Algoritme

For at lave algoritmen til at f.eks. at planlægge et skema, er der forskellige metoder. Hver metode har fordele og ulemper og er derfor egnet til forskellige problemer.

**Brute Force:**

Brute-force algoritmen går i sin simple form ud på, at tjekke alle mulige kombinationer af problemet, og ligesom i den genetiske algoritme, beregne en fitness for hver mulig løsning. Da alle mulige løsninger bliver afprøvet, er det sikkert, at den bedste løsning bliver fundet i forhold til den valgte fitness.

Hvis der f.eks. skal findes en divisor for et heltal n, ville brute-force metoden gå ud på at gå gennem alle tal mellem 1 og n, og tjekke om tallet kan divideres uden, at producere en rest.

Selvom denne metode er sikker på den optimale løsning, vil et problem med mange parametre kræve længere tid at køre, sammenlignet med andre metoder som genetisk algoritme, da vi i genetisk algoritme hurtigt sorterer mange løsninger fra, uden at gå igennem dem enkeltvis.

Der er visse applikationer hvor brute-force kan være meget brugbart, hvis man F.eks. vil cracke et password. Når man skal cracke et password kan man ikke bruge genetiske algoritmer, for du får ikke et output der fortæller om ens løsning er tæt på at være rigtig. Med brute-force er det sikkert, at man får den rigtige løsning, da brute-force tester alle løsninger.

**Genetiske algoritmer:**

Genetiske algoritmer er en metode til at optimere en løsning til et givent problem. Genetiske algoritmer er en række retningslinjer som udefra disse danner flere løsninger til et problem og finde den bedste løsning til problemet.

Genetiske algoritmer er en metode, som er inspireret af Charles Darwins teori om naturlig selektion, som omhandler hvordan de biologiske arter udvikler sig over tid, ved at tilpasse sig miljøet og derved bliver bedre egnet til, at overleve og formere sig i miljøet. Teorien danner grundlag i, at populationen af en given art har forskellige kromosomer og at der over tid vil ske små ændringer i kromosomerne hos individerne. Disse små ændringer i kromosomerne vil over længere tid, føre til større ændringer hos individerne. På daværende tidspunkt var Darwins teori meget kontroversiel, dette er dog ikke tilfældet for softwarebrug af genetiske algoritmer, da en algoritme er noget lettere at forklare, end en biologisk ændring i en art over længere tid.

En genetisk algoritme består af følgende dele

1. Individer som er mulige løsninger til problemet.
2. Fitness som er en egenskab hos individerne, som bliver udregnet i forhold til, hvor god en løsning individerne er til problemet.
3. Et individ består af flere kromosomer.
4. En population som består af en mængde af individer.
5. Generationer som indikere hvor lang tid algoritmer forløber over.

En population bliver muteret ved følgende operationer.

Selektion:

Operatoren vælger individer til reproduktionen, jo større fitness individerne har, jo større sandsynlighed er der for, at de bliver valgt til reproduktion. Reproduktion kombiner 2 individer og danner ud fra deres kromosomer et nyt individ.

Crossover:

Operatoren vælger 2 tilfældige kromosomer og blander dem så der bliver dannet 2 nye kromosomer som er en kombination af de 2 kromosomer fx strengene 1110100 1011111 bliver krydset og danner de 2 nye strenge 1010100 1111111.

Mutation:

Operationen flipper tilfældigt nogle stykker af kromosomet fx strengen 1001000 bliver muteret i dens tredje position til 1011000. Mutation kan ske i alle positioner i strengen med en hvis sandsynlighed. For en simpel algoritme kan fungere skal der være givet et defineret problem og et individ som en mulig

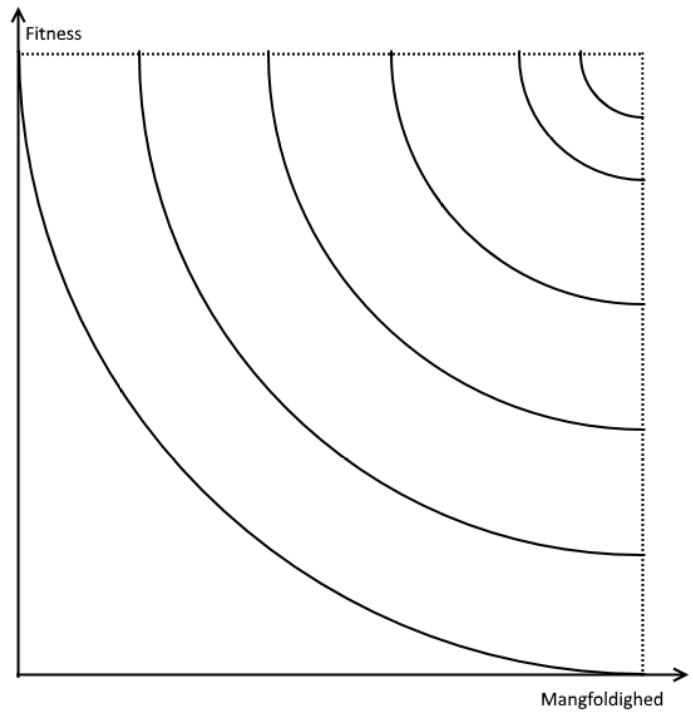
Mangfoldighed

Når man prøver at finde den optimale løsning, er det vigtigt at man ikke bliver fanget på et lokalt højdepunkt, men at man kan komme over disse til det globale højdepunkt. Eksempelvis, hvis et problem havde 5 parametre, og man udregnede fitness heraf. Heri vil der forekomme selektion, crossover og mutation. Dette gør at vi nu kan udregne endnu en fitness for den givne population, men hvad nu hvis, at man hænger fast.

Hvis at man ved at ændre parametre kun en lille smule for hver gang, gør at man nu hænger fast, da tallene omkring disse parametre ikke gør at der kommer en højre fitness. Man har ved hjælp af parametrene muligheden for at udregne den højeste fitness, alt afhængigt af hvor meget parametrene skal justeres, da minimale ændringer vil ikke nødvendigvis resultater i den højeste fitness, hvorimod store ændringer i parametrene vil muligvis udregne det højeste fitness. Altså vil vi i starten gerne finde løsninger, som lægger spredt ud over hele problemet, og så derefter fjerne denne mangfoldighed igen for at ’zoome’ ind på den endelige optimale løsning.

Det man gerne vil have er et individ med, ikke bare stor fitness, men et som heller ikke ligner de andre. Hvis at man kigger på figur **XX**, vil man finde det individ der ligger på linjerne.

Både ved brug af brute force og generisk algoritme, er målet at finde den optimale løsning. Denne beregnes ud fra et fitness niveau, hvor der på hver løsning laves en udregning ud fra valgte parametre. Denne returnerer et fitnessniveau. Herefter gemmes løsningen med det bedste fitnessniveau. Det bedste fitnessniveau kan være defineret ved den højest mulige eller lavest mulige værdi. [[1]](#footnote-1)



Kilde til mangfoldighed: <https://www.youtube.com/watch?v=kHyNqSnzP8Y>

løsning til problemet.

1. Der bliver tilfældet generet en population af n-individer med l-kromosomer
2. De genetiske algoritme operatorer udsætter populationen for mutation og derved vil der forekomme ændringer hos kromosomerne i individerne eller danne nye individer.
3. Der bliver beregnet fitness for hvert n-individ i populationen. Fitness bestemmer sandsynligheden for individet overlever i populationen.
4. Processen gentages fra trin 2 i x antal givende generationer eller indtil et givet kriterium er opfyldt.

Algoritmen er færdig når x antal generationer er kørt igennem eller et givet kriterium er opfyldt og der vil være en population med potentielle løsninger til problemet, hvor der herudfra vil blive valgt en løsning til problemet som typisk er løsningen med højst fitness. Genetiske algoritmer kan bruges til at optimere problemet med skemalægning, udefra en række retningslinjer som bliver bestemt. Det vil udmunde i en optimal løsning til skemalægningsproblemet, hvis der bliver givet de korrekte retningslinjer.

1. https://www.tutorialspoint.com/genetic\_algorithms/genetic\_algorithms\_fitness\_function.htm [↑](#footnote-ref-1)