Ryan Helmlinger

Period 3

5/17/16

Distance: 89101.89074840903



Path:

[1, 3, 6, 11, 2, 7, 5, 4, 8, 10, 15, 12, 13, 9, 14, 20, 29, 31, 38, 42, 49, 62, 65, 71, 108, 104, 113, 134, 132, 125, 143, 147, 151, 140, 142, 137, 131, 119, 112, 107, 90, 89, 70, 88, 59, 57, 52, 48, 37, 33, 30, 32, 35, 61, 64, 82, 97, 111, 114, 117, 130, 139, 155, 175, 166, 149, 135, 103, 99, 96, 78, 87, 122, 106, 94, 101, 102, 128, 129, 141, 138, 146, 153, 154, 174, 194, 183, 173, 145, 148, 152, 177, 198, 201, 202, 211, 217, 218, 223, 219, 225, 232, 238, 255, 259, 287, 290, 286, 283, 299, 311, 288, 260, 273, 304, 276, 226, 241, 264, 291, 310, 323, 341, 355, 344, 359, 373, 406, 405, 383, 380, 363, 347, 377, 376, 409, 389, 356, 333, 308, 320, 307, 275, 258, 254, 250, 216, 206, 136, 116, 144, 164, 158, 172, 185, 193, 197, 188, 214, 222, 228, 204, 210, 221, 249, 278, 285, 293, 297, 302, 318, 309, 279, 282, 303, 315, 327, 319, 358, 361, 388, 393, 403, 415, 467, 495, 468, 461, 440, 452, 464, 482, 524, 532, 547, 566, 516, 531, 540, 553, 565, 546, 545, 549, 593, 608, 594, 595, 667, 634, 652, 692, 712, 723, 734, 724, 694, 683, 693, 679, 661, 668, 635, 636, 648, 670, 656, 639, 597, 582, 583, 561, 550, 525, 521, 491, 484, 506, 510, 529, 538, 552, 573, 592, 599, 606, 609, 629, 628, 605, 570, 555, 563, 574, 620, 598, 587, 626, 647, 649, 662, 689, 680, 684, 676, 675, 653, 641, 603, 569, 596, 568, 554, 528, 496, 474, 469, 471, 456, 466, 427, 382, 398, 407, 397, 396, 385, 362, 349, 366, 367, 354, 350, 399, 391, 394, 404, 444, 425, 445, 428, 429, 420, 441, 417, 430, 433, 431, 437, 487, 502, 505, 477, 457, 434, 450, 435, 458, 475, 478, 497, 486, 507, 530, 533, 534, 541, 604, 610, 616, 624, 619, 615, 631, 640, 643, 646, 645, 644, 627, 625, 638, 654, 651, 655, 658, 666, 672, 674, 660, 669, 657, 673, 677, 681, 686, 690, 682, 701, 695, 688, 678, 685, 696, 704, 705, 709, 708, 720, 719, 718, 710, 713, 726, 725, 722, 716, 711, 706, 699, 698, 700, 703, 702, 707, 714, 717, 727, 732, 733, 729, 728, 730, 731, 721, 715, 697, 691, 687, 663, 659, 664, 671, 665, 650, 642, 637, 607, 577, 581, 571, 562, 586, 588, 564, 548, 511, 503, 488, 492, 489, 514, 498, 499, 508, 493, 485, 479, 453, 459, 447, 446, 451, 448, 439, 424, 411, 390, 392, 438, 436, 442, 422, 416, 410, 418, 408, 368, 364, 340, 332, 336, 330, 301, 306, 312, 294, 292, 331, 343, 365, 352, 381, 395, 401, 387, 372, 374, 353, 338, 337, 322, 325, 326, 348, 375, 384, 419, 454, 465, 470, 462, 426, 402, 412, 443, 476, 480, 494, 500, 490, 504, 526, 536, 523, 537, 535, 557, 542, 556, 560, 578, 600, 601, 611, 579, 584, 575, 580, 589, 617, 612, 621, 590, 567, 572, 602, 630, 613, 585, 576, 618, 622, 632, 633, 623, 614, 591, 558, 559, 551, 517, 512, 539, 543, 544, 527, 518, 509, 513, 519, 522, 520, 515, 501, 481, 483, 473, 463, 472, 460, 455, 449, 421, 379, 357, 314, 271, 248, 242, 240, 229, 257, 247, 256, 246, 263, 270, 281, 289, 313, 317, 296, 295, 269, 262, 237, 253, 245, 266, 265, 261, 252, 236, 234, 268, 272, 280, 305, 335, 346, 345, 371, 400, 378, 370, 369, 360, 386, 414, 423, 432, 413, 351, 342, 334, 324, 321, 316, 298, 329, 339, 328, 300, 284, 274, 277, 267, 244, 239, 243, 251, 235, 233, 230, 203, 189, 207, 199, 186, 200, 150, 159, 176, 156, 160, 157, 161, 163, 168, 179, 167, 178, 224, 231, 227, 220, 205, 126, 120, 121, 124, 123, 208, 190, 180, 191, 195, 212, 215, 192, 184, 171, 170, 196, 187, 169, 165, 162, 105, 109, 110, 118, 133, 181, 213, 209, 182, 127, 115, 91, 86, 81, 80, 79, 85, 84, 93, 77, 47, 46, 36, 45, 58, 68, 67, 76, 75, 74, 73, 72, 63, 54, 44, 50, 53, 56, 66, 100, 95, 83, 98, 92, 69, 60, 55, 51, 43, 39, 40, 34, 41, 28, 23, 24, 26, 17, 19, 21, 25, 22, 18, 27, 16]

My genetics algorithms program for the traveling salesman problem starts with an initial population size of 15. My program partially uncrosses each person of the starting population and each child produced. It partially uncrosses them by going through each city and checking for better swaps, but then the program never returns to previous cities to double check for another improvement. My fitness function is the distance and I scale it for the probability of finding the parents by subtracting each distance by the lowest distance found so far in the program. I combine to find the parents by taking a certain number of cities in the order of the first parent, and then add the remaining needed cities in the order listed in the second parent. The program did mutations by swapping two values in the list and it mutates 12.5% of the time. After 1200 seconds, my genetic algorithms method returns the individual with the best fitness.