

# Cvičení 1 – Algoritmus Metropolis

**Jméno:** Kateřina Fořtová (xforto00)

## Problém N dam

**Charakteristika experimentu:** Jak lze rozmístit  $n$  dam na šachovnici o rozměrech  $n \times n$  tak, aby se vzájemně neohrožovaly?

### Řešení fitness funkce:

```
def fitness (qs):  
    confs = 0  
  
    # (i - j) = k - 1 or i + j = k + 1. for conflict on diagonals  
    for queen1 in range(len(qs)):  
        for queen2 in range(len(qs)):  
            if (queen1 == queen2):  
                continue  
  
            if (queen1 - qs[queen1] == queen2 - qs[queen2] or queen1 + qs[queen1] == queen2 + qs[queen2]):  
                confs = confs + 1  
  
    return confs
```

### Upravení části Metropolis algoritmu pro předčasné ukončení iterací po fitness = 0:

```
if is_accepted:  
    prevq = newq[:]  
    prevfit = newfit  
    print >> sys.stderr, i, newfit  
  
    if (newfit == 0):  
        break
```

### Nastavení různých parametrů pro maximálně 10 000 iterací:

SIZE = 8

MAX\_ITER = 10000

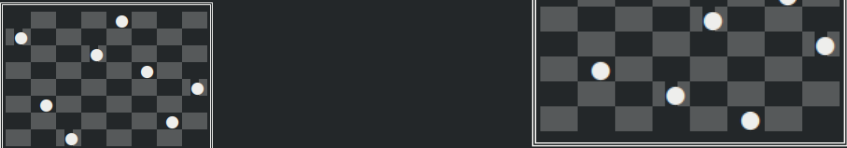
PARAM = 0.5 (levý sloupec) a 0.25 (pravý sloupec)

```
katerina@katerina-ASUS:~/Documents/EVO/cv1$ python Nqueens.py
[0, 1, 2, 3, 4, 5, 6, 7]
1 10
2 10
3 8
4 8
5 8
6 6
7 4
8 4
9 4
10 4
11 4
12 4
13 4
14 4
15 2
16 2
17 2
18 2
19 2
20 2
21 2
22 2
23 2
24 2
25 2
26 2
27 2
28 2
29 2
30 2
31 2
32 2
33 2
34 2
35 2
36 2
37 2
38 2
39 2
40 2
41 2
42 2
43 2
44 0

Nalezene reseni, fitness = 0
Genotyp
[2, 0, 6, 4, 7, 1, 3, 5]
Fenotyp

katerina@katerina-ASUS:~/Documents/EVO/cv1$ python Nqueens.py
[0, 1, 2, 3, 4, 5, 6, 7]
0 6
1 6
2 2
15 2
27 2
38 2
43 2
44 0

Nalezene reseni, fitness = 0
Genotyp
[4, 0, 3, 5, 7, 1, 6, 2]
Fenotyp
```



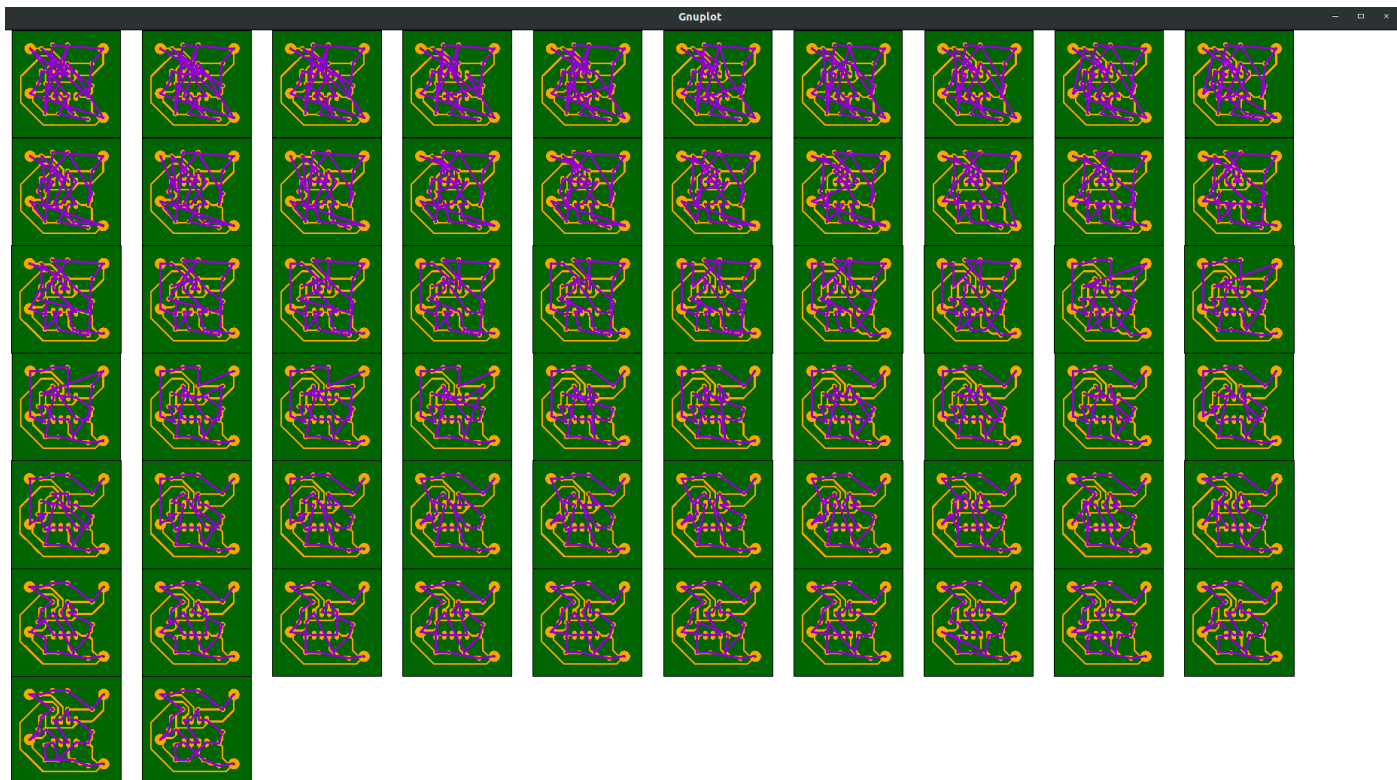
## TSP

**Charakteristika experimentu:** V daném ohodnoceném úplném grafu najděte nejkratší hamiltonovskou kružnici.  
Př.: vrtání desek plošných spojů -- co nejkratší dráha vrtáku:

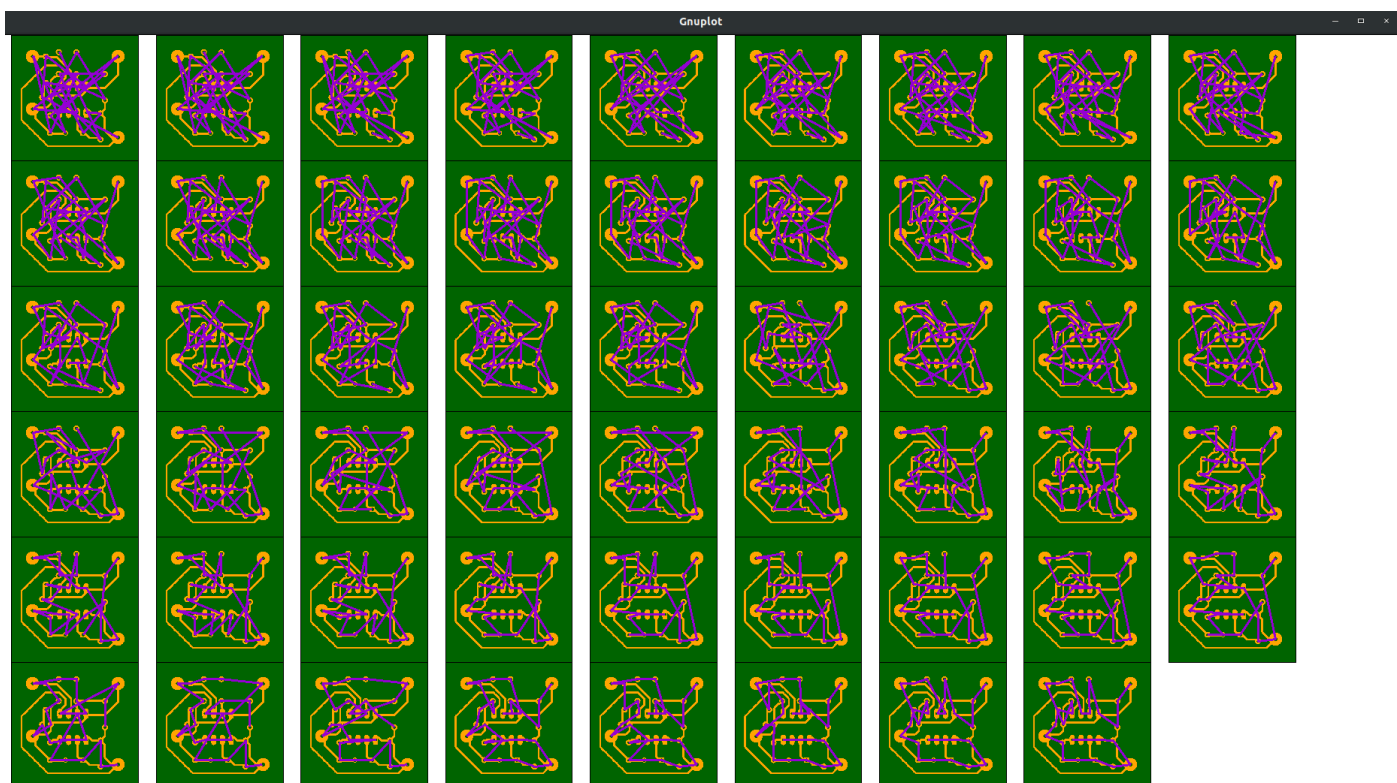
Využití Simulovaného žíhání - **nalezeno řešení po nejméně iteracích (využito parametru cool - mění se váha akceptace přijmutí řešení)**



Využití Metropolis - někdy akceptuje horší řešení:



Využití Hill Climbing - vždy akceptuje pouze lepší řešení:



**Zhodnocení:** Při problematice N dam mělo na velikost potřebných iterací vliv jak PARAM, který využíval Metropolis algoritmus, tak počet dam, které chceme rozmístit (při vyšším počtu dam bylo nutné provést více iterací algoritmu). Při využití simulovaného žihání u problému TSP bylo nalezeno nejlepší řešení v nejkratším čase ve srovnání s ostatními dvěma metodami. Jedná se o algoritmus, kdy se mění váha akceptace přijetí řešení pomocí parametru cool.