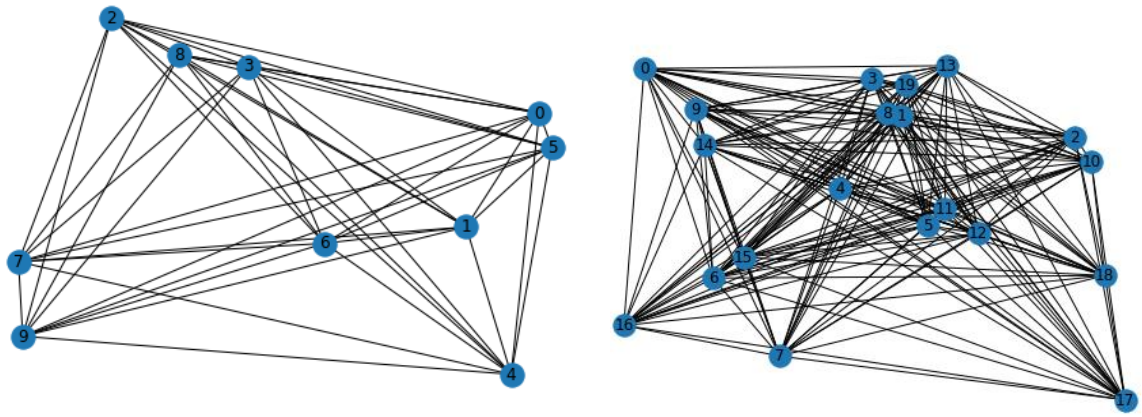
Exercise 1

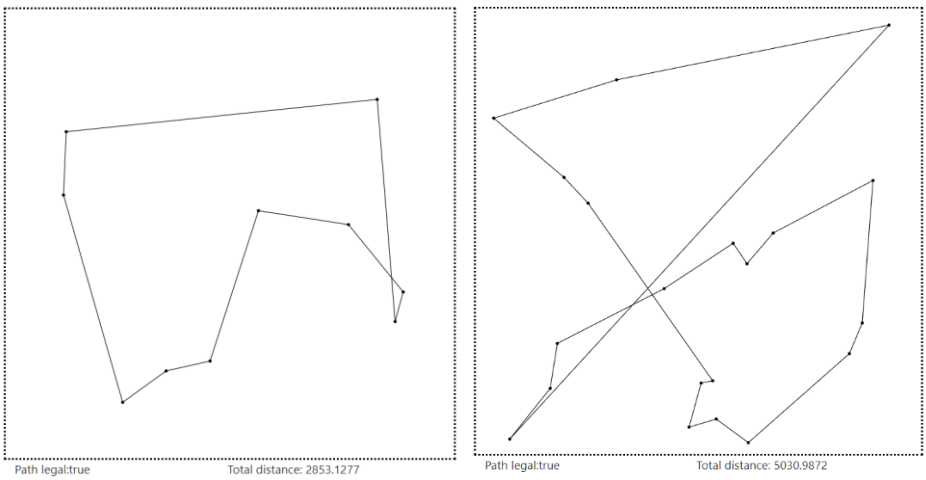
According to the exercise, we had to implement Nearest Neighbour algorithm to the cities provided. It was assumed that, there exists a direct road between any pair of cities. The graph generated for 10 and 20 cities were as follows:



(a) 10 Cities (b) 20 Cities

Figure 1: Graphs for the Cities

According to the implementation of nearest neighbour algorithm, the tour is started from a node and all nodes are traversed exactly once any considering the shortest path to reach any node and finally end the tour by returning to the starting node. The tour path for 10 cities using NN was (0->5->1->6->3->8->2->7->9->4->0) and the path cost was 2853.1277. Similarly, the tour path for 20 cities was (0->9->14->4->5->11->12->18->10->2->13->19->3->8->1->15->6->16->7->17->0) and path cost was 5030.9872. The tours were visualized using the web tool as follows.



(a) 10 Cities (b) 20 Cities

Figure 2: Visualization of Tour for the Cities

Time to execute the tour for 10 and 20 cities were 0.00013937699986854568 and 0.0003111820001322485 seconds respectively.

Similarly for 100 cities the tour path using NN was (0->13->84->12->38->3->75->52->67->55->41->40->76->60->28->16->25->53->56->46->69->6->30->18->22->74->80->62->17->95->99->10->68->45->2->81->32->93->20->91->23->96->44->36->90->35->70->72->79->59->33->61->58->98->77->37->9->34->63->48->71->88->7->97->94->21->26->49->8->29->54->65->87->64->43->4->27->11->31->92->66->39->5->83->82->89->15->51->57->86->24->50->47->19->14->73->1->42->78->85->0) and the cost of the path was 9962.2720. The visualization is as follows:

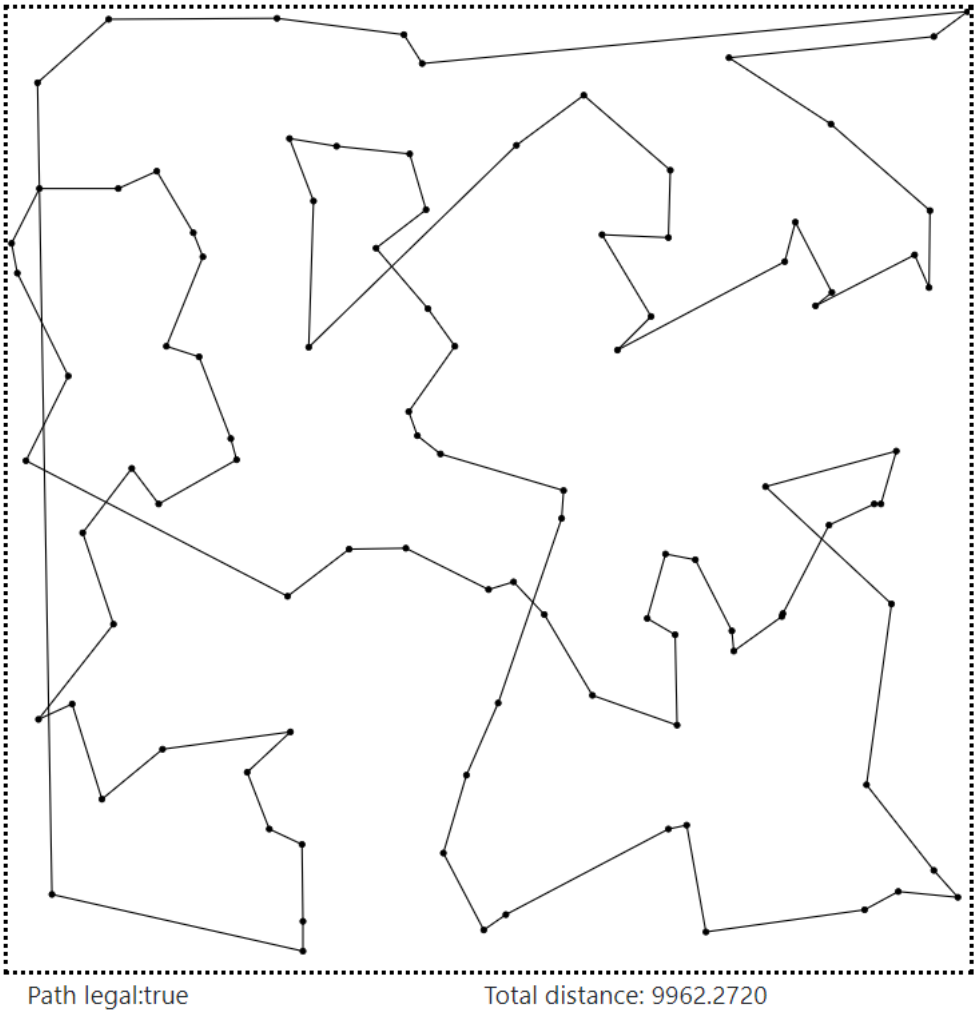


Figure 3: Visualization of Tour for 100 cities

Time to execute the tour for 100 cities was 0.005799821000096017 seconds.

Exercise 2

In this exercise, we were supposed to run the 2-opt local search optimization technique for the cities provided as in exercise1. In 2-opt local search optimization at first a random tour through all the cities is generated and the tour cost is computed. Later this tour path is altered by randomly swapping nodes and checking if the path cost is less than the original tour path cost generated earlier. This alteration is performed for a number of iterations. I have considered 10^5 iterations for this.

For 10 Cities:

The tour path for 10 cities was found to be (0->5->1->4->6->9->7->2->8->3->0) and its path cost was 2638.595865875728 which is less than that using NN (2853.1277). The number of swaps required was 17 and execution time was 1.7012065560002156 seconds. The visualization is as follows:

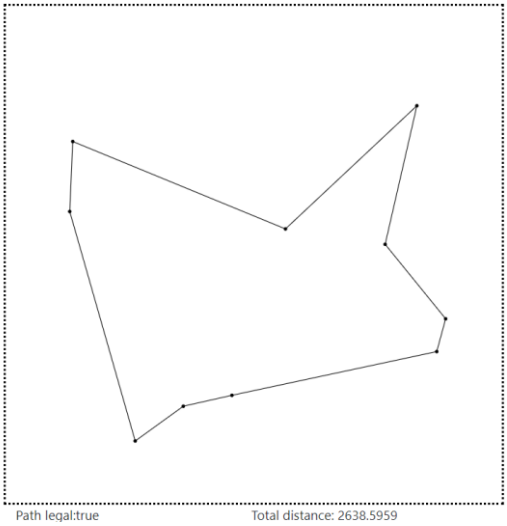


Figure 4: Visualization of 2-opt for 10 cities

For 20 Cities:

The tour path for 20 cities was found to be (0->10->17->18->12->2->5->11->4->16->15->7->6->9->3->13->19->1->8->14->0) and its path cost was 6410.3840 which is greater than that using NN (5030.9872). The number of swaps required was 11 and execution time was 2.9492801570004303 seconds. The visualization is as follows:

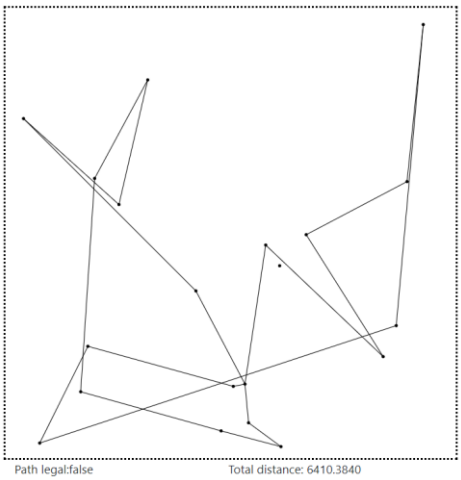


Figure 5: Visualization of 2-opt for 20 cities

For 100 Cities:

The tour path for 100 cities was found to be (0->78->45->80->4->94->18->9->14->66->74->62->3->68->37->11->71->97->55->34->32->2->59->24->22->83->40->92->81->13->53->77->89->38->79->82->16->35->5->39->27->8->21->12->36->50->84->15->17->69->43->58->63->90->73->99->19->51->57->26->86->25->70->96->7->20->46->65->47->61->75->93->48->95->10->42->87->29->91->30->6->1->56->72->23->41->64->98->49->44->67->31->88->85->54->60->76->33->28->52->0) and its path cost was 51730.136173317136 which is much greater than that using NN (9962.2720). The number of swaps required was 13 and execution time was 13.407980826000312 seconds. The visualization is as follows:

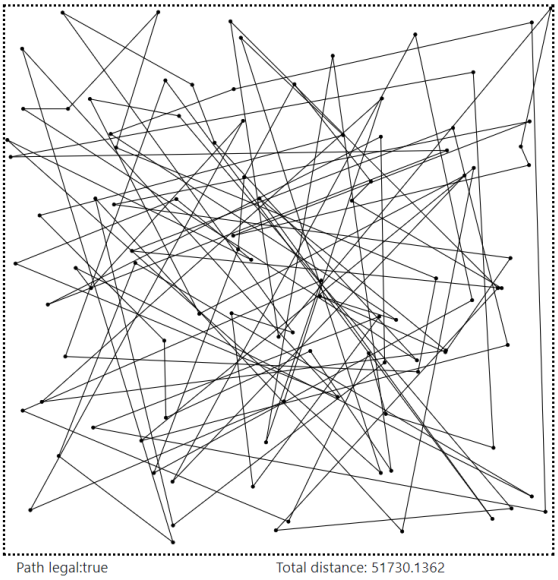


Figure 6: Visualization of 2-opt for 100 cities

For 1000 Cities:

The tour path for 1000 cities was found to be (0->460->128->429->735->841->740->745->21->559->774->407->589->629->664->539->557->458->734->487->172->951->880->290->344->416->239->126->816->275->342->457->245->669->41->328->154->992->957->357->686->688->9->433->869->164->55->863->779->283->277->313->844->477->746->995->111->409->134->873->588->944->848->651->574->138->13->658->479->507->712->60->182->203->719->470->106->152->529->58->830->57->44->243->516->226->582->906->434->683->909->292->359->5->713->107->61->845->956->423->595->174->594->364->864->370->341->381->318->982->432->305->149->282->377->744->798->440->826->610->361->572->767->585->837->378->915->977->715->878->904->375->967->476->764->727->300->139->22->881->4->258->984->505->65->18->603->990->625->7->506->793->920->662->220->541->824->865->188->35->187->445->66->738->543->93->71->833->709->846->593->453->530->757->166->265->204->162->524->795->418->86->406->369->726->807->577->6->191->368->97->2->31->379->346->197->513->941->316->796->532->480->276->175->241->325->934->978->620->296->437->96->820->802->760->421->254->638->26->624->444->500->949->893->544->791->228->264->991->412->303->400->122->994->236->320->257->493->88->85->198->428->806->291->402->685->847->942->839->465->235->955->534->147->598->961->570->969->581->647->965->568->931->985->451->98->714->945->212->350->472->900->621->391->45->829->703->584->732->30->339->144->926->640->660->39->853->124->834->550->986->123->249->866->742->396->446->15->533->818->627->737->947->510->321->728->167->102->165->971->710->964->586->223->91->948->75->37->69->113->207->579->729->156->564->315->327->393->561->828->319->641->278->385->196->811->486->323->822->578->109->281->331->475->888->100->141->656->523->274->653->77->527->137->430->358->467->404->925->556->195->354->988->622->491->20->284->179->119->399->324->632->334->158->84->963->386->308->910->983->565->526->511->531->644->923->643->520->979->797->287->921->591->781->616->655->857->808->521->914->998->535->835-

>519->929->351->962->916->489->731->886->752->14->270->933->103->765->143->716->768->903->552->517->115->932->230->773->163->678->209->665->405->424->118->758->389->708->804->326->725->76->420->233->832->790->415->114->64->668->759->587->221->363->675->634->435->939->294->260->812->240->674->663->483->250->528->116->189->789->891->38->549->989->503->452->855->981->441->912->161->512->794->560->36->253->843->10->234->590->210->930->547->130->700->761->372->892->279->867->575->786->562->936->17->871->353->301->877->676->883->938->463->461->754->132->427->766->142->679->537->280->248->689->999->242->935->455->92->397->973->49->775->273->730->392->601->566->558->704->856->875->8->255->504->266->67->360->237->555->87->382->442->666->636->976->411->823->782->722->33->889->924->699->701->468->897->261->52->670->693->184->388->819->466->329->809->169->478->345->928->858->563->12->617->602->895->447->972->717->177->652->190->449->946->27->762->649->514->352->338->376->783->842->943->695->135->336->232->623->398->639->831->635->580->157->850->692->854->987->170->246->540->95->362->213->975->872->151->74->355->211->185->70->862->438->659->618->600->83->295->682->383->684->569->739->271->471->183->571->426->262->218->849->501->32->785->343->365->792->285->417->780->868->312->642->799->79->723->78->876->645->112->25->456->805->919->155->525->724->425->117->80->954->554->462->859->677->827->269->180->576->24->901->108->131->502->885->661->605->879->251->637->508->671->913->322->150->413->542->748->548->214->615->736->667->317->104->770->800->884->484->628->902->408->968->298->390->690->573->952->496->772->225->743->306->40->801->101->694->776->199->72->619->613->611->333->153->899->492->890->229->439->966->741->815->431->176->960->173->302->205->309->454->422->311->788->63->494->750->958->219->443->522->16->646->604->53->787->68->654->129->474->778->753->387->896->238->307->687->259->148->861->289->706->490->607->340->673->608->252->28->56->771->121->288->186->485->515->206->450->898->905->367->90->546->817->146->553->414->482->711->51->648->272->94->733->159->59->769->749->231->244->73->630->950->29->481->401->721->217->403->495->498->980->384->599->50->89->567->202->168->46->227->201->268->62->612->469->680->181->626->337->332->314->894->304->672->366->631->707->110->755->127->247->536->410->997->718->47->216->705->993->860->348->136->99->592->720->927->473->215->803->297->970->509->838->606->349->54->518->310->633->697->499->263->497->224->917->193->814->922->583->171->356->140->160->609->133->907->810->836->691->545->192->597->882->756->825->448->380->937->293->200->851->887->178->813->940->784->696->911->918->596->419->43->436->373->840->657->19->908->698->681->347->874->870->650->34->1->538->852->3->81->763->23->751->459->821->335->777->488->996->974->222->120->394->194->145->125->371->953->286->374->299->747->614->82->48->464->959->702->11->42->256->330->105->551->395->267->208->0) and its path cost was 512048.4945 which is much greater than that using NN (28164.0098). The number of swaps required was 12 and execution time was 317.81478120799875 seconds. The visualization is as follows:

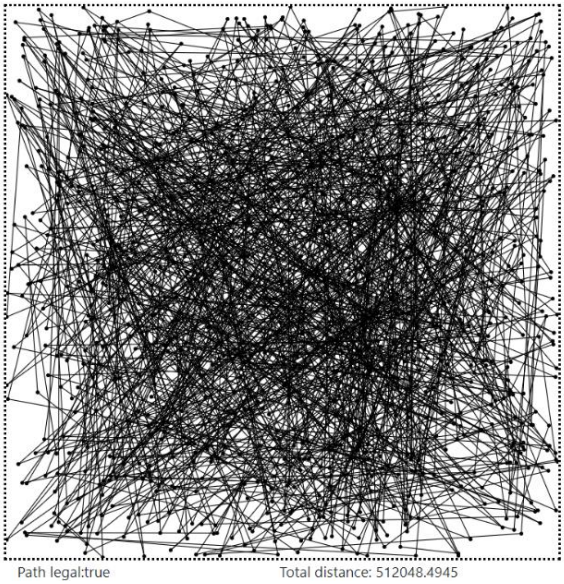


Figure 7: Visualization of 2-opt for 1000 cities

Therefore, it was observed that except for the city size 10, NN gives better results than 2-opt local search which is probably because of the number of iterations. I think if the number of iterations is increased then 2-opt local search could have given better results than that in NN but still it would depend on the random tour generated at first.

Exercise 3

In exercise 3, we use the same implementation of 2-opt local search as like in exercise 2 but this time we start with the tour path generated by the NN algorithm in exercise 1 instead of a random tour path and compare the results.

For 10 Cities:

The tour generated by NN for 10 cities was (0->5->1->6->3->8->2->7->9->4->0) and the path cost was 2853.1277. After applying 2-opt local search starting with this tour there seemed to be no change in the path and path cost for 10^5 iterations although we obtained a better path with less cost in exercise 2 which was (0->5->1->4->6->9->7->2->8->3->0) and its path cost was 2638.595865875728. The total execution time for NN and 2-opt search was 2.513959676001832 seconds.

For 20 Cities:

The tour generated by NN for 20 cities was (0->9->14->4->5->11->12->18->10->2->13->19->3->8->1->15->6->16->7->17->0) and path cost was 5030.9872. After applying 2-opt local search starting with this tour the result seemed to improve after 10^5 iterations and the tour was found to be (0->9->14->8->1->3->19->13->2->10->18->12->11->5->4->15->6->16->7->17->0) which had cost 4880.11731900225 . The total execution time for NN and 2-opt search was 4.113318652998714 seconds.

For 100 Cities:

The tour generated by NN for 20 cities was (0->13->84->12->38->3->75->52->67->55->41->40->76->60->28->16->25->53->56->46->69->6->30->18->22->74->80->62->17->95->99->10->68->45->2->81->32->93->20->91->23->96->44->36->90->35->70->72->79->59->33->61->58->98->77->37->9->34->63->48->71->88->7->97->94->21->26->49->8->29->54->65->87->64->43->4->27->11->31->92->66->39->5->83->82->89->15->51->57->86->24->50->47->19->14->73->1->42->78->85->0) and path cost was 9962.2720. After applying 2-opt local search starting with this tour the result seemed to improve after 10^5 iterations and the tour was found to be (0->13->84->12->38->3->75->52->67->55->41->40->76->60->28->16->25->53->56->46->69->6->30->18->22->74->80->62->17->95->99->10->68->45->2->81->32->93->20->91->23->96->44->36->90->35->70->72->79->59->33->61->58->98->77->37->9->34->63->48->71->88->7->97->94->21->26->49->4->43->64->87->65->54->29->8->27->11->31->92->66->39->5->83->82->89->15->51->57->86->24->50->47->19->14->73->1->42->78->85->0) which had cost 9958.30524492126. The total execution time for NN and 2-opt search was 17.6561636539991 seconds.

Explanation:

In exercise 2, the start tour for 2-opt search was chosen randomly, so it was very likely to produce bad tours which costed more. Although swappings were done and checked for the tour with minimum

cost still the choosing of nodes to be swapped were random. Therefore, it was very likely to generate bad tour results at the end even after 10^5 iterations.

On the other hand, in exercise 1, from each node the shortest path is chosen to reach another node and hence the NN algorithm tends to give better results. However, NN algorithm is a greedy algorithm and does not always give optimal results. In exercise 3, the start tour is the tour generated from NN algorithm which already tends to optimal tour. Hence making swappings on this tour for 10^5 iterations generates betters results than NN algorithm. Finally, it can be said that combination of NN and 2-opt local search gives better results as we have seen in exercise 3.

Exercise 4

I have used simulated annealing for the optimization. Simulated annealing is a probabilistic technique to for approximating the global optimum of a given function [1]. The search life span of a simulated annealing is based on its temperature. Initially it is "hot" and thus accepts degrading solutions. Gradually the temperature decreases and its probability to accept bad solutions reduces. In my code I have used the satsp library to solve the graph of cities for the TSP problem.

It is same like as 2-opt optimization except for the fact that a temperature is set. The 2-opt optimization is performed for a number of times until a stopping condition is met.

For 10 cities:

For 10 cities I found the path to be [0, 3, 8, 2, 7, 9, 6, 4, 1, 5] which has a cost of 2638.595865875754 which is the lowest as we found in ex2. The time required was 1.7088990689999264 seconds.

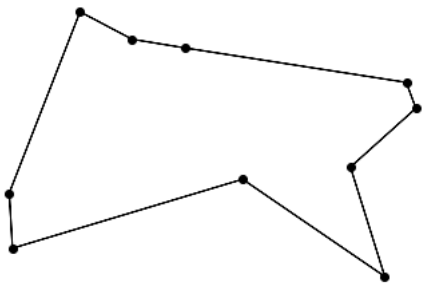


Figure 8: Visualization of Simulated Annealing for 10 cities

For 20 cities:

For 20 cities I found the path to [0, 3, 8, 1, 19, 13, 2, 10, 11, 5, 12, 18, 17, 7, 16, 6, 15, 4, 14, 9] which has a cost of 4103.718791905329 which is the lower than the cost we found in ex1, ex2 and ex3. The time required was 5.523534074000054 seconds.

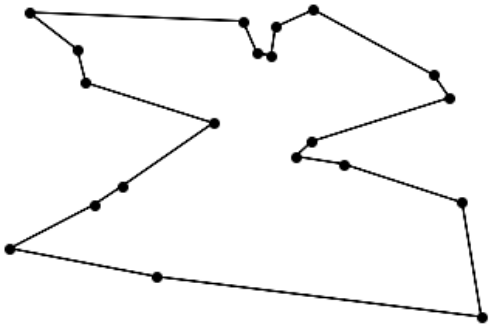


Figure 9: Visualization of Simulated Annealing for 20 cities

For 100 cities:

For 100 cities I found the path [0, 13, 84, 12, 3, 38, 75, 52, 85, 55, 67, 41, 62, 17, 95, 94, 21, 26, 49, 8, 29, 54, 87, 64, 43, 4, 16, 28, 60, 76, 40, 80, 74, 53, 25, 56, 46, 69, 6, 22, 18, 30, 78, 42, 1, 73, 14, 65, 27, 11, 66, 5, 39, 92, 31, 50, 24, 47, 19, 86, 51, 57, 82, 83, 89, 15, 70, 35, 90, 36, 72, 20, 93, 97, 7, 10, 99, 68, 32, 81, 23, 91, 96, 44, 79, 59, 33, 61, 58, 98, 77, 9, 37, 2, 45, 88, 71, 48, 34, 63] which has a cost of 8325.655850800602 which is the lower than the cost we found in ex1,ex2 and ex3. The time required was 19.332885238000017 seconds.

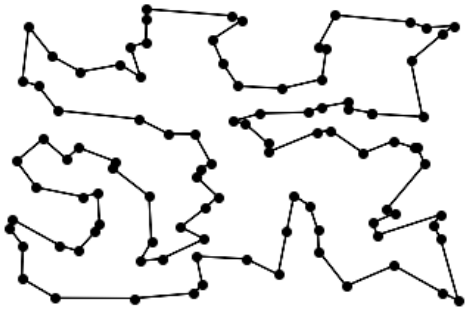


Figure 10: Visualization of Simulated Annealing for 100 cities

For 1000 cities:

For 1000 cities I found the path [0, 226, 447, 929, 351, 514, 890, 421, 502, 872, 993, 658, 113, 748, 247, 56, 191, 805, 162, 818, 65, 240, 512, 823, 666, 376, 983, 698, 806, 204, 152, 998, 394, 856, 534, 154, 554, 461, 580, 896, 159, 574, 333, 145, 960, 301, 389, 577, 344, 38, 220, 6, 92, 484, 525, 693, 846, 68, 505, 681, 617, 272, 865, 457, 455, 88, 174, 227, 873, 5, 862, 231, 605, 69, 342, 257, 281, 877, 358, 125, 968, 124, 956, 654, 689, 604, 481, 309, 528, 228, 136, 690, 224, 750, 258, 925, 463, 289, 764, 16, 871, 967, 12, 324, 506, 122, 382, 503, 146, 550, 521, 660, 759, 870, 211, 634, 221, 435, 989, 523, 132, 64, 388, 729, 428, 966, 37, 782, 10, 905, 84, 613, 284, 94, 239, 860, 76, 103, 755, 821, 879, 742, 902, 887, 415, 711, 399, 762, 594, 137, 717, 544, 919, 756, 119, 721, 692, 148, 186, 201, 572, 932, 253, 488, 422, 116, 295, 237, 746, 274, 223, 683, 630, 529, 963, 305, 811, 153, 718, 595, 141, 216, 252, 596, 996, 941, 341, 824, 918, 927, 45, 208, 844, 326, 135, 934, 840, 172, 646, 610, 910, 140, 908, 579, 109, 763, 766, 527, 921, 541, 564, 808, 401, 78, 24, 498, 473, 357, 282, 761, 242, 785, 402, 197, 767, 150, 866, 70, 426, 378, 699, 283, 995, 219, 625, 709, 101, 982, 332, 439, 903, 90, 464, 981, 749, 834, 741, 900, 275, 349, 497, 259, 27, 962, 841, 961, 778, 849, 885, 753, 803, 408, 43, 108, 901, 187, 39, 789, 657, 928, 752, 730, 815, 722, 954, 876, 912, 472, 705, 82, 682, 243, 719, 622, 783, 139, 93, 571, 381, 459, 123, 987, 303, 196, 713, 943, 618, 312, 878, 570, 545, 482, 559, 626, 338, 427, 615, 532, 744, 7, 384, 671, 623, 588, 179, 11, 624, 677, 791, 569, 799, 697, 325, 838, 334, 621, 480, 883, 672, 662, 13, 470, 813, 434, 77, 359, 951, 627, 655, 339, 499, 30, 486, 483, 310, 138, 555, 316, 578, 533, 620, 404, 429, 790, 957, 704, 990, 812, 546, 894, 869, 710, 331, 772, 42, 398, 603, 601, 269, 882, 407, 952, 999, 370, 774, 880, 443, 775, 647, 593, 41, 958, 362, 73, 120, 199, 509, 734, 800, 833, 114, 297, 267, 379, 848, 71, 780,

636, 406, 198, 906, 495, 85, 649, 396, 745, 673, 675, 635, 343, 712, 118, 656, 260, 977, 924, 158, 936, 144, 183, 708, 205, 315, 832, 920, 440, 629, 327, 392, 474, 134, 51, 300, 727, 568, 265, 678, 234, 203, 465, 549, 386, 380, 616, 453, 270, 185, 363, 117, 664, 736, 520, 984, 133, 264, 565, 760, 336, 797, 738, 200, 33, 888, 436, 189, 771, 279, 997, 192, 937, 688, 262, 551, 235, 737, 543, 355, 724, 539, 31, 232, 467, 633, 504, 449, 127, 323, 632, 393, 410, 369, 466, 597, 418, 313, 2, 100, 599, 715, 368, 831, 786, 684, 126, 854, 926, 490, 909, 290, 413, 974, 317, 142, 293, 20, 63, 49, 567, 592, 650, 702, 641, 128, 516, 478, 542, 245, 893, 676, 695, 784, 777, 202, 165, 830, 639, 450, 229, 166, 922, 881, 176, 437, 411, 4, 851, 739, 209, 491, 213, 160, 222, 897, 607, 845, 861, 271, 586, 330, 732, 471, 452, 217, 277, 726, 907, 456, 412, 687, 611, 858, 454, 978, 288, 614, 458, 667, 944, 703, 28, 97, 768, 589, 795, 518, 898, 501, 36, 696, 121, 819, 874, 535, 522, 241, 87, 163, 110, 184, 425, 164, 524, 582, 445, 553, 340, 487, 444, 111, 475, 254, 294, 356, 670, 994, 307, 311, 57, 758, 494, 914, 707, 236, 619, 62, 320, 723, 129, 215, 188, 321, 731, 964, 350, 835, 581, 884, 988, 462, 304, 40, 348, 430, 306, 249, 322, 728, 720, 225, 390, 807, 287, 83, 60, 652, 66, 940, 190, 500, 175, 280, 233, 178, 899, 147, 853, 868, 563, 52, 701, 493, 1, 105, 372, 955, 801, 792, 942, 337, 364, 373, 648, 365, 576, 852, 451, 403, 556, 584, 600, 970, 916, 802, 273, 286, 485, 725, 361, 892, 23, 329, 54, 538, 366, 302, 608, 663, 810, 345, 519, 915, 61, 104, 67, 864, 816, 950, 644, 513, 959, 992, 953, 194, 314, 441, 933, 292, 836, 828, 255, 433, 820, 911, 409, 319, 46, 21, 855, 207, 590, 585, 787, 22, 949, 9, 3, 548, 328, 99, 53, 405, 86, 75, 645, 352, 91, 886, 298, 537, 81, 278, 917, 193, 34, 248, 747, 383, 904, 743, 432, 638, 765, 256, 296, 691, 489, 562, 251, 930, 171, 706, 889, 946, 857, 510, 793, 419, 230, 945, 442, 716, 850, 740, 102, 842, 653, 167, 798, 829, 839, 822, 143, 788, 299, 424, 19, 751, 353, 115, 733, 659, 18, 679, 58, 971, 420, 74, 642, 177, 387, 417, 875, 566, 714, 431, 96, 469, 669, 206, 972, 25, 526, 212, 180, 460, 95, 8, 575, 531, 80, 938, 776, 55, 354, 827, 360, 517, 496, 583, 530, 867, 400, 291, 606, 965, 598, 79, 991, 923, 552, 859, 169, 969, 173, 979, 35, 149, 106, 754, 560, 631, 602, 557, 661, 779, 157, 507, 895, 238, 847, 89, 374, 948, 438, 218, 508, 668, 558, 665, 757, 980, 773, 492, 367, 561, 98, 986, 515, 863, 32, 796, 931, 680, 973, 814, 735, 131, 685, 335, 843, 17, 573, 891, 347, 769, 640, 214, 261, 540, 246, 318, 416, 266, 686, 837, 151, 244, 26, 210, 285, 468, 825, 397, 536, 781, 47, 385, 975, 913, 375, 651, 643, 804, 414, 308, 391, 276, 547, 628, 161, 50, 587, 156, 371, 29, 477, 809, 591, 268, 395, 700, 170, 195, 935, 770, 377, 155, 448, 985, 107, 168, 181, 346, 44, 817, 250, 48, 112, 939, 976, 14, 511, 794, 609, 130, 637, 263, 476, 479, 446, 612, 674, 694, 826, 59, 72, 15, 182, 423, 947] which has a cost of 30474.834953225596 which is the lower than the cost we found in ex2 and ex3. The time required was 254.55815948500003 seconds.

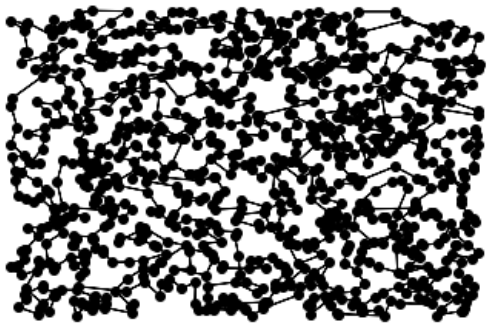


Figure 9: Visualization of Simulated Annealing for 1000 cities