

# **UMINT GA Projekt 1 Dokumentácia**

**Ladislav Rajcsányi a Daniel Malenka**

**Marec 2020**

## Obsah

1. <a href="#">Naša úloha</a>	3
2. <a href="#">Použité funkcie</a>	3
<a href="#">Vlastné funkcie</a>	3
<a href="#">coord_dots_input</a>	3
<a href="#">center</a>	3
<a href="#">calculate_obstacles</a>	4
<a href="#">draw_obstacles</a>	5
<a href="#">custom_converter</a>	6
<a href="#">calculate_t_and_u</a>	7
<a href="#">collision_check</a>	8
<a href="#">fitness_test</a>	8
<a href="#">display_indiv</a>	9
<a href="#">Funkcie z Genetického toolboxu</a>	10
<a href="#">genrpop</a>	10
<a href="#">crossover</a>	10
<a href="#">mutx</a>	11
<a href="#">muta</a>	11
<a href="#">selbest</a>	12
<a href="#">seltourn</a>	12
3. <a href="#">Zakódovanie reťazca v populácii</a>	13
4. <a href="#">Výpočet a realizácia účelovej (fitness) funkcie</a>	13
5. <a href="#">Genetický algoritmus</a>	13
6. <a href="#">Dosiahnuté výsledky po 10 spustení</a>	16
7. <a href="#">Dosiahnuté výsledky pri modifikácii GA</a>	18

## 1. Naša úloha

Riešili sme **2. úlohu**. Mali sme **navrhnuť genetický algoritmus, ktorý nájde optimálnu dráhu mobilného robota v 2D prostredí s prekážkami**. Počiatočný bod (start) sme mali v ľavom dolnom rohu a cieľový bod (stop) v pravom hornom rohu priestoru. Mali sme navrhnuť priestor tak, aby sme tam mali tri prekážky, ktoré mohli byť čísla alebo písmená. Veľkosť prostredia a písmen bola ľubovoľná. Dráha mala byť čo najkratšia a bez kolízií.

Riešenia sú priložené vo forme spustiteľných programov v Matlabe, v priečinku Figures sa nachádzajú grafy, ktoré sme vytvorili pomocou programu.

## 2. Použité funkcie

### Vlastné funkcie

**coord\_dots\_input** – načítanie súradníc

#### Charakteristika:

Táto funkcia nám umožní načítať súradnice počiatočného a cieľového bodu. Okrem toho môžeme aj načítať aj počet prechodových bodov.

#### Syntax:

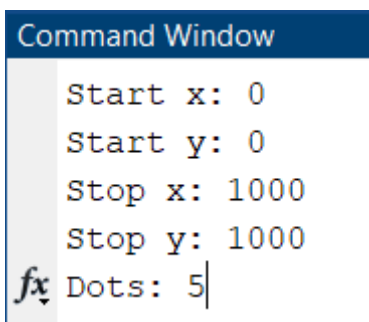
`[start,stop,dots]= coord_dots_input();`

start – súradnice počiatočného bodu (Robot) [ start\_x , start\_y]

stop – súradnice cieľového bodu (Robot) [ stop\_x , stop\_y]

dots – počet prechodových bodov

#### Príklad:



```
Command Window
Start x: 0
Start y: 0
Stop x: 1000
Stop y: 1000
fx Dots: 5
```

**center** – výpočet počiatočného bodu prvej prekážky

#### Charakteristika:

Táto funkcia je schopná v prípade načítania súradníc vypočítať počiatočný bod prvej prekážky (L) a veľkosť prekážok, takým spôsobom, že po vykreslení , písmená budú v strede priestoru a ich veľkosť bude závisieť od načítaných hodnôt.

#### Syntax:

`[x,y,s] = center(start,stop)`

x – x-ová súradnica počiatočného bodu prvej prekážky

y – y-ová súradnica počiatočného bodu prvej prekážky

s – size – veľkosť prekážok

start – súradnice počiatového bodu (Robot) [ start\_x , start\_y]

stop – súradnice cieľového bodu (Robot) [ stop\_x , stop\_y]

#### Príklad:

(Pre výpočty otvorte zložku center.m)

```
start =[0 0]
```

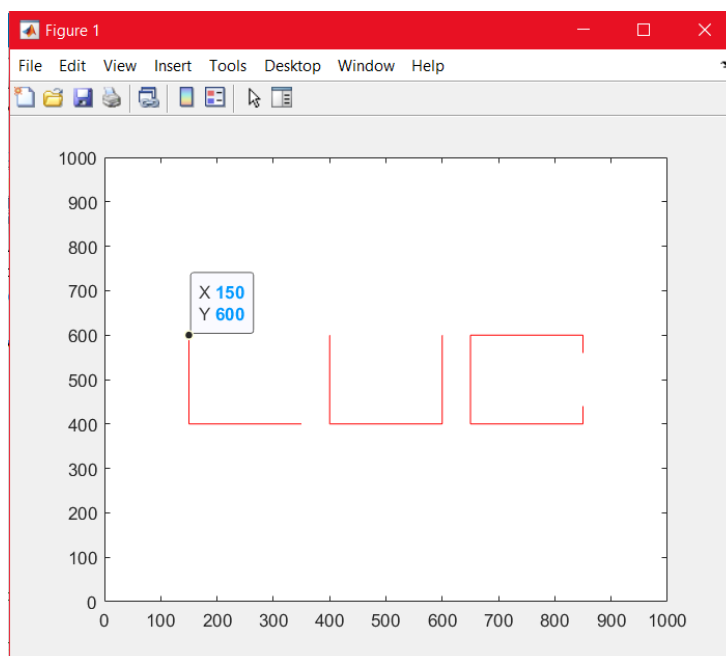
```
stop = [1000 1000]
```

```
[x,y,s]=center(start,stop)
```

```
x=150
```

```
y = 600
```

```
s = 200
```



#### calculate\_obstacles – výpočet prekážok

##### Charakteristika:

Z hodnôt čo nám vrátila funkcia **center** nám vypočíta, kde sa nachádzajú prekážky

##### Syntax:

```
[L,U,C] = calculate_obstacles(start,stop)
```

L – Matica, ktorá obsahuje body prekážky L

U – Matica, ktorá obsahuje body prekážky U

C – Matica, ktorá obsahuje body prekážky C

start – súradnice počiatového bodu (Robot) [ start\_x , start\_y]

stop – súradnice cieľového bodu (Robot) [ stop\_x , stop\_y]

**Príklad:**

```
start =[0 0]
```

```
stop = [1000 1000]
```

```
[L,U,C] = calculate_obstacles(start,stop)
```

```
L=[ 150 600
```

```
    150 400
```

```
    350 400]
```

```
U=[ 400 600
```

```
    400 400
```

```
    600 400
```

```
    600 600]
```

```
C= [ 850 560
```

```
    850 600
```

```
    650 600
```

```
    650 400
```

```
    850 400
```

```
    850 440]
```

**draw\_obstacles – vykreslenie prekážok****Charakteristika:**

Vykreslí nám prekážky, ktoré nám vypočítala funkcia **calculate\_obstacles**.

**Syntax:**

```
draw_obstacles(start,stop)
```

start – súradnice počiatočného bodu (Robot) [ start\_x , start\_y]

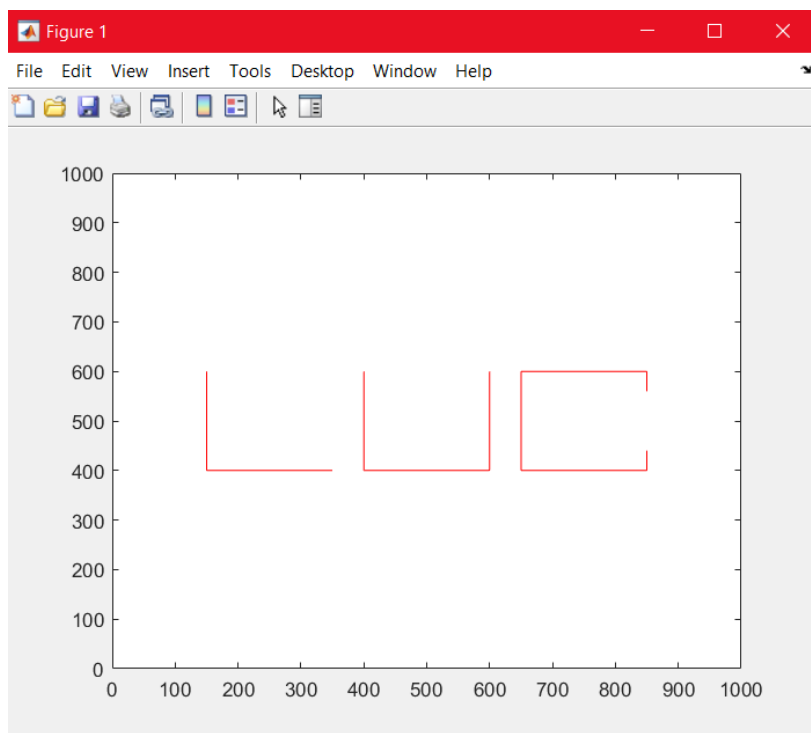
stop – súradnice cieľového bodu (Robot) [ stop\_x , stop\_y]

**Príklad:**

```
start =[0 0]
```

```
stop = [1000 1000]
```

```
draw_obstacles(start,stop)
```



draw\_obstacles(start,stop)

### custom\_converter – funkcia na usporiadanie súradníc jedinca do matice

#### Charakteristika:

Pre ľahšie overenie kolízie a vykreslenie trasy sme potrebovali funkciu, ktorá nám usporiada súradnice jedinca do matice typu  $N \times 2$ , kde  $N$  je polovica prvkov z jedinca.

#### Syntax:

`a = custom_converter(indiv)`

`a` – Súradnice jedinca usporiadané do matice

`indiv` – Jedinec, ktorého chceme usporiadať

#### Príklad:

Povedzme, že budeme mať 5 prechodových bodov, potom náš jedinec po vygenerovaní bude vyzeráť :

`indiv =`

Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
0	0	103,6	36,8	288,5	823	497,4	37,6	966	195,2	677,2	219,9	1000	1000

`a = custom_converter(indiv)`

a =

x	y	
0	0	Start
103,6	36,8	Bod1
288,5	823	Bod2
497,4	37,6	Bod3
966	195,2	Bod4
677,2	219,9	Bod5
1000	1000	Stop

**calculate\_t\_and\_u** – implementácia vzorca potrebné na overenie kolízie

**Charakteristika:**

Na stránke, ktorú sme mali ako pomôcku bol vzorec na výpočet t a u:

$$t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$u = -\frac{\begin{vmatrix} x_1 - x_2 & x_1 - x_3 \\ y_1 - y_2 & y_1 - y_3 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = -\frac{(x_1 - x_2)(y_1 - y_3) - (y_1 - y_2)(x_1 - x_3)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)},$$

**Syntax:**

[t,u] = calculate\_t\_and\_u(x1,y1,x2,y2,x3,y3,x4,y4)

t – Hodnota t, ktorú sme dostali zo vzorca

u - Hodnota u, ktorú sme dostali zo vzorca

x1,y1 – počiatočný bod aktuálnej časti trasy robota

x2,y2 – cieľový bod aktuálnej časti trasy robota

x3,y3 – počiatočný bod aktuálnej časti prekážky

x4,y4 – cieľový bod aktuálnej časti prekážky

**Príklad:**

x1=755,6

y1=704,7

x2=112,1

y2=899,6

x3=150

y3= 600

x4=150

y4=400

[t,u] = calculate\_t\_and\_u(x1,y1,x2,y2,x3,y3,x4,y4)

t= 0,9411

u=-1,4406

**collision\_check – funkcia na overenie kolízie**

**Charakteristika:**

Pomocou **calculate\_obstacles** vypočítame polohy prekážok a po usporiadaní jedinca s **custom\_converter**, prekontrolujeme jednotlivé časti trasy so všetkými prekážkami a overujeme, či nedošlo ku kolízii . To dosiahneme porovnaním hodnoty t a u , čo sme dostali z **calculate\_t\_and\_u**. Ak  $0.0 \leq t \leq 1.0$  a  $0.0 \leq u \leq 1.0$ , potom to znamená, že je tam kolízia.

**Syntax:**

[collision] = collision\_check(start, stop, indiv)

collision – počet kolízií

start – súradnice počiatočného bodu (Robot) [ start\_x , start\_y]

stop – súradnice cieľového bodu (Robot) [ stop\_x , stop\_y]

indiv – Jedinec, ktorého prekontrolujeme

## **Fitness funkcia :**

**fitness\_test – fitness funkcia, ktorá vypočíta dĺžku trasy**

**Charakteristika:**

Funguje podobne, ako testfn3 , čo sme používali naposledy, ale museli sme ju trochu modifikovať. Používali sme vzorec :

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ktorý slúži na výpočet vzdialenosti dvoch bodov. Ak nastala kolízia, čo overujeme funkciou **collision\_check** , daného jedinca penalizujeme metódou mŕtvej pokuty. To znamená, že



v prípade kolízie pripočítame veľké číslo k „fitness hodnote“ daného jedinca. Tým pádom pri výbere bude mať menšiu šancu ako ostatní .

### Syntax:

`[Fit] = fitness_test(Pop)`

Fit – obsahuje „fitness hodnoty“ jedincov

Pop – populácia s ktorými pracujeme

### display\_indiv – vykreslenie jedinca

#### Charakteristika:

Túto funkciu používame na zobrazenie jedinca. **Custom\_converter** nám uľahčí robotu, a môžeme uložiť vykreslené prechodové body (a) a vykreslené prechody (b), môžeme ich vrátiť, a potom keď potrebujeme môžeme výkres vymazať aby sme mohli vykresliť druhého/lepšieho jedinca. Týmto spôsobom je zabezpečený vizuálny rozvoj jedinca.

### Syntax:

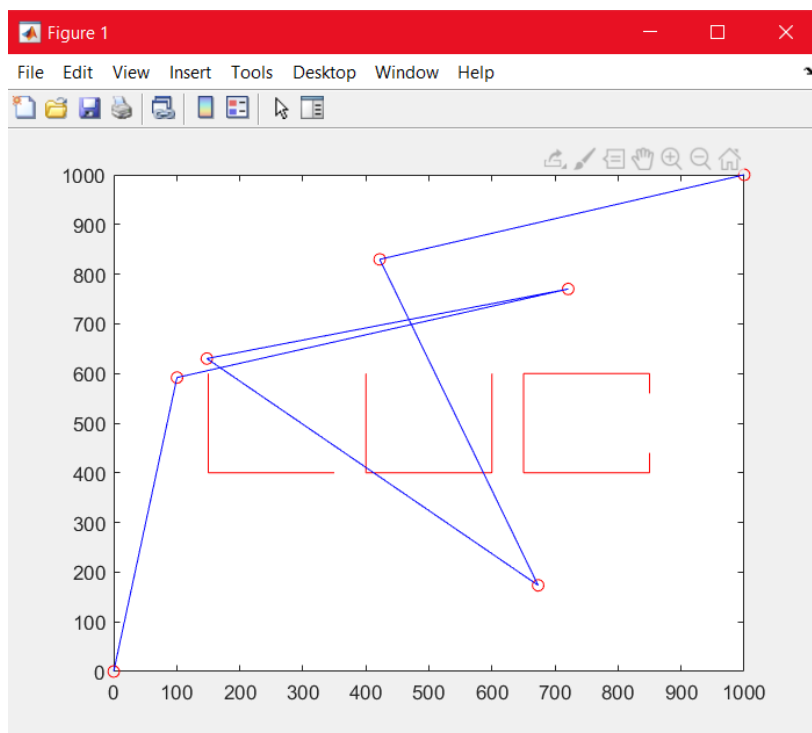
`[a,b]=display_indiv(indiv);`

a – vykreslené prechodové body jedinca

b – vykreslené prechody jedinca

indiv – jedinec, ktorého chceme zobraziť

### Príklad:



display\_indiv(indiv)

## Funkcie z Genetického toolboxu

### genrpop - vygenerovanie náhodnej reálnočíselnej populácie<sup>1</sup>

#### Charakteristika:

Funkcia vygeneruje populáciu zvoleného počtu reťazcov. Jednotlivé gény reťazcov obsahujú náhodné reálne čísla z definovaného rozsahu. Rozsah sa definuje pre každý gén samostatne pomocou matice Space. Prvý riadok tejto matice určuje dolné ohraničenia a druhý riadok horné ohraničenia jednotlivých génov.

#### Syntax:

Newpop=genrpop(popsiz,Space)

Newpop - nová, zmutovaná populácia

popsiz - počet reťazcov generovanej populácie

Space - matica obmedzení, ktorej 1. riadok je vektor minimálnych a 2. riadok je vektor maximálnych prípustných hodnôt génov

### crossov - kríženie <sup>2</sup>

#### Charakteristika:

Funkcia vytvorí novú populáciu reťazcov, ktorá vznikne skrížením všetkých reťazcov starej populácie 1- až 4-bodovým krížením. Krížené sú všetky reťazce (ak je ich párny počet, ak nie tak len jedince ktoré vytvorili páry). Výber párov je buď náhodný alebo sú vybrané susedné reťazce v populácii podľa voľby parametra sel.

#### Syntax:

Newpop=crossov(Oldpop,num,sel)

Newpop - matica skríženej (výstupnej) populácie

Oldpop - pôvodná (vstupná) populácia

num - počet bodov kríženia (miest rozdelenia) od 1 do 4

sel - spôsob výberu dvojíc: 0 - náhodný 1 - susedné dvojice v populácii

---

<sup>1,2</sup> TOOLBOX - GENETICKÉ ALGORITMY pre riešenie optimalizačných problémov v prostredí Matlab - POUŽÍVATEĽSKÁ PRÍRUČKA  
Ivan Sekaj

### **mutx - obyčajná mutácia (globálna mutácia) <sup>3</sup>**

#### **Charakteristika:**

Funkcia zmutuje populáciu reťazcov s intenzitou úmernou parametru rate (z rozsahu od 0 do 1). Mutovaných je len niekoľko génov v rámci celej populácie. Mutované hodnoty sú zmenené na náhodné hodnoty z priestoru definovaného ohraničeniami pomocou dvojriadkovej matice. Prvý riadok matice určuje dolné ohraničenia jednotlivých génov reťazcov a druhý riadok ich horné ohraničenia.

#### **Syntax:**

`Newpop=mutx(Oldpop,rate,Space)`

Newpop - nová, zmutovaná populácia

Oldpop - stará populácia

Space - matica obmedzení, ktorej 1.riadok je vektor minimálnych a 2. riadok je vektor maximálnych prípustných mutovaných hodnôt

rate - miera početnosti mutovania génov v populácii (od 0 do 1)

### **muta - aditívna mutácia (lokálna mutácia)<sup>4</sup>**

#### **Charakteristika:**

Funkcia zmutuje populáciu reťazcov s intenzitou úmernou parametru rate (z rozsahu od 0 do 1). Mutovaných je len niekoľko génov v rámci celej populácie. Mutácie vzniknú pripočítaním alebo odpočítaním náhodných čísel ohraničených veľkostí k pôvodným hodnotám náhodne vybraných génov celej populácie. Absolútne hodnoty prípustných veľkostí aditívnych mutácií sú ohraničené hodnotami vektora Amp. Po tejto operácii sú ešte výsledné hodnoty génov ohraničené (saturované) na hodnoty prvkov matice Space. Prvý riadok matice určuje dolné ohraničenia a druhý riadok horné ohraničenia jednotlivých génov.

#### **Syntax:**

`Newpop=muta(Oldpop,rate,Amp,Space)`

Newpop - nová, zmutovaná populácia

Oldpop - stará populácia

Amp - vektor ohraničení prípustných aditívnych hodnôt mutácií

Space - matica obmedzení, ktorej 1. riadok je vektor minimálnych a 2. riadok je vektor maximálnych prípustných mutovaných hodnôt

rate - miera početnosti mutovania génov v populácii (od 0 do 1)

---

<sup>3,4</sup> TOOLBOX - GENETICKÉ ALGORITMY pre riešenie optimalizačných problémov v prostredí Matlab - POUŽÍVATEĽSKÁ PRÍRUČKA  
Ivan Sekaj

### **selbest - výber najlepších jedincov<sup>5</sup>**

#### **Charakteristika:**

Funkcia skopíruje zo vstupnej populácie do výstupnej populácie určené počty reťazcov v závislosti od ich hodnôt účelovej funkcie. O tom, ktoré reťazce budú kopírované, rozhoduje vektor Nums, ktorého prvky určujú počty vybraných reťazcov nasledovne: prvá hodnota určuje koľkokrát sa skopíruje najúspešnejší reťazec do výstupnej populácie, druhá hodnota určuje koľkokrát sa skopíruje 2. najúspešnejší reťazec do výstupnej populácie atď. Najúspešnejšími reťazcami sú chápané jedince s najmenšími hodnotami účelovej funkcie.

#### **Syntax:**

`Newpop=selbest(Oldpop,Objpop,Nums);`

Newpop - nová (výstupná) populácia

Oldpop - stará (vstupná) populácia

Objpop - vektor hodnôt účelovej funkcie starej populácie

Nums - vektor, ktorého prvky určujú, koľkokrát sa reťazec na príslušnom poradí úspešnosti skopíruje do cieľovej populácie

### **seltourn – turnajový výber<sup>6</sup>**

#### **Charakteristika:**

Funkcia vyberie zo vstupnej populácie do výstupnej populácie určený počet reťazcov. Z populácie sa vyberú dva náhodné jedince a lepší z nich sa zapíše do novej populácie. Oidva jedince sa vrátia naspäť do starej populácie a výber sa opakuje až kým nie je vybraný potrebný počet reťazcov.

#### **Syntax:**

`Newpop=seltourn(Oldpop,Objpop,Num);`

Newpop - nová (výstupná) populácia

Oldpop - stará (vstupná) populácia

Objpop - vektor hodnôt účelovej funkcie starej populácie

Num – počet vybraných reťazcov

---

<sup>5,6</sup> TOOLBOX - GENETICKÉ ALGORITMY pre riešenie optimalizačných problémov v prostredí Matlab - POUŽÍVATEĽSKÁ PRÍRUČKA  
Ivan Sekaj

### 3. Zakódovanie reťazca v populácii

Pomocou **coord\_dots\_input** načítame súradnice počiatočného a cieľového bodu. A podľa toho môžeme zdefinovať Space, ktorú budeme používať na generovanie jedincov. Riešili sme to takým spôsobom, že počet génov bude závisieť od počtu prechodových bodov, takže náš Space vyzeralo následne:

```
space=[start(1) start(2) start(1)*ones(1,dots) start(2)*ones(1,dots) stop(1) stop(2);
       start(1) start(2) stop(1)*ones(1,dots) stop(2)*ones(1,dots) stop(1) stop(2)];
```

Pomocou neho sme zabezpečili, že počiatočný aj cieľový bod bude vo všetkých prípadoch rovnaký a už sme boli schopný pri používaní funkcie **genrpop** vygenerovať 50 jedincov. V prípade, že sme mali 5 prechodových bodov, jedinci obsahovali 14 súradníc:

Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
0	0	103,6	36,8	288,5	823	497,4	37,6	966	195,2	677,2	219,9	1000	1000

### 4. Výpočet a realizácia účelovej (fitness) funkcie

Ako sme už spomínali vyššie, naša účelová (fitness) funkcia sa podobá na fitness funkciu **testfn3**, čo sme použili pri hľadaní globálneho minima Schwefelovej funkcie. Ale trochu sme ju museli modifikovať. Po prvé sme museli zistiť súradnice počiatočného a cieľového bodu, pretože vo funkcii **fitness\_test** (naša fitness funkcia) sme chceli zavolať **collision\_check**, ktorý berie ako vstupný parameter tieto súradnice, aby nám otestoval, či nedošlo ku kolízii. Ak tam bola kolízia (**collision** (čo nám vrátil **collision\_check**)>0) penalizovali sme jedinca metódou mŕtvej pokuty, teda k „fitness“ hodnote sme pripočítali obrovské číslo (v našom prípade 1000000) až potom sme vypočítali dĺžku trasy pomocou vzorca :

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

### 5. Genetický algoritmus

V každej iterácii sme vždy najprv vybrali 15 najlepších (používali sme **selbest** s nastavením [8,4,3], čo znamená, že najlepšieho jedinca sem prekopírovali 8-krát, 2. najlepšieho 4-krát a 3. najlepšieho 3-krát). Do „best15“ sme ich prekopírovali preto, lebo nechceli by sme ich stratiť pri mutácii a krížení. 35 jedincov, ktorí nám chýbali k tomu, aby sme mali tých pôvodných 50 jedincov sem vybrali pomocou **seltourn**. Nazvali sme ich skupina „work“. Urobili sme na nich kríženie pomocou **crossover** (work,1,0). Kde „1“ sme nastavili preto, aby sme mali len jeden bod kríženia, a „0“ aby sme vybrali dvojicu náhodne. Na jednoduchom príklade môžeme ukázať ako to vyzeralo:

	Before CROSSOV													
	Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
R1	0	0	103,6	36,8	288,5	823	497,4	37,6	966	195,2	677,2	219,9	1000	1000
R2	0	0	451,9	47	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
	After CROSSOV													
	Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
P1	0	0	103,6	47	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
P2	0	0	451,9	36,8	288,5	823	497,4	37,6	966	195,2	677,2	219,9	1000	1000

R1,R2 – sú rodičia (pôvodný stav) P1,P2 – sú potomkovia (nový stav)

Po crossov naša „work“ skupina vyzerala nasledovne :

0	0	103,6	47	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
0	0	509	216,9	939,1	742,9	857	164,4	51,5	682,6	448,2	58,2	1000	1000
0	0	472,7	71	484,3	490,7	903,6	434,9	427,3	762,8	366,3	641,2	1000	1000
0	0	23,3	453,1	347,6	430,5	230,4	824,7	465,5	512,2	164,9	544,2	1000	1000
0	0	538,1	773,3	534,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	70,2	384,8	113	201,4	857	134,4	738,4	892,8	874,9	323,9	1000	1000
0	0	540,5	822	873,1	498,8	434,9	573	220	882,9	172,8	180,1	1000	1000
0	0	888,8	570,1	954,2	30,8	299,3	251,9	271	541,7	103,6	809,7	1000	1000
0	0	451,9	47	303,9	617,8	54,8	800,9	440,8	393,7	66,2	717,4	1000	1000
0	0	137,1	962,5	867,3	532,6	738,8	37,6	966	195,2	677,2	219,9	1000	1000
0	0	299,9	13,8	160,4	843,7	528,6	540,4	387,4	36,4	87,7	712	1000	1000
0	0	103,6	36,8	288,5	823	497,4	868,4	804,5	436,6	451,6	941,9	1000	1000
0	0	924,4	564,8	872,7	175,8	599,3	497,2	504,7	369,4	174,8	2,7	1000	1000
0	0	268,3	716,2	47,1	451,7	173,7	950	564,3	576,2	560	651,6	1000	1000
0	0	70,2	384,8	113	201,4	86,3	859,2	781,4	983	874,2	976,6	1000	1000
0	0	538,1	773,3	534,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	23,3	453,1	347,6	430,5	230,4	824,7	465,5	512,2	164,9	641,2	1000	1000
0	0	538,1	773,3	534,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	44,1	639,2	778,2	437,7	309,7	167,1	51,5	682,6	448,2	58,2	1000	1000
0	0	991,7	956,6	503,1	168,1	989,4	256,8	776,5	657,1	578,9	623,2	1000	1000
0	0	870,5	822	873,1	498,8	434,9	573	317,4	595,7	65,9	543,5	1000	1000
0	0	967,8	693,8	47,1	451,7	869,1	675	832,2	76,1	480	136,5	1000	1000
0	0	924,4	564,8	872,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	343,5	867,7	635,2	234,6	350,8	439	440,7	862,6	377,7	895,4	1000	1000
0	0	268,3	716,2	50,2	797,4	173,7	950	564,3	576,2	560	651,6	1000	1000
0	0	145,6	560	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
0	0	384,2	139,9	520,3	575,1	165,2	723	987,9	798,6	128,3	616,3	1000	1000
0	0	845,4	667,3	497,3	753,9	247,5	881,8	317,4	595,7	65,9	543,5	1000	1000
0	0	268,3	716,2	50,2	797,4	869,1	675	832,2	76,1	677,2	219,9	1000	1000
0	0	967,8	693,8	50,2	797,4	869,1	675	832,2	76,1	480	136,5	1000	1000
0	0	206,9	770,7	850,6	489,6	697,8	268,2	929,7	346,1	880,9	200,2	1000	1000
0	0	451,9	36,8	288,5	823	497,4	37,6	966	195,2	677,2	219,9	1000	1000
0	0	248,8	795,9	206,9	232,5	296,6	941,8	625,7	276,1	66,6	577,1	1000	1000
0	0	103,6	36,8	288,5	823	497,4	37,6	966	195,2	480	136,5	1000	1000
0	0	268,3	716,2	50,2	797,4	869,1	675	832,2	76,1	480	136,5	1000	1000

Potom sme spustili - mutx(work,0.1,space), čo je globálna mutácia, ktorá nám umožní veľké modifikácie, teda veľké skoky v priestore. 0.1 na rate sme si vybrali preto , lebo maximálna odporúčaná mutácia je 10% (a keďže rate je definovaný medzi 0 a 1 , potom 0.1 = 10%). Space sme tam potrebovali kvôli tomu, aby nová mutovaná hodnota neprekročila interval zadefinovaný v space .

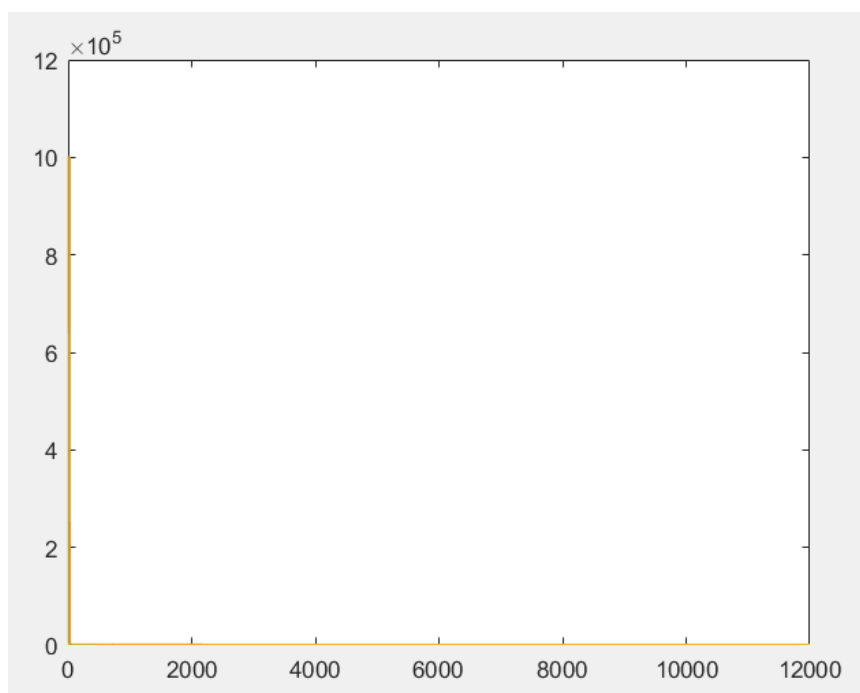
Work after MUTX													
0	0	768	47	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
0	0	509	216,9	146,9	742,9	857	164,4	51,5	682,6	448,2	58,2	1000	1000
0	0	472,7	71	484,3	490,7	467,4	238,6	427,3	762,8	366,3	641,2	1000	1000
0	0	23,3	453,1	347,6	430,5	230,4	984,3	465,5	512,2	164,9	544,2	1000	1000
0	0	538,1	773,3	534,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	70,2	384,8	113	201,4	408,4	164,4	738,4	892,8	874,9	323,9	1000	1000
0	0	870,5	822	873,1	498,8	434,9	573	220	882,9	172,8	180,1	1000	1000
0	0	888,8	570,1	954,2	30,8	299,3	251,9	271	541,7	103,6	809,7	1000	1000
0	0	451,9	47	303,9	617,8	54,8	800,9	440,8	154,3	66,2	717,4	1000	1000
0	0	137,1	962,5	867,3	532,6	738,8	37,6	966	195,2	677,2	219,9	1000	1000
0	0	299,9	13,8	453,4	843,7	528,6	540,4	387,4	36,4	87,7	712	1000	1000
0	0	103,6	36,8	288,5	823	497,4	868,4	804,5	436,6	451,6	941,9	1000	1000
0	0	924,4	564,8	872,7	175,8	599,3	497,2	504,7	369,4	174,8	2,7	1000	1000
0	0	268,3	716,2	47,1	816,4	173,7	950	504,9	576,2	901,8	651,6	1000	1000
0	0	70,2	384,8	57,6	201,4	86,3	859,2	781,4	983	874,2	976,6	1000	1000
0	0	538,1	773,3	534,7	721,7	864,4	196	952,4	293,1	839,1	19,3	1000	1000
0	0	23,3	453,1	347,6	430,5	230,4	824,7	465,5	512,2	164,9	641,2	1000	1000
0	0	538,1	773,3	534,7	808,7	599,3	497,2	504,7	369,4	174,8	2,7	1000	1000
0	0	363	639,2	778,2	437,7	309,7	167,1	51,5	682,6	448,2	58,2	1000	1000
0	0	991,7	956,6	375,2	168,1	989,4	256,8	776,5	657,1	578,9	623,2	1000	1000
0	0	870,5	822	873,1	498,8	434,9	573	950,3	595,7	65,9	543,5	1000	1000
0	0	967,8	693,8	47,1	451,7	869,1	675	832,2	76,1	480	136,5	1000	1000
0	0	924,4	564,8	872,7	721,7	864,4	196	952,4	293,1	839,1	291,7	1000	1000
0	0	343,5	867,7	635,2	878,8	350,8	439	440,7	862,6	377,7	895,4	1000	1000
0	0	571	716,2	50,2	797,4	173,7	950	564,3	576,2	560	651,6	1000	1000
0	0	145,6	560	468,7	223	364,8	428,9	591,3	187,8	416,4	43	1000	1000
0	0	384,2	139,9	520,3	575,1	165,2	723	987,9	798,6	128,3	895	1000	1000
0	0	845,4	667,3	497,3	753,9	247,5	881,8	317,4	595,7	65,9	543,5	1000	1000
0	0	217,1	716,2	50,2	797,4	869,1	33,9	832,2	76,1	677,2	451,9	1000	1000
0	0	967,8	693,8	50,2	111	869,1	675	832,2	76,1	480	136,5	1000	1000
0	0	206,9	770,7	850,6	489,6	697,8	320,6	929,7	346,1	880,9	200,2	1000	1000
0	0	451,9	36,8	288,5	823	233,1	601,5	966	328,1	677,2	219,9	1000	1000
0	0	248,8	795,9	206,9	232,5	296,6	941,8	927,4	276,1	66,6	577,1	1000	1000
0	0	103,6	36,8	288,5	823	497,4	37,6	966	195,2	776,2	684,8	1000	1000
0	0	268,3	716,2	50,2	797,4	869,1	675	832,2	76,1	480	136,5	1000	1000

Aby sme dostali čo najbližšie k najkratšej trase a čo najrýchlejšie sme potrebovali spustiť aj muta(work,**0.1,Amp,space**), čo je lokálna mutácia, ktorá nám umožní menšie modifikácie, teda, aby sme sa dostali čo najbližšie k prekážkam (bez kolízií) pre kratšiu trasu. Takisto ako v prípade mutx, preto sme si vybrali 0.1, lebo je to maximálna odporúčaná mutácia.

Amp sme nastavili na [0,0,50\*ones(1,dots),50\*ones(1,dots),0,0], aby sa nám súradnice počiatočného a cieľového bodu nezmenili, ale okrem toho bol schopný aj pracovať ľubovoľným počtom prechodových bodov. Space je tam z toho istého dôvodu ako v prípade mutx.

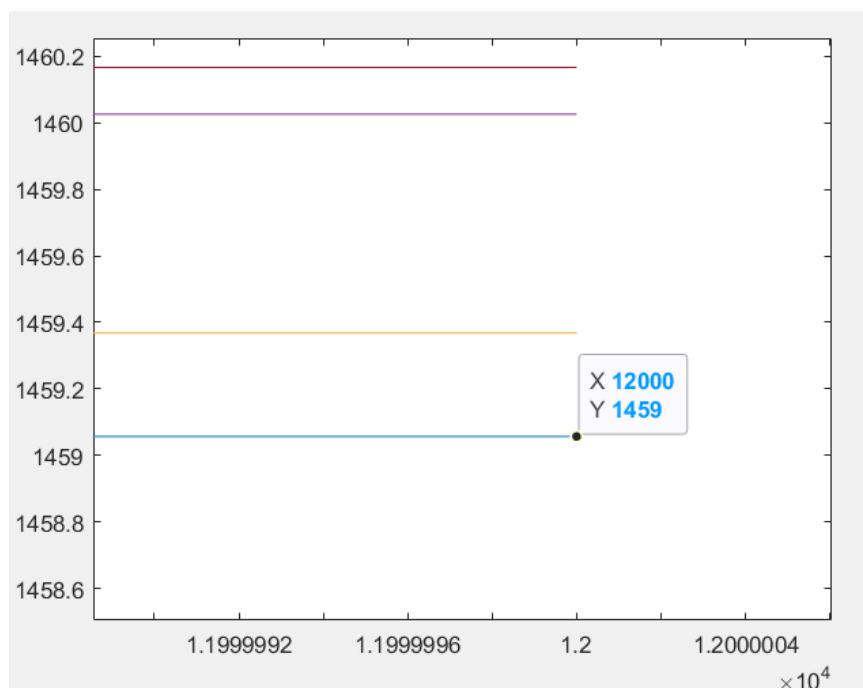
Before MUTA													
Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
0	0	363	639,2	778,2	437,7	309,7	167,1	51,5	682,6	448,2	58,2	1000	1000
After MUTA													
Start_x	Start_y	Bod1_x	Bod1_y	Bod2_x	Bod2_y	Bod3_x	Bod3_y	Bod4_x	Bod4_y	Bod5_x	Bod5_y	Stop_x	Stop_y
0	0	363	639,2	817,4	437,7	309,7	167,1	128,3	682,6	448,2	58,2	1000	1000

## 6. Dosiahnuté výsledky po 10 spustení



Figures – Fittrend\_10\_Amp\_50

Po 10 spustení sme dostali takýto fittrend, čo nám moc neprezradí, ale, keď ju trošku zväčšíme:

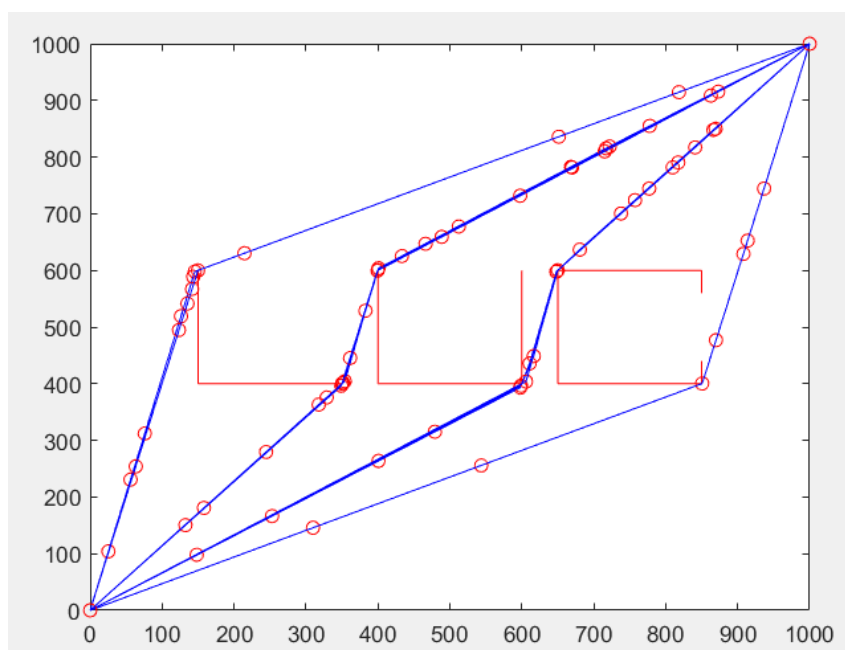


Figures – Fittrend\_10\_Amp\_50

Môžeme zistiť, že najlepší jedinec má dĺžku 1459 a je značený modrou farbou, (túto hodnotu sme dosiahli v **10280. generácii**) a vidíme nad ním je žltá farba, čo znamená, že z desiatich spustení sme tam dostali dvakrát dĺžku 1459.

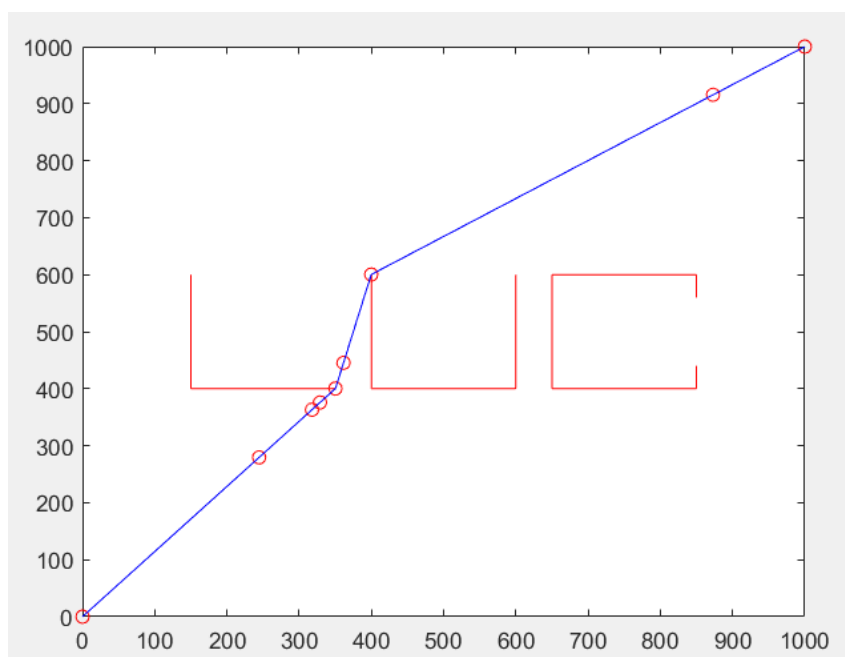


Takto vyzerali vykreslené trasy. Tieto boli najlepšie v danom spustení.



Figures – Robot\_10\_Amp\_50

Keďže sme už videli ktorá bola najlepšia dĺžka bolo dobré vedieť aj to, ktorá trasa z desiatich je tá najlepšia .



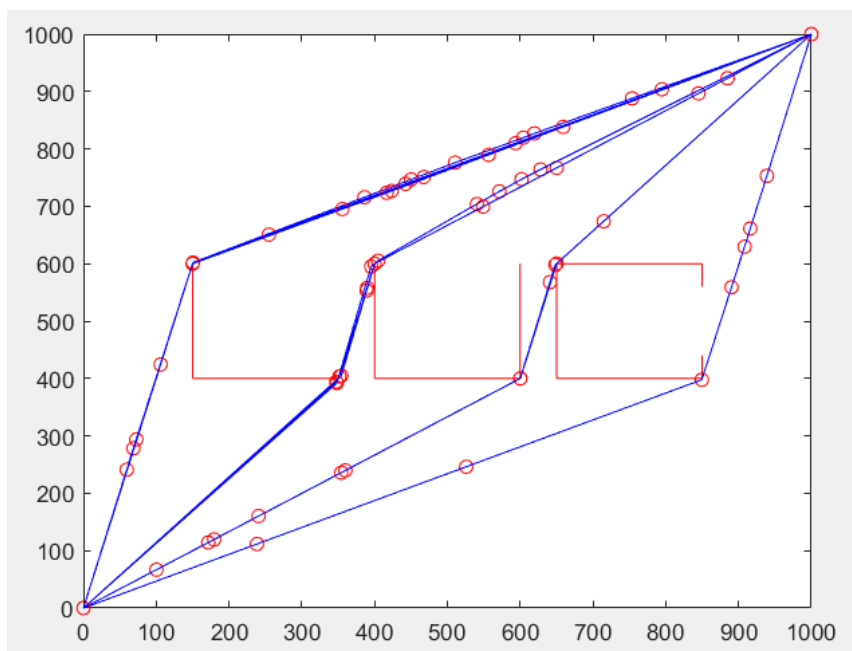
Figures – Best\_of\_10\_Amp\_50

A teda táto trasa bola úplne najlepšia z desiatich a mala dĺžku :

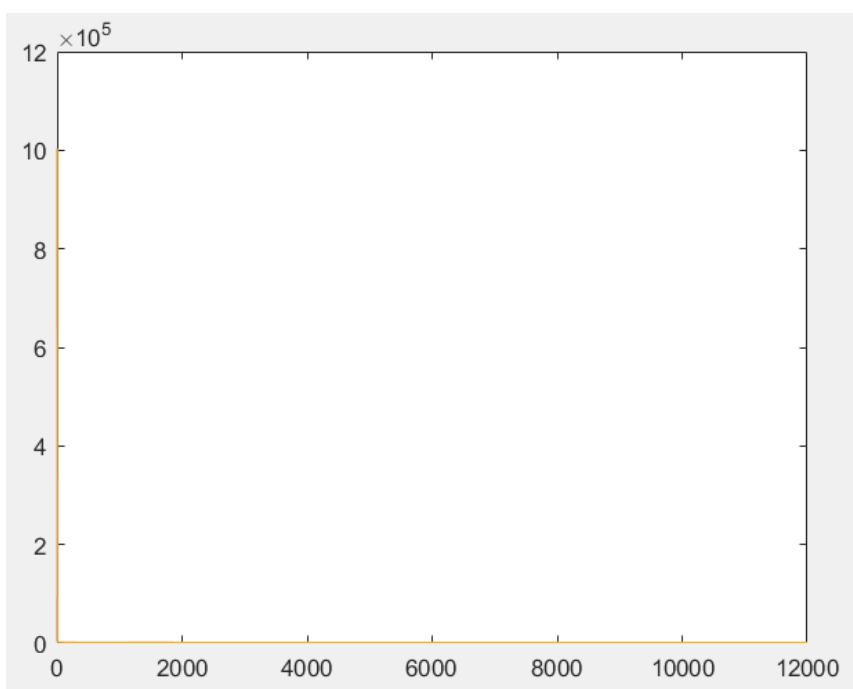
Najlepšie riešenie má dĺžku  
1.4591e+03

## 7. Dosiahnuté výsledky pri modifikácii GA

Keby sme Amp zmenili na  $[0,0,100*\text{ones}(1,\text{dots}),100*\text{ones}(1,\text{dots}),0,0]$  dostali by sme trasy:



Figures – Robot\_10\_Amp\_100

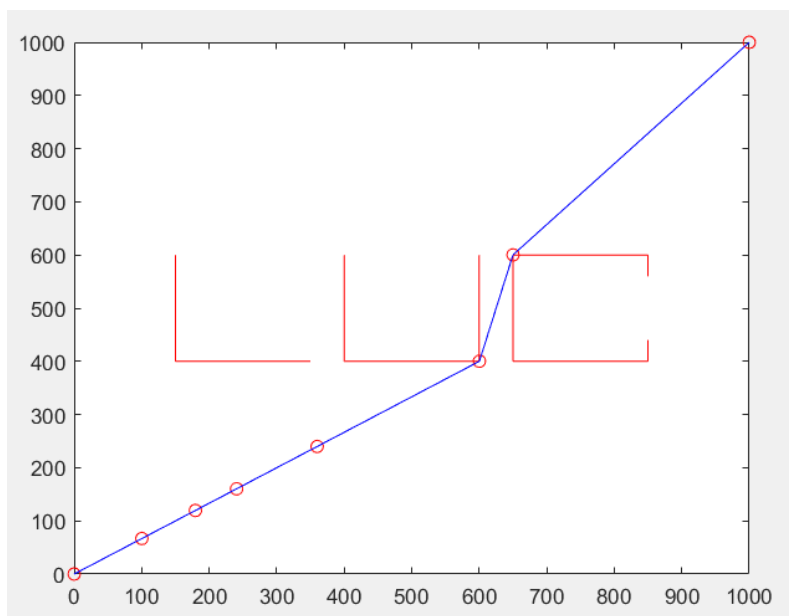


Figures – Fittrend\_10\_Amp\_100

Po zvážení sme zistili, že najlepšie riešenie v tomto prípade je okolo 1459 presnejšie:

Najlepšie riešenie má dĺžku  
1.4589e+03

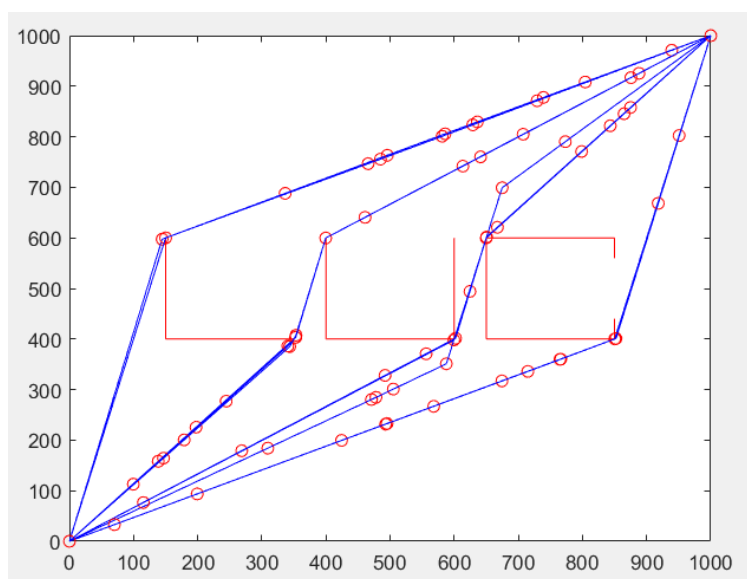
Túto hodnotu sme dosiahli v **6231**. generácii, a „najhoršie“ riešenie mala dĺžku 1559. Na rozdiel od predchádzajúceho príkladu, sme dostali do najkratšej dĺžky rýchlejšie, ale dĺžku 1459 sme dosiahli trikrát.



Figures – Best\_of\_10\_Amp\_100

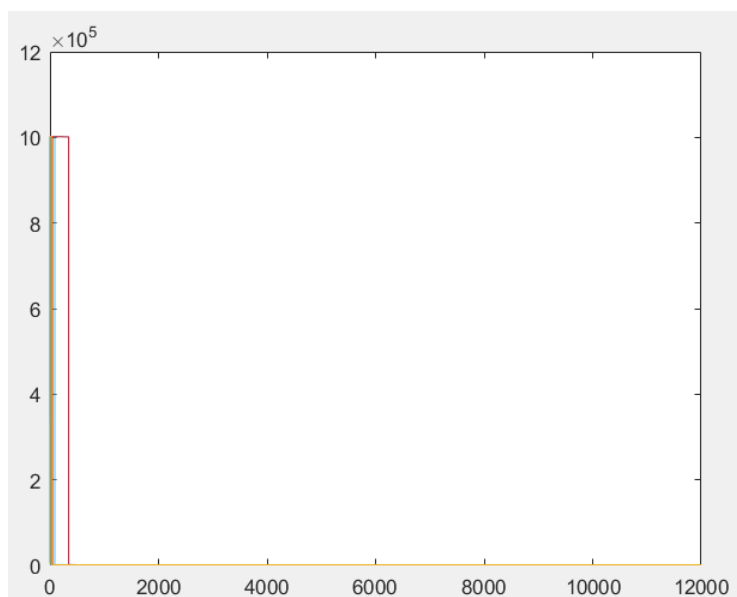
- Najlepšia trasa v prípade  $\text{Amp}=[0,0,100*\text{ones}(1,\text{dots}),100*\text{ones}(1,\text{dots}),0,0]$

Keby sme tam nemali crossov dostali by sme tieto trasy:



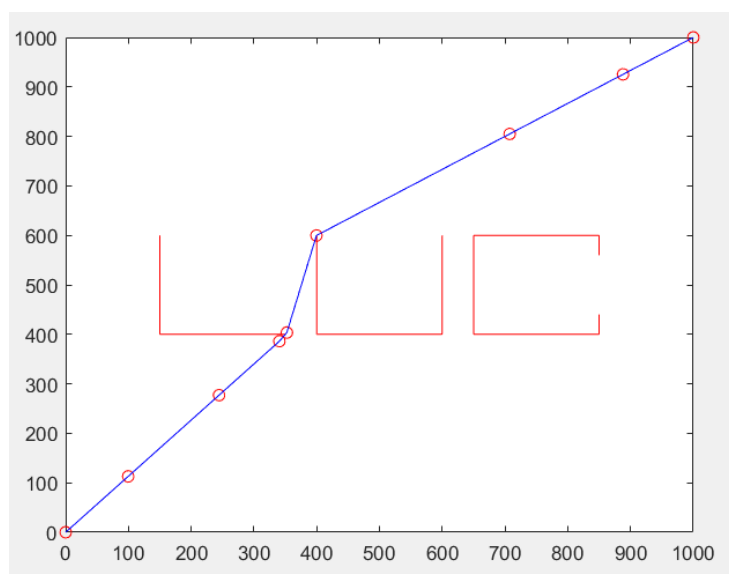
Figures – Robot\_10\_No\_Crossover

Fittrend v prípade, že nepoužívame crossov .



Figures – Fittrend\_10\_No\_Crossover

Najlepšie riešenie bolo okolo 1459.3 , čo je trochu horšie ako sme dosiahli v normálnom prípade.

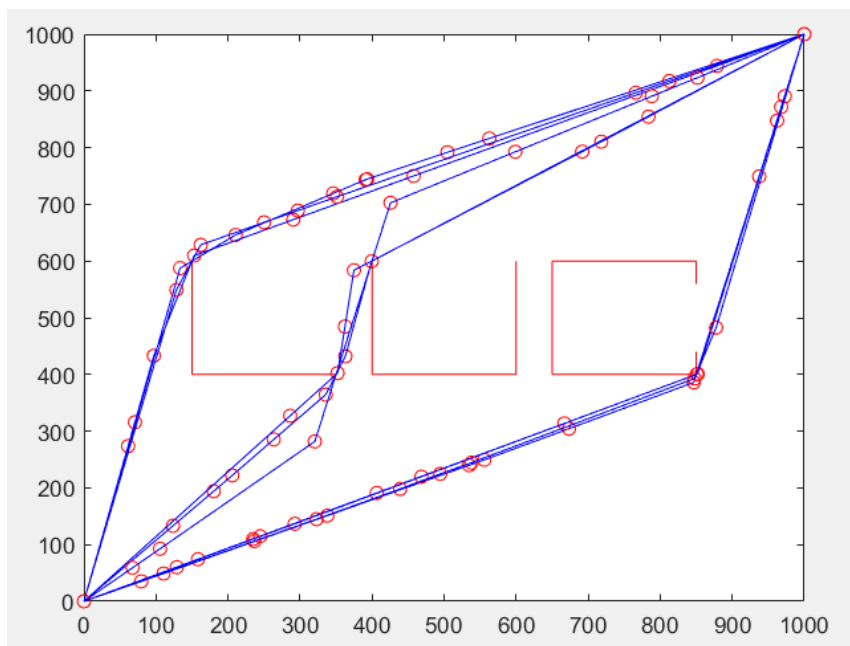


Figures – Best\_of\_10\_No\_Crossover

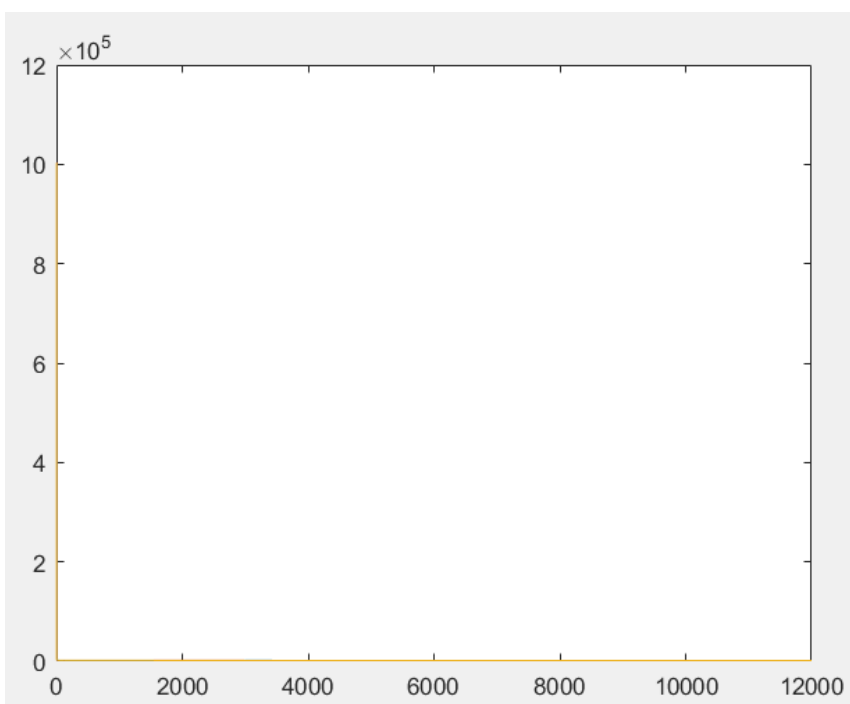
Najlepší jedinec bez použitia crossov s dĺžkou:

Najlepšie riešenie má dĺžku  
1.4593e+03

Keby sme nepoužívali lokálnu mutáciu dostali by sme trasy:

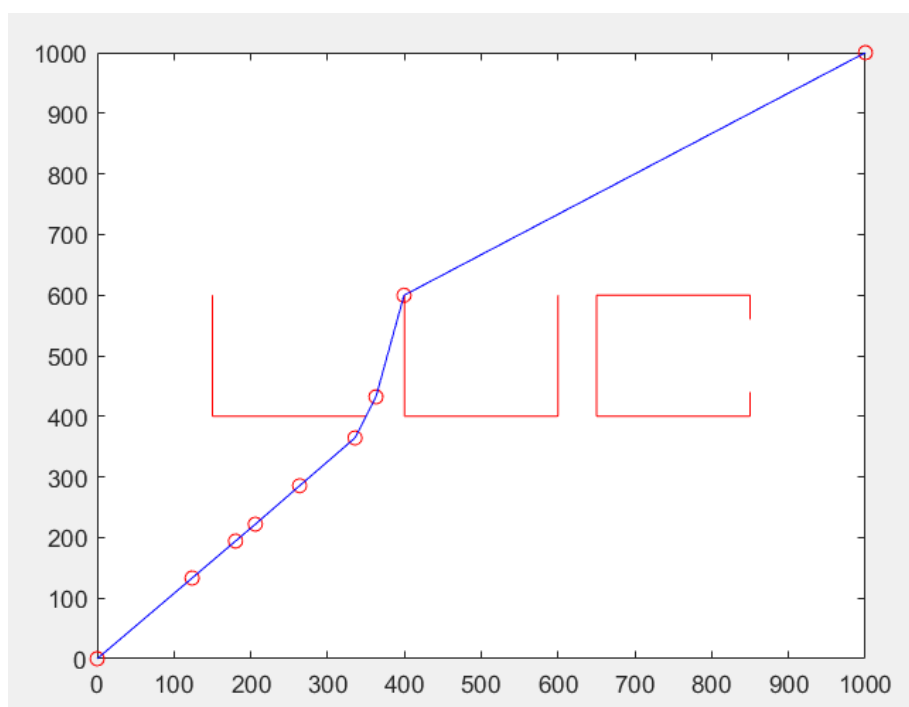


Figures – Robot\_10\_No\_Muta



Figures – Fittrend\_10\_No\_Muta

Všetky riešenia , ktoré sme dostali mali dĺžku nad 1460, to znamená, že bez muta nevieme dosiahnuť úplne najkratšiu trasu .



Figures – Best\_of\_10\_No\_Muta

Najlepšie riešenie bez používania muta má dĺžku 1462.

Najlepšie riešenie má dĺžku  
1.4617e+03