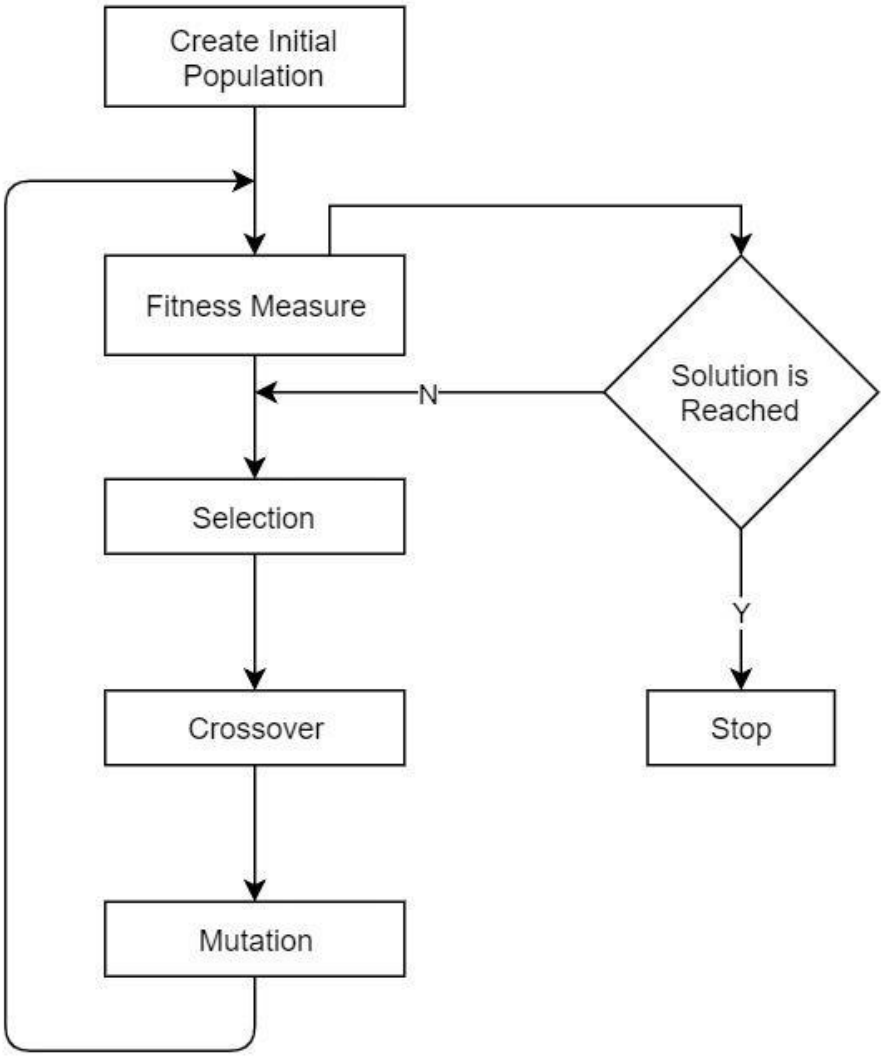
Problém batohu

Klasický scénář pod kterým si můžeme problém batohu představit je, že zloděj se snaží nakrást co nejvíce věcí. Jeho batoh má určitou kapacitu, co se do něj vejde a zloděj se snaží vybrat takovou kombinaci věcí, která bude mít největší celkovou cenu. Každá věc má určitou váhu a cenu, podle kterých se zloděj rozhoduje. V této seminární práci se zaměřuji na 0-1 batoh, což znamená, že v batohu dává věc buď je nebo není. Jinými slovy, zloděj nevezme jen polovinu televize.

Genetický algoritmus

Genetický algoritmus je heuristický postup, který se snaží se snaží aplikací principů evoluční biologie nalézt řešení problémů. V těchto algoritmech se používají techniky napodobující evoluční procesy jako dědičnost, mutace, přirozený výběr a křížení. Princip práce genetického algoritmu je postupná tvorba generací různých řešení daného problému. Při řešení se uchovává tzv. populace, jejíž každý jedinec představuje jedno řešení daného problému. Jak populace probíhá evolucí, řešení se zlepšují. Tradičně je řešení reprezentováno binárními čísly, řetězci nul a jedniček. V přechodu do nové generace je pro každého jedince spočtena tzv. fitness funkce, která vyjadřuje kvalitu řešení reprezentovaného tímto jedincem. Podle této kvality jsou stochasticky vybráni jedinci, kteří jsou modifikováni (pomocí mutací a křížení), čímž vznikne nová populace. Tento postup se iterativně opakuje, čímž se kvalita řešení v populaci postupně vylepšuje.



Průběh algoritmu můžeme popsat slovně:

- **Inicializace**: vytvoření nulté populace (náhodně generovaná)
- **Počátek cyklu**: vybereme z první populace nejlepší jedince (zde maximální zisk)
- **Křížení jedinců**
- **Mutace jedinců**

Po mutaci se vracíme na začátek, kde se znovu počítá fitness dané populace a cyklus se opakuje. Algoritmus končí po zvoleném počtu generací.

Vytvoření tabulek vah a cen a celikost batohu

K výběru je 50 věcí, kde každá z nich má svojí váhu a cenu (**W, V**). Velikost batohu jsem nastavil na 1000.

```
clear; close all; clc
n=50  %počet věcí
```

```
n = 50
```

```
W=[80 82 85 70 72 70 66 50 55 25 50 55 40 48 50 32 22 60 30 32 ...
    40 38 35 32 25 28 30 22 50 30 45 30 60 50 20 65 20 25 30 10 ...
    20 25 15 10 10 10 4 4 2 1];       %váha každé věci
V=[220 208 198 192 180 180 165 162 160 158 155 130 125 122 120 ...
    118 115 110 105 101 100 100 98 96 95 90 88 82 80 77 75 73 72 ...
    70 69 66 65 63 60 58 56 50 30 20 15 10 8 5 3 1];     %hodnota každé věci
CW=1000;   %Velikost batohu
```

Počáteční nastavení

Velikost populace a počet iterací jsem nastavil na 1000. Dále jsem nastavil pravděpodobnost křížení (**pc**) na hodnotu 0,85 a pravděpodobnost mutace (**pm**) na 0,1.

```
popsize=1000;               %velikost populace
popbest=zeros(popsize, n)   %proměnná pro uložení nejlepší populace
```

```
popbest = 1000×50
     0     0     0     0     0     0     0     0     0     0     0     0 ···
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     ⋮
```

```
pop=initpop(popsize,n)      %počáteční populace
```

```
pop = 1000×50
     1     0     0     1     0     0     0     1     1     0     1     1     0 ···
     0     0     1     0     0     1     0     1     1     1     1     1     1
```

```
0    1    1    1    0    1    1    0    1    1    0    1    0
1    0    1    1    0    1    0    0    1    1    1    0    1
0    1    1    0    1    1    1    0    0    1    1    1    0
1    0    1    0    0    0    0    1    0    1    1    1    1
0    1    1    1    1    0    1    1    0    1    0    0    1
0    0    0    1    0    0    1    1    1    1    1    0    0
0    0    0    0    0    1    1    1    1    1    1    0    1
1    0    1    1    1    1    0    1    0    0    1    0    1
⋮
```

```
t=1000;                     %počet iterací

pc=0.85;                %pravděpodobnost křížení
pm=0.1;                 %pravděpodobnost mutace

%uložení nejlepšího z populace
y=zeros(1,popsize);   %ulož maximální hodnotu
g=zeros(1,popsize);   %ulož maximální váhu
n=zeros(1,popsize);   %ulož pozici
```

Výpočet

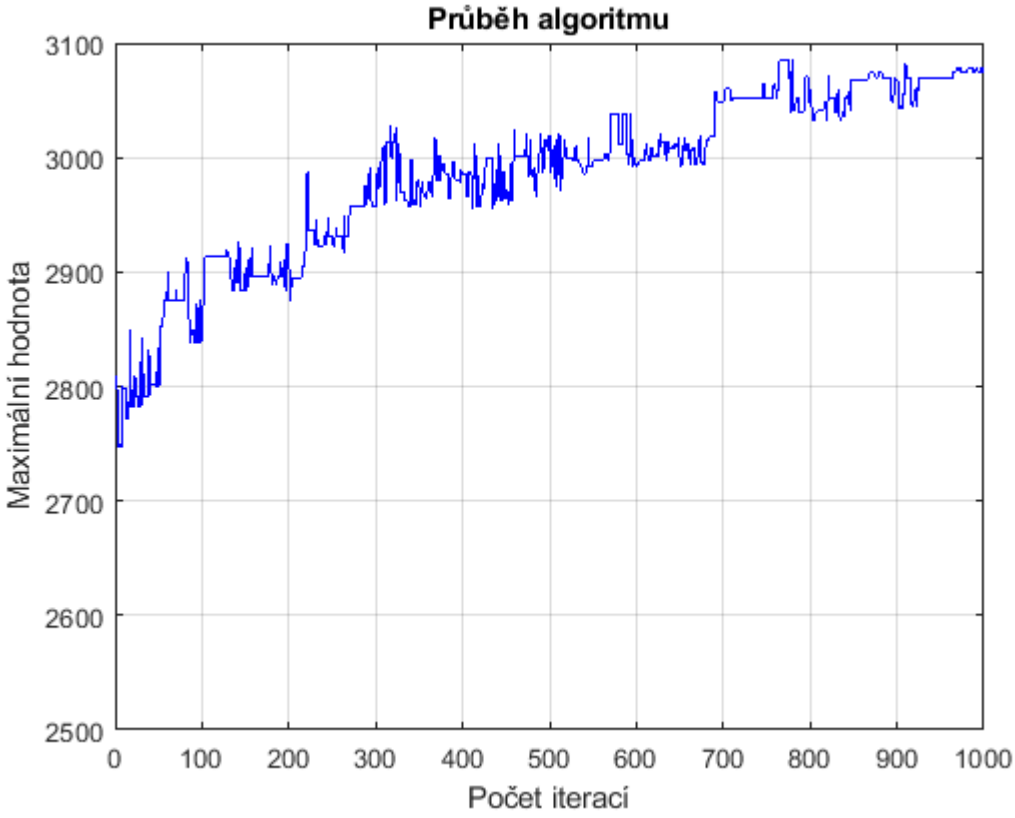
```
for i=1:1000
    [fitvalue]=calobjvalue(pop,V,W,CW); %výpočet účelové funkce
    [bestweight,bestvalue, bestpop]=best(pop,fitvalue,W);
    %výpočet optimální váhy, hodnoty

    y(i)=max(bestvalue);    %maximální hodnota z ceny
    g(i)=max(bestweight);   %maximální hodnota z váhy
    n(i)=i;                 %pozice
    popbest(i,:)=bestpop;    %nejlepší jedinec v populaci
    [newpop]=selection(pop,fitvalue);  %výběr nejlepších jednotlivců
                                       %pro křížení

    [newpop1]=crossover(newpop,pc);    %křížení
    [newpop2]=mutation(newpop1,pm);     %mutace
    pop=newpop2;
end
pop;
```

Vizualizace

```
i=1:1000;
plot(y(i),'-b')
xlabel('Počet iterací')
ylabel('Maximální hodnota');
title('Průběh algoritmu');
grid on
xlim([0 1000])
ylim([2500 3100])
```




```
[z, index]=max(y);
po=n(index)          %optimální pozice
```

```
po = 765
```

```
W=g(index)          %optimální váha
```

```
W = 999
```

```
V=z                 %Nejlepší hodnota
```

```
V = 3085
```

```
BEST=popbest(po,:);    %Optimální řešení
```

Závěr

Algoritmus s pravděpodobností křížení 0,85 a pravděpodobností mutace 0,1 se mi osvědčil nejvíce. Nejlepší hodnoty dostala 765. populace, která nevyužila celou kapacitu batohu, ale jen 999 jednotek, ale měla největší cenu a to 3085.