Лекция 1. Моделирование и Оптимизация в Julia: Оптимизация нелинейных и линейных задач с непрерывной обл. опр. (глобальная и локальная оптимизация)

Дисциплина: Б1.О.01 Математические методы принятия решений

Направление подготовки: 01.04.02 Прикладная математика и информатика

Направленность (профиль) подготовки: Математическое моделирование

Основная цель: Научить студентов применять пакеты языка программирования Julia, предназначенные для нахождения глобальных и локальных оптимумов в линейных и нелинейных задачах математического программирования. На первом занятии рассматриваются задачи с переменными, непрерывно меняющимися в своей области определения.

Основные умения и навыки. После изучения материала лекции и разбора примеров студенты должны уметь:

- пользоваться специализированными пакетами BlackBoxOptim.jl, Optim.jl, Convex.jl, JuMP.jl для формализации задачи, построения модели и нахождения глобального или локального оптимального решения;
- устанавливать необходимые пакеты в своем проекте;
- подключать solver (решатель) к модели и задавать его параметры;
- решать задачи, аналогичные рассмотренным на лекции.

Содержание лекции:

Часть 1: Пакеты BlackBoxOptim.jl, Optim.jl и Convex.jl. Глобальная и локальная оптимизация.

Часть 2: Знакомство с JuMP – предметно-ориентированным языком моделирования для математической оптимизации, встроенным в Julia. Изучение правил построения моделей. Подключение решателей Ipopt, HiGHS, GLPK, SCS. Вычислительные эксперименты.

Примеры, рассмотренные на лекции:

- Минимизация функции Розенброка.
- Задачи линейного программирования.
- Задача минимизации нормы при наличии ограничений.
- Транспортная задача

Лекция проводится в компьютерном классе, оборудованном проектором. На каждом компьютере имеется все необходимое программное обеспечение.

Лекция 1. Моделирование и Оптимизация в Julia:

Оптимизация нелинейных и линейных задач с непрерывной обл. опр. (глобальная и локальная оптимизация)

Часть 1: Пакеты BlackBoxOptim.jl, Optim.jl и Convex.jl

Глобальная и локальная оптимизация

BlackBoxOptim — пакет глобальной оптимизации для Julia. Он поддерживает как многоцелевые, так и одноцелевые задачи оптимизации и ориентирован на (мета) эвристические / стохастические алгоритмы. Не требует, чтобы оптимизируемая функция была дифференцируемой. Загрузить пакет:

using Pkg

Pkg.add("BlackBoxOptim")

Пример 1.1: Минимизация функции Розенброка:

$$f(x) = (a - x_1)^2 + b \cdot (x_2 - x_1^2)^2$$

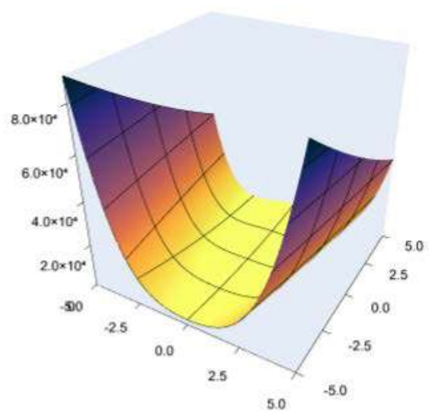
$$x_1 \in (-5, 5), \quad x_2 \in (-5, 5)$$

при $a = 1$, $b = 100$. Глобальный минимум в точке (a, a^2).

In [2]:
```
# график функции (поверхность можно перемещать, увеличивать и вращать):
using Plots; plotlyjs()
x = y = collect(-5:0.1:5)
g(x, y) = (1.0 - x)^2 + 100.0 * (y - x^2)^2

surface(x, y, g, c=cgrad(:thermal, rev = true), legend = false)
wireframe!(x, y, g)
```

Out[2]:



In [3]:
```
# Сохранение рисунка:
savefig("rosenbrock.png")
```

Out[3]:
```
"C:\\Users\\Olga1\\rosenbrock.png"
```

In [4]:
```
# Задание функции:
Rosenbrock(x) = (1.0 - x[1])^2 + 100.0 * (x[2] - x[1]^2)^2
```

Out[4]:
```
Rosenbrock (generic function with 1 method)
```

```
Out[4]: Rosenbrock (generic function with 1 method)
```

In [6]:
```
# Оптимизация:
using BlackBoxOptim

res = bboptimize(Rosenbrock; SearchRange = (-5.0, 5.0), NumDimensions = 2)

best_fitness(res) < 0.001
```

```
Starting optimization with optimizer DiffEvoOpt{FitPopulation{Float64}, RadiusLimitedSelector, BlackBoxOptim.Adap
tiveDiffEvoRandBin{3}, RandomBound{ContinuousRectSearchSpace}}
0.00 secs, 0 evals, 0 steps

Optimization stopped after 10001 steps and 0.01 seconds
Termination reason: Max number of steps (10000) reached
Steps per second = 909188.60
Function evals per second = 918552.30
Improvements/step = 0.21620
Total function evaluations = 10104


Best candidate found: [1.0, 1.0]

Fitness: 0.000000000
```

Out[6]:
```
true
```

Оптимизация с пакетами Optim.jl, Optimization.jl

Optim полностью написан на языке Julia (в отличие от большинства других решателей).

Optim имеет доступ к функциям автоматического дифференцирования через пакеты в JuliaDiff.

Загрузить пакет:

using Pkg

Pkg.add("Optim")

Optim — это пакет Julia, реализующий различные алгоритмы для одномерной и многомерной оптимизации.

Пример 1.2.1: Минимизация функции Розенброка в пакете Optim (параметры по умолчанию - алгоритм Nelder-Mead):

In [7]:
```
using Optim
f(x) = (1.0 - x[1])^2 + 100.0 * (x[2] - x[1]^2)^2
x0 = [0.0, 0.0]
optimize(f, x0)
```

Out[7]:
```
 * Status: success

 * Candidate solution
    Final objective value:     3.525527e-09

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    60
    f(x) calls:    117
```

Пример 1.2.2: Минимизация функции Розенброка в пакете Optim (параметры по умолчанию, алгоритм BFGS):

In [8]:
```
using Optim
rosenbrock(x) =  (1.0 - x[1])^2 + 100.0 * (x[2] - x[1]^2)^2
result = optimize(rosenbrock, zeros(2), BFGS())
```

Out[8]:
```
 * Status: success
```

```
 * Candidate solution
    Final objective value:     5.471417e-17

 * Found with
    Algorithm:     BFGS

 * Convergence measures
    |x - x'|               = 3.47e-07 ≰ 0.0e+00
    |x - x'|/|x'|          = 3.47e-07 ≰ 0.0e+00
    |f(x) - f(x')|         = 6.59e-14 ≰ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 1.20e+03 ≰ 0.0e+00
    |g(x)|                 = 2.33e-09 ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    16
    f(x) calls:    53
    ∇f(x) calls:   53
```

Пример 1.2.4: Минимизация функции Розенброка в пакете Optim методом Optim.IPNewton()

In [9]:

```
using Optimization, OptimizationOptimJL
rosenbrock(x, p) = (p[1] - x[1])^2 + p[2] * (x[2] - x[1]^2)^2
cons = (res, x, p) -> res .= [x[1]^2 + x[2]^2]
x0 = zeros(2)
p = [1.0, 100.0]
prob = OptimizationFunction(rosenbrock, Optimization.AutoForwardDiff(); cons = cons)
prob = Optimization.OptimizationProblem(prob, x0, p, lcons = [-5.0], ucons = [10.0])
sol = solve(prob, IPNewton())
```

Out[9]:

```
u: 2-element Vector{Float64}:
 0.9999999992669327
 0.9999999985109471
```

Пример 1.2.5: Метод ParticleSwarm выполняет в Optim глобальную оптимизацию для задач с геометрическими ограничениями или без них. Работает как с нижними и верхними границами, установленными lb и ub, так и без них в Optimization.OptimizationProblem:

In [10]:

```
using Optimization, OptimizationOptimJL
rosenbrock(x, p) = (p[1] - x[1])^2 + p[2] * (x[2] - x[1]^2)^2
x0 = zeros(2)
p = [1.0, 100.0]
ffff = OptimizationFunction(rosenbrock)
prob = Optimization.OptimizationProblem(ffff, x0, p, lb = [-1.0, -1.0], ub = [1.0, 1.0])
sol = solve(prob, Optim.ParticleSwarm(lower = prob.lb, upper = prob.ub, n_particles = 100))
```

Out[10]:

```
u: 2-element Vector{Float64}:
 1.0
 1.0
```

In [ ]:

```

```

Пример 1.2.6: Метод Optim.SAMIN() выполняет в Optim глобальную оптимизацию для задач с геометрическими ограничениями:

In [11]:

```
using Optimization, OptimizationOptimJL
rosenbrock(x, p) = (1 - x[1])^2 + 100 * (x[2] - x[1]^2)^2
x0 = zeros(2)
p = [1.0, 100.0]
f2 = OptimizationFunction(rosenbrock, Optimization.AutoForwardDiff())
prob = Optimization.OptimizationProblem(f2, x0, p, lb = [-1.0, -1.0], ub = [1.0, 1.0])
sol = solve(prob, Optim.SAMIN())
```

```
================================================================================
SAMIN results
NO CONVERGENCE: MAXEVALS exceeded

     Obj. value:           0.02876

       parameter      search width
         0.84048           2.00000
         0.71216           2.00000
================================================================================
```

Out[11]:

```
u: 2-element Vector{Float64}:
 0.8404760605783965
 0.7121592069032203
```

Решение задач методами выпуклой оптимизации (Convex)

In [12]:
```
using Convex
using SCS
using LinearAlgebra
```

Пример 1.3. Задача линейного программирования:

$$65x_1 + 70x_2 + 60x_3 + 120x_4 \to \max$$

$$4x_1 + 2x_2 + 2x_3 + 8x_4 \le 4800,$$

$$2x_1 + 10x_2 + 6x_3 \le 2400,$$

$$x_1 + 2x_3 + x_4 \le 1500,$$

$$x_i \ge 0,\ i = \overline{1,4},\quad x_2 \le 100,\quad x_1 + x_4 \le 200$$

In [15]:
```
x = Variable(4)
c = [65; 70; 60; 120]
A = [4 2 2 8;2 10 6 0;1 0 2 1]
b = [4800; 2400; 1500]
p = Convex.maximize(dot(c, x)) # или c' * x
p.constraints += A * x <= b
p.constraints += [x >= 0; x[2] <= 100; x[1] + x[3] <= 200]
Convex.solve!(p, SCS.Optimizer; silent_solver = true)

println("Оптимальное значение функционала задачи:   ", round(p.optval, digits = 3))
println("Решение:    ", round.(evaluate(x), digits = 5))
```

```
Оптимальное значение функционала задачи:   82000.0
Решение:    [0.0, 100.0, 200.0, 525.0]
```

Пример 1.4.:

$$\|Ax - b\|^2 \to \min \quad \text{при условии } x \ge 0$$

In [16]:
```
# Генерируем данные к задаче случайным образом
n = 4;  m = 5
A = randn(n, m); b = randn(n, 1)
println("Размерность матрицы A: $n на $m")
println("Размерность вектора b: $m на 1")
```

```
Размерность матрицы A: 4 на 5
Размерность вектора b: 5 на 1
```

In [17]:
```
# A и b:
for i=1:n
    for j=1:m
        print(round(A[i,j], digits=3)," ")
    end
    println("         ", round(b[i], digits=3))
end
```

```
-1.086 -0.617 -2.591 1.354 -1.032          0.513
-0.954 -0.034 1.78 -0.715 1.826          0.686
-0.298 0.832 -1.438 -1.594 0.704          0.77
-1.825 0.561 -0.315 -0.422 -0.166          -0.028
```

In [18]:
```
# Построение модели и решение:

using Convex, SCS

# Создаем вектор-столбец (m x 1)
x = Variable(m)
# Требуется минимизировать ||Ax - b||^2 при условии x >= 0
```

```
# Используем: minimize(objective, constraints)
problem = minimize(sumsquares(A * x - b), [x >= 0])

# Решим задачу вызвав solve!
Convex.solve!(problem, SCS.Optimizer; silent_solver = true)

# Статус задачи:
problem.status # :Optimal, :Infeasible, :Unbounded etc.

# Оптимальное значение функционала:
println("Оптимальное значение функционала: ",problem.optval)
println("Решение:    ", round.(evaluate(x), digits = 5))
```

```
Оптимальное значение функционала: 0.9991923038246311
Решение:    [-0.0, 0.00829, -0.0, 0.0, 0.25617]
```

Часть 2: Основы моделирования в JuMP

JuMP — является предметно-ориентированным языком моделирования для математической оптимизации, встроенным в Julia. В настоящее время он поддерживает ряд открытых и коммерческих решателей (Artelys Knitro, BARON, Bonmin, Cbc, Clp, Couenne, CPLEX, ECOS, FICO Xpress, GLPK, Gurobi, Ipopt, MOSEK, NLopt, SCS, HiGHS, SCIP и др.).

Можно рассматривать JuMP как набор вспомогательных пакетов для математической оптимизации в Julia.

JuMP позволяет легко формулировать постановку задачи математического программирования, в том числе для целочисленных и смешанных задач; задач выпуклого программирования; задач динамической оптимизации и др.

LP = линейное программирование;

QP = квадратичное программирование;

SOCP = коническое программирование второго порядка (включая задачи с выпуклыми квадратичными ограничениями и / или целью);

MILP = Смешанное и целочисленное линейное программирование;

НЛП = Нелинейное программирование;

MINLP = смешанно-целочисленное нелинейное программирование;

SDP = полуопределенное программирование;

MISDP = смешанно-целочисленное полуопределенное программирование;

JuMP используется совместно со специализированными пакетами для решения конкретных классов задач.

Решатели (Solver), рассмотренные на занятии: Ipopt, HiGHS, GLPK, SCS

In [1]:
```
using JuMP
using Ipopt
```

Пример 2.1.: Снова функция Розенброка:

$$f(x) = (1.0 - x_1)^2 + 100.0 \cdot (x_2 - x_1^2)^2 \to \min$$

$$x_1 \in (-5, 5), \quad x_2 \in (-5, 5)$$

In [2]:
```
# функция Розенброка
model1 = Model(Ipopt.Optimizer)
@variable(model1, -5.0 <= x[i = 1:2] <= 5.0)
@NLobjective(model1, Min, (1.0 - x[1])^2 + 100.0 * (x[2] - x[1]^2)^2)
set_start_value.(x[1], -5.0)
set_start_value.(x[2], -5.0)
optimize!(model1)
println("solution = ", (value(x[1]),value(x[2])))
```

```
******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit https://github.com/coin-or/Ipopt
******************************************************************************

This is Ipopt version 3.13.4, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).
```

```
Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        3

Total number of variables............................:        2
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        2
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  8.6780381e+04 0.00e+00 9.72e+01  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  3.0733341e+04 0.00e+00 4.11e+01  -1.0 1.35e+00    -  3.80e-02 1.00e+00f  1
   2  7.0928973e+03 0.00e+00 1.19e+01  -1.0 1.35e+00    -  2.68e-01 1.00e+00f  1
   3  2.3397159e+02 0.00e+00 1.88e+00  -1.0 5.41e+00    -  1.00e+00 1.00e+00f  1
   4  1.4371479e+01 0.00e+00 3.17e-01  -1.0 2.74e-01    -  1.00e+00 1.00e+00f  1
   5  6.3325208e+00 0.00e+00 2.89e-02  -1.7 1.03e-01    -  1.00e+00 1.00e+00f  1
   6  5.5004449e+00 0.00e+00 6.56e-02  -2.5 8.01e-01    -  9.01e-01 1.00e+00f  1
   7  4.3645757e+00 0.00e+00 2.79e-02  -2.5 3.46e-01    -  1.00e+00 1.00e+00f  1
   8  3.8395224e+00 0.00e+00 4.83e-02  -3.8 5.82e-01    -  9.08e-01 1.00e+00f  1
   9  2.8313531e+00 0.00e+00 1.05e-02  -3.8 1.04e-01    -  1.00e+00 1.00e+00f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10  2.5804793e+00 0.00e+00 2.57e-02  -3.8 7.14e-01    -  1.00e+00 5.00e-01f  2
  11  1.7633177e+00 0.00e+00 6.06e-03  -3.8 8.67e-02    -  1.00e+00 1.00e+00f  1
  12  1.6686281e+00 0.00e+00 2.45e-02  -3.8 5.29e-01    -  1.00e+00 5.00e-01f  2
  13  9.9001933e-01 0.00e+00 3.29e-03  -3.8 6.75e-02    -  1.00e+00 1.00e+00f  1
  14  7.8786327e-01 0.00e+00 6.75e-03  -3.8 5.19e-01    -  1.00e+00 2.50e-01f  3
  15  5.6596492e-01 0.00e+00 9.73e-03  -3.8 1.71e-01    -  1.00e+00 1.00e+00f  1
  16  3.6119492e-01 0.00e+00 3.41e-03  -3.8 1.01e-01    -  1.00e+00 1.00e+00f  1
  17  2.9913382e-01 0.00e+00 1.37e-02  -3.8 1.93e-01    -  1.00e+00 1.00e+00f  1
  18  1.2458468e-01 0.00e+00 7.21e-04  -3.8 9.33e-02    -  1.00e+00 1.00e+00f  1
  19  7.8515974e-02 0.00e+00 7.41e-03  -5.7 3.18e-01    -  9.56e-01 5.00e-01f  2
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  20  3.1700263e-02 0.00e+00 1.08e-03  -5.7 1.01e-01    -  1.00e+00 1.00e+00f  1
  21  1.8629837e-02 0.00e+00 7.21e-03  -5.7 1.83e-01    -  1.00e+00 1.00e+00f  1
  22  2.1951174e-03 0.00e+00 1.27e-04  -5.7 4.84e-02    -  1.00e+00 1.00e+00f  1
  23  3.6236571e-04 0.00e+00 1.23e-03  -5.7 8.29e-02    -  1.00e+00 1.00e+00f  1
  24  1.0749823e-06 0.00e+00 1.91e-06  -5.7 6.65e-03    -  1.00e+00 1.00e+00f  1
  25  1.1730666e-10 0.00e+00 7.07e-07  -8.6 2.07e-03    -  1.00e+00 1.00e+00f  1
  26  3.5499914e-14 0.00e+00 7.89e-13  -8.6 4.15e-06    -  1.00e+00 1.00e+00f  1

Number of Iterations....: 26

                                   (scaled)                 (unscaled)
Objective...............:   5.9154691681460067e-17    3.5499913571877819e-14
Dual infeasibility......:   7.8887792671990457e-13    4.7342142138314910e-10
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   2.5059045470702524e-09    1.5038434367877998e-06
Overall NLP error.......:   2.5059045470702524e-09    1.5038434367877998e-06


Number of objective function evaluations             = 48
Number of objective gradient evaluations             = 27
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 26
Total CPU secs in IPOPT (w/o function evaluations)   =      0.768
Total CPU secs in NLP function evaluations           =      1.151

EXIT: Optimal Solution Found.
solution = (0.9999998116908037, 0.9999996227526742)
```

In [3]:

```
# Общая информация по процессу решения задачи:
solution_summary(model1)
```

Out[3]:

```
* Solver : Ipopt

* Status
  Termination status : LOCALLY_SOLVED
  Primal status      : FEASIBLE_POINT
  Dual status        : FEASIBLE_POINT
  Message from the solver:
  "Solve_Succeeded"

* Candidate solution
  Objective value      : 3.549991357187782e-14

* Work counters
```

```
Solve time (sec)   : 2.15200
```

Пример 2.2. Задача линейного программирования:

In [4]:

```julia
model2 = Model(Ipopt.Optimizer)

 @variable(model2,x>=0)
 @variable(model2,y>=0)
 @constraint(model2, 5*x+6*y <= 60)
 @constraint(model2, -5*x+3*y<= 15)
 @constraint(model2, 2*x-7*y <= 14)
 @constraint(model2, x+2*y >= 2)
 @objective(model2, Max, x+2*y)

 print(model2)
 optimize!(model2)

 println("Optimal objective value= $(objective_value(model2))")
 println("x= ",value(x))
 println("y= ",value(y))
```

$$
\begin{aligned}
\max \quad & x + 2y \\
\text{Subject to} \quad & x + 2y \geq 2.0 \\
& 5x + 6y \leq 60.0 \\
& -5x + 3y \leq 15.0 \\
& 2x - 7y \leq 14.0 \\
& x \geq 0.0 \\
& y \geq 0.0
\end{aligned}
$$

```
This is Ipopt version 3.13.4, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        8
Number of nonzeros in Lagrangian Hessian.............:        0

Total number of variables............................:        2
                     variables with only lower bounds:        2
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        4
        inequality constraints with only lower bounds:        1
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        3

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  2.9999970e-02 1.97e+00 2.60e-01  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  1.0629696e-01 1.89e+00 1.54e+00  -1.0 5.47e+00    -  1.91e-02 4.83e-02h  1
   2  2.0683374e+00 0.00e+00 1.64e+00  -1.0 6.44e+00    -  2.93e-01 1.00e+00f  1
   3  2.2611643e+00 0.00e+00 1.10e+00  -1.0 8.86e-01    -  4.49e-01 1.00e+00f  1
   4  7.5111137e+00 0.00e+00 7.98e-01  -1.0 1.92e+01    -  3.02e-01 1.00e+00f  1
   5  1.3717778e+01 0.00e+00 1.24e-01  -1.0 2.18e+01    -  5.34e-01 1.00e+00f  1
   6  1.7909990e+01 0.00e+00 1.92e-02  -1.0 1.90e+01    -  7.05e-01 9.78e-01f  1
   7  1.8639214e+01 0.00e+00 1.28e-03  -1.7 9.02e+00    -  1.00e+00 9.37e-01f  1
   8  1.8661006e+01 0.00e+00 2.83e-08  -2.5 9.14e-02    -  1.00e+00 1.00e+00f  1
   9  1.8666367e+01 0.00e+00 1.50e-09  -3.8 3.67e-02    -  1.00e+00 1.00e+00f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10  1.8666663e+01 0.00e+00 1.84e-11  -5.7 2.05e-03    -  1.00e+00 1.00e+00f  1
  11  1.8666667e+01 0.00e+00 2.51e-14  -8.6 2.57e-05    -  1.00e+00 1.00e+00f  1

Number of Iterations....: 11

                                   (scaled)                 (unscaled)
Objective...............:  -1.8666666848321928e+01    1.8666666848321928e+01
Dual infeasibility......:   2.5063284780912909e-14    2.5063284780912909e-14
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   2.5080611021185267e-09   -2.5080611021185267e-09
Overall NLP error.......:   2.5080611021185267e-09    2.5063284780912909e-14


Number of objective function evaluations             = 12
Number of objective gradient evaluations             = 12
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 12
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 1
Number of Lagrangian Hessian evaluations             = 1
Total CPU secs in IPOPT (w/o function evaluations)   =      0.001
Total CPU secs in NLP function evaluations           =      0.067
```

```
EXIT: Optimal Solution Found.
Optimal objective value= 18.666666848321928
x= 2.000000023178471
y= 8.333333412571728
```

Пример 2.3. Транспортная задача (без условия целочисленности решения):

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}x_{ij} \rightarrow \min$$

$$\sum_{j=1}^{m} x_{ij} = supply_i, \quad i = \overline{1, n},$$

$$\sum_{i=1}^{n} x_{ij} = demand_j, \quad j = \overline{1, m},$$

$$x_{ij} \geq 0, \quad i = \overline{1, n}, \quad j = \overline{1, m}$$

In [5]:

```
# Данные задачи:
using DataFrames
dataTransportation =  DataFrame("RetailStore1" => [10, 16, 18, 30],
                        "RetailStore2" => [13,11,20,100],
                        "RetailStore3" => [15,12,17,70],
                        "RetailStore4" => [14,19,21,50],
                        "Supply" => [60, 150, 40, 250])
```

Out[5]: 4 rows × 5 columns

	RetailStore1	RetailStore2	RetailStore3	RetailStore4	Supply
	Int64	Int64	Int64	Int64	Int64
1	10	13	15	14	60
2	16	11	12	19	150
3	18	20	17	21	40
4	30	100	70	50	250

In [6]:

```
n=3
m=4
A = [10 13 15 14; 16 11 12 19; 18 20 17 21]
costs = A
supply = [60,150,40]
demand = [30,100,70,50]
sum(supply) == sum(demand)
```

Out[6]:

```
true
```

In [7]:

```
using JuMP
using GLPK

model3 = Model(GLPK.Optimizer)
@variable(model3,x[1:n,1:m]>=0)
@variable(model3,y>=0)
@constraint(model3, sum(x[1:3,j] for j in 1:4) .== supply)
@constraint(model3, sum(x[i,1:4] for i in 1:3) .== demand)
@objective(model3, Min, sum(costs.*x))

optimize!(model3)

println(model3)
println(value.(x))
```

```
Min 10 x[1,1] + 16 x[2,1] + 18 x[3,1] + 13 x[1,2] + 11 x[2,2] + 20 x[3,2] + 15 x[1,3] + 12 x[2,3] + 17 x[3,3] + 1
4 x[1,4] + 19 x[2,4] + 21 x[3,4]
Subject to
 x[1,1] + x[1,2] + x[1,3] + x[1,4] == 60.0
 x[2,1] + x[2,2] + x[2,3] + x[2,4] == 150.0
 x[3,1] + x[3,2] + x[3,3] + x[3,4] == 40.0
 x[1,1] + x[2,1] + x[3,1] == 30.0
 x[1,2] + x[2,2] + x[3,2] == 100.0
 x[1,3] + x[2,3] + x[3,3] == 70.0
 x[1,4] + x[2,4] + x[3,4] == 50.0
```

```
 x[1,1] >= 0.0
 x[2,1] >= 0.0
 x[3,1] >= 0.0
 x[1,2] >= 0.0
 x[2,2] >= 0.0
 x[3,2] >= 0.0
 x[1,3] >= 0.0
 x[2,3] >= 0.0
 x[3,3] >= 0.0
 x[1,4] >= 0.0
 x[2,4] >= 0.0
 x[3,4] >= 0.0
 y >= 0.0

[30.0 0.0 0.0 30.0; 0.0 100.0 50.0 0.0; 0.0 0.0 20.0 20.0]
```

In [ ]:

Лекция 2. Моделирование и Оптимизация в Julia: Оптимизация целочисленных и смешанных задач линейного программирования

Дисциплина: Б1.О.01 Математические методы принятия решений

Направление подготовки: 01.04.02 Прикладная математика и информатика

Направленность (профиль) подготовки: Математическое моделирование

Основная цель: Научить студентов применять пакеты языка программирования Julia для решения целочисленных и смешанных задач линейного программирования.

Основные умения и навыки. После изучения материала лекции и разбора примеров студенты должны уметь:

- формализовать текстовое задание в виде задачи целочисленного линейного программирования;
- описывать модели с целочисленными или смешанными переменными, используя для этого функции пакета JuMP;
- пользоваться решателями GLPK, HiGHS, SCIP и понимать их особенности и ограничения;
- решать задачи, аналогичные рассмотренным на лекции.

Содержание лекции: Построение моделей и решение оптимизационных задач.

Примеры, рассмотренные на лекции:

- Задача о рюкзаке.
- Задачи целочисленного линейного программирования.
- Задача о выборе инвестиций.
- Задача о наборе персонала.
- Задача о башнях.
- Задача о составлении расписания.

Лекция 2. Моделирование и Оптимизация в Julia:

Оптимизация целочисленных и смешанных задач линейного программирования

Solvers: GLPK, HiGHS, SCIP

In [ ]:

```

```

Пример 1. Задача о рюкзаке:

$$\sum_{i=1}^{n} c_i x_i \to \max$$

$$\sum_{i=1}^{n} w_i x_i \leq \text{capacity},$$

$$x_i \in \{0; 1\}, \quad i = \overline{1, n}.$$

In [1]:

```
capacity = 20

using DataFrames
items = [1,2,3,4,5,6,7,8,9,10]
utilitys = [1,4,7,4,2,9,11,4,3,5]
weights = [4,7,3,1,3,8,4,2,3,6]
df = DataFrame(; Items_n = items, Utility_c = utilitys, Weight_w = weights)
show(df)
```

```
10×3 DataFrame
 Row │ Items_n  Utility_c  Weight_w
     │ Int64    Int64      Int64
─────┼──────────────────────────────
   1 │       1          1         4
   2 │       2          4         7
   3 │       3          7         3
   4 │       4          4         1
   5 │       5          2         3
   6 │       6          9         8
   7 │       7         11         4
   8 │       8          4         2
   9 │       9          3         3
  10 │      10          5         6
```

In [2]:

```
# Построение модели и решение:
using GLPK
using JuMP
utilities = [1, 4, 7, 4, 2, 9, 11, 4, 3, 5]
weights = [4, 7, 3, 1, 3, 8, 4 , 2, 3, 6]
capacity = 20
knapsack = Model(GLPK.Optimizer)
@variable(knapsack, x[1:10], Bin)
@objective(knapsack, Max, sum(utilities .* x))
@constraint(knapsack, sum(weights .* x) <= capacity)
optimize!(knapsack)
println(knapsack)
println(value.(x))
```

```
Max x[1] + 4 x[2] + 7 x[3] + 4 x[4] + 2 x[5] + 9 x[6] + 11 x[7] + 4 x[8] + 3 x[9] + 5 x[10]
Subject to
 4 x[1] + 7 x[2] + 3 x[3] + x[4] + 3 x[5] + 8 x[6] + 4 x[7] + 2 x[8] + 3 x[9] + 6 x[10] <= 20.0
 x[1] binary
 x[2] binary
 x[3] binary
 x[4] binary
 x[5] binary
 x[6] binary
 x[7] binary
 x[8] binary
```

```
 x[9] binary
 x[10] binary

[0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0]
```

Пример 2. Задача целочисленного линейного программирования

In [3]:

```julia
using JuMP
using HiGHS
```

In [4]:

```julia
# Сравните с решением задачи 2.2 без условия целочисленности решения из лекции 1.
model2 = Model(HiGHS.Optimizer)

 @variable(model2,x>=0,Int)
 @variable(model2,y>=0,Int)
 @constraint(model2, 5*x+6*y <= 60)
 @constraint(model2, -5*x+3*y<= 15)
 @constraint(model2, 2*x-7*y <= 14)
 @constraint(model2, x+2*y >= 2)
 @objective(model2, Max, 3*x+y)

 print(model2)
 optimize!(model2)

 println("Termination status: $(termination_status(model2))")
 if termination_status(model2) == MOI.OPTIMAL
 println("Optimal objective value= $(objective_value(model2))")
 println("x= ",round(Int64,value(x)))
 println("y= ",round(Int64,value(y)))
 else
 println("No optimal solution available")
 end
```

$$
\begin{aligned}
\max \quad & 3x + y\\
\text{Subject to} \quad & x + 2y \geq 2.0\\
& 5x + 6y \leq 60.0\\
& -5x + 3y \leq 15.0\\
& 2x - 7y \leq 14.0\\
& x \geq 0.0\\
& y \geq 0.0\\
& x \in \mathbb{Z}\\
& y \in \mathbb{Z}
\end{aligned}
$$

```
Termination status: OPTIMAL
Optimal objective value= 31.0
x= 10
y= 1

Presolving model
4 rows, 2 cols, 8 nonzeros
4 rows, 2 cols, 8 nonzeros
Objective function is integral with scale 1

Solving MIP model with:
   4 rows
   2 cols (0 binary, 2 integer, 0 implied int., 0 continuous)
   8 nonzeros

Solving root node LP relaxation

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |       Wo
rk
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | LpIters
Time

         0       0         0   0.00%   -33.2340426     inf                  inf        0      0      0         2
0.0s
 R       0       0         0   0.00%   -31             -31                0.00%        6      1      0         4
0.0s

Solving report
  Status            Optimal
  Primal bound      -31
  Dual bound        -31
  Solution status   feasible
```

```
                    31 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
Timing              0.02 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
Nodes               1
LP iterations       4 (total)
                    0 (strong br.)
                    2 (separation)
                    0 (heuristics)
```

Пример 3. Задача о выборе инвестиций:

Совет директоров должен принять решение о выборе инвестиций. Имеется n=7 проектов, в которые можно инвестировать. Суммарный объем инвестиций не должен превышать M у.е. При положительном решении по проекту j минимальный объем инвестиций в проект j равен доле a от M. На множестве инвестиций заданы условия:

Тип инвестиций	Условия вложения
1	-
2	если есть инвестиции в 1
3	если есть инвестиции в 2
4	если есть инвестиции и в 1, и в 2
5	если нет инвестиций в 1 или 2
6	если нет инвестиций ни в 2, ни в 3 в первый проект
7	если есть инвестиции в 2, и нет в 3

Известны r_j – доход от вложений 1 у.е. в j-й проект, c_j – фиксированные расходы при вложении j. Компания стремится получить максимальную прибыль от вложений. Требуется построить математическую модель и найти оптимальное решение.

Математическая модель:

$$\sum_{j=1}^{n} (r_j y_j - c_j x_j) \rightarrow \max$$

$$aMx_j \leq y_j \leq Mx_j, \quad j = \overline{1, n},$$

$$Ax \leq b,$$

$$y_j \geq 0, \quad x_j \in \{0; 1\}, \quad j = \overline{1, n}.$$

y_j - величина инвестиций в i-й проект,

$$x_j = \begin{cases} 1, & \text{если есть инвестиции в проект } j, \\ 0, & \text{иначе} \end{cases}$$

In [5]:

```julia
n = 7
M = 200
a = 1/7
r = [6.1,6.2,7.5,4.9,7.1,6.0,5.3]
c = [0.5,0.2,0.6,0.2,0.1,0.7,0.2]
A = [-1 1 0 0 0 0 0; 0 -1 1 0 0 0 0; -1 0 0 1 0 0 0; 0 -1 0 1 0 0 0; 1 1 0 0 1 0 0; 0 0.5 0.5 0 0 1 0; 0 -1 0 0 
b = [0; 0; 0; 0; 2; 1; 0; 1]
```

Out[5]:

```
8-element Vector{Int64}:
 0
 0
 0
 0
 2
 1
 0
 1
```

In [6]:

```julia
using JuMP
using HiGHS
```

In [7]:

```julia
model3 = Model(HiGHS.Optimizer)
@variable(model3, x[j=1:7], Bin)
@variable(model3, y[j=1:7] >= 0.0)
@constraint(model3, sum(y[j] for j in 1:7) <= M)
for j = 1:7
    @constraint(model3, y[j] <= M * x[j])
    @constraint(model3, y[j] >= (M / 7) * x[j])
end
for i = 1:8
@constraint(model3, sum(A[i,j] * x[j] for j in 1:7) <= b[i])
end
@objective(model3, Max, sum(r[j]*y[j] - c[j]*x[j] for j in 1:7))

# write_to_file(model3, "model3.lp")

# Если не удается решить задачу имеющимися решателями, то можно сохранить модель в файл.lp
# и решить в любом другом подходящем некоммерческом или коммерческом решателе.
# Например, в SCIP  https://www.scipopt.org/

 #print(model3)
 optimize!(model3)

 println("Termination status: $(termination_status(model3))")
 if termination_status(model3) == MOI.OPTIMAL
 println("Optimal objective value= $(objective_value(model3))")
 println("x= ",value.(x))
 println("y= ",value.(y))
 else
 println("No optimal solution available")
 end
```

```
Termination status: OPTIMAL
Optimal objective value= 1421.557142857143
x= [1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0]
y= [28.57142857142857, 28.57142857142857, 142.8571428571429, 0.0, 0.0, 0.0, 0.0]

Presolving model
23 rows, 14 cols, 53 nonzeros
23 rows, 14 cols, 54 nonzeros

Solving MIP model with:
   23 rows
   14 cols (7 binary, 0 integer, 0 implied int., 7 continuous)
   54 nonzeros

( 0.0s) Starting symmetry detection
( 0.0s) No symmetry present

Solving root node LP relaxation

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |       Wo
rk
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | LpIters
Time

         0       0         0   0.00%   -1438.98889     inf                  inf        0      0      0         7
0.0s
 R       0       0         0   0.00%   -1432.17143     -1419.9            0.86%       13      1      0         9
0.0s
 C       0       0         0   0.00%   -1421.55714     -1421.55714        0.00%       18      3      0        10
0.0s

Solving report
  Status            Optimal
  Primal bound      -1421.55714286
  Dual bound        -1421.55714286
  Solution status   feasible
                    1421.55714286 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    2.84217094304e-14 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     10 (total)
                    0 (strong br.)
                    3 (separation)
                    0 (heuristics)
```

Пример 4: Задача о наборе персонала

Ресторан работает 7 дней в неделю. Повара работают 6 часов в день, 5 дней подряд и затем 2 дня отдыхают. У всех поваров одинаковая зарплата. Приготовление каждого блюда занимает определенное время, и для каждого дня недели установленно общее необходимое количество часов для приготовления пищи. Данные о ежедневном количестве рабочих часов приведены в таблице:

День недели	Требуемое количество часов
понедельник	150
вторник	200
среда	400
четверг	300
пятница	700
суббота	800
воскресенье	300

Администратору нужно принять решение о найме поваров. Нужно решить, какое количество поваров нанять, и в какие дни они должны работать, чтобы суммарные затраты на оплату труда были наименьшими. Требуется построить математическую модель и найти оптимальное решение.

Обозначим через x_j - количество поваров, которые на первую смену выходят на работу в день недели с номером j (понедельник - первый день).

In [ ]:

```
using JuMP
using HiGHS
```

In [8]:

```
A = [1 0 0 1 1 1 1; 1 1 0 0 1 1 1; 1 1 1 0 0 1 1; 1 1 1 1 0 0 1; 1 1 1 1 1 0 0; 0 1 1 1 1 1 0; 0 0 1 1 1 1 1]
b = [150; 200; 400; 300; 700; 800; 300]
```

Out[8]:

```
7-element Vector{Int64}:
 150
 200
 400
 300
 700
 800
 300
```

In [9]:

```
model4 = Model(HiGHS.Optimizer)
@variable(model4, x[j=1:7]>=0, Int)
for i = 1:7
@constraint(model4, sum(A[i,j] * x[j] for j in 1:7) >= b[i] / 6)
end
@objective(model4, Min, sum(x[j] for j in 1:7))

# write_to_file(model4, "model4.lp")

 print(model4)
 optimize!(model4)

 println("Termination status: $(termination_status(model4))")
 if termination_status(model4) == MOI.OPTIMAL
 println("Optimal objective value= $(objective_value(model4))")
 println("x= ",value.(x))
 else
 println("No optimal solution available")
 end
```

$$
\begin{aligned}
\min\quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7\\
\text{Subject to}\quad & x_1 + x_4 + x_5 + x_6 + x_7 \geq 25.0\\
& x_1 + x_2 + x_5 + x_6 + x_7 \geq 33.33333333333333\\
& x_1 + x_2 + x_3 + x_6 + x_7 \geq 66.66666666666666\\
& x_1 + x_2 + x_3 + x_4 + x_7 \geq 50.0\\
& x_1 + x_2 + x_3 + x_4 + x_5 \geq 116.66666666666666\\
& x_2 + x_3 + x_4 + x_5 + x_6 \geq 133.33333333333331\\
& x_3 + x_4 + x_5 + x_6 + x_7 \geq 50.0\\
& x_1 \geq 0.0\\
& x_2 \geq 0.0\\
& x_3 \geq 0.0\\
& x_4 \geq 0.0\\
& x_5 \geq 0.0\\
& x_6 \geq 0.0
\end{aligned}
$$

$$x_7 \geq 0.0$$
$$x_1 \in Z$$
$$x_2 \in Z$$
$$x_3 \in Z$$
$$x_4 \in Z$$
$$x_5 \in Z$$
$$x_6 \in Z$$
$$x_7 \in Z$$

```
Termination status: OPTIMAL
Optimal objective value= 134.0
x= [0.0, 84.0, 0.0, 0.0, 50.0, 0.0, 0.0]

Presolving model
7 rows, 7 cols, 35 nonzeros
7 rows, 7 cols, 35 nonzeros
Objective function is integral with scale 1

Solving MIP model with:
   7 rows
   7 cols (0 binary, 7 integer, 0 implied int., 0 continuous)
   35 nonzeros

Solving root node LP relaxation

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |       Wo
rk
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | LpIters
Time

 T       0       0         0   0.00%   -inf            134             9999.00%        0      0      0         2
0.0s

Solving report
  Status            Optimal
  Primal bound      134
  Dual bound        134
  Solution status   feasible
                    134 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     2 (total)
                    0 (strong br.)
                    0 (separation)
                    0 (heuristics)
```

Пример 5: Задача о башнях

Имеется конечное множество клиентов $i = 1..n$ и конечное множество возможных мест $j = 1..m$, где можно размещать башни сотовой связи. Известны расстояния между клиентами и башнями. Каждый клиент прикрепляется к ближайшей башне, если ближайших башен несколько, то к любой из них. Нужно разместить ровно p башен так, чтобы после прикрепления клиентов к башням разница между максимальным и минимальным числом клиентов, закрепленных за башнями, была минимальной.

Введем переменные:

$$y_i = \begin{cases} 1, & \text{если в } j\text{-м пункте размещается башня,} \\ 0, & \text{иначе;} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{если клиент } i \text{ обслуживается башней в } j\text{-м пункте,} \\ 0, & \text{иначе.} \end{cases}$$

u - максимальное количество клиентов, обслуживаемых одной башней,

l - минимальное количество клиентов, обслуживаемых одной башней.

$$u - l \to \min$$

$$x_{ij} \leq y_j, \quad i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, m\},$$

$$\sum_{j=1}^{m} y_j = p,$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i \in \{1, \ldots, n\},$$

$$\sum_{i=1}^{n} x_{ij} \le u, \quad j \in \{1, \ldots, m\},$$

$$l \le \sum_{i=1}^{n} x_{ij} + n(1 - y_j), \quad j \in \{1, \ldots, m\},$$

$$\sum_{l=1}^{m} c_{il}x_{il} + (M_i - c_{ij})y_j \le M_i, \quad i \in \{1, \ldots, n\}, \ j \in \{1, \ldots, m\},$$

$$x_{ij} \in \{0; 1\}, \quad i \in \{1, \ldots, n\}, \ j \in \{1, \ldots, m\},$$

$$y_j \in \{0; 1\}, \quad j \in \{1, \ldots, m\},$$

$$u, l \in Z, \quad u, l \ge 0,$$

где $M_i = \max_j c_{ij}, \ i \in \{1, \ldots, n\}$.

In [ ]:
```
using JuMP
using HiGHS
```

In [10]:
```
n = 4
m = 3
p = 2
c = [3 4 2; 1 5 6; 2 7 3; 8 2 5]
M = zeros(n)
for i = 1:n
    M[i] = maximum(c[i,j] for j in 1:m)
end
```

In [11]:
```
modelTower = Model(HiGHS.Optimizer)
@variable(modelTower,x[1:n,1:m],Bin)
@variable(modelTower,y[1:m],Bin)
@variable(modelTower,u>=0,Int)
@variable(modelTower,l>=0,Int)
@objective(modelTower, Min,u-l)
for i = 1:n
for j= 1:m
@constraint(modelTower, x[i,j]<=y[j])
@constraint(modelTower,sum(c[i,l]*x[i,l] for l in 1:m)+ (M[i]-c[i,j])*y[j] <= M[i])
end
@constraint(modelTower,sum(x[i,j] for j in 1:m) == 1)
end
for j =1:m
@constraint(modelTower, sum(x[i,j] for i in 1:n) <= u)
@constraint(modelTower, sum(x[i,j] for i in 1:n) + n*(1-y[j])>= l)
end
@constraint(modelTower,sum(y[j] for j in 1:m) == p)

optimize!(modelTower)

println("статус задачи: $(termination_status(modelTower))")
    if termination_status(modelTower) == MOI.OPTIMAL
    println("Оптимальное значение задачи: $(objective_value(modelTower))")
    for i = 1:n
       for j= 1:m
       print("x[$i,$j] = ", value(x[i,j]), ", ")
       end
       println(" ")
    end
    for j = 1:m
       print("y[$j] = ", value(y[j]), ", ")
    end
    else
    println("Допустимого решения нет")
end
```

```
статус задачи: OPTIMAL
Оптимальное значение задачи: 0.0
x[1,1] = 0.0, x[1,2] = 0.0, x[1,3] = 1.0,
x[2,1] = 0.0, x[2,2] = 1.0, x[2,3] = 0.0,
x[3,1] = 0.0, x[3,2] = 0.0, x[3,3] = 1.0,
x[4,1] = 0.0, x[4,2] = 1.0, x[4,3] = 0.0,
```

```
y[1] = 0.0, y[2] = 1.0, y[3] = 1.0,
Presolving model
35 rows, 17 cols, 116 nonzeros
32 rows, 6 cols, 56 nonzeros
6 rows, 5 cols, 16 nonzeros
0 rows, 0 cols, 0 nonzeros
Presolve: Optimal

Solving report
  Status            Optimal
  Primal bound      0
  Dual bound        0
  Solution status   feasible
                    0 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.00 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             0
  LP iterations     0 (total)
                    0 (strong br.)
                    0 (separation)
                    0 (heuristics)
```

Задача 6. Задача о составлении расписания

Имеется n работ и одна машина для выполнения этих работ. В каждый момент времени может выполняться только одна работа. Каждая работа выполняется без прерывания. Известны следующие целочисленные величины: p_j – длительность выполнения работы j, d_j – крайний срок завершения работы j, w_j – коэффициент важности работы j, $j = 1, \ldots, n$. Требуется найти допустимое расписание с наименьшей взвешенной суммой времен завершения работ. Требуется построить математическую модель и найти оптимальное решение.

Математическая модель:

$$y_{i,j} = \begin{cases} 1, & \text{если работа } i \text{ выполняется раньше работы } j, \\ 0, & \text{иначе;} \end{cases}$$

$t_j -$ время завершения работы j;

$$J = \sum_{t=1}^{n} w_j \cdot t_j \quad \rightarrow \quad \min$$

$$t_j \leq d_j$$

$$t_j \geq p_j$$

$$y_{i,j} + y_{j,i} = 1 \quad \text{для всех } i = \overline{1,n},\ j = \overline{1,n},\ i \neq j$$

$$y_{j,j} = 0 \quad \text{для всех } j = \overline{1,n}$$

$t_j \leq t_i - p_i$, если работа j завершилась раньше, чем работа i

$t_i \leq t_j - p_j$, если работа j завершилась позже, чем работа i

Последние два условия перепишем в следующем виде:

$$t_j \leq t_i - p_i + d_j y_{i,j}, \quad i = \overline{1,n},\ j = \overline{1,n},\ i \neq j$$

$$t_i \leq t_j - p_j + d_i(1 - y_{i,j}), \quad i = \overline{1,n},\ j = \overline{1,n},\ i \neq j$$

$$t_j \in \mathrm{Z},\ t_j \geq 0 \text{ для всех } j = \overline{1,n}$$

$$y_{i,j} \in \{0, 1\} \text{ для всех } i = \overline{1,n},\ j = \overline{1,n}$$

In [ ]:

```
using JuMP
using HiGHS
```

In [12]:

```
n = 10
```

```
# чем больше w_j,  тем выше приоритет работы j
w = [2,5,8,1,9,3,4,8,3,10]
d = [34,35,22,36,18,20,35,36,39,48]
p = [4,1,5,2,6,2,3,4,1,8]
```

Out[12]:

```
10-element Vector{Int64}:
 4
 1
 5
 2
 6
 2
 3
 4
 1
 8
```

In [13]:

```
model6 = Model(HiGHS.Optimizer)
@variable(model6, y[i=1:n,j=1:n]>=0, Bin)
@variable(model6, t[j=1:n]>=0, Int)
for j = 1:n
    @constraint(model6, t[j] <= d[j])
    @constraint(model6, t[j] >= p[j])
end

for i = 1:n
    for j = 1:n
        if i != j
         @constraint(model6, y[i,j] + y[j,i] == 1)
         @constraint(model6, t[j]<= t[i] -  p[i] + d[j]*y[i,j] )
         @constraint(model6,   t[i]<= t[j] -  p[j] + d[i]* (1-y[i,j]))
        else  @constraint(model6, y[i,i]  == 0)
        end
    end
end
@objective(model6, Min, sum(w[j]*t[j] for j in 1:n))

#print(model6)
optimize!(model6)
```

```
Presolving model
180 rows, 55 cols, 540 nonzeros
88 rows, 55 cols, 264 nonzeros
88 rows, 55 cols, 264 nonzeros
Objective function is integral with scale 1

Solving MIP model with:
   88 rows
   55 cols (45 binary, 10 integer, 0 implied int., 0 continuous)
   264 nonzeros

( 0.0s) Starting symmetry detection
( 0.0s) No symmetry present

Solving root node LP relaxation

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |       Wo
rk
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | LpIters
Time

         0       0         0   0.00%   242             inf                  inf        0      0      0        45
0.0s
 L       0       0         0   0.00%   436.2896756     996               56.20%     7113     67     53       711
0.1s
 L       0       0         0   0.00%   436.2896756     921               52.63%     6666     33     53       711
0.2s
 L     101      12        38  54.05%   436.9941533     905               51.71%     4995    122    435      6881
0.4s
 T     377      30       147  62.59%   436.9941533     864               49.42%     4935    136   1264     14840
0.7s
 T     394      30       154  62.60%   436.9941533     861               49.25%     4937    136   1288     14900
0.7s
 T     412      30       161  62.60%   436.9941533     859               49.13%     4939    136   1323     14961
0.7s
 T     415      28       163  62.64%   436.9941533     857               49.01%     4941    136   1336     14963
0.7s
 L    1263      97       507  66.03%   436.9941533     853               48.77%     9599    116   2379     21493
1.0s
 T    4997     357      2043  77.28%   559.1248224     851               34.30%     5947    125   7143     50456
2.7s
 L    5454     380      2234  77.41%   563.6915747     849               33.61%     6269     92   7794     54578
2.9s
     15544     789      6698  87.33%   721.1626016     849               15.06%     3317     82   5745    140489
8.0s
```

```
Solving report
  Status            Optimal
  Primal bound      849
  Dual bound        849
  Solution status   feasible
                    849 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            11.94 (total)
                    0.00 (presolve)
                    0.00 (postsolve)
  Nodes             22053
  LP iterations     207004 (total)
                    10786 (strong br.)
                    29838 (separation)
                    12390 (heuristics)
```

In [14]:

```
println("статус задачи: $(termination_status(model6))")
    if termination_status(model6) == MOI.OPTIMAL
    println("Оптимальное значение задачи: $(objective_value(model6))")
    for i = 1:n
       for j= 1:n
       print("y[$i,$j] = ", value(y[i,j]), ", ")
       end
       println(" ")
    end
    for j = 1:n
       print("t[$j] = ", value(t[j]), ", ")
    end
    else
    println("Допустимого решения нет")
end
```

```
статус задачи: OPTIMAL
Оптимальное значение задачи: 849.0
y[1,1] = 0.0, y[1,2] = 0.0, y[1,3] = 0.0, y[1,4] = 1.0, y[1,5] = 0.0, y[1,6] = 0.0, y[1,7] = 0.0, y[1,8] = 0.0, y
[1,9] = 0.0, y[1,10] = 0.0,
y[2,1] = 1.0, y[2,2] = 0.0, y[2,3] = 1.0, y[2,4] = 1.0, y[2,5] = 1.0, y[2,6] = 1.0, y[2,7] = 1.0, y[2,8] = 1.0, y
[2,9] = 1.0, y[2,10] = 1.0,
y[3,1] = 1.0, y[3,2] = 0.0, y[3,3] = 0.0, y[3,4] = 1.0, y[3,5] = 1.0, y[3,6] = 1.0, y[3,7] = 1.0, y[3,8] = 0.0, y
[3,9] = 0.0, y[3,10] = 1.0,
y[4,1] = 0.0, y[4,2] = 0.0, y[4,3] = 0.0, y[4,4] = 0.0, y[4,5] = 0.0, y[4,6] = 0.0, y[4,7] = 0.0, y[4,8] = 0.0, y
[4,9] = 0.0, y[4,10] = 0.0,
y[5,1] = 1.0, y[5,2] = 0.0, y[5,3] = 0.0, y[5,4] = 1.0, y[5,5] = 0.0, y[5,6] = 1.0, y[5,7] = 1.0, y[5,8] = 0.0, y
[5,9] = 0.0, y[5,10] = 1.0,
y[6,1] = 1.0, y[6,2] = 0.0, y[6,3] = 0.0, y[6,4] = 1.0, y[6,5] = 0.0, y[6,6] = 0.0, y[6,7] = 1.0, y[6,8] = 0.0, y
[6,9] = 0.0, y[6,10] = 1.0,
y[7,1] = 1.0, y[7,2] = 0.0, y[7,3] = 0.0, y[7,4] = 1.0, y[7,5] = 0.0, y[7,6] = 0.0, y[7,7] = 0.0, y[7,8] = 0.0, y
[7,9] = 0.0, y[7,10] = 1.0,
y[8,1] = 1.0, y[8,2] = 0.0, y[8,3] = 1.0, y[8,4] = 1.0, y[8,5] = 1.0, y[8,6] = 1.0, y[8,7] = 1.0, y[8,8] = 0.0, y
[8,9] = 0.0, y[8,10] = 1.0,
y[9,1] = 1.0, y[9,2] = 0.0, y[9,3] = 1.0, y[9,4] = 1.0, y[9,5] = 1.0, y[9,6] = 1.0, y[9,7] = 1.0, y[9,8] = 1.0, y
[9,9] = 0.0, y[9,10] = 1.0,
y[10,1] = 1.0, y[10,2] = 0.0, y[10,3] = 0.0, y[10,4] = 1.0, y[10,5] = 0.0, y[10,6] = 0.0, y[10,7] = 0.0, y[10,8]
= 0.0, y[10,9] = 0.0, y[10,10] = 0.0,
t[1] = 34.0, t[2] = 1.0, t[3] = 11.0, t[4] = 36.0, t[5] = 17.0, t[6] = 19.0, t[7] = 22.0, t[8] = 6.0, t[9] = 2.0,
t[10] = 30.0,
```

In [15]:

```
# пример записи решения в файл

outfile = "solution_y_t_model6.txt"
open(outfile, "w") do f
    for i = 1:n
       for j= 1:n
       println(f, "y[$i,$j] =        ", value(y[i,j]))
       end
       println(" ")
    end
    for j = 1:n
       println(f, "t[$j] = ", value(t[j]))
    end
end
```

Лекция 3. Моделирование и Оптимизация в Julia: Продвинутый уровень: оптимизация бинарных задач; задачи большой размерности; чтение задачи из файла

Дисциплина: Б1.О.01 Математические методы принятия решений

Направление подготовки: 01.04.02 Прикладная математика и информатика

Направленность (профиль) подготовки: Математическое моделирование

Основная цель: Научить студентов применять пакеты языка программирования Julia для решения нестандартных целочисленных и смешанных задач линейного программирования.

Основные умения и навыки. После изучения материала лекции и разбора примеров студенты должны уметь:

- получать данные о задаче из текстового файла и записывать в файл решение;
- выделять отдельные ограничения задачи, имена переменных, параметры и коэффициенты, используя для этого функции пакета JuMP;
- пользоваться решателями GLPK, HiGHS, SCIP для решения (частичного решения) задач;
- использовать эвристики и решать задачи, аналогичные рассмотренным на лекции.

Содержание лекции: Рассматриваются задачи:

- Задача о выборе мест для размещения базовых станций (особенности задачи – данные в файле; большая размерность: 151637 бинарных переменных, 1323337 ограничений; наличие конфликтного множества).
- Транспортная задача (особенности – имеются несвязанные пункты отправки и приема товара; данные читаются из файла).
- Задача о p-медиане (большие вычислительные сложности, даже при небольшой размерности задачи)

Лекция 3. Моделирование и Оптимизация в Julia:

Продвинутый уровень: оптимизация бинарных задач; задачи большой размерности; чтение задачи из файла

In [1]:
```julia
using JuMP
using HiGHS
```

Задача 1. О выборе мест для размещения базовых станций

(данные в файле, 151637 бинарных переменных, 1323337 ограничений)

Введем переменные ($i \in I, \quad j \in J$), где I - множество мест, где можно разместить базовые станции, J - общее множество клиентов; клиент с номером j может быть обслужен базовой станцией, если она размещена в месте с номером i из I_j (I_j, - заданные множества):

$$y_i = \begin{cases} 1, & \text{если базовая станция размещена в месте } i, \\ 0, & \text{иначе;} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{если клиент } j \text{ обеспечиваются связью от хотя бы одной базовой станции,} \\ 0, & \text{otherwise.} \end{cases}$$

Постановка задачи:

$$\begin{aligned}
&\min \sum_{i \in I} y_i, \\
&\sum_{i \in I_j} y_i \geq 1 && \forall j \in H, \\
&\sum_{i \in I_j} y_i \geq x_j && \forall j \in J \setminus H, \\
&\sum_{j \in J \setminus H} x_j \geq p, && (1) \\
&y_{i_1} + y_{i_2} \leq 1 && \forall (i_1, i_2) \in V, \\
&y_i \in \{0, 1\} && \forall i \in I, \\
&x_j \in \{0, 1\} && \forall j \in J \setminus H.
\end{aligned}$$

Цель задачи - минимизировать количество функционирующих базовых станций. Клиентов из множества H необходимо обязательно обеспечить связью. Необходимо обеспечить не менее p клиентов из множества J/H. Задача с конфликтным множеством: заданы пары номеров базовых станций, которые не могут функционировать одновременно.

In [2]:
```julia
model1 = read_from_file("Problem1.mps")   # файл Problem1.mps расположен в текущей директории
```

Out[2]:
```
A JuMP Model
Minimization problem with:
Variables: 151637
Objective function type: AffExpr
`AffExpr`-in-`MathOptInterface.GreaterThan{Float64}`: 1323331 constraints
`AffExpr`-in-`MathOptInterface.LessThan{Float64}`: 6 constraints
`VariableRef`-in-`MathOptInterface.Interval{Float64}`: 151503 constraints
`VariableRef`-in-`MathOptInterface.ZeroOne`: 134 constraints
Model mode: AUTOMATIC
CachingOptimizer state: NO_OPTIMIZER
Solver name: No optimizer attached.
```

In [3]:
```julia
# Выбираем решатель, устанавливаем опции "presolve" (предварительный разбор задачи решателем) и максимальное врем
set_optimizer(model1, HiGHS.Optimizer)
set_optimizer_attribute(model1, "presolve", "on")
set_optimizer_attribute(model1, "time_limit", 100.0)
optimize!(model1)
println("Статус процесса решения: $(termination_status(model1))")
```

```
Статус процесса решения: OPTIMAL
```

```
Presolving model
23496 rows, 21326 cols, 75980 nonzeros
21193 rows, 21237 cols, 68987 nonzeros
Objective function is integral with scale 1

Solving MIP model with:
   21193 rows
   21237 cols (65 binary, 0 integer, 0 implied int., 21172 continuous)
   68987 nonzeros

( 3.0s) Starting symmetry detection
( 3.1s) No symmetry present

Solving root node LP relaxation

        Nodes      |    B&B Tree     |            Objective Bounds              |  Dynamic Constraints |       Wo
rk
     Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |   Cuts   InLp Confl. | LpIters
Time

         0       0         0   0.00%   72.91994625     inf                  inf        0      0      0     22524
19.8s

Solving report
  Status            Optimal
  Primal bound      73
  Dual bound        73
  Solution status   feasible
                    73 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            36.36 (total)
                    2.97 (presolve)
                    0.00 (postsolve)
  Nodes             1
  LP iterations     23533 (total)
                    0 (strong br.)
                    1009 (separation)
                    0 (heuristics)
```

In [4]:

```
# Чтобы увидеть найденное решение x, y, нужно определить под каким именем эти переменные были записаны в модели
# (т.е. в исходном файле "Problem1.mps"). Из неравенства (1) можно найти имена соответствующие переменным x.
# Тогда остальные переменные - это y.

# Определяем ограничения вида больше или равно, среди них определяем ограничение вида (1)
# переменные, входящие в ограничение (1), - это переменные x

greater_constraints_all = all_constraints(model1, AffExpr, MOI.GreaterThan{Float64})

E_constraints = [c for c in greater_constraints_all if constraint_object(c).set != MOI.GreaterThan(0.0)
                                                       && constraint_object(c).set != MOI.GreaterThan(1.0)]

# Из правой части этого ограничения знаем значение p:
constraint_object(E_constraints[1]).set
```

Out[4]:

```
MathOptInterface.GreaterThan{Float64}(135621.9)
```

In [5]:

```
# Определяем названия переменных в задаче (они заданы в исходном файле):
# Разделяем имена переменных x и y (все x входят в неравенство c ):
XX = collect(linear_terms(constraint_object(E_constraints[1]).func))
x_ref = Dict{Int64, VariableRef}()
X = Set()

for i = 1:length(XX)
    x_ref[i] = XX[i][2]
    push!(X,x_ref[i])
end

Y = setdiff(all_variables(model1),X)
lx = length(X)
ly = length(Y)

# Вывод решения на экран (т.к. у нас слишком много переменных, выведем на экран значения только переменных y, ос
println("статус задачи: $(termination_status(model1))")
println("Найденное значение задачи: $(objective_value(model1))")
println("Найденное количество базовых станций: $(objective_value(model1))")
println("Количество клиентов, обслуживаемых станциями : $(objective_value(model1))")
println("Значения y_i,  i=1..$ly :")
for y in  Y
    println("$y = ", value(y))
```

```julia
end
#println("Значения x_j,  j=1..$lx :")
#for x in  X
#    println("$x = ", value(x))
#end
```

```
статус задачи: OPTIMAL
Найденное значение задачи: 73.0
Найденное количество базовых станций: 73.0
Количество клиентов, обслуживаемых станциями : 73.0
Значения y_i,  i=1..134 :
C0000000 = 0.0
C0000001 = 1.0
C0000002 = 1.0
C0000003 = 1.0
C0000004 = 1.0
C0000005 = 1.0
C0000006 = 0.0
C0000007 = 0.0
C0000008 = 1.0
C0000009 = 0.0
C0000010 = 0.0
C0000011 = 1.0
C0000012 = 1.0
C0000013 = 0.0
C0000014 = 1.0
C0000015 = 1.0
C0000016 = 1.0
C0000017 = 1.0
C0000018 = 0.0
C0000019 = 1.0
C0000020 = 1.0
C0000021 = 0.0
C0000022 = 0.0
C0000023 = 0.0
C0000024 = -0.0
C0000025 = 0.0
C0000026 = 1.0
C0000027 = 1.0
C0000028 = 0.0
C0000029 = 1.0
C0000030 = 1.0
C0000031 = 0.0
C0000032 = 0.0
C0000033 = 0.0
C0000034 = 1.0
C0000035 = 0.0
C0000036 = 1.0
C0000037 = 0.0
C0000038 = -0.0
C0000039 = 1.0
C0000040 = 1.0
C0000041 = 1.0
C0000042 = 1.0
C0000043 = 1.0
C0000044 = 0.0
C0000045 = 1.0
C0000046 = 1.0
C0000047 = 1.0
C0000048 = 0.0
C0000049 = 1.0
C0000050 = 1.0
C0000051 = 0.0
C0000052 = 1.0
C0000053 = 0.0
C0000054 = 0.0
C0000055 = 1.0
C0000056 = 0.0
C0000057 = 1.0
C0000058 = 0.0
C0000059 = 1.0
C0000060 = 1.0
C0000061 = -0.0
C0000062 = 1.0
C0000063 = 1.0
C0000064 = 0.0
C0000065 = 1.0
C0000066 = 0.0
C0000067 = 1.0
C0000068 = 1.0
C0000069 = 1.0
C0000070 = 1.0
C0000071 = 0.0
C0000072 = 1.0
C0000073 = 0.0
C0000074 = 0.0
C0000075 = 1.0
C0000076 = 0.0
```

```
C0000077 = 1.0
C0000078 = 1.0
C0000079 = 0.0
C0000080 = 1.0
C0000081 = 0.0
C0000082 = 1.0
C0000083 = 0.0
C0000084 = 1.0
C0000085 = 1.0
C0000086 = 0.0
C0000087 = 1.0
C0000088 = 1.0
C0000089 = 1.0
C0000090 = 0.0
C0000091 = 0.0
C0000092 = 0.0
C0000093 = 0.0
C0000094 = 0.0
C0000095 = 1.0
C0000096 = 0.0
C0000097 = 0.0
C0000098 = 0.0
C0000099 = 1.0
C0000100 = 1.0
C0000101 = 0.0
C0000102 = 1.0
C0000103 = 0.0
C0000104 = 0.0
C0000105 = 0.0
C0000106 = 1.0
C0000107 = 0.0
C0000108 = 0.0
C0000109 = -0.0
C0000110 = 0.0
C0000111 = 0.0
C0000112 = 1.0
C0000113 = 0.0
C0000114 = 1.0
C0000115 = 1.0
C0000116 = 0.0
C0000117 = 1.0
C0000118 = 1.0
C0000119 = 0.0
C0000120 = 1.0
C0000121 = 0.0
C0000122 = 1.0
C0000123 = 0.0
C0000124 = 1.0
C0000125 = 1.0
C0000126 = -0.0
C0000127 = 1.0
C0000128 = 1.0
C0000129 = 1.0
C0000130 = 0.0
C0000131 = 1.0
C0000132 = 1.0
C0000133 = 1.0
```

In [6]:

```
# Запись решения в файл

outfile = "solution-model1-HiGHS.txt"
open(outfile, "w") do f
    println(f, "Значения y_i,  i=1..$ly :")
    for y in  Y
       println(f, "$y =                     ", value(y))
    end
    println(f, "Значения x_j,  j=1..$lx :")
    for x in  X
       println(f, "$x =                                    ", value(x))
    end
end
```

Задача 2. Транспортная задача (имеются несвязанные пункты отправки и приема товара)

In [7]:

```
using JuMP
import DelimitedFiles
import HiGHS
```

In [8]:

```
# имитация получения данных из файла transp.txt:
```

```
open(joinpath(@__DIR__, "transp.txt"), "w") do io
    print(
        io,
        """
             . FRA  DET LAN WIN  STL  FRE  LAF SUPPLY
          GARY  39   14  11  14   16   82    8   1400
          CLEV  27    .  12   .   26   95   17   2600
          PITT  24   14  17  13   28   99   20   2900
        DEMAND 900 1200 600 400 1700 1100 1000      0
        """,
    )
    return
end
```

In [9]:

```
# Достаем данные из файла transp.txt:
# Обратите внимание, что точка на месте (i,j) означает, что либо из пункта отправки i нельзя доставить груз в пун
# либо пункт приема j не может принять груз от пункта i. Это нужно учесть при составлении модели.
function read_data(filename::String)
    data = DelimitedFiles.readdlm(filename)
    rows, columns = data[2:end, 1], data[1, 2:end]
    return Containers.DenseAxisArray(data[2:end, 2:end], rows, columns)
end

data = read_data(joinpath(@__DIR__, "transp.txt"))
```

Out[9]:

```
2-dimensional DenseAxisArray{Any,2,...} with index sets:
    Dimension 1, Any["GARY", "CLEV", "PITT", "DEMAND"]
    Dimension 2, Any["FRA", "DET", "LAN", "WIN", "STL", "FRE", "LAF", "SUPPLY"]
And data, a 4×8 Matrix{Any}:
  39    14      11   14       16    82      8  1400
  27      "."   12     "."    26    95     17  2600
  24    14      17   13       28    99     20  2900
 900  1200     600  400     1700  1100   1000     0
```

In [10]:

```
function solve_transportation_problem(data::Containers.DenseAxisArray)
    # Get the set of supplies and demands
    O, D = axes(data)
    # Drop the SUPPLY and DEMAND nodes from our sets
    O, D = setdiff(O, ["DEMAND"]), setdiff(D, ["SUPPLY"])
    model = Model(HiGHS.Optimizer)
    set_silent(model)
    @variable(model, x[o in O, d in D] >= 0)
    # Remove arcs with "." cost by fixing them to 0.0.
    for o in O, d in D
        if data[o, d] == "."
            fix(x[o, d], 0.0; force = true)
        end
    end
    @objective(
        model,
        Min,
        sum(data[o, d] * x[o, d] for o in O, d in D if data[o, d] != "."),
    )
    @constraint(model, [o in O], sum(x[o, :]) <= data[o, "SUPPLY"])
    @constraint(model, [d in D], sum(x[:, d]) == data["DEMAND", d])
    optimize!(model)
    # Pretty print the solution in the format of the input
    print("    ", join(lpad.(D, 7, ' ')))
    for o in O
        print("\n", o)
        for d in D
            if isapprox(value(x[o, d]), 0.0; atol = 1e-6)
                print("      .")
            else
                print(" ", lpad(value(x[o, d]), 6, ' '))
            end
        end
    end
    return
end
```

Out[10]:

```
solve_transportation_problem (generic function with 1 method)
```

In [11]:

```
# Решаем задачу:
solve_transportation_problem(data)
```

```
        FRA    DET    LAN    WIN    STL    FRE    LAF
GARY      .      .      .      .  300.0 1100.0      .
CLEV      .      .  600.0      . 1000.0      . 1000.0
```

```
PITT  900.0 1200.0      .  400.0  400.0      .      .
```

Задача 3: Задача о p-медиане

Задача о p-медиане возникает во многих приложениях, например, размещение предприятий бытового обслуживания, складов, пунктов автосервиса на дорогах, коммутаторов в телефонной сети и др. Задача о p-медиане принадлежит классу NP-трудных.

Цель состоит в поиске местоположений p объектов в сети при минимизации затрат. Математическая модель задачи p-медианы:

$$\sum_i \sum_j a_i d_{ij} x_{ij} \to \min$$

$$\sum_j x_{ij} = 1, \quad \forall i,$$

$$x_{ij} - y_j \le 0, \quad \forall i, j$$

$$\sum_{j=1}^{p} y_j = p,$$

$$x_{ij},\ y_j \in \{0; 1\} \quad i = \overline{1, n}, \quad j = \overline{1, p}.$$

Переменные модели:

$$y_j = \begin{cases} 1, & \text{if demand node } i \text{ assigned to facility located at } j, \\ 0, & \text{otherwise;} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{если клиент } i \text{ обслуживается башней в } j\text{-м пункте,} \\ 0, & \text{otherwise.} \end{cases}$$

Данные взяты из статьи: Satman, Mehmet & Akadal, Emre. (2023). Optimization With Julia. 10.26650/B/SS28ET06.2023.006.11.

In [12]:

```julia
using JuMP, GLPK
n = 7 # number of customers
p = 7 # number of facilities
d = [
11 12 29 14 34 28 48;
12 22 33 18 34 12 32;
29 33 33 26 26 40 50;
14 18 26 44 15 30 32;
34 34 26 15 33 15 35;
28 12 40 30 15 22 17;
48 32 50 32 35 17 11
]
model = Model(GLPK.Optimizer)
@variable(model, x[1:n, 1:p], Bin)
@objective(model, Min, sum(d.*x))
for i in 1:n
@constraint(model, sum(x[i, :]) == 1)
end
for j in 1:p
@constraint(model, sum(x[:, j]) == 1)
end
optimize!(model)
println("Result: ", termination_status(model))
println("Minimum Distance: ", objective_value(model))
x_opt = value.(x)
for i in 1:n
j = findfirst(x_opt[i, :] .== 1)
println("Customer $i -> Facility $j")
end
```

```
Result: OPTIMAL
Minimum Distance: 109.0
Customer 1 -> Facility 1
Customer 2 -> Facility 6
Customer 3 -> Facility 3
Customer 4 -> Facility 5
Customer 5 -> Facility 4
Customer 6 -> Facility 2
Customer 7 -> Facility 7
```