Optimalizácia mravčou kolóniou

Daniel Kuchta

Zadanie domácej úlohy je vyriešiť Capacited Vehicle Routing Problem pomocou algoritmu Ant Colony Optimization s jedným depom, z ktorého vozidlá doručujú zásielky zákazníkom. Cieľom je teda nájsť čo najkratšiu trasu s čo najmenším počtom potrebných vozidiel. Problém obsahuje celkovo 3 rôzne datasety, ktoré obsahujú súradnice depa, miest, requestov, samotné požiadavky a kapacitu vozidla.

Riešenie problému

Ideou riešenia je použiť ACO s ďalšími podmienkami tak, aby to riešilo CVHR problém - teda rešpektovalo kapacity vozidiel a vždy keď je to nutné, pridalo ďalšie vozidlo.

Algoritmus ostal prakticky nezmenený, resp. pridal som logiku na handlovanie pridávania vozidiel podľa potreby a výber miest podľa zvyšnej kapacity vozidla. Počet mravcov je rovný počtu požiadaviek, pričom každý mravec začína v jednom z miest a do svojej trasy počas behu priberie depo. Patrične som upravil fitness funkciu aby zohľadňovala aj penalizáciu za každé vozdilo nad minimálny možný počet.

Kód z cvičenia som použil ako backbone, ale upravoval som ho na viacerých miestach. Zmeny som robil v plotovaní výsledných trás, kde som pridal vykresľovanie trás jednotlivých vozidiel každú inou farbou pre lepšiu vizualizáciu, ďalej vo fitness function, kde som pridal 20% penalizáciu za každé vozidlo navyše oproti minimálnemu možnému počtu vozidiel a pridanie cesty do depa za každé vozidlo z jeho posledného bodu trasy a tiež v generovaní riešení, kde som pridal logiku pridávania vozidiel a ošetrenie výberu ďalšieho zákazníka s ohľadom na zvyšnú kapacitu vozidla. Nejaké drobné zmeny som ešte robil prakticky v celom kóde, ale tie nebudem bližšie popisovať.

Nastavenie a ladenie parametrov som vzhľadom na dĺžku behu algoritmu robit na 2. datasete, vyžadujúcom najmenej 5 vozidiel. Nasleduje tabuľka prehľadu fitness vzhľadom na rôzne kombinácie parametrov pri 1000 iteráciách:

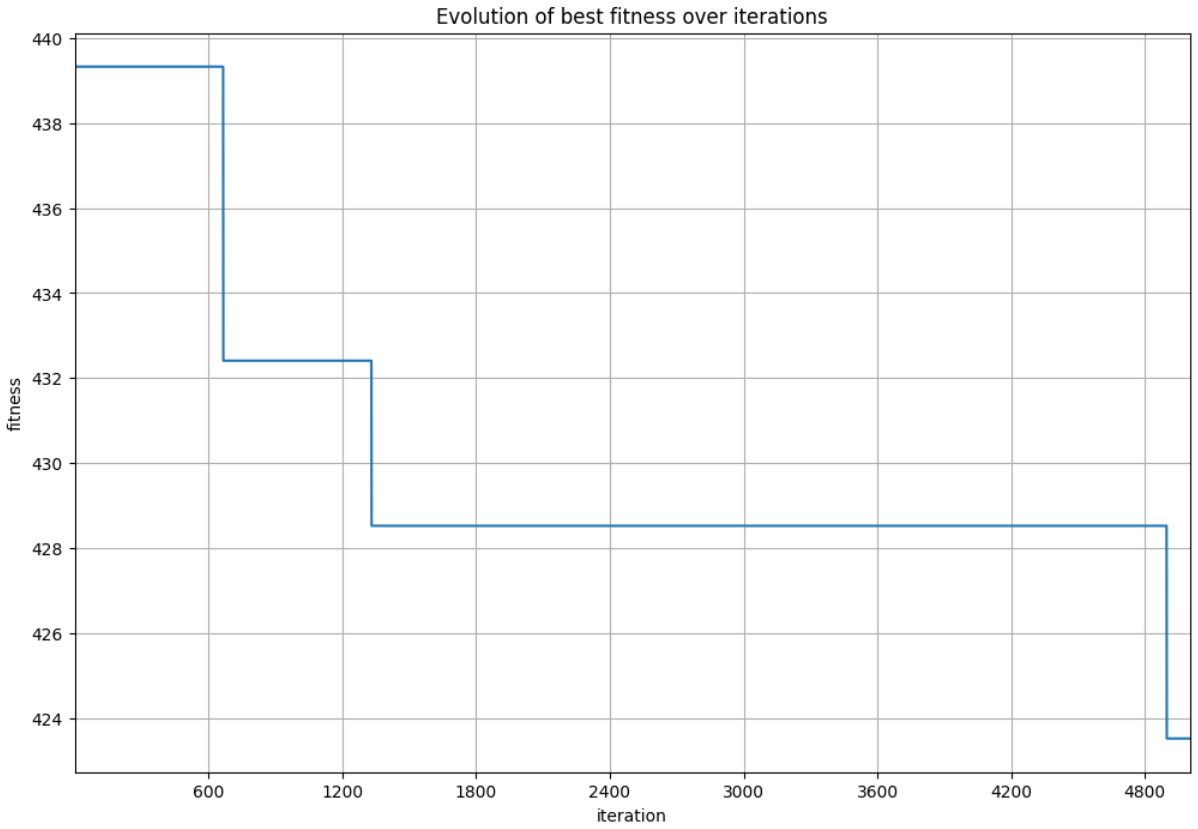
Q	rho	alpha	beta	fitness	distance	vehicles used	min required vehicles	time
50	0.6	1	1	295.202	295.202	5	5	34.9
100	0.6	1	1	302.255	302.255	5	5	35.2
200	0.6	1	1	296.627	296.627	5	5	35.2
50	0.7	1	1	293.747	293.747	5	5	35.4
100	0.7	1	1	304.880	304.880	5	5	35.3
200	0.7	1	1	278.444	278.444	5	5	35.5
50	0.8	1	1	293.044	293.044	5	5	35.5
100	0.8	11	1	295.093	295.093	5	5	35.3
200	0.8	1	1	297.323	297.323	5	5	35.8
50	0.6	1	2	269.458	269.458	5	5	35.3
100	0.6	1	2	282.214	282.214	5	5	35.8
200	0.6	1	2	282.460	282.460	5	5	35.8
50	0.7	1	2	265.912	265.912	5	5	35.3
100	0.7	1	2	278.611	278.611	5	5	35.3
200	0.7	1	2	272.774	272.774	5	5	35.2
50	0.8	1	2	270.359	270.359	5	5	35.3
100	0.8	1	2	273.410	273.410	5	5	35.4
200	0.8	1	2	282.804	282.804	5	5	35.5
50	0.6	1	3	263.126	263.126	5	5	35.5
100	0.6	1	3	267.940	267.940	5	5	35.7
200	0.6	1	3	269.764	269.764	5	5	35.5
50	0.7	1	3	269.636	269.636	5	5	36.1
100	0.7	1	3	268.208	268.208	5	5	35.6
200	**0.7**	**1**	**3**	**259.983**	**259.983**	**5**	**5**	**35.6**
50	0.8	1	3	274.099	274.099	5	5	36.1
100	0.8	1	3	268.047	268.047	5	5	35.8
200	0.8	1	3	265.470	265.470	5	5	35.2

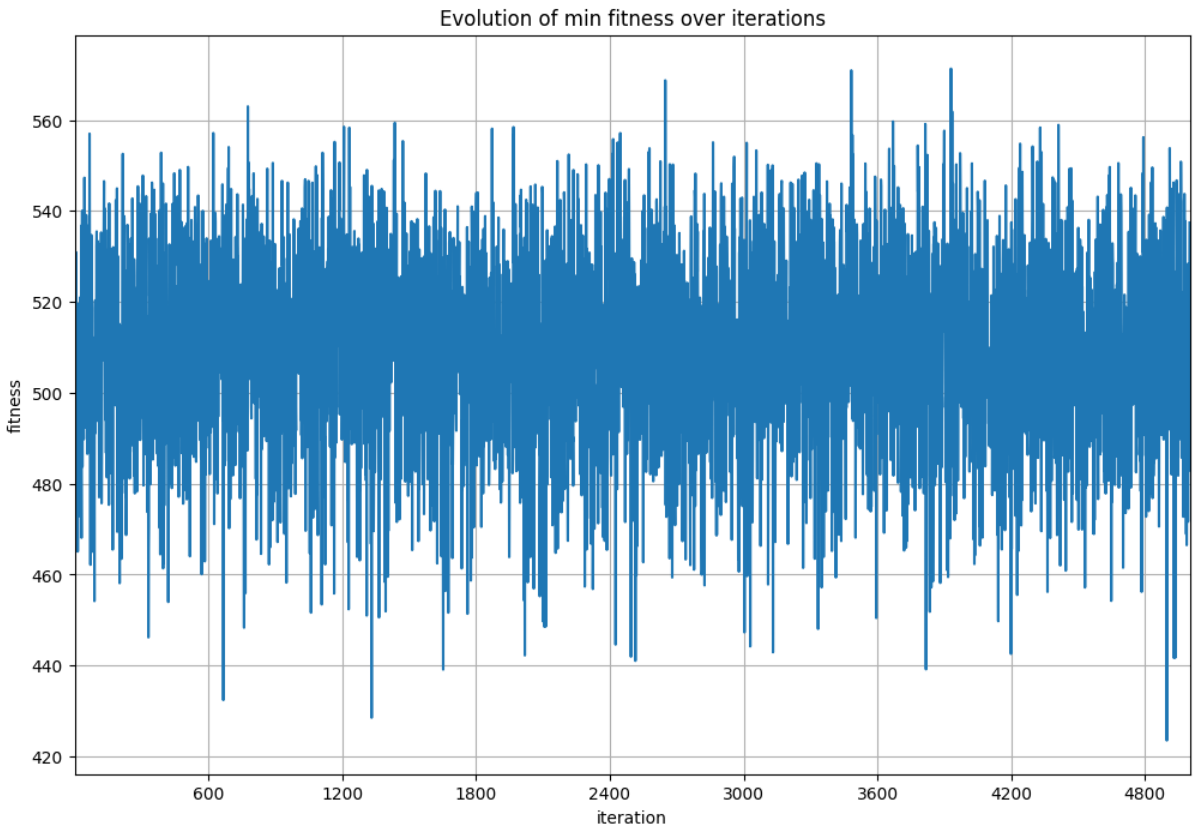
Výsledky

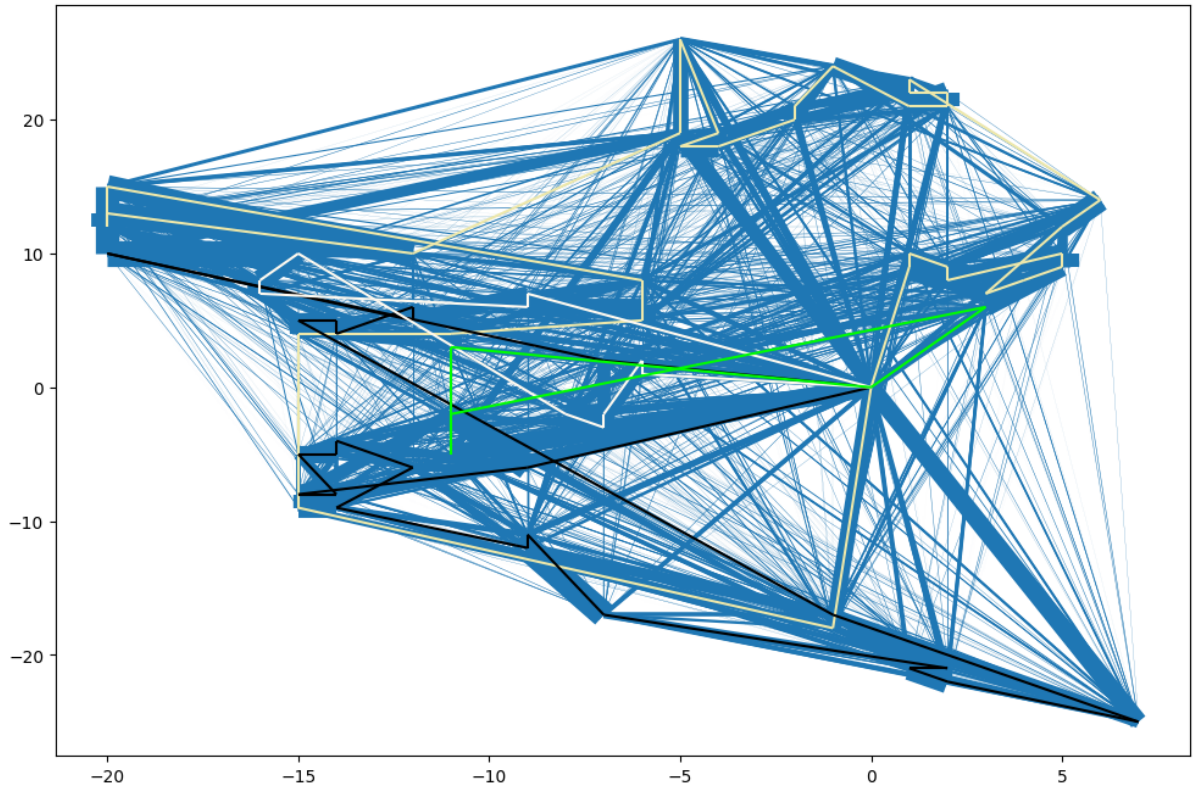
Dataset 1

Fitness: 423.51818335480164
Distance: 423.51818335480164
Vehicle count: 4
Počet iterácií: 5000
Minimum required vehicles: 4
Time: 1449.0
Routes:

[25, 26, 27, 24, 23, 22, 31, 30, 29, 28, 54, 47, 44, 43, 45, 46, 49, 48, 50, 52, 51, 53, 71, 40, 39, 38, 41, 42, 56, 55, 62, 65, 17, 4, 1]
[19, 18, 14, 3, 16, 20, 2, 13, 7, 72, 11, 5, 9, 8, 10, 6, 66, 64, 63, 60, 59, 70, 35, 1]
[32, 33, 37, 36, 34, 69, 68, 67, 57, 58, 1]
[21, 15, 12, 61, 1]



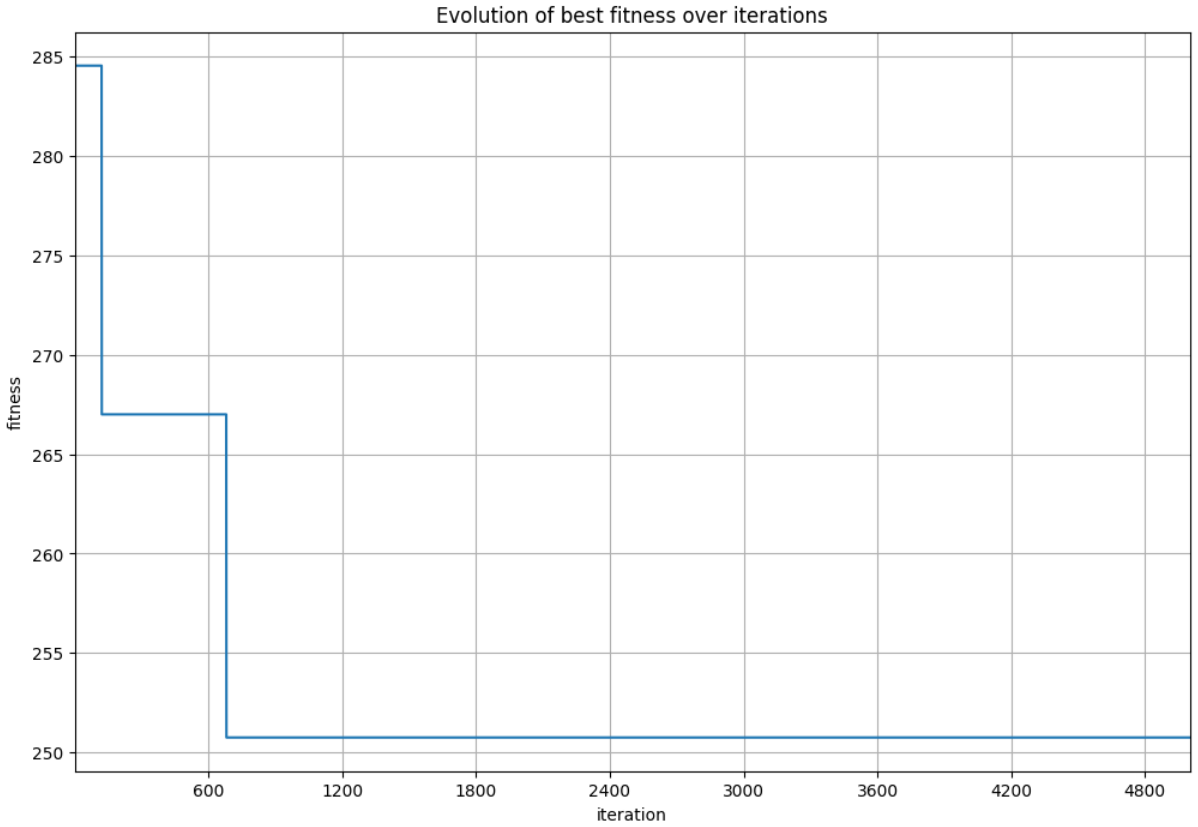

Evolution of min fitness over iterations
fitness
iteration
560
540
520
500
480
460
440
420
600
1200
1800
2400
3000
3600
4200
4800

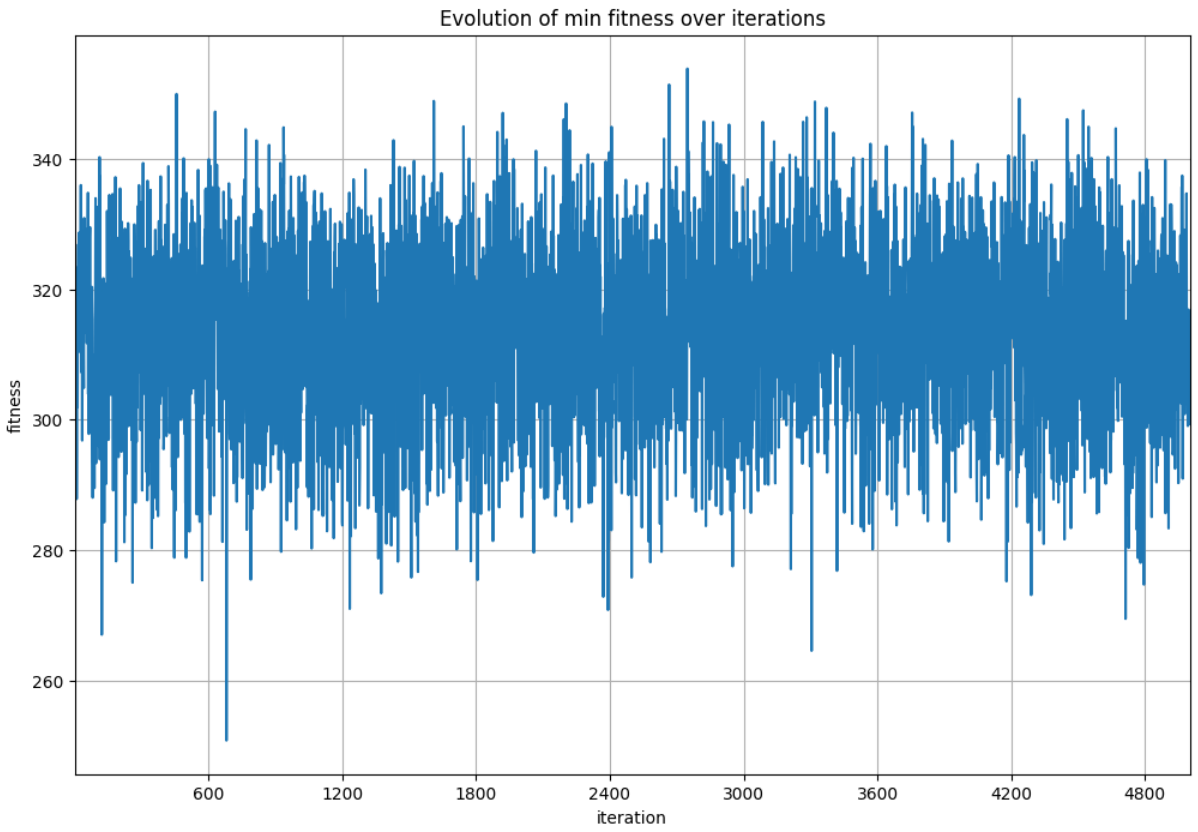

20
10
0
−10
−20
−20
−15
−10
−5
0
5

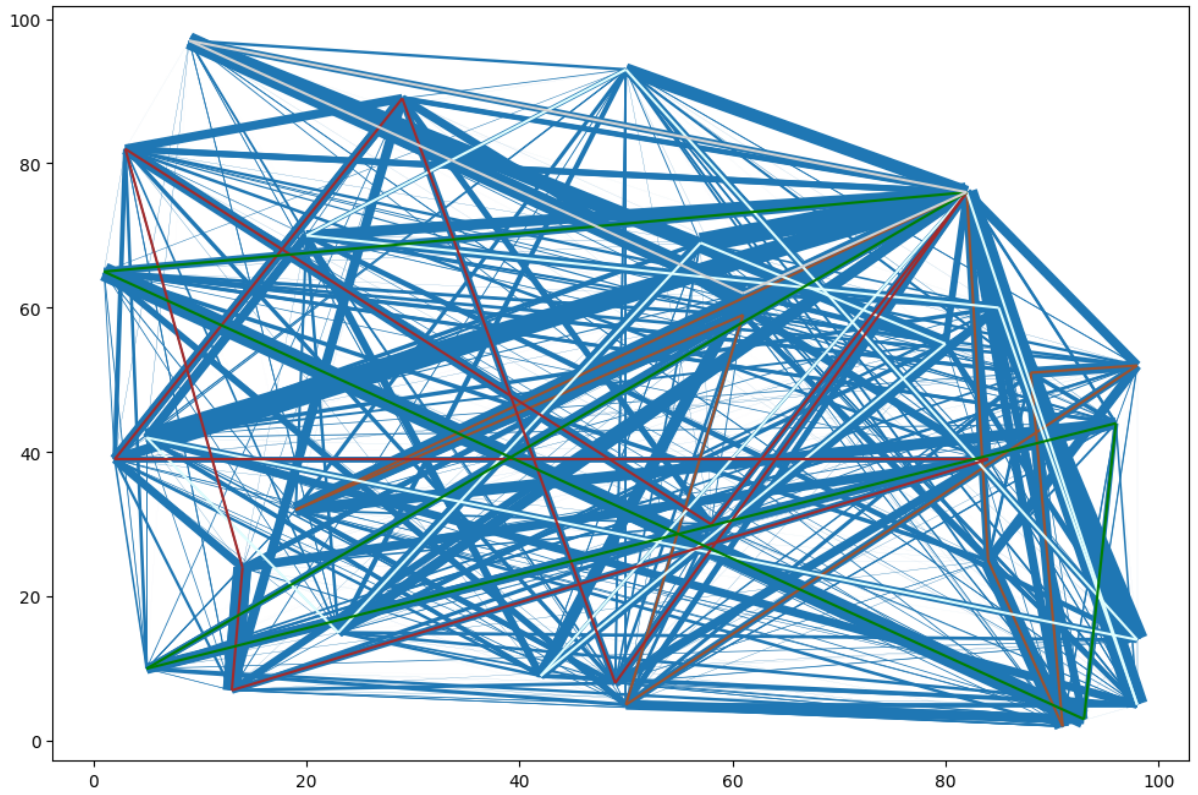
Dataset 2

Fitness: 250.75096065124785
Distance: 250.75096065124785
Vehicle count: 5
Počet iterácií: 5000
Minimum required vehicles: 5
Time: 180.5
Routes:

[14, 18, 17, 13, 3, 15, 19, 1]
[32, 31, 30, 21, 22, 23, 29, 28, 27, 24, 1]
[7, 11, 9, 5, 8, 10, 6, 4, 1]
[12, 2, 20, 16, 1]
[25, 26, 1]




Evolution of min fitness over iterations
fitness
iteration
340
320
300
280
260
600
1200
1800
2400
3000
3600
4200
4800

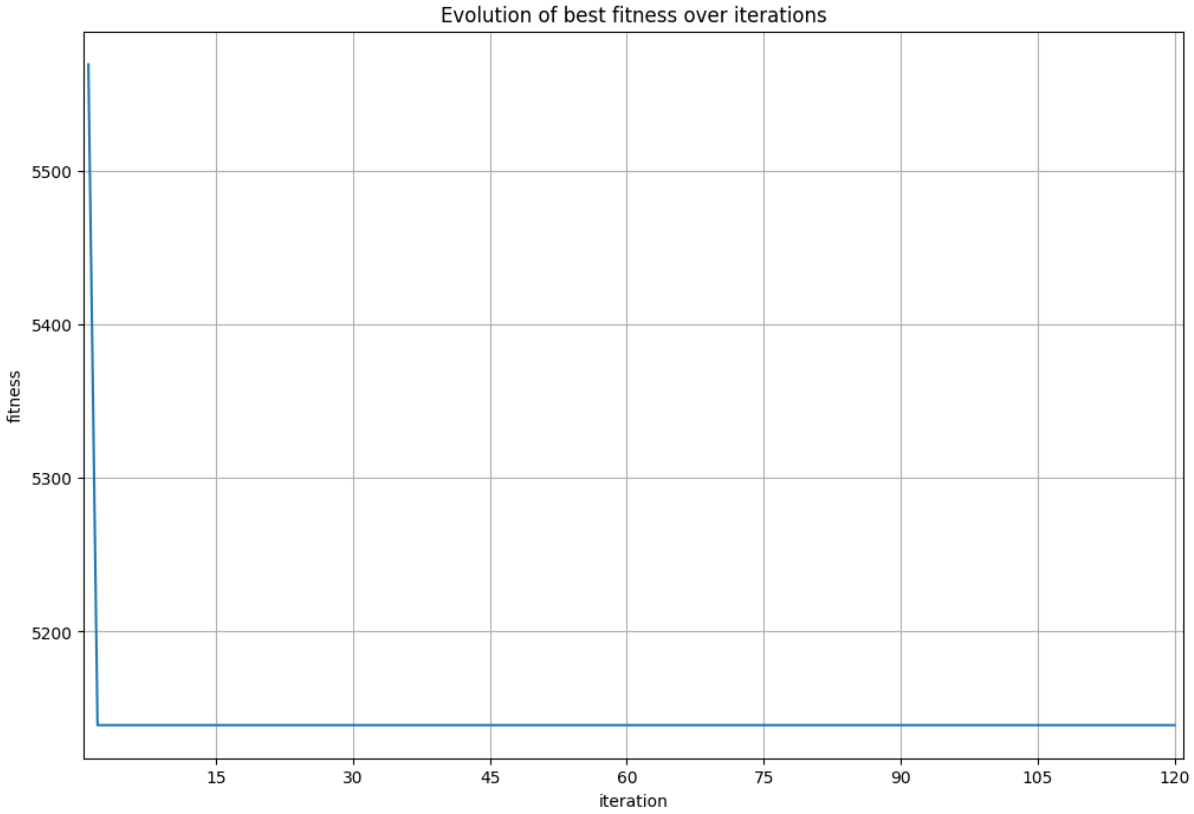

100
80
60
40
20
0
0
20
40
60
80
100

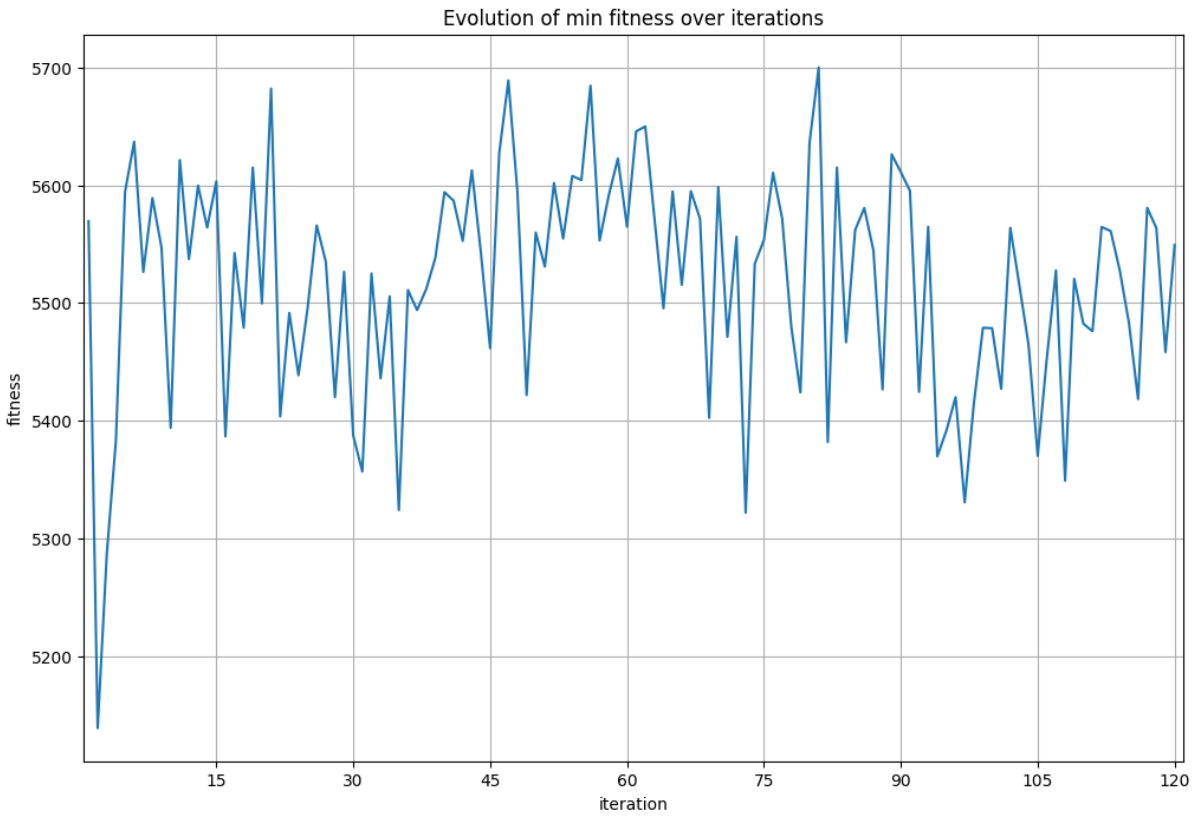
Dataset 3

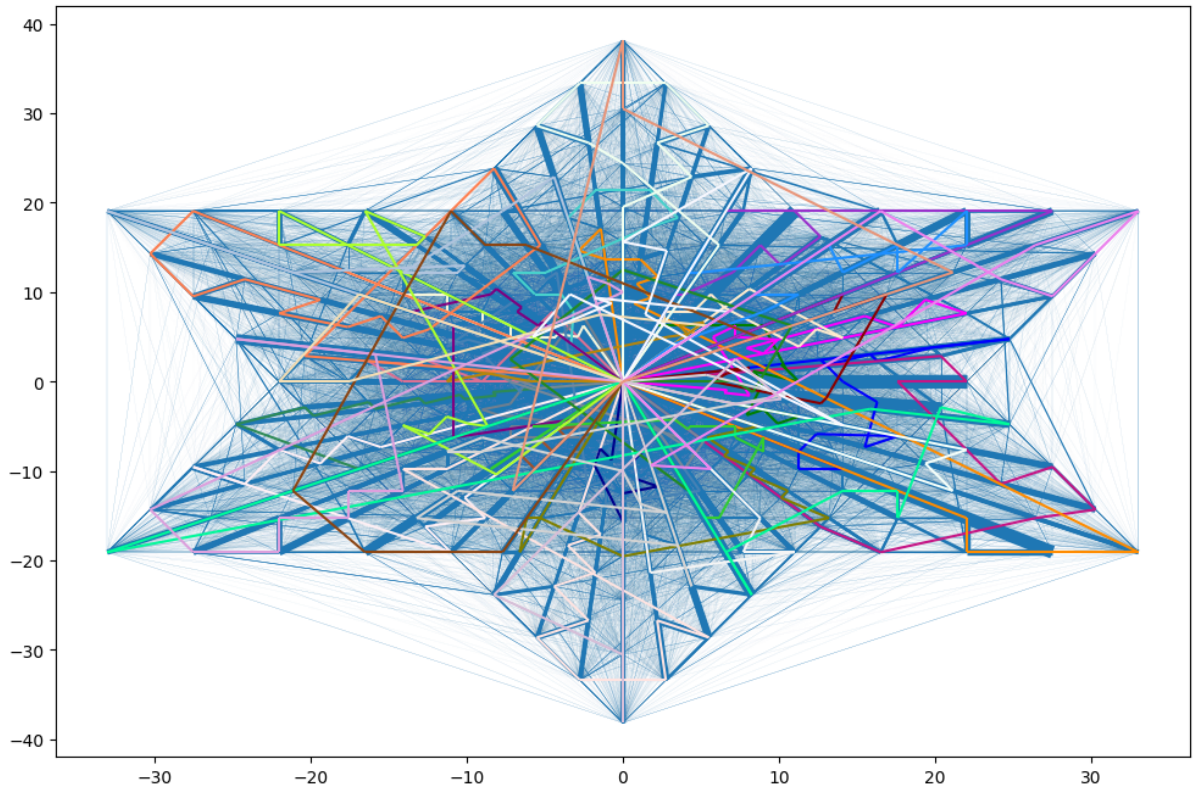
Fitness: 5139.066252048158
Distance: 3670.761608605827
Vehicle count: 40
Minimum required vehicles: 38
Počet iterácií: 120
Time: 5768.6
Routes:

[244, 330, 270, 219, 124, 123, 63, 421]
[378, 318, 313, 258, 253, 198, 241, 354, 197, 256, 137, 180, 239, 240, 421]
[157, 102, 162, 217, 222, 277, 282, 337, 342, 397, 402, 190, 130, 421]
[373, 361, 301, 317, 376, 377, 312, 418, 413, 421]
[136, 77, 76, 17, 121, 133, 181, 138, 193, 85, 26, 421]
[54, 114, 15, 75, 24, 84, 25, 80, 79, 20, 200, 259, 260, 319, 421]
[360, 419, 420, 372, 28, 87, 88, 147, 148, 153, 212, 213, 421]
[230, 285, 225, 170, 165, 110, 105, 50, 108, 49, 45, 44, 103, 43, 91, 31, 421]
[224, 283, 284, 343, 344, 403, 365, 398, 305, 338, 421]
[22, 82, 3, 89, 34, 36, 95, 155, 96, 156, 215, 216, 276, 421]
[159, 99, 39, 90, 30, 97, 42, 100, 41, 101, 160, 211, 171, 112, 172, 286, 421]
[37, 150, 64, 66, 126, 46, 106, 166, 226, 173, 421]
[231, 232, 291, 292, 351, 352, 411, 412, 417, 357, 416, 229, 228, 421]
[21, 81, 86, 141, 122, 182, 194, 139, 140, 199, 254, 421]
[5, 98, 65, 125, 158, 218, 185, 421]
[12, 118, 178, 233, 238, 293, 298, 353, 358, 191, 290, 345, 350, 421]
[18, 78, 73, 1, 195, 294, 187, 127, 205, 145, 421]
[4, 210, 184, 2, 74, 62, 134, 421]
[146, 201, 242, 302, 314, 374, 362, 381, 326, 385, 421]
[52, 51, 111, 131, 71, 176, 117, 57, 56, 421]
[272, 273, 268, 267, 208, 262, 243, 183, 202, 142, 29, 421]
[38, 70, 69, 129, 214, 269, 209, 154, 94, 149, 421]
[227, 167, 107, 47, 48, 53, 113, 58, 59, 60, 119, 179, 120, 346, 295, 421]
[396, 401, 400, 341, 281, 340, 280, 221, 220, 161, 271, 331, 391, 339, 390, 421]
[6, 246, 306, 406, 366, 297, 237, 421]
[55, 115, 175, 235, 40, 68, 67, 174, 61, 421]
[395, 336, 335, 334, 389, 394, 10, 104, 163, 164, 223, 278, 421]
[72, 132, 192, 252, 257, 316, 421]
[369, 309, 249, 329, 274, 275, 382, 332, 421]
[310, 370, 255, 315, 375, 414, 359, 300, 299, 355, 415, 421]
[14, 144, 204, 203, 263, 323, 264, 324, 265, 384, 325, 261, 206, 266, 421]
[321, 248, 333, 392, 393, 388, 387, 328, 327, 188, 421]
[13, 234, 168, 169, 109, 11, 177, 236, 287, 347, 296, 356, 421]
[399, 279, 288, 289, 348, 349, 408, 409, 404, 8, 93, 152, 421]
[7, 322, 303, 363, 250, 407, 405, 421]
[9, 189, 364, 304, 32, 83, 421]
[23, 143, 92, 33, 35, 151, 196, 27, 207, 128, 386, 421]

[186, 116, 307, 367, 380, 379, 320, 383, 421]
[16, 135, 19, 410, 311, 371, 251, 421]
[247, 308, 368, 245, 421]




Evolution of min fitness over iterations
fitness
iteration
5700
5600
5500
5400
5300
5200
15
30
45
60
75
90
105
120


40
30
20
10
0
−10
−20
−30
−40
−30
−20
−10
0
10
20
30

Záver

Ukázalo sa, že ACO algoritmus dokáže pomerne efektívne hľadať near-optimal riešenia Capacited Vehicle Routing problému, čo je prínosné, nakoľko väčšie inštancie tohto problému sú exaktnými algoritmami prakticky neriešiteľné. Zaujímavé bolo tiež skúmať aký vplyv majú na hľadanie najlepšieho riešenia parametre algoritmu.