David Napravnik

5. HW TSP

Co jsem zkusil a jak to dopadlo

- Vytvareni jedince hladovym algoritmem (`create_short_ind`)
    - prvni mesto je nahodne
    - pridavame nejblizsi zatim nenavstivene mesto
- Vlatni crossover (`edge_recombination_cross`)
- Chytra heuristika / mutace (`opt2_mutate`)

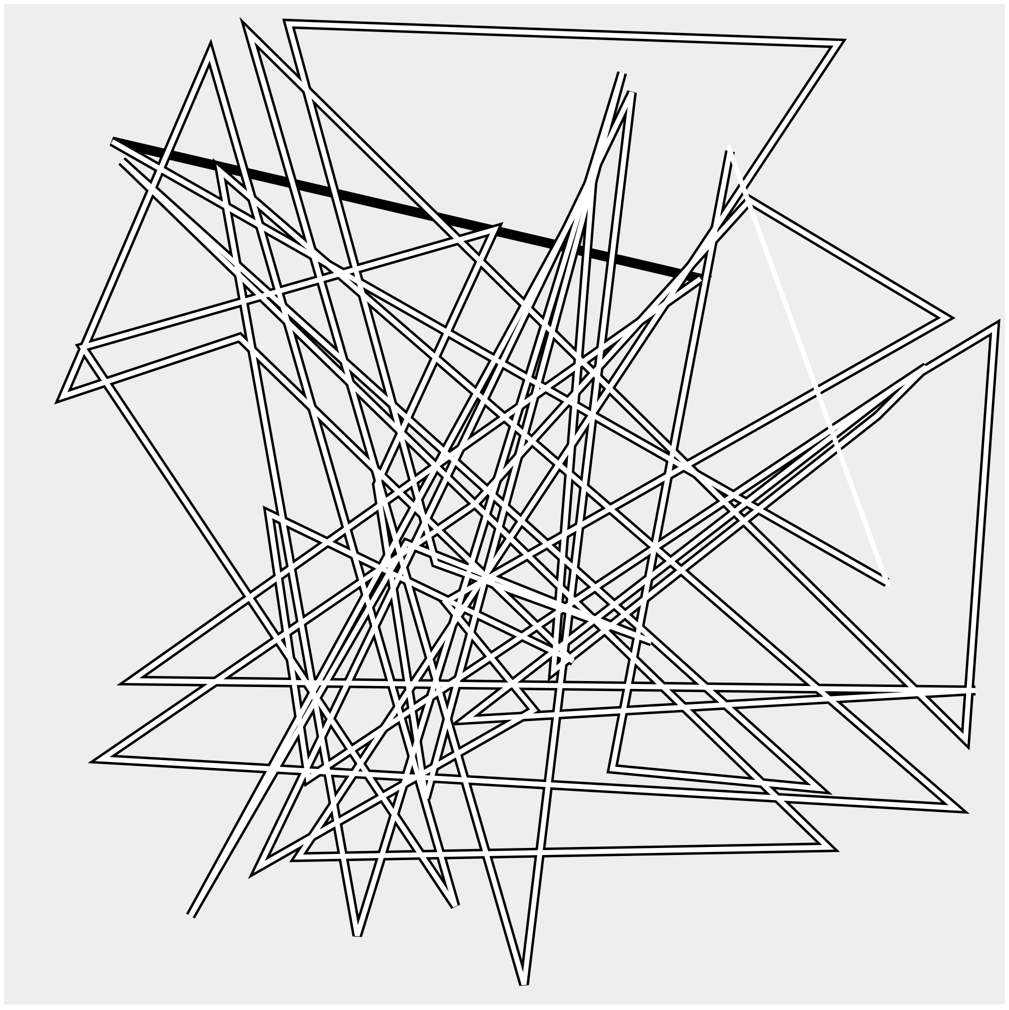
Nejlepsi beh

TSP jsem pustil pres noc na 10'000 generaci a ono jelikoz bezi rychle, tak mi vydalo 60 behu, z toho 7 bylo 158418 km a **zadne lepsi**.
Tady me vsak prekvapil **miniaturni rozdil** (3e-11), zpusobeny float aritmetikou (dukaz totoznosti cest v obrazku nize).
Takze jsem vpodstate nasel pouze lepsi vychozi mesto.

```
// best 1
158418.84201141924
[9, 12, 88, 57, 72, 13, 60, 76, 81, 78, 10, 14, 82, 84, 28, 85, 83, 2, 27, 0,
42, 55, 64, 63, 39, 53, 17, 91, 58, 16, 54, 66, 87, 40, 92, 35, 29, 77, 25, 50,
56, 65, 22, 32, 4, 38, 19, 3, 44, 90, 6, 33, 48, 21, 36, 47, 41, 79, 61, 5, 34,
93, 20, 15, 95, 80, 8, 75, 46, 49, 74, 18, 94, 30, 43, 59, 52, 70, 24, 86, 62,
7, 51, 98, 23, 1, 96, 73, 99, 31, 89, 37, 45, 71, 97, 68, 11, 67, 69, 26]
```

```
// best 2
158418.84201141927
[72, 13, 60, 76, 81, 78, 10, 14, 82, 84, 28, 85, 83, 2, 27, 0, 42, 55, 64, 63,
39, 53, 17, 91, 58, 16, 54, 66, 87, 40, 92, 35, 29, 77, 25, 50, 56, 65, 22, 32,
4, 38, 19, 3, 44, 90, 6, 33, 48, 21, 36, 47, 41, 79, 61, 5, 34, 93, 20, 15, 95,
80, 8, 75, 46, 49, 74, 18, 94, 30, 43, 59, 52, 70, 24, 86, 62, 7, 51, 98, 23,
1, 96, 73, 99, 31, 89, 37, 45, 71, 97, 68, 11, 67, 69, 26, 9, 12, 88, 57]
```

Obe cesty pres sebe (prvni cesta je cerna cara s tloustkou 2px a druha cesta je bila cara s tloustkou 1px):



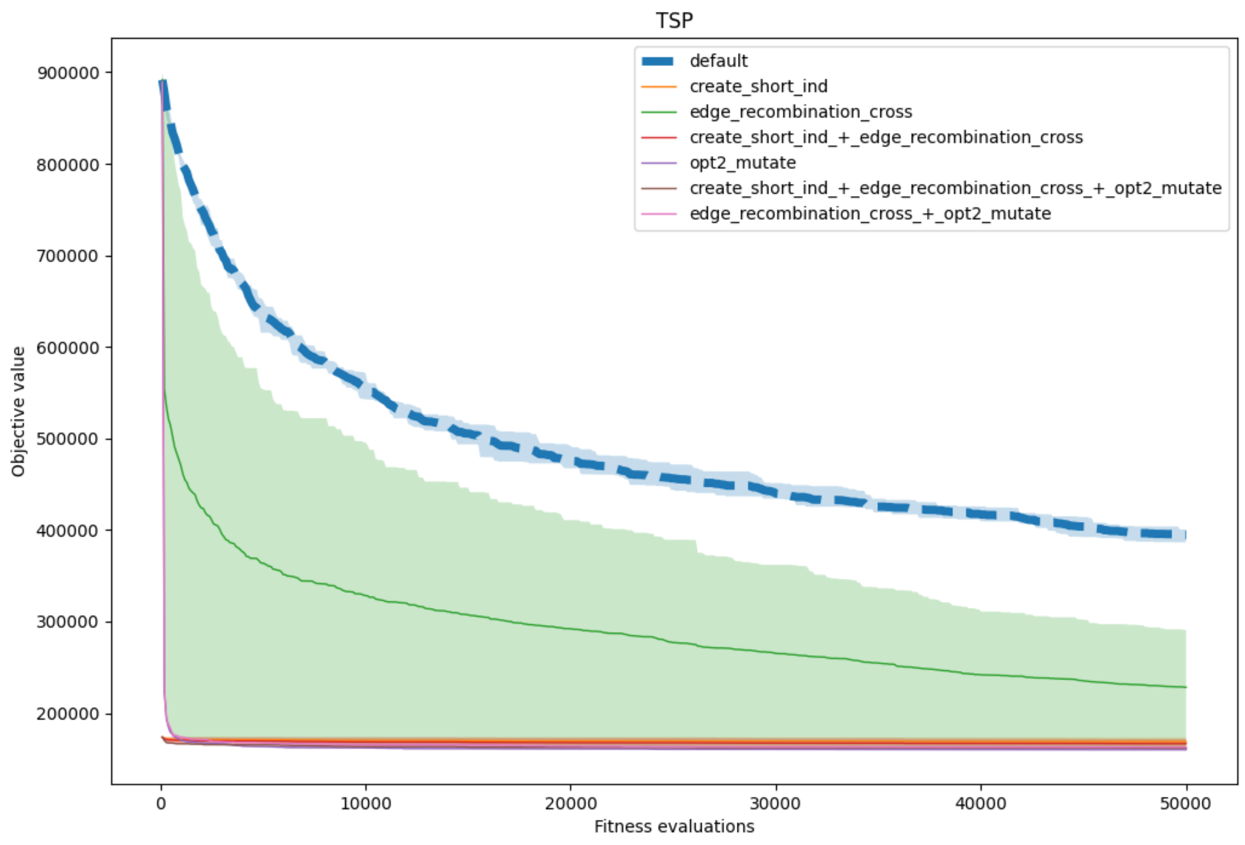
Code

```python
def create_short_ind(ind_len, cities):
    ind = [-1 for _ in range(ind_len)]
    ind[0] = random.randrange(0, ind_len)
    for i in range(1, ind_len):
        shortestIndex = -1
        shortestValue = 1e6
        for j in range(ind_len):
            if j not in ind:
                dist = distance(cities[ind[i-1]], cities[j])
                if dist < shortestValue:
                    shortestIndex = j
                    shortestValue = dist
        ind[i] = shortestIndex
    return ind
```

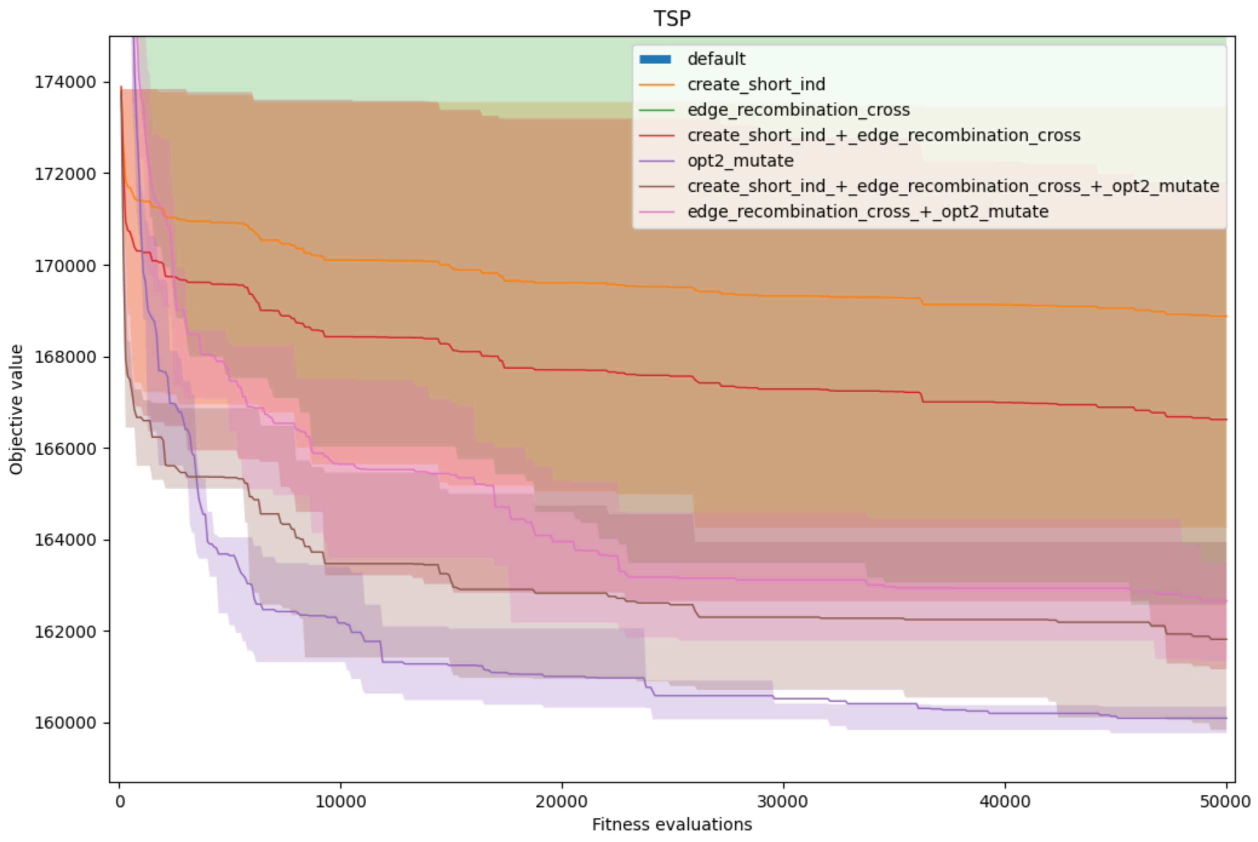
```python
def edge_recombination_cross(p1, p2):
    neighbor_list = {}
    for i, _ in enumerate(p1):
        neighbor_list[p1[i]] = set()
        neighbor_list[p2[i]] = set()
    for i, _ in enumerate(p1):
        neighbor_list[p1[i]].add(p1[(i + 1) % len(p1)])
        neighbor_list[p1[i]].add(p1[(i - 1) % len(p1)])
        neighbor_list[p2[i]].add(p2[(i + 1) % len(p2)])
        neighbor_list[p2[i]].add(p2[(i - 1) % len(p2)])
    child = []
    x = random.choice([p1[0], p2[0]])
    child.append(x)
    for node in neighbor_list:
        neighbor_list[node].discard(x)
    while len(child) < len(p1):
        if not neighbor_list[x]:
            z = random.choice([node for node in neighbor_list if node not in child])
        else:
            min_neighbors = min(len(neighbor_list[node]) for node in neighbor_list[x])
            neighbors_with_min = [node for node in neighbor_list[x] if len(neighbor_list[node])
            z = random.choice(neighbors_with_min)
        child.append(z)
        for node in neighbor_list:
            neighbor_list[node].discard(z)
        x = z
    return [child, child]
```

```python
def opt2_mutate(p, max_len, cities):
    o = p[:]
    found_improvement = False
    n = len(p)
    for i in range(n - 1):
        for j in range(i + 1, n):
            length_delta = -distance(cities[o[i]], cities[o[(i + 1) % n]])
            length_delta -= distance(cities[o[j]], cities[o[(j + 1) % n]])
            length_delta += distance(cities[o[i]], cities[o[j]])
            length_delta += distance(cities[o[(i + 1) % n]], cities[o[(j + 1) % n]])
            if length_delta < 0:
                o[i+1:j+1] = reversed(o[i+1:j+1])
                found_improvement = True
    if found_improvement:
        return o
    else:
        return swap_mutate(p, max_len, cities)
```

Grafy



Priblizeni na y-ove ose, tak aby bylo videt i lepsi behy:


TSP
default
create_short_ind
edge_recombination_cross
create_short_ind_+_edge_recombination_cross
opt2_mutate
create_short_ind_+_edge_recombination_cross_+_opt2_mutate
edge_recombination_cross_+_opt2_mutate
Objective value
Fitness evaluations