In[•]:=
```
numberOfContainers = 35; (*число контейнеров*)
numberOfPlatforms = 10; (*число платформ*)
numberOfPreferences = 2; (*число приоритетов*)
numberOfBatches = 3; (*число партий*)
weights = {1/3, 2/3};(*веса для критериев оптимизации*)
```

In[•]:=
```
subsets = Subsets[Range@numberOfContainers, {2, 3}] ;
(*варианты расстановки контейнеров на платформах*)
share = 0.5;
feasibleSubsetsForPlatforms = Table[RandomSample[subsets,
    RandomInteger[{1, Floor[share Length@subsets]}]], {p, numberOfPlatforms}];
 (*множество множеств случайных подмножеств контейнеров,
характеризующие платформы,
проще говоря возможные расстановки контейнеров на платформах*)
containers = Union@Flatten@feasibleSubsetsForPlatforms;
(*список задействованных контейнеров
 (необходимо тк какие-то могут не входить в множество допустимых)*)
getVariante = RandomVariate@
    MultinomialDistribution[Length@containers, ConstantArray[1 / #, #] &@#] &;
getCharacterictics = RandomSample@Flatten@
     MapIndexed[ConstantArray[#2⟦1⟧, #1] &, DeleteCases[#, 0]] &;
priors = getCharacterictics@getVariante@numberOfPreferences;
(*приоритет каждого из контейнеров*)
batches = getCharacterictics@getVariante@numberOfBatches - 1;
(*партия каждого из контейнеров*)
setOfContainers = Association[#⟦1⟧ → <|"Приоритет" → #⟦2⟧, "Партия" → #⟦3⟧|> & /@
    Transpose[{containers, priors, batches}]];
 (*0 партия - беспартийник*)(*информация по каждому из контейнеров,
какой у него приоритет и партия*)
distances = AssociationThread[#, RandomReal[{0, 100}, Length@#]] &@
   DeleteDuplicates@Flatten[feasibleSubsetsForPlatforms, 1];
 (*характеризуем каждое подмножество возможной комбинаци контейнеров расстоянием*)
setsForPriors = GroupBy[containers, setOfContainers[#, "Приоритет"] &];
(*группировка контейнеров по приоритетам*)
setsForBatches = KeyDrop[GroupBy[containers, setOfContainers[#, "Партия"] &], 0];
(*группировка контейнеров по партиям, при этом беспартийников нет*)
```

In[•]:=
```
all = Array[x, #] & /@
   Thread[{Range@numberOfPlatforms, Length[#] & /@ feasibleSubsetsForPlatforms}] ;
varsX = Last[#] & /@ all;
varsy1 = Array[y1, numberOfPreferences - 1];
varsy2 = Array[y2, numberOfBatches - 1];
vars = Join[Flatten@varsX, varsy1, varsy2];
```

```
In[∘]:= objFun1 = Total@Flatten[Table[
              Length[#] & /@ feasibleSubsetsForPlatforms⟦i⟧ * varsX⟦i⟧, {i, numberOfPlatforms}]];
        objFun2 = Total[Flatten@varsX * Flatten@Table[Table[Values[distances]⟦
                  Flatten[Position[Keys[distances], feasibleSubsetsForPlatforms⟦j⟧⟦i⟧]]⟧,
                 {i, Length@feasibleSubsetsForPlatforms⟦j⟧}], {j, numberOfPlatforms}]];
        objFun = Dot[weights, {-objFun1, objFun2}];

In[∘]:= c = Last@CoefficientArrays[objFun, vars];
        c1 = Last@CoefficientArrays[objFun1, vars];
        c2 = Last@CoefficientArrays[objFun2, vars];

In[∘]:= subsetsWithC = Select[subsets, MemberQ[#]] & /@ containers ;
        (* каждый элемент множества - множество контейнеров,
        где упомянается контейнер с, по сути это SC_c*)

In[∘]:= (*= - '0', ≥ - '1', ≤ - '-1'*)

In[∘]:= (*первое ограничение*)
        subsetsWithCFromfeasibleSubsetsForPlatforms =
          Select[Flatten[feasibleSubsetsForPlatforms, 1], MemberQ[#]] & /@ containers;
         (* каждый элемент множества - допустимый сценарий,где упомянается контейнер с*)
        aa = Intersection[subsetsWithCFromfeasibleSubsetsForPlatforms⟦#⟧, subsetsWithC⟦#⟧] & /@
           containers;

In[∘]:=  con1 = Table[Total[varsX⟦#⟦1⟧⟧⟦#⟦2⟧⟧ & /@ Flatten[
               Position[feasibleSubsetsForPlatforms, #] & /@ aa⟦i⟧, 1]], {i, numberOfContainers}];
        rhs1 = ConstantArray[{1, -1}, numberOfContainers];

In[∘]:= (*второе ограничение*)
        con2 = Total@# & /@ varsX;
        rhs2 = ConstantArray[{1, -1}, numberOfPlatforms];

In[∘]:=

        (*третье ограничение*)
        kk = Table[setsForPriors[i], {i, numberOfPreferences - 1}];
        kk11 = Table[Flatten[
            Intersection[subsetsWithCFromfeasibleSubsetsForPlatforms⟦#⟧, subsetsWithC⟦#⟧] & /@
             kk⟦i⟧, 1], {i, Length@kk}];
        M =
          99 999;

In[∘]:= con3 =
          Table[Total[varsX⟦#⟦1⟧⟧⟦#⟦2⟧⟧ & /@ Flatten[Position[feasibleSubsetsForPlatforms, #] & /@
                kk11⟦i⟧, 1]], {i, numberOfPreferences - 1}] + M * varsy1;
        rhs3 = {Length[#], 1} & /@
           kk;
```

```
In[ ]:= (*четвертое ограничение*)
        kk2 = Table[setsForPriors[i], {i, 2, numberOfPreferences}];
        kk22 = Table[Flatten[
            Intersection[subsetsWithCFromfeasibleSubsetsForPlatforms[[#]], subsetsWithC[[#]]] & /@
             kk2[[i]], 1], {i, Length@kk2}];

In[ ]:= con4 =
          Table[Total[varsX[[#[[1]]]][[#[[2]]]] & /@ Flatten[Position[feasibleSubsetsForPlatforms, #] & /@
                kk22[[i]], 1]], {i, Length@kk22}] + M * varsy1;
        rhs4 = ConstantArray[{M, -1}, Length@kk2];

In[ ]:=
        (*пятое ограничение*)
        kk3 = Values[setsForBatches];
        kk33 = Table[Flatten[
            Intersection[subsetsWithCFromfeasibleSubsetsForPlatforms[[#]], subsetsWithC[[#]]] & /@
             kk3[[i]], 1], {i, Length@kk3}];

In[ ]:= con5 = Table[Total[varsX[[#[[1]]]][[#[[2]]]] & /@
              Flatten[Position[feasibleSubsetsForPlatforms, #] & /@ kk33[[i]], 1]],
            {i, Length@kk33}] - (Length[#] & /@ kk3 * Reverse@varsy2);
        rhs5 = ConstantArray[{0, 0}, Length@kk3];

In[ ]:=
        lu  = Join[ConstantArray[{0, 1}, Length[Flatten[feasibleSubsetsForPlatforms, 1]]],
           ConstantArray[{0, 1}, numberOfPreferences - 1],
           ConstantArray[{0, 1}, numberOfBatches - 1]];
        domain = ConstantArray[Integers, Length[Flatten[feasibleSubsetsForPlatforms, 1]] +
            numberOfPreferences - 1 + numberOfBatches - 1];
        m = Last@CoefficientArrays[Join[con1, con2, con3, con4, con5], vars];
        b = Join[rhs1, rhs2, rhs3, rhs4, rhs5];
```

LinearProgramming

```
In[ ]:= sol = LinearProgramming[c, m, b, lu, domain];

In[ ]:= positions = Flatten[Position[varsX, #] & /@
            DeleteCases[Take[sol * vars, Length[Flatten[feasibleSubsetsForPlatforms, 1]]], 0], 1];

In[ ]:= cont = feasibleSubsetsForPlatforms[[#[[1]]]][[#[[2]]]] & /@ positions;
        (*номера расставленных контейнеров*)

In[ ]:= platf = #[[1]] & /@ positions;(*номера задействованных платформ*)

In[ ]:= Thread[{platf, cont}](*сопоставление поставленных контейнеров платформе*)

Out[ ]= {{1, {14, 15, 29}}, {2, {8, 23, 32}}, {3, {2, 12, 33}}, {4, {4, 11, 20}}, {5, {5, 16, 30}},
         {6, {19, 28, 31}}, {7, {10, 22, 24}}, {8, {1, 6, 35}}, {9, {7, 25, 27}}, {10, {3, 17, 26}}}

In[ ]:= Length[Flatten[cont]] (*число расставленных контейнеров на платформах*)

Out[ ]= 30
```

```
In[•]:= Length[platf] (*число задействованных платформ*)

Out[•]= 10
```

GurobiOptimization

```
In[•]:= Get[StringJoin[NotebookDirectory[], "\\Gurobi-main\\GurobiOptimization.wl"]];
        directory = "C:\\gurobi912\\win64\\bin\\";

In[•]:= solGurobi = GurobiOptimization[Normal /@ {-c1, c2}, Normal@m,
           b, lu, domain, directory, MultiObjOpt → {PriorityOpt → {2, 1}}];

In[•]:= positionsGurobi = Flatten[Position[varsX, #] & /@ DeleteCases[
              Take[solGurobi * vars, Length[Flatten[feasibleSubsetsForPlatforms, 1]]], 0], 1];

In[•]:= contGurobi = feasibleSubsetsForPlatforms⟦#⟦1⟧⟧⟦#⟦2⟧⟧ & /@ positionsGurobi;
        (*номера расставленных контейнеров*)

In[•]:= platfGurobi = #⟦1⟧ & /@ positionsGurobi; (*номера задействованных платформ*)

In[•]:= Thread[{platfGurobi, contGurobi}] (*сопоставление поставленных контейнеров платформе*)

Out[•]= {{1, {14, 15, 29}}, {2, {8, 23, 32}}, {3, {2, 12, 33}}, {4, {19, 28, 31}}, {5, {4, 11, 20}},
         {6, {5, 16, 30}}, {7, {10, 22, 24}}, {8, {1, 6, 35}}, {9, {7, 25, 27}}, {10, {3, 17, 26}}}

In[•]:= Length[Flatten[contGurobi]] (*число расставленных контейнеров на платформах*)

Out[•]= 30

In[•]:= Length[platfGurobi] (*число задействованных платформ*)

Out[•]= 10
```