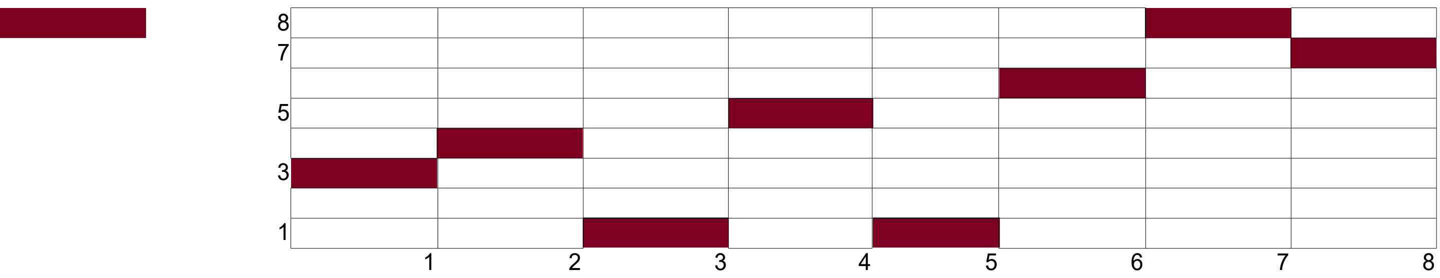
Introductory steps:
-what problem (task formulation): **8 Queens Problem** (place 8 queens on a (8x8) chessboard in a non-attacking configuation)
- search space: the set of all the chessboard configurations???
- genome: (slightly) complex
=> we can limit the search space to configurations containing only one queen in one column
- genome: 8-digit sequences, as:

3 4 1 5 1 6 8 7 **(genome)**
\- - - - - - - -



(individual)

- fitness function: 28 - attacks

(fitness f. Should be non-negative for the roulette wheel method)

We decided to take N=6 individuals to the population (too few for the computer implementation)

GENERATION OF THE INITIAL POPULATION

We use www.random.org to generate 6 8-digit genomes
Genome of individual 1 (Ind_1):

5	4	5	2	1	5	3	8

Genome of Ind_2:

5	6	7	7	2	6	3	6

Genome of Ind_3:

1	3	1	2	6	3	4	3

Genome of Ind_4:

3	2	4	5	5	2	3	7

Genome of Ind_5:

8	5	8	7	4	5	7	4

Genome of Ind_6:

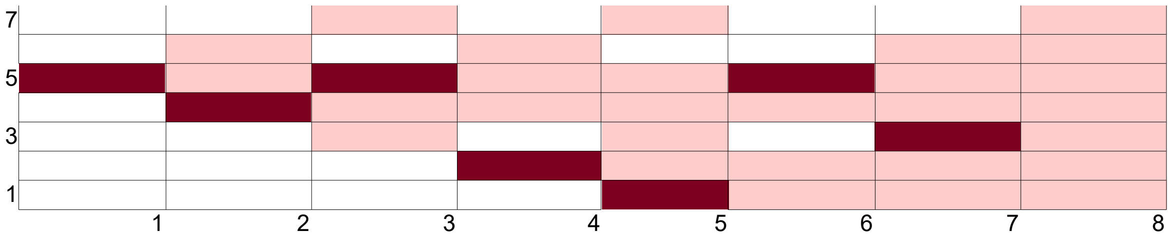
8	6	7	8	3	5	2	6

SELECTION

a) Evaluation (if any individual has the maximal fitness, we quit)

Ind_1:

1)5
2)3
3)1
4) 1
5) 1
6)0
7)0
11 attacks

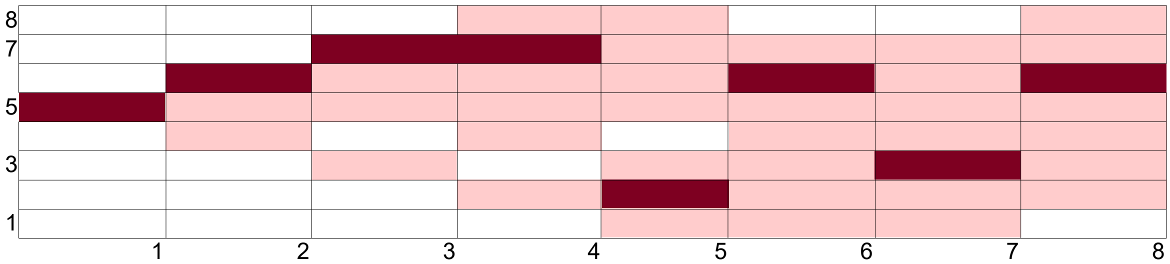


5 4 5 2 1 5 3 8

F_1=f(Ind_1)=**28-attacks(Ind_1)**=28-11=**17**

Ind_2:

8 attacks
2
3
2
0
0
1
0

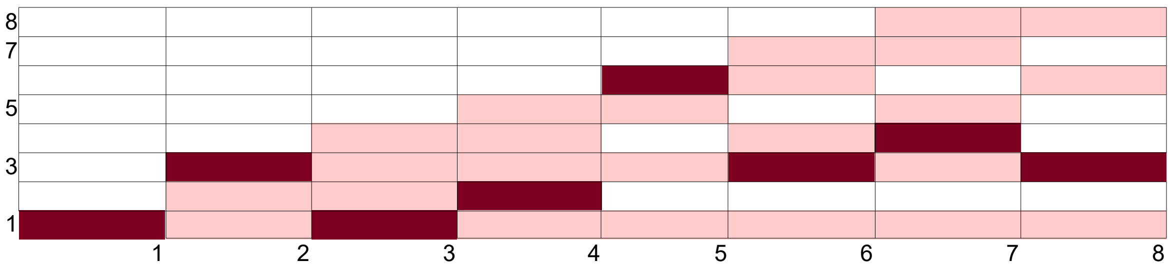


5 6 7 7 2 6 3 6

F2=f(Ind_2)=28-8=20

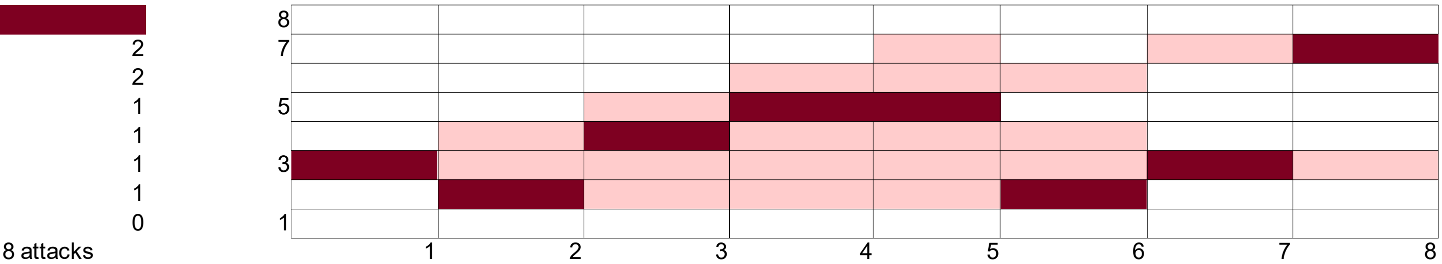
Ind3:

1
3
1
0
2
2
1
10 attacks



F3=f(Ind_3)=28-10=18

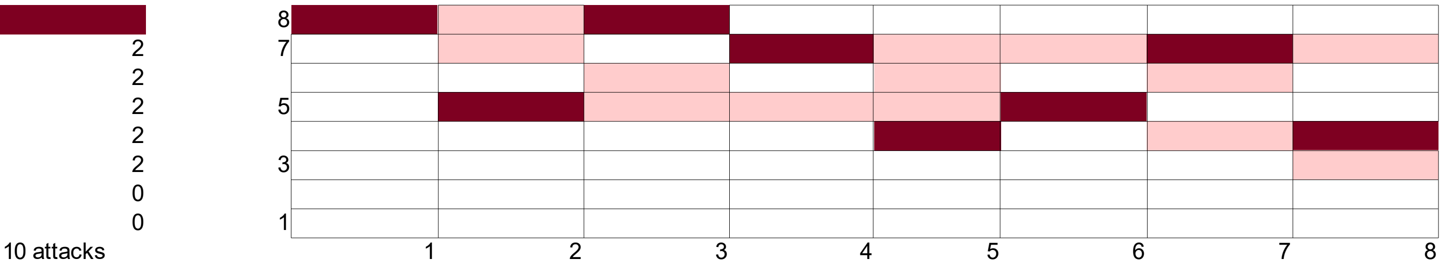
Ind4:



3 2 4 5 5 2 3 7

F4=f(Ind_4)=28-8=20

Ind5

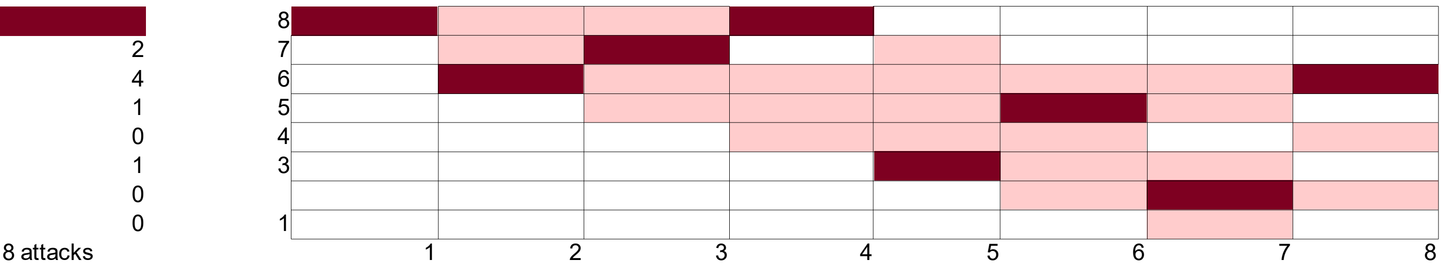


8 5 8 7 4 5 7 4

F5=28-10 =18

Ind 6

8 6 7 8 3 5 2 6



8 6 7 8 3 5 2 6

F6=28-8=20

b) Creating the roulette wheel

F1=17, F2=20 F3=18 F4=20 F5=18 F6=20

F=F1+F2+...+F6=53+60=113 (total fitness)

S=[0,F)=[0,113) ← total interval

subintervals

S1=[0,F1)=**[0,17)** **|S1|=F1 (the length of S1)**

S2=[F1,F1+F2)=**[17,37) |S2|=F2**

S3=[F1+F2,F1+F2+F3)=**[37,55) |S3|=F3**

S4=**[55,75)**

S5=[75,**93)**

S6=**[93,113)** **c) Running the roulette wheel**

Draw 6 random numbers r1,r2,...,r6 from [0,1]

→ r1=0.76, r2=0.96, r3=0.98,r4=0.29, r5=0.65, r6=1.00

Then we calculate:

R1=r1*F, R2=r2*F,..., R6=r6*F (so R1,R2,..,R6 are from **S**)

R1=85,88 belongs to S5 → Ind 5

R2=108,48 belongs S6 → Ind 6

R3>R2 => belongs to S6 → Ind 6

R4=34,77 belongs to S2 → Ind 2

R5=73,4 belongs to S4 → Ind 4

R6=113 → Ind 6

Hence the **Mating Pool** is: **Ind 5** **Ind 6** **Ind 6** **Ind 2** **Ind 4** **Ind 6**

CROSSOVER

Pair I (Ind 5---Ind 6)

draw random r1,r2 from {1,...,7}-> r1=r2=4 ->

parent G5	8	5	8	7	| |	4	5	7	4
parent G6	8	6	7	8	||	3	5	2	6
offspring	8	5	8	7		4	5	7	4
offspring	8	6	7	8		3	5	2	6

Pair I (Ind 6---Ind 2)

draw random r3,r4 from {1,...,7}-> r3=3, r4=7 ->

parent G6 8 6 7 | 8 3 5 2 | 6

parent G2 5 6 7 | 7 2 6 3 | 6

offspring 8 6 7 7 2 6 3 6

offspring 5 6 7 8 3 5 2 6

pair III (Ind 4---Ind 6)

draw random r5,r5 from {1,...,7}-> r5=1, r6=3 ->

parent G4 3 | 2 4 | 5 5 2 3 7

parent G6 8 | 6 7 | 8 3 5 2 6

offspring 3 6 7 5 5 2 3 7

offspring 8 2 4 8 3 5 2 6

Hence we've obtained the following set of

Offsprings

O1 8 5 8 7 4 5 7 4

O2 8 6 7 8 3 5 2 6

O3 8 6 7 7 2 6 3 6

O4 5 6 7 8 3 5 2 6

O5 3 6 7 5 5 2 3 7

O6 8 2 4 8 3 5 2 6

Mutation

p_m – **mutation probability (for the single gene)**

here (for demonstration purposes) we assume p_m=1/15 (for computer implementation try p_m from [0.002,0.01]**)**

Since there are **48**=6*8 **genes ,** we need 48 random numbers

r_1^1	r_1^2	r_1^3	r_1^4	r_1^5	r_1^6	r_1^7	r_1^8
r_2^1	r_2^2	r_2^3	r_2^4	r_2^5	r_2^6	r_2^7	r_2^8
r_3^1	r_3^2	r_3^3	r_3^4	r_3^5	r_3^6	r_3^7	r_3^8
r_4^1	r_4^2	r_4^3	r_4^4	r_4^5	r_4^6	r_4^7	r_4^8
r_5^1	r_5^2	r_5^3	r_5^4	r_5^5	r_5^6	r_5^7	r_5^8
r_6^1	r_6^2	r_6^3	r_6^4	r_6^5	r_6^6	r_6^7	r_6^8

from the set of integers: **{1,2,3,...,15}** (as **p_m=1/15**). We assume that if r_i=1 then it means that the gene number i will be mutated.

Here are my random numbers

13	5	1	7	12	14	4	7
4	4	9	12	7	7	3	2
6	8	9	7	5	3	7	2
1	1	6	5	11	2	9	1
8	7	1	1	6	15	2	15
14	1	1	7	5	7	14	8

r_1^3,r_4^1,r_4^2,r_4^8,r_5^3,r_5^4,r_6^2,r_6^3=1

---> the genes O1(3), O4(1), O4(2), O4(8), O5(3), O5(4),O6(2), O6(3) should be mutated.

We generate new random values for these 8 genes:

2 4 7 2 8 5 3 6

2

O1	8	5	8	7	4	5	7	4
N1	**8**	**5**	**2**	**7**	**4**	**5**	**7**	**4**

4 7 2

O4	5	6	7	8	3	5	2	6
N4	**4**	**7**	**7**	**8**	**3**	**5**	**2**	**2**

8 5

O5	3	6	7	5	5	2	3	7
N5	3	6	8	5	5	2	3	7

3 **6**

O6 8 2 4 8 3 5 2 6

N6 8 3 6 8 3 5 2 6

New population: **N1,N2,N3,N4,N5,N6**

N1	**8**	**5**	**2**	**7**	**4**	**5**	**7**	**4**
N2	**8**	**6**	**7**	**8**	**3**	**5**	**2**	**6**
N3	**8**	**6**	**7**	**7**	**2**	**6**	**3**	**6**
N4	**4**	**7**	**7**	**8**	**3**	**5**	**2**	**2**
N5	**3**	**6**	**8**	**5**	**5**	**2**	**3**	**7**
N6	**8**	**3**	**6**	**8**	**3**	**5**	**2**	**6**

Then we replace the old population with the new one, that is:

G1=N1, G2=N2, …,G6=N6

… and repeat the steps (until finding a non-attacking configuration)

„Paper version": max. 4 points ; take N=6, p_m=1/15; do 5 iterations of the algorithm