EC-Grid problem:

- **Population Model** : Steady state model with probability 90%
- **Chromosome length** : 20 Bits
- **Population size** : 400
- **Maximum number of generations :** 4000
- **Crossover probability :** 90%
- **mutation probability :** 0.1% (per individual, not per position)
- **Selection pressure** : in parent selection (select from 70% best)
- **Crossover method** : one-point & uniform

Representation:

Array with size 20 , each element determine location of number I as (x,y) tuple in grid environment.

results are for successful runs were solution found in determined generation limit. Algorithm time contains time for printing each generation info.

Uniform Crossover :

Generation Number	Algorithm Time (sec)
1059	2.79
827	1.98
1285	3.2
419	1.04
621	1.65
645	1.64
728	1.86
550	1.44
558	1.47
998	2.43

Results :

Average Generation Number : 769
Average Algorithm Time : 1.78 sec

One point Crossover:

Generation Number	**Algorithm Time (sec)**
1940	3.73
3011	5.43
836	1.53
1667	3.09
2242	4.20
1174	2.18
1559	2.83
3277	6.18
1632	2.95
2132	3.84

Results :

Average Generation Number : 1947
Average Algorithm Time : 3.596 sec

success rate of one-point was less than uniform.
convergence speed to local optimum solution is high

10 last best chromosome:

('generation:', 24, ' best fit:', 5.732050807568877)
('generation:', 25, ' best fit:', 3.414213562373095)
('generation:', 26, ' best fit:', 3.414213562373095)
('generation:', 27, ' best fit:', 2.0)
('generation:', 28, ' best fit:', 2.414213562373095)
('generation:', 29, ' best fit:', 2.0)
('generation:', 30, ' best fit:', 2.0)
('generation:', 31, ' best fit:', 2.0)
('generation:', 32, ' best fit:', 1.0)
('generation:', 33, ' best fit:', 0.0)

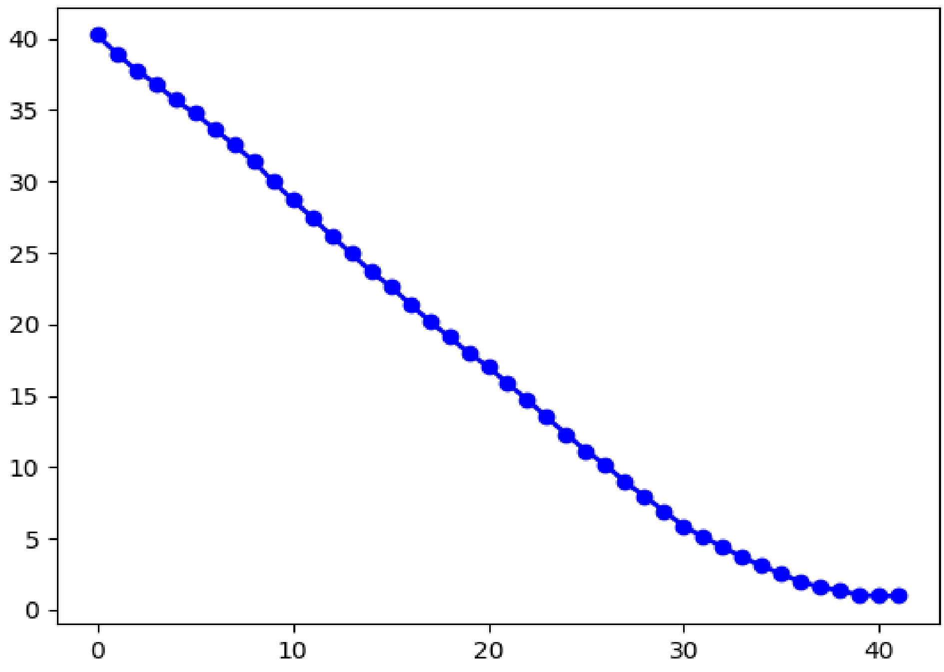

Illustration 1: average fitness per generation

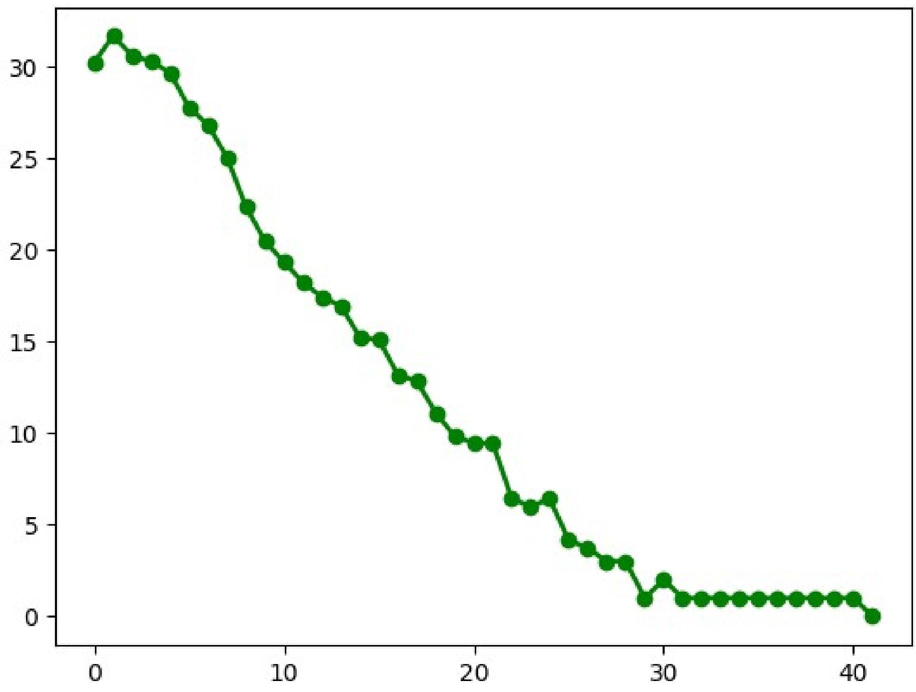

Illustration 2: best fitness per generations

Robot problem:

- **Population Model** : Generational model
- **Chromosome length** : 243 Bits
- **Population size** : 300
- **Maximum number of generations :** 1000
- **Crossover probability :** 100%
- **mutation probability :** 0.5% (per chromosome of individual)
- **Selection pressure** : in parent selection (select from 70% best) with Roulette wheel method .
- **Crossover method** : one-point & uniform

Representation:

Array with size 243 witch each elements of that is a possible robot action from the action list below :

["n","s","e","w","r","st","b"] :

n : go north

s : go south
e : go east
w : go west
r : choose random direction
st : stay up
b : bend down to pick up a can

results are for successful runs were solution found in determined generation limit. Algorithm time contains time for printing each generation info.

One point Crossover :

Generation Number	Algorithm Time (sec)
850	45.75
504	26.76
475	26.33
104	5.48
263	13.77
207	11.28
328	17.53
117	6.38
204	10.98
849	44.27

Results :

Average Generation Number : 390
Average Algorithm Time : 20.853 sec

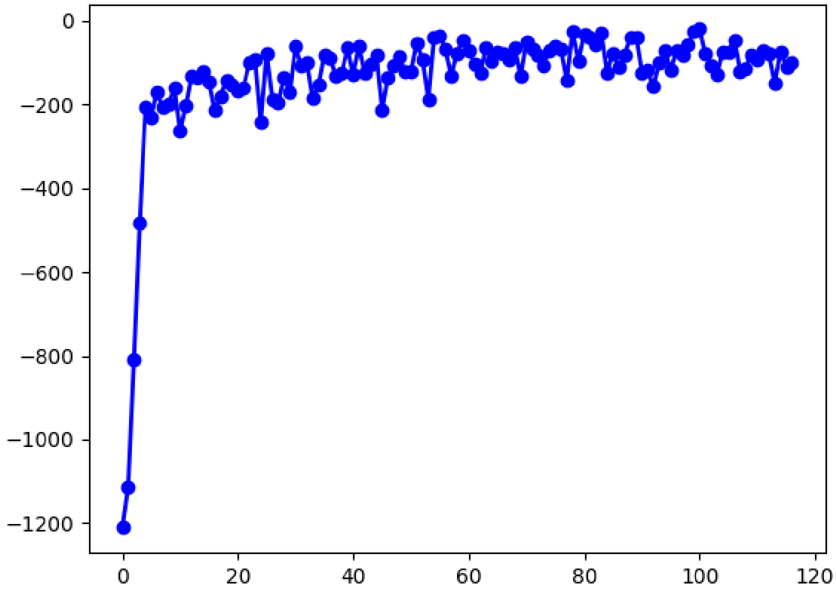

Illustration 3: average fitness per generation

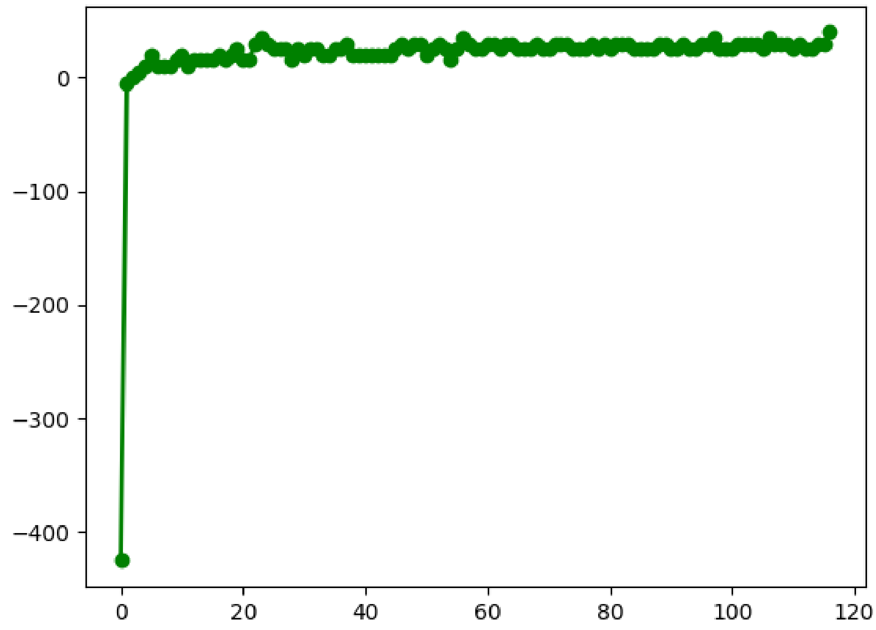

Illustration 4: best fitness per generation

STD String problem:

- **Population Model** : Steady state model with probability 90%
- **Chromosome length** : equal to target chromosome length
- **Population size** : 900
- **Maximum number of generations :** 300
- **Crossover probability :** 90%
- **mutation probability :** 0.1% (per individual, not per position)
- **Selection pressure** : in parent selection (select from 70% best)
- **Crossover method** : uniform

Representation:

string with size equal to target string size , each character can be alphabet (a-z) or digits (0-9).

results are for successful runs were solution found in determined generation limit. Algorithm time contains time for printing each generation info.

Uniform Crossover :

Generation Number	Algorithm Time (sec)
47	0.47
45	0.42
42	0.45
45	0.45
47	0.46
43	0.42
46	0.45
43	0.42
43	0.42
46	0.44

Results :

Average Generation Number : 45

Average Algorithm Time : 44 sec

convergence speed to local optimum solution is high such that for generation number in (40-50) , search algorithm stuck into local optimum.

because of high speed diversity lost we have to increase population size at the beginning.

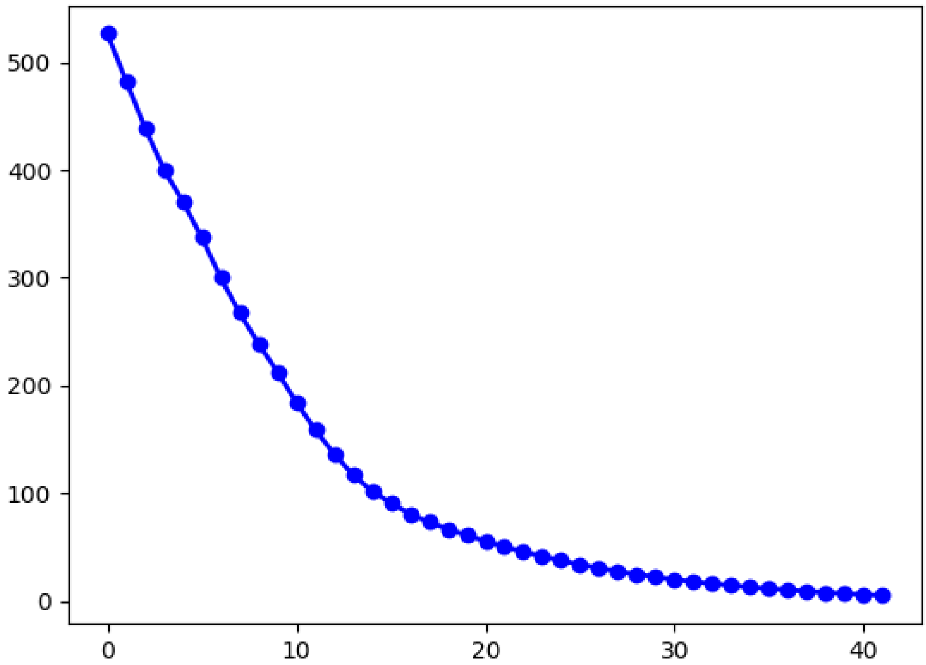

Illustration 5: average fitness per generation

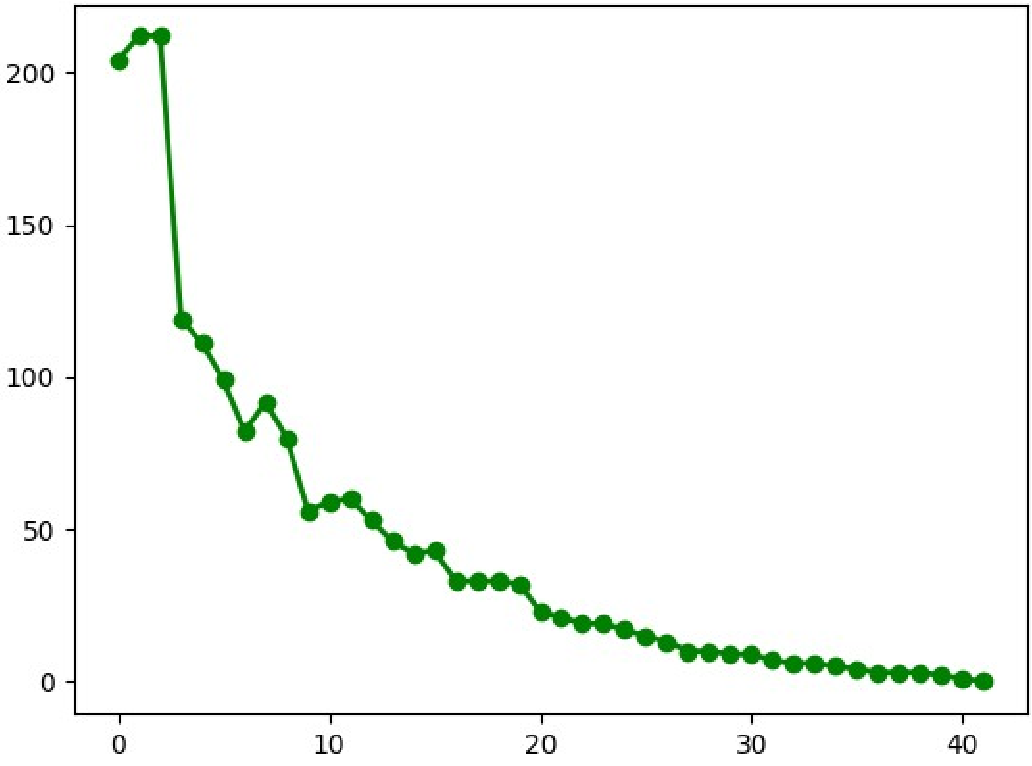

Illustration 6: best fitness per generation