```
In[1]:= data = Import[FileNameJoin[{NotebookDirectory[], "r101.txt"}], "Table"];

In[2]:= vehicles = data[[5, 1]];

In[3]:= q = data[[5, 2]];

In[4]:= depotAndCustomersData = data[[10 ;;]];

In[5]:= depotAndCustomers = depotAndCustomersData[[All, 1]];

In[6]:= customers = Rest[depotAndCustomers];

In[7]:= coords = depotAndCustomersData[[All, 2 ;; 3]];

In[8]:= locations = Association[Thread[depotAndCustomers → coords]];

In[9]:= demand = Association[Thread[depotAndCustomers → depotAndCustomersData[[All, 4]]]];

In[10]:= timeWindows = Association[Thread[depotAndCustomers → depotAndCustomersData[[All, 5 ;; 6]]]];

In[11]:= serviceTime = Association[Thread[depotAndCustomers → depotAndCustomersData[[All, 7]]]];

In[12]:= timeMatrix = N[DistanceMatrix[coords]];

In[13]:= time = Association[
    Table[i ↔ j → timeMatrix[[i + 1, j + 1]], {i, depotAndCustomers}, {j, depotAndCustomers}]];
```

Код для декодирования индивида

In[14]:=
```
Clear[customersToRoutes]
customersToRoutes[customers_] := Module[{
   routes = {},
   currentRoute = {},
   currentVehicleCapacity = 0,
   currentTime = 0,
   lastCustomer = 0,
   customerDemand, possibleVehicleCapacity, possibleTime
  },
  Do[
   customerDemand = demand[customer];
   possibleVehicleCapacity = currentVehicleCapacity + customerDemand;
   possibleTime = Max[currentTime + time[lastCustomer ↦ customer],
      timeWindows[customer][[1]]] + serviceTime[customer];
   If[
    (possibleVehicleCapacity ≤ q) ∧
     (possibleTime + time[customer ↦ 0] ≤ timeWindows[0][[2]]) ∧
     (possibleTime ≤ timeWindows[customer][[2]]),
    AppendTo[currentRoute, customer];
    currentVehicleCapacity = possibleVehicleCapacity;
    currentTime = possibleTime,
    AppendTo[routes, currentRoute];
    currentRoute = {customer};
    currentVehicleCapacity = customerDemand;
    currentTime =
     Max[time[0 ↦ customer], timeWindows[customer][[1]]] + serviceTime[customer];
   ];
   lastCustomer = customer,
   {customer, customers}];
  If[Length[currentRoute] ≥ 1, AppendTo[routes, currentRoute]]
 ]
```

In[16]:=
```
Clear[f]
f[routes_] := Module[{alpha = 100, beta = 0.001},
  alpha * Length[routes] + beta * Total[
     time[#[[1]] ↦ #[[2]]] & /@ Partition[Flatten[Join[{0}, Riffle[routes, 0], {0}]], 2, 1]]
 ]
```

Код для генерации начальной популяции

In[18]:=
```
Clear[knn]
knn[customers_] := Module[{
   sample = RandomSample[customers],
   route = {},
   randomChoice
  },
  randomChoice = RandomChoice[sample];
  AppendTo[route, randomChoice];
  sample = Delete[sample, Position[sample, randomChoice]];
  While[Length[sample] ≠ 0,
   randomChoice = MinimalBy[{locations[route[[-1]]], locations[#], #} & /@ sample,
      EuclideanDistance[#[[1]], #[[2]]] &][[1, 3]];
   AppendTo[route, randomChoice];
   sample = Delete[sample, Position[sample, randomChoice]];
  ];
  route
 ]
```

In[20]:=
```
Clear[populationGeneration]
populationGeneration[populationSize_, customers_, phase2_ : False] := Module[{
   p = IntegerPart[populationSize * 0.1],
   population,
   finalPopulation = {},
   currentPopulation,
   candidatePopulation
  },
  population = Join[
    Table[knn[customers], {p}],
    Table[RandomSample[customers], {populationSize - p}]
   ];
  If[
   phase2,
   currentPopulation = customersToRoutes /@ population;
   candidatePopulation =
    customersToRoutes /@ (Join[Rest[#], {First[#]}] & /@ population);
   Do[
    If[
     f@currentPopulation[[i]] < f@candidatePopulation[[i]],
     AppendTo[finalPopulation, Flatten[currentPopulation[[i]]]],
     AppendTo[finalPopulation, Flatten[candidatePopulation[[i]]]]
    ],
    {i, Length@currentPopulation}];
   finalPopulation
   ,
   population
  ]
 ]
```

Турнирная селекция

In[22]:=
```
Clear[parentPool]
parentPool[populationDecoding_, numberOfParents_, probability_] :=
 Module[{randomChoice},
  Table[
   randomChoice = RandomChoice[populationDecoding, 4];
   If[RandomReal[] ≤ probability,
    MinimalBy[randomChoice, f][[1]], RandomChoice[randomChoice]
   ],
   {numberOfParents}]
 ]
```

Оценка индивида в поколение через ранжирование по

Парето

In[24]:=
```
Clear[vectorOfRoutes]
vectorOfRoutes[routes_] := {Length[routes], Total[
   time[#[[1]] ↔ #[[2]] ] & /@ Partition[Flatten[Join[{0}, Riffle[routes, 0], {0}]], 2, 1]]}
```

In[26]:=
```
Clear[ranking]
ranking[population_] := Module[{
    ranked = {},
    unranked = population,
    vectors = Association[Thread[population → vectorOfRoutes /@ population]],
    currentRank = 1, vector1, nondominated, vector2
   },
   While[Length[unranked] ≥ 2,
    Do[
     vector1 = vectors[unranked[[i]]];
     nondominated = True;
     Do[
      If[i ≠ j,
       vector2 = vectors[unranked[[j]]];
       If[(vector1[[1]] ≥ vector2[[1]]) ∧ (vector1[[2]] ≥ vector2[[2]]) ∧
          (vector1[[1]] > vector2[[1]] ∨ vector1[[2]] > vector2[[2]]),
        nondominated = False
       ]
      ],
      {j, Length@unranked}
     ];
     If[nondominated, AppendTo[ranked, {unranked[[i]], currentRank}]],
     {i, Length@unranked}
    ];
    unranked = Complement[unranked, ranked[[All, 1]]];
    currentRank += 1;
   ];
   AppendTo[ranked, {unranked[[1]], currentRank}];
   #[[1]] → #[[2]] & /@ ranked
  ]
```

Кроссовер

In[28]:=
```
Clear[getCrossover]
getCrossover[parentPool_] := Module[{
    parents = Partition[parentPool, 2],
    children = {},
    parent1, parent2, routeFrom1, routeFrom2, newParent2, newParent1
   },
   Do[
    parent1 = parents[[k, 1]];
    parent2 = parents[[k, 2]];
    routeFrom1 = parent1[[RandomChoice[Range[Length@parent1]]]];
    routeFrom2 = parent2[[RandomChoice[Range[Length@parent2]]]];
    newParent2 = Select[Flatten[parent2], ! MemberQ[routeFrom1, #] &];
    newParent1 = Select[Flatten[parent1], ! MemberQ[routeFrom2, #] &];
    Do[
     newParent2 = Flatten[MinimalBy[customersToRoutes /@ Table[
           Insert[newParent2, routeFrom1[[j]], i], {i, Length[newParent2] + 1}], f][[1]]],
     {j, Length@routeFrom1}
    ];
    Do[
     newParent1 = Flatten[MinimalBy[customersToRoutes /@ Table[
           Insert[newParent1, routeFrom2[[j]], i], {i, Length[newParent1] + 1}], f][[1]]],
     {j, Length@routeFrom2}
    ];
    AppendTo[children, newParent1];
    AppendTo[children, newParent2]
    , {k, Length@parents}];
   children
  ]
```

Мутация с вероятностями p_2opt, p_2hopt, p_4opt

In[30]:=
```
Clear[inverse]
inverse[permutation_, i1_, j1_] := Module[
  {p = permutation, sortedij = Sort[{i1, j1}], i, j, oldi},
  i = sortedij[[1]];
  j = sortedij[[2]];
  If[
   j - i == Length[p] - 1,
   oldi = p[[i]]; p[[i]] = p[[j]]; p[[j]] = oldi; p,
   Join[p[[1 ;; i - 1]], Reverse[p[[i ;; j]]], p[[j + 1 ;; Length@p]]]
  ]
 ]
```

In[32]:=
```
Clear[insert]
insert[permutation_, i_, j_] := Module[
  {p = permutation, element = permutation[[j]]},
  p = Drop[p, {j}]; Insert[p, element, i]
 ]
```

In[34]:=
```
Clear[swap]
swap[permutation_, i_, j_] := Module[
  {p = permutation, element = permutation[[i]]},
  p[[i]] = p[[j]]; p[[j]] = element; p
 ]
```

In[36]:=
```
Clear[getMutation]
getMutation[children_, probabilityMutation_, probabilityMethods_] := Module[{
    newChildren,
    random,
    n = IntegerPart[Length[children] * probabilityMutation],
    range, permutation, p
   },
  newChildren = children[[n + 1 ;;]];
  range = Range[Length[children[[1]]] - 3];
  Do[
   random = RandomChoice[range];
   p = RandomReal[];
   permutation = Which[
     p ≤ probabilityMethods[[1]], swap[children[[i]], random, random + 2],
     p ≤ probabilityMethods[[1]] + probabilityMethods[[2]],
     inverse[children[[i]], random, random + 2],
     True, insert[children[[i]], random, random + 2]
    ];
   AppendTo[newChildren, permutation],
   {i, n}
  ];
  newChildren
 ]
```

ГА для VRPTW

In[38]:=
```
Clear[ga]
ga[populationSize_, customers_, n_, probabilities_] := Module[{
    population, parents, children, mutationChildren,
    currentBest, best = {}, vectorProbanilities = probabilities
   },
  population = customersToRoutes /@ populationGeneration[populationSize, customers];
  Do[
   parents = parentPool[population, 100, vectorProbanilities[[1]]];
   children = getCrossover[parents];
   mutationChildren =
    getMutation[children, vectorProbanilities[[2]], vectorProbanilities[[3 ;; 5]]];
   population = Join[RandomSample[parents, populationSize - Length[mutationChildren]],
     customersToRoutes /@ mutationChildren];
   currentBest = MinimalBy[population, f][[1]];
   AppendTo[best, currentBest],
   {i, n}];
  best
 ]
```

In[40]:=
```
populationSize = 100;
```

In[41]:=
```
gaSteps = 20;
```

In[42]:=
```
solution = ga[populationSize, customers[[ ;; 20]], gaSteps, {0.8, 0.1, 0.5, 0.2, 0.3}];
```

Перебор вероятностей

In[43]:=
```
Clear[enumeration]
enumeration[populationSize_, customers_, gaSteps_, set_] :=
 MinimalBy[{f[Last[ga[populationSize, customers, gaSteps, #]]], #} & /@ set, First][[1, 2]]
```

In[45]:=
```
setOfProbabilities = Flatten /@ Tuples[{Range[0.7, 0.9, 0.1],
    Range[0.1, 0.3, 0.1], Normalize[#, Total] & /@ Table[RandomReal[], {2}, {3}]}]
```

Out[45]=
```
{{0.7, 0.1, 0.139412, 0.509982, 0.350606}, {0.7, 0.1, 0.801549, 0.167512, 0.0309387},
 {0.7, 0.2, 0.139412, 0.509982, 0.350606}, {0.7, 0.2, 0.801549, 0.167512, 0.0309387},
 {0.7, 0.3, 0.139412, 0.509982, 0.350606}, {0.7, 0.3, 0.801549, 0.167512, 0.0309387},
 {0.8, 0.1, 0.139412, 0.509982, 0.350606}, {0.8, 0.1, 0.801549, 0.167512, 0.0309387},
 {0.8, 0.2, 0.139412, 0.509982, 0.350606}, {0.8, 0.2, 0.801549, 0.167512, 0.0309387},
 {0.8, 0.3, 0.139412, 0.509982, 0.350606}, {0.8, 0.3, 0.801549, 0.167512, 0.0309387},
 {0.9, 0.1, 0.139412, 0.509982, 0.350606}, {0.9, 0.1, 0.801549, 0.167512, 0.0309387},
 {0.9, 0.2, 0.139412, 0.509982, 0.350606}, {0.9, 0.2, 0.801549, 0.167512, 0.0309387},
 {0.9, 0.3, 0.139412, 0.509982, 0.350606}, {0.9, 0.3, 0.801549, 0.167512, 0.0309387}}
```

In[46]:=
```
(*enumeration[populationSize,customers[[;;20]],gaSteps,setOfProbabilities]*)
```

In[47]:=
```
(*{0.7`,0.1`,0.5943097852588924`,0.27629064661873204`,0.1293995681223755`} *)
```

Алгоритм подбора вероятностей SPSA

In[48]:=
```
Clear[spsa]
spsa[populationSize_, customers_, gaSteps_, initialProbabilities_, spsaSteps_] := Module[
  {
   c = 0.5,
   a = 0.3,
   theta = initialProbabilities,
   delta, solution1, solution2, folution1, folution2, g
  },
  Do[
   delta = RandomVariate[NormalDistribution[0, 0.5], 5];
   solution1 = ga[populationSize, customers, gaSteps, theta + delta * c];
   solution2 = ga[populationSize, customers, gaSteps, theta + delta * c];
   folution1 = f[Last[solution1]];
   folution2 = f[Last[solution2]];
   g = (folution1 - folution2) * (1/(2 * c * #) & /@ delta);
   theta = theta - g * a;
   theta = Join[theta[[1 ;; 2]], Normalize[theta[[3 ;; 5]], Total]];
   a = a * 0.9;
   c = c * 0.9;
   , {h, spsaSteps}];
  theta
 ]
```

In[50]:=
```
spsaSteps = 10;
```

In[51]:=
```
(*probs=
  spsa[populationSize,customers[[;;20]],gaSteps,{0.5,0.5,0.33,0.33,0.33},spsaSteps];*)
```

In[52]:=
```
(*probs*)
```

In[53]:=
```
(*{0.5985117377218948`,0.5262062078561438`,
 0.2463389174834077`,0.31021972535358094`,0.4434413571630113`} *)
```

Визуализация решения

In[54]:=
```
Clear[manipulate]
manipulate[solution_] := Module[{graphs},
  graphs =
   (DirectedEdge @@@ Partition[Flatten[Join[{0}, Riffle[#, 0], {0}]], 2, 1]) & /@ solution;
  Manipulate[
   Graph[
    graphs[[step]],
    VertexLabels → "Name",
    VertexCoordinates → Table[i → locations[i], {i, 1, Length[Flatten[solution]]}]
   ],
   {step, 1, Length@solution, 1}]
 ]
```

In[56]:=
```
manipulate[solution]
```

Out[56]=