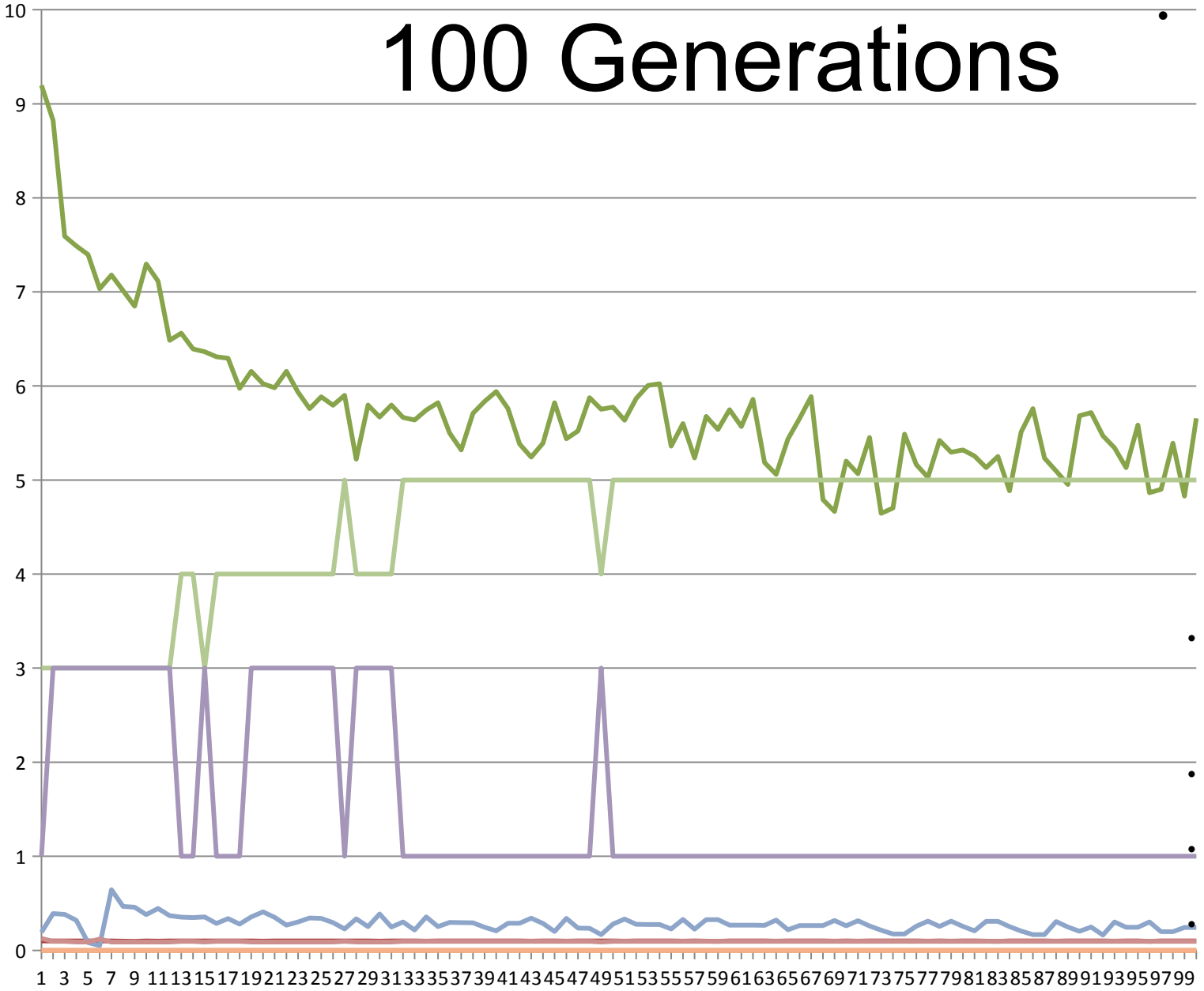


- Vi ser at energi forbruk av beste genom reduseres og stabiliserer seg.



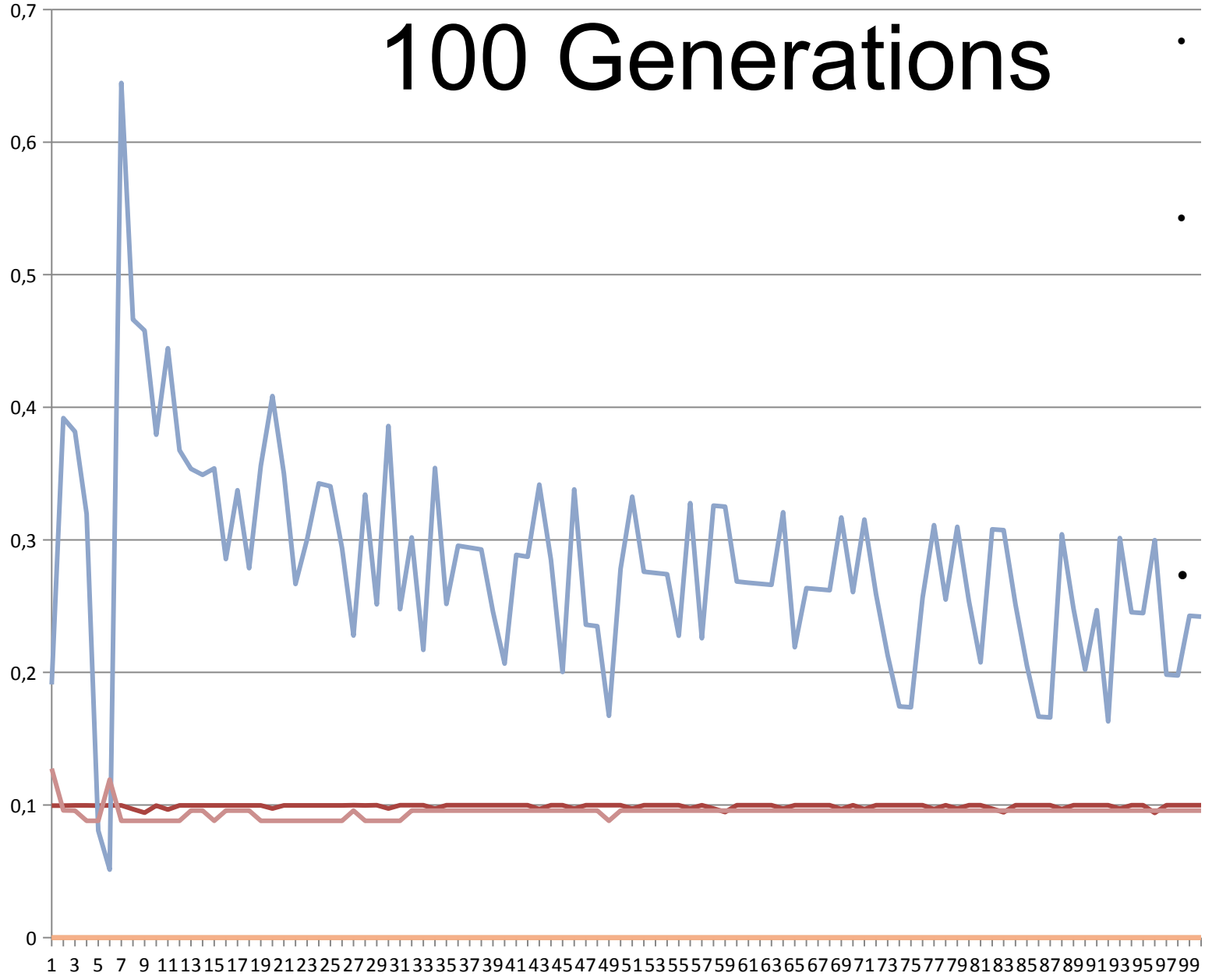
- Basert på input data ser vi en kraftig vekst med påfølgende fall i populasjon, før stabilitet.

100 Generations



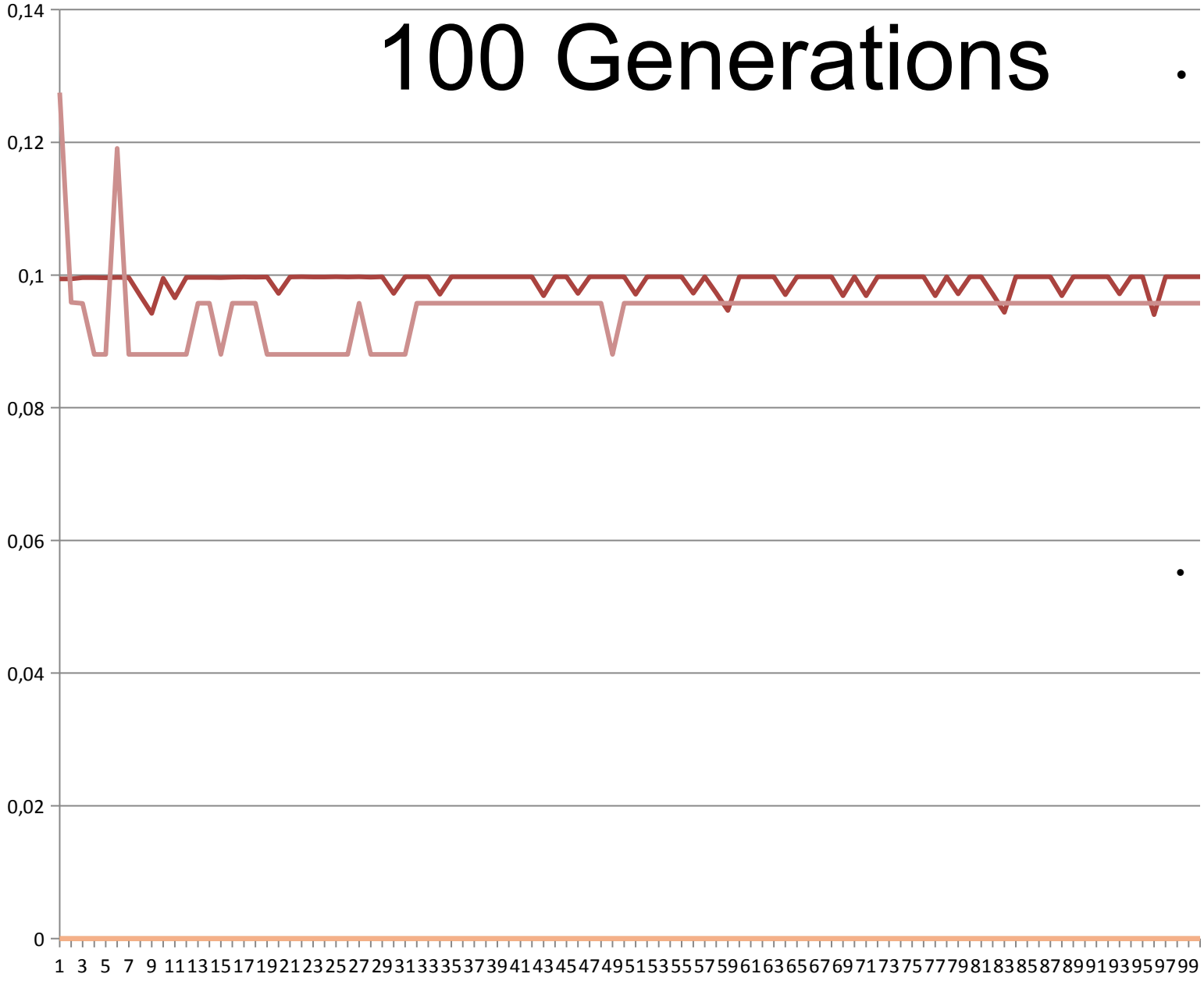
- Vi ser at energi forbruk av beste genom reduseres og stabiliserer seg.
- Vi ser at antall kunder øker hos beste leverandør (spesialisering).
- Pris strategi lik 1 er fixed-price.
- Pris strategi lik to er spot-price.
- Pris strategi lik tre er hybrid-price.

100 Generations



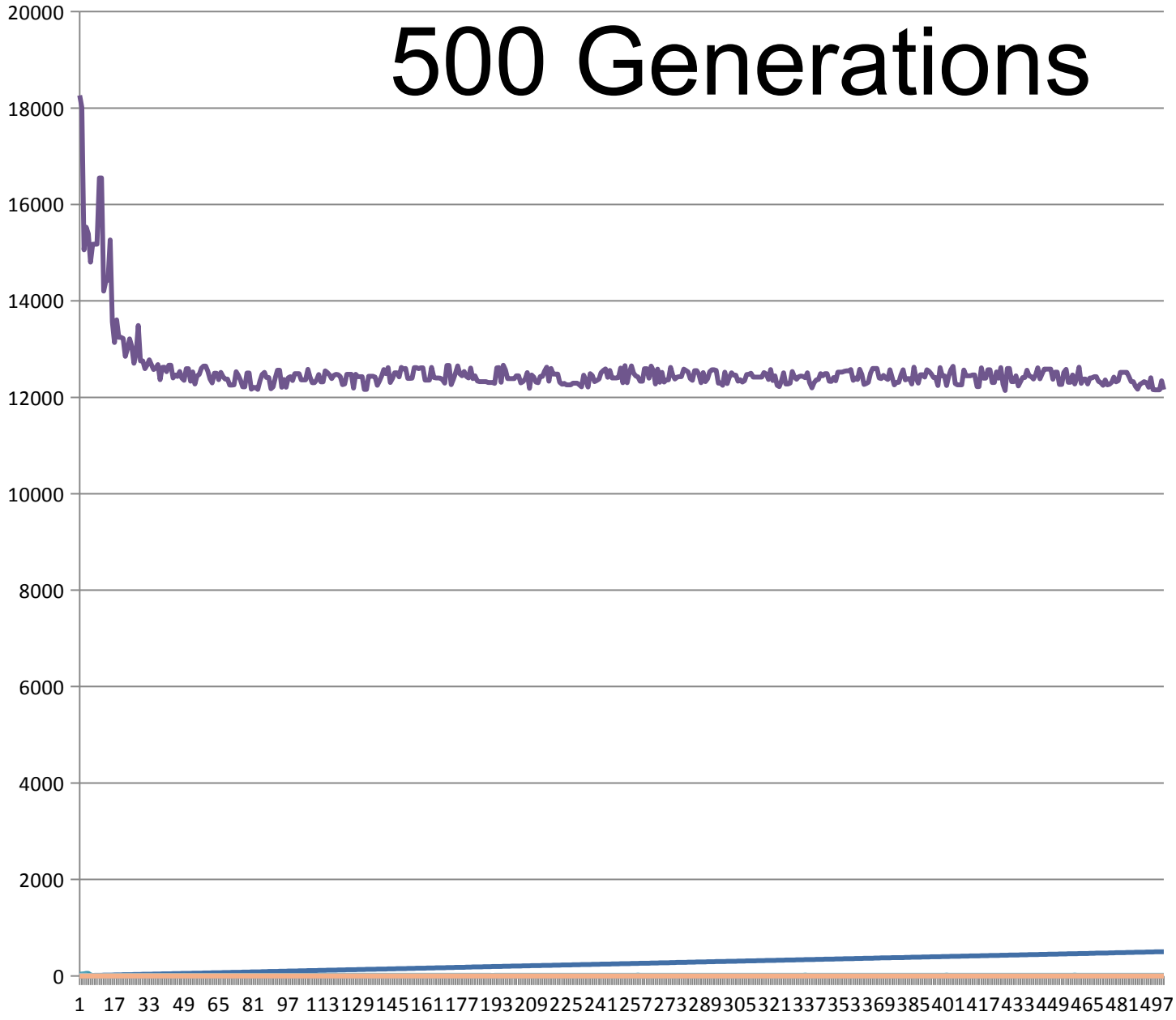
- I starten ser vi hvordan leverandør's saldo er veldig ustabil.
- Oppførselen reflekterer godt hva vi såg for populasjonen.
- Vi merker oss at leverandøren s saldo er synkende men også stabiliserende .

100 Generations

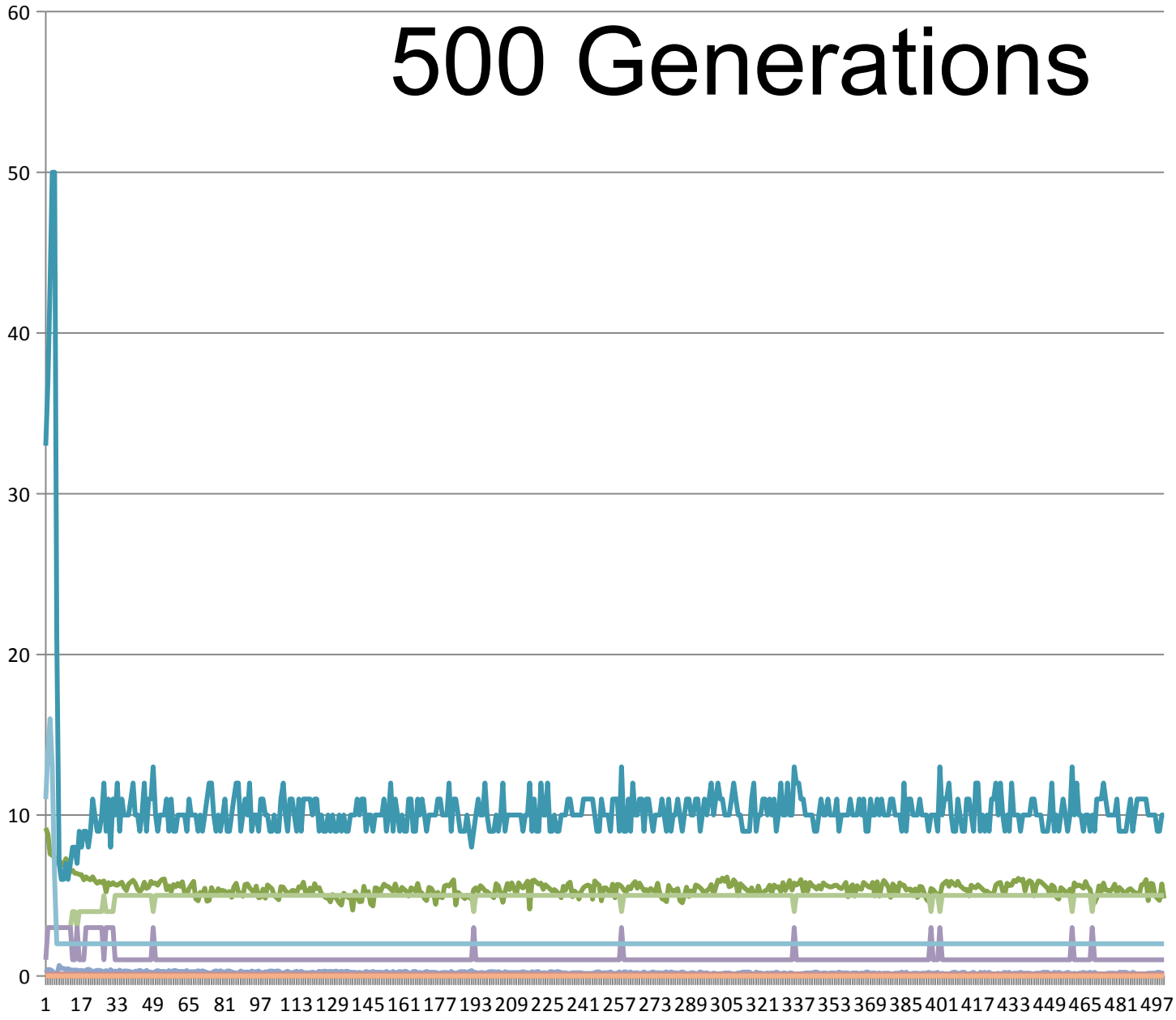


- Vi ser at de beste prosumenter ligger stabilt opp mot 0.1

- Videre ser vi at leverandør prøver seg litt med høye priser i starten, men finner ut at en lavere pris gir bedre resultat, og stabiliserer seg.

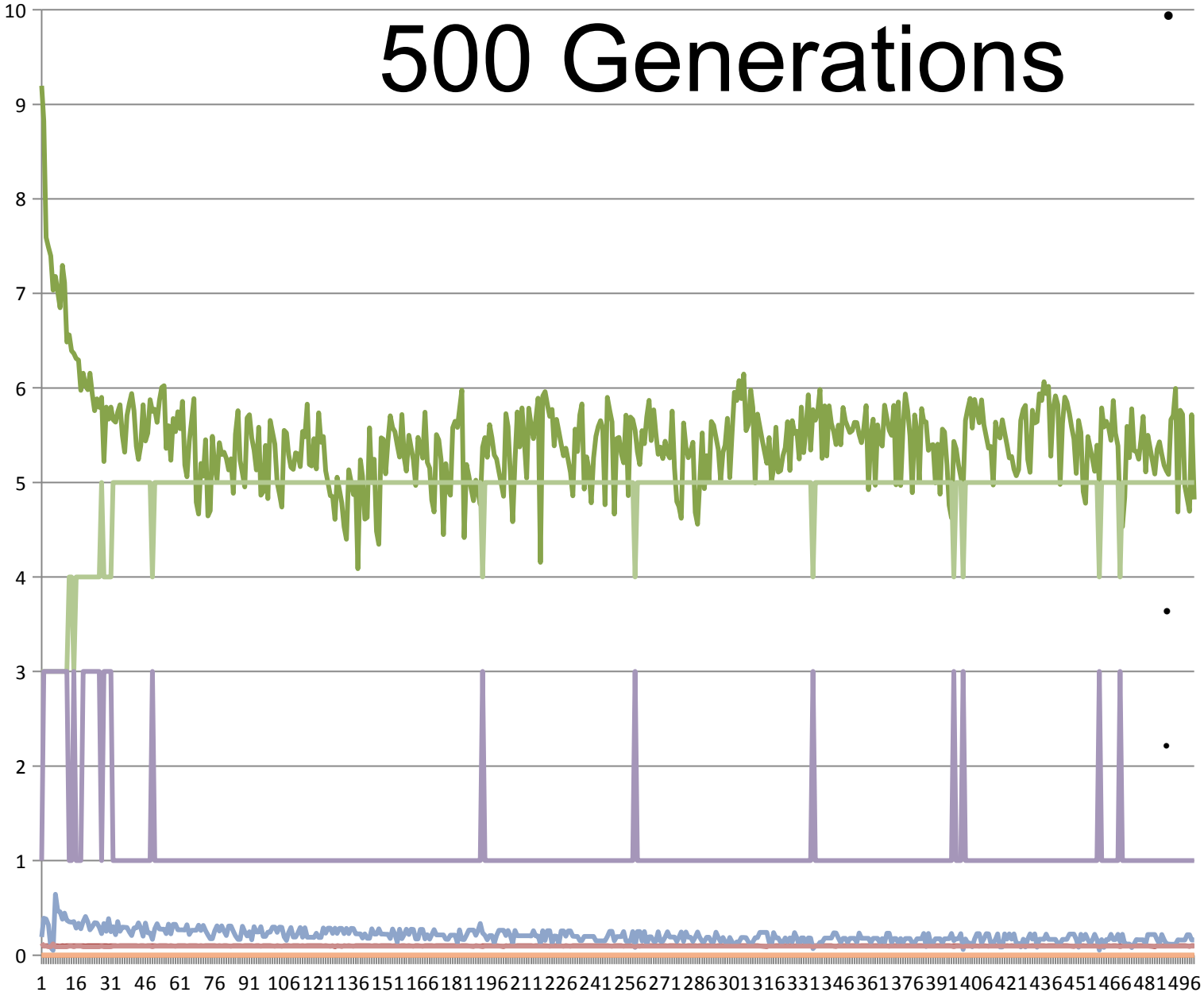


- Vi merker oss et bedre inntrykk av den tendens vi såg etter bare 100 generasjoner mtp stabilisering av energi forbruk hos beste prosument.



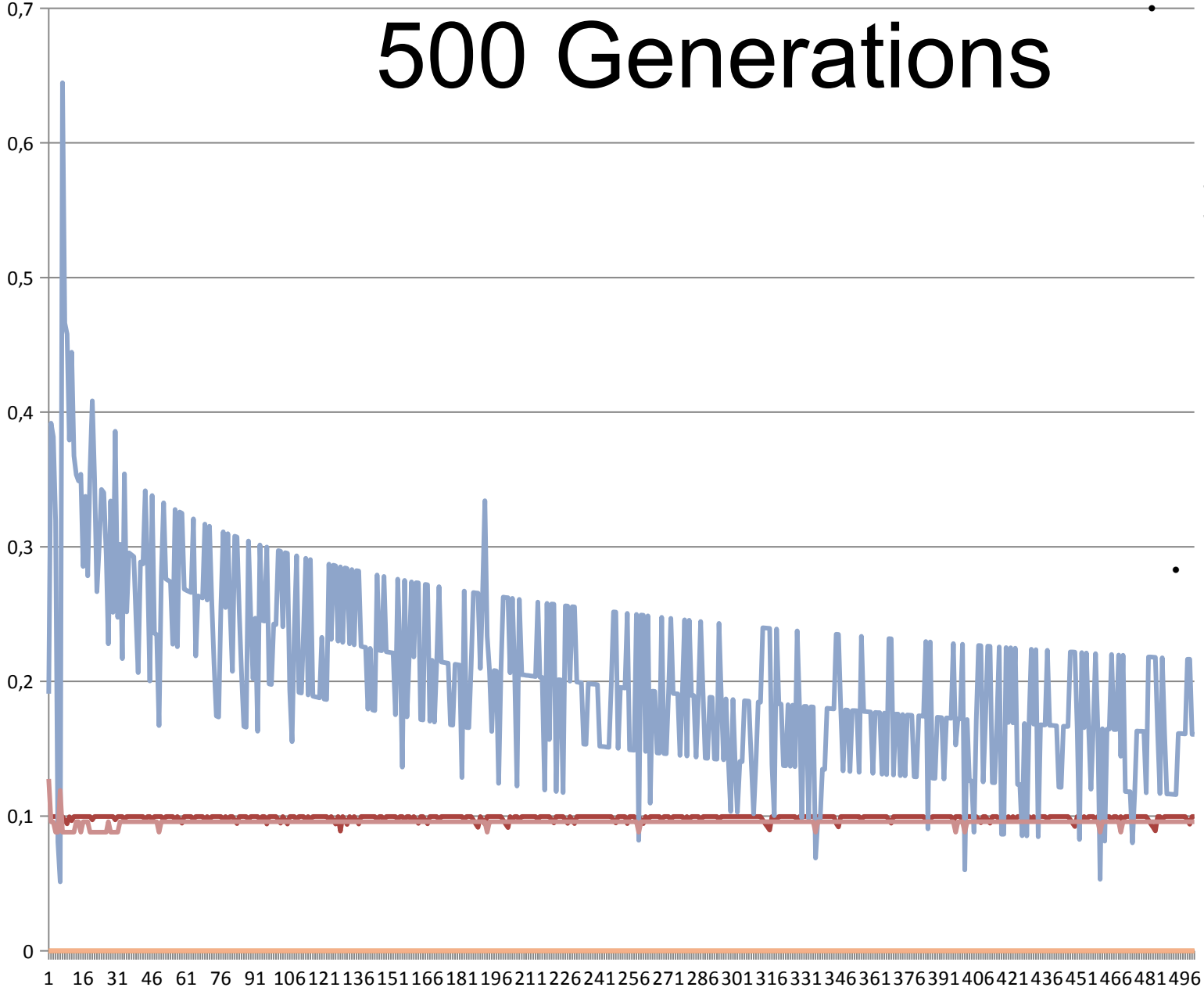
- Igjen kan vi over 500 generasjoner merke oss hvordan populasjonen stabiliserer seg over tid.
- Dette kan man begrynne med at leverandør populasjonens størrelse og leveranse kapasitet finner et komplimentært forhold til populasjonen av kunder.

500 Generations



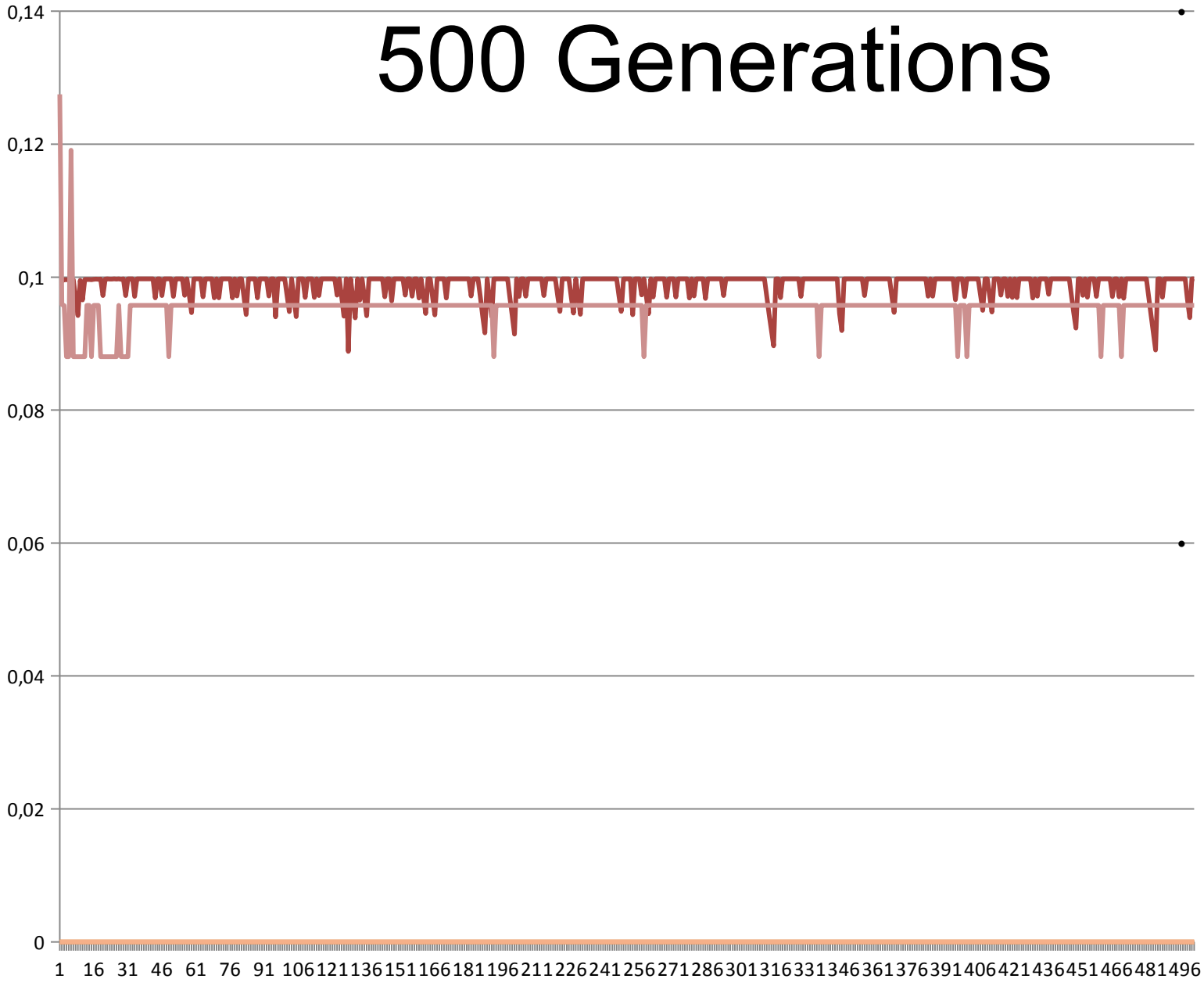
- Vi ser hvordan beste prosument sin økonomiske kapasitet stabiliserer seg.
- Beste leverandør har stort sett fem kunder, noen ganger fire.
- Videre ser vi hvordan den faste prisen stort sett er best, men hvor hybrid pris noen ganger vinner. Spot-pris vinner aldri... Her kunne man nok ha lekt med parametre noe for å få et mer variabelt resultat.

500 Generations



Leverandørens saldo synker med en logaritmisk stabilisering. Dette er et direkte resultat av vår faste utgift, som fungerer logaritmisk for å ivareta leverandører med høy alder.

Ved eksperimentering såg vi at en lineær fast utgift fikk nye leverandører en fordel, siden deres saldo ikke var likt påvirket av utgiften. Dette skapte kaos og ødela spesialisering.



Vi ser at prosumentens saldo hele tiden kjemper om å holde seg opp mot det som virker som toppen (0.1). Mens lett kaotiske istagger pryder hele historien.

Leverandører sitt pristilbud stabiliserer seg over tid, med noen pris-fall i ny og ne. Disse fallene er identisk med når en hybrid pris-strategi vinner. Dette tyder på at i vår algoritme for saldo hos leverandører er det ikke alltid at best pris alene blir vinneren.

Diskusjon

- Dette utviklet seg til en oppgave som gav mye rom for diskusjon, eksperimentering og kreativitet. Man måtte finne egne algoritmer for kalkulasjon av saldo og andre elementer i den genetiske algoritmen, og måtte samtidig eksperimentere med input parametre for å se samspillet mellom individer av prosumenter og leverandører.
- Det hadde vært veldig spennende å ha noe mer tid til oppgaven, da omfanget ble noe stort for tidsfristen som ble satt (jeg jobbet nesten utelukkende på GA i denne siste oppgaven, med tall ut i konsoll vinduet).
- Å tweake seg fram til gode parametre ville tatt mer tid enn vi hadde til rådighet, og man kan derfor anta at resultatene vi ser er langt fra optimale. Dessuten burde de nok vært dratt ut fra kildekoden og lagt i datafiler eller gjort tilgjengelig i brukergrensesnittet, noe omfanget i oppgaven også gjorde vanskelig.
- Jeg ønsket å løse oppgaven på en noe unik måte, og fant løsningen hvor man grafisk ser saldo/fitness av alle prosumenter som en interessant måte å få oversikt over populasjonens tilstand på. Hadde man kunne trykket på disse individene for å lese mer detaljert info om dem, ville denne angrepsvinkelen virkelig fått fram sin styrke.
- Tanken var at ved å grafisk fremstille hvert individ, ville man få en intuitiv fremstilling av algoritmen i bunnen, uten å trenge forstå den. Hadde man videre kunne leke seg med input parametrene, ville en person uten forståelse for GA og underliggende algoritme kunne leke seg fram til optimale løsninger (ved å spille). Disse målene fikk jeg ikke tid til å realisere til det fulle.
- Alt i alt en kjempespennende oppgave som jeg gjerne skulle ha lekt mer med!