E-kursuse " Praktiline optimiseerimine " materjalid

Aine maht 6 EAP

Peep Miidla (Tartu Ülikool), 2012

Praktiline optimiseerimine, ainekava

Aine MTMM.00.306 „Praktiline optimiseerimine“ läbimise eest saab üliõpilane 6 ainepunkti. Oluline rõhk on praktilisel osal, see baseerub rakenduspaketil **MATLAB**. Kursuse omandamiseks vajalik materjal on täielikult kättesaadav Moodle keskkonnas. Kursuse eduka läbimise eelduseks on siiski aktiivne iseseisev töö paketi **MATLAB** sisseehitatud vahendite, abimaterjalide ja demode tundmaõppimisel – ühtki arvutitarkvara ei saa omandada pelgalt õppejõu jutu põhjal. Vastavalt ÕIS-is sätestatule on üliõpilaste iseseisva töö mahuks semestri jooksul 92 tundi. Õpetamine toimub segaõppe (*Blended Learning*) vormis, kus kirjalikud materjalid seletatakse auditooriumis lahti.

Kohustuslike ülesannete lahendused tuleb vormistada paketi **MATLAB** failidena, nn m-failidena, mis on vastavalt vormistusjuhendile varustatud piisavate kommentaaridega. Lahendusi kontrollitakse nii, et kõigepealt avatakse fail toimetiaknas, seejärel, kui olulisi süntaksivigu ei avastata, käivitatakse käsuaknast. Lahenduste üleslaadimise tähtajad on näha Moodle keskkonna vastavate vahendite juures.

Teemade loetelu:

- Rakenduspaketi **MATLAB** tundmaõppimine.
- Optimiseerimisülesande üldine püstitus. Ülesannete liigitus.
- Lineaarse planeerimise ülesanne. Vektor-maatrikskuju. Simpleksmeetod.
- Duaalne lineaarse planeerimise ülesanne.
- Täisarvulise lineaarse planeerimise ülesanne.
- Rändkaupmehe ülesanne. Võrkplaneerimise ülesanne.
- Mittelineaarse planeerimise ülesanded.
- Ruutplaneerimise ülesanne. Lagrange'i kordajate meetod.
- Heuristilised meetodid.
- Sipelgaalgoritm. Geneetilised algoritmid. Tehisnärvivõrgud.
- Esitlusettekanded.
- Teisi vahendeid optimiseerimisülesannete lahendamiseks - ettekannetest.

peep.miidla@ut.ee

The course MTMM.00.306 "Practical Optimization" gives for student six credits. An important focus is on the practical part which is based on **MATLAB** Package. The necessary course material is accessible in the Moodle environment. However, for successful and complete passing of the course active individual work is needed with built-in tools of package **MATLAB**, support materials and demos - no computer software cannot be acquired solely on the basis of teacher's talk. According to the rules, it is provided for the students' independent work within a volume of 92 hours. Instruction is being blended learning in the form of written materials explained in class.

The solutions of mandatory assignments must be recorded as programs of the package **MATLAB**, so-called m-files, which is in accordance with vormistusjuhend and with adequate comments. The solutions tested so that the file will be opened in the Editor, then, if syntax errors are not discovered runned in the Command Window. Solutions for the upload dates are for the respective funds at the Moodle environment. Number of mandatory assignments is 5.

peep.miidla@ut.ee

Eesmärk ja õpiväljundid

Kursuse eesmärgiks on üliõpilastele baasteadmiste andmine tänapäevaste optimiseerimismeetodite tundmaõppimiseks ja rakendamiseks, samuti iseseisva töö oskuste treenimine.

The goal of the course is giving basic knowledge to the students in studing modern methods of optimization and training of skills of individual work.

Peale kursuse läbimist üliõpilane
- Tunneb põhilisi optimiseerimisülesandeid;
- Tunneb optimiseerimisülesannete lahendamise meetodeid;
- Oskab kasutada rakenduspaketti MATLAB;
- Tunneb rakenduspaketi MATLAB lisavahendit *Optimization Toolbox;*
- Omab suuremat vilumust iseseisva töö esitlemiseks;
- Oskab kasutada õpikeskkonda MOODLE.

After passing the course students
- Knows main problems of otimization;
- Knows main methods for solving problems of otimization;
- Can use MATLAB, language of technical computing;
- Knows how to use the *Optimization Toolbox;*
- Has better skills for presentation of individuaal work;
- Knows how to use MOODLE environment.

peep.miidla@ut.ee

Sissejuhatus ainesse „Praktiline optimiseerimine“ Moodle keskkonnas.

Aine MTMM.00.306 „Praktiline optimiseerimine“ läbimise eest saab üliõpilane 6 ainepunkti (EAP). Oluline rõhk on praktilisel osal, see baseerub rakenduspaketil **MATLAB**. Selle osa omandamiseks vajalik materjal on täielikult kättesaadav Moodle’i keskkonnas. Kursuse eduka omandamise ja läbimise eelduseks on aktiivne iseseisev töö paketi **MATLAB** sisseehitatud vahendite, abimaterjalide ja demode tundmaõppimisel – ühtki arvutitarkvara ei saa omandada pelgalt õppejõu jutu põhjal. Vastavalt ÕIS-is sätestatule on üliõpilaste iseseisva töö mahuks semestri jooksul 92 tundi. Õpetamine toimub segaõppe (*Blended Learning*) vormis, kus kirjalikud materjalid seletatakse auditooriumis üksikasjalikult lahti. Kevadel 2012 toimub auditoorne töö esmaspäeviti kell 12-14 arvutikassis Liivi 2 – 003 ja kolmapäeviti kell 16-18 Liivi 2 auditooriumis 402.

Kohustuslike ülesannete lahendused tuleb vormistada paketi **MATLAB** failidena, nn m-failidena, mis on vastavalt vormistusjuhendile varustatud piisavate kommentaaridega. Lahendusi kontrollitakse nii, et kõigepealt avatakse fail toimetiaknas, seejärel, kui olulisi süntaksivigu ei avastata, käivitatakse käsuaknast. Lahenduste üleslaadimise tähtajad on näha Moodle keskkonna vastavate vahendite juures, vaikimisi on need fikseeritud kevadsemestri arvestusliku lõpuga. Kohustuslikke ülesandeid on kokku 2, aga need on sisuliselt iseseisvad uurimisprobleemid.

Kursuse teemade kohta on olemas rohkesti inglisekeelset veebimaterjali. Muude kõrval kasutame John W. Chinneck’i materjale tema loal:

http://www.sce.carleton.ca/faculty/chinneck/po.html

Kursuse teemad:

Tarkvarapaketi **MATLAB** tutvustus. Optimization Toolbox. Lineaarse planeerimise ülesanne. Täisarvulise lineaarse planeerimise ülesanne. Rändkaupmehe ülesanne. Mittelineaarse planeerimise ülesanne. Ruutplaneerimise ülesanne. Heuristilised meetodid.

Peep Miidla, vastutav õppejõud. Liivi 2-419, tel 7375492.

peep.miidla@ut.ee

Esitlused

Aprillis tuleb kõikidel aine kuulajatel teha esitlus. Selle teemad võib valida alltoodud nimistust või valida ise vastavalt oma tööle ja huvidele. Esitluse pikkus on 30-40 minutit, selle põhiosa tuleks vormistada esitlusprogrammiga (*PowerPoint, Impress, Prezi*), lisaks võib kasutada programmipaketti **MATLAB**, veebilehitsejaid jm. Esitluse teema ja kuupäev tuleb valida ja kinnitada õppejõu juures veebruarikuu jooksul.

Alltoodud teemade hulgas on soovitusi teha ülevaateid olemasolevatest veebis leiduvatest optimiseerimisvahenditest. Kui selline valitakse, peab esitlusena valmima juhend vastava veebipaiga võimaluste kohta, kaasa arvatud konkreetsete näiteülesannete lahendused. Soovitatavad ülesanded on lineaarse planeerimise ülesanne (*Linear Programming*) ja rändkaupmehe ülesanne (*Travelling Salesperson*).

- Ülevaade veebipaigast **IBM® ILOG® CPLEX® Optimization Studio, Version 12 Release 2 Information Center**

  http://publib.boulder.ibm.com/infocenter/cosinfoc/v12r2/index.jsp

- Excel Solver

  http://www.solver.com/xlsplatforme.htm

- Ülevaade veebipaigast **TracePro 7.0 Interactive Optimizer**

  www.lambdares.com

- Ülevaade veebipaigast **Monte Carlo Simulation – Introduction**

  http://www.solver.com/simulation/monte-carlo-simulation/index.html

- Ülevaade veebipaigast **Remote Interactive Optimization Testbed**

  http://riot.ieor.berkeley.edu/

- Ülevaade veebipaigast **AMPL, A Mathematical Programming Language**

  http://www.ampl.com/

- Ülevaade veebipaigast **IBM ILOG CPLEX Optimization Studio**

  http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/?S

- Ülevaade veebipaigast **GLPK (GNU Linear Programming Kit)**

  http://www.gnu.org/software/glpk/

- Ülevaade veebipaigast **Clp (Coin-or linear programming)**

https://projects.coin-or.org/Clp

- Ülevaade veebipaigast **Netlib**, optimiseerimise alamhulk

http://www.netlib.org/

- Ülevaade veebipaigast **Gurobi Optimizer**

http://www.gurobi.com/

- Maailma võimsamad arvutid

http://www.top500.org/

- Ülevaade veebipaigast **Solving Constraint Integer Programs SCIP**

http://scip.zib.de/scip.shtml

- Ülevaade veebipaigast **TOMLAB Optimization Environment**

http://tomopt.com/tomlab/

- Ülevaade veebipaigast **OPERA TB - A MATLAB Toolbox for Operational Analysis**

http://user.it.uu.se/~matsh/opt/opera.html

- Travelling Salesman Problem (TSP) art

http://www.cgl.uwaterloo.ca/~csk/projects/tsp/

http://www.tsp.gatech.edu/data/ml/monalisa.html

- Spordiväljaku valgustamise ülesanne. Vt. viidet abifailidele.

- Ülevaade veebipaigast **MProbe an assistant for mathematical programming**

http://www.sce.carleton.ca/faculty/chinneck/mprobe.html

- Interior-Point Methods for linear programming probleem

http://www.neos-guide.org/NEOS/index.php/Interior-Point_Methods

- Seljakoti pakkimise ülesanne, The Knapsack probleem

http://www.win.tue.nl/~wscor/OW/2V300/H4.pdf

- Rändkaupmehe ülesande lahendamine metaheuristilise meetodiga. Valida meetod, mida ei vaadelda kursusel.

peep.miidla@ut.ee

Optimiseerimisülesanne

Optimiseerimisülesande üldine püstitus on järgmine.

On antud hulk $\mathbf{X}$ elementidega $\mathbf{x} \in \mathbf{X}$. Olgu antud funktsioon $\mathbf{f} : \mathbf{X} \to \mathbb{R}$, nõndanimetatud **sihifunktsioon** (*Objective Function*); $\mathbb{R}$ on reaalarvude hulk. Veel olgu antud funktsioonid $\mathbf{g}_i \to \mathbb{R}$, i = 1,2,…,m, niinimetatud **kitsendusfunktsioonid** (*Constraint functions*). Hulka $\mathbf{\Omega} = \{x \in \mathbf{X} \mid \mathbf{g}_i(x) \le 0,\ i = 1,2,\ldots,m\}$ nimetatakse **lubatavate lahendite hulgaks** (*Feasible Set*).

Leida $\max \mathbf{f}(x)$ üle kõigi elementide $\mathbf{x} \in \mathbf{\Omega}$. (*)

Elementi $\mathbf{x}^* \in \mathbf{X}$, mis realiseerib funktsiooni **f** maksimumi lubatavate lahendite hulgal $\mathbf{\Omega}$, nimetatakse optimiseerimisülesande (*) **lahendiks**. Optimiseerimisülesannetel on sageli rohkem kui üks lahend. Mittelahenduvus tähendab seda, et lubatavate lahendite hulk $\mathbf{\Omega}$ on tühi või tõkestamata.

Sõltuvalt sihifunktsiooni **f**, hulga **X**, ja kitsenduste $\mathbf{g}_i$ (i = 1,2,…,m), st. lubatavate lahendite hulga olemusest, nimetatakse ülesannet (*) lineaarseks planeerimisülesandeks, ruutplaneerimise ülesandeks, mittelineaarseks planeerimisülesandeks, poollõpmatuks planeerimisülesandeks, poolmääratud planeerimisülesandeks, mitme sihifunktsiooniga planeerimisülesandeks, diskreetseks planeerimisülesandeks jne. Optimiseerimisülesannete ja –meetodite valdkond on kiiresti arenev, Paralleelselt modifitseerub ka vastav terminoloogia.

Mõnikord formuleeritakse minimiseerimisülesanne, milles nõutakse sihifunktsiooni miinimumi leidmist. See on lihtsasti ja üldisust kahandamata viidav kujule (*): kui mingi element **x** maksimiseerib funktsiooni **f** väärtuse hulgal $\mathbf{\Omega}$, siis minimiseerib seesama element funktsiooni –**f** väärtuse samal hulgal.

Sageli on $\mathbf{X} = \mathbb{R}^n$, see tähendab, et optimiseerimismuutuja **x**, seega ka lubatavad lahendid, on n – mõõtmelised vektorid.

Väga oluliseks optimiseerimisülesannete klassiks on lineaarse planeerimise ülesanne:

Maksimiseerida $\mathbf{c}^T\mathbf{x}$ kitsendustel $\mathbf{Ax} \le \mathbf{b}$; $\mathbf{Bx} = \mathbf{beq}$, $\mathbf{x}_{min} \le \mathbf{x} \le \mathbf{x}_{max}$.

Sel juhul üldises seades (*) on sihifunktsiooniks $\mathbf{f}(x) = \mathbf{c}^T\mathbf{x}$ (vektorite **c** ja **x** skalaarkorrutis) ja lubatavate lahendite hulgaks $\mathbf{\Omega} = \{\mathbf{x} \in \mathbf{X} = \mathbb{R}^n \mid \mathbf{Ax} \le \mathbf{b}, \mathbf{Bx} = \mathbf{beq}, \mathbf{x}_{min} \le \mathbf{x} \le \mathbf{x}_{max}\}$. Algandmeteks on **A** ja **B** - etteantud maatriksid vastavate dimensioonidega (kindlasti on nendes **n** veergu); **b**, **beq**, ja **c** - etteantud vektorid vastavate pikkustega. Võime kirjutada, et lubatavate lahendite hulga $\mathbf{\Omega}$ määravad kitsendusfunktsioonid $(\mathbf{g}_1(\mathbf{x}),\ldots,\mathbf{g}_m(\mathbf{x}))^T = \mathbf{Ax}$, $(\mathbf{h}_1(\mathbf{x}),\ldots,\mathbf{h}_k(\mathbf{x}))^T = \mathbf{Bx}$.

peep.miidla@ut.ee

Veebiviiteid *Links*

The Mathematical Programming Glossary
http://glossary.computing.society.informs.org/
http://glossary.computing.society.informs.org/ver2/mpgwiki/index.php/Main_Page

Linear Programming FAQ
http://www.neos-guide.org/NEOS/index.php/Linear_Programming_FAQ

Nonlinear Programming FAQ
http://www.neos-guide.org/NEOS/index.php/Nonlinear_Programming_FAQ

MPL On-Line Tutorial
http://www.maximal-usa.com/mpltutor/

Practical Optimization: A Gentle Introduction by John W. Chinneck
http://www.sce.carleton.ca/faculty/chinneck/po.html

J. E. Beasley OR-Notes
http://people.brunel.ac.uk/~mastjjb/jeb/or/tutorial.html

Maksimiseeritakse ristküliku, kolmnurga ja kuusnurga ümber joonistatud samasuguse kujundi pindala.
http://www.math.dartmouth.edu/~klbooksite/appfolder/306unit/Optimization.html

Linear Programming: Foundations and Extensions
http://www.princeton.edu/~rvdb/LPbook/

Web Resources for Students of Mathematical Programming, Optimization, and Operations Research
http://www.sce.carleton.ca/faculty/chinneck/StudentOR.html

Operations Research Models and Methods
http://www.me.utexas.edu/~jensen/ORMM/student.html

The Math Forum's Internet Math Library
http://mathforum.org/library/topics/operations_research/

Deterministic Modeling: Linear Optimization with Applications
http://home.ubalt.edu/ntsbarsh/opre640a/partviii.htm

The traveling salesman problem
http://users.encs.concordia.ca/~chvatal/tsp/tsp.html

Harjutusülesandeid

1. Sisestada vabalt valitud vektor pikkusega 5.

(i) Liita vektori igale elemendile 15.

(ii) Liita paarituarvulise indeksiga elementidele 6.

(iii) Leida ruutjuur igast elemendist.

(iv) Leida iga elemendi ruut.

2. Moodustada vektor elementidega

$$x_n = (-1)^{n+1}/(2n-1) .$$

Alustades vektorist n=3 lisada sellele elemente, kuni vektori pikkuseks on 10.

2. Sisestada vabalt valitud elementidega 4x4 maatriks A ning genereerida käsuga **rand** 4x4 maatriks B.

(i) Leida A*B, A/B, A\B, A .*B, A ./B, A .\B.

(ii) Uurida saadud maatrikseid käskudega **flipud**, **fliplr**, **rot90**, **flipdim.**

(iii) Eraldada leitud maatriksite 2. veerud ning moodustada neist käsuga **cat** uus maatriks.

(iv) Saadud maatriks transponeerida ja jätta sellest välja kaks vähima reasummaga rida.

(v) Lahendada lineaarne võrrandisüsteem, mille maatriksiks on eelmises punktis tekkinud maatriks. Vabaliikmed valida ise. Enne lahendamist veenduda, et maatriksi determinant erineb nullist.

3. Sisestada kaks nullidest ja ühtedest koosnevat vektorit , mõlemad 10-elemendilised.

Olgu nendeks näiteks u ja v.

(i) Leida **and**(u,v), **or**(u,v), **xor**(u,v), **not**(u), **not**(v). Selgitada tulemusi.

(ii) Leida u > v, u < v, u == v, (u > v) | (v < u), (u > v) & (v < u). Selgitada tulemusi.

(iii) Leida u > 3, v < 3, u((u < 2) | (v >= 1)), v((u < 2) | (u >= 0)). Selgitada tulemusi.

Ülesandeid tuleb iseseisvalt veebist otsida. Ainult harjutamine teeb meistriks.

peep.*miidla***@ut.ee**

Maatriksid ja vektorid paketis **MATLAB**

Tähtsaim paketi **MATLAB** poolt toetatav andmetüüp on maatriks.

- [1 2 3] või [1, 2, 3] on reavektor;
- [1; 2; 3] on veeruvektor;
- [1 2; 3 4; 5 6] on 3 × 2 maatriks;
- [1:4] on sama mis [1 2 3 4]; koolon tähendab siin „ühest neljani sammuga 1“;
- [1:0.2:2] on sama mis [1 1.2 1.4 1.6 1.8 2]; koolon tähendab „ühest kaheni sammuga 0.2“;
- A’ annab transponeeritud maatriksi;
- A(2, 3) on maatriksi A teise rea kolmanda veeru element;
- A(1, :) on maatriksi A esimene rida vektorina;
- A(2, [1 3]) on vektor [A(2,1), A(2,3)];
- A([1 2], [3 4]) on maatriksi A alammaatriks [A(1,3) A(1,4);A(2,3) A(2,4)] ;
- **ones**(2,3) on ühtedest koosnev 2 × 3 maatriks;
- **zeros**(2,3) on nullidest koosnev 2 × 3 maatriks;
- **eye**(n) on n×n maatriks mille peadiagonaalil on ühed ja mujal nullid;
- **eye**(n,m) on n×m maatriks mille diagonaalil on ühed ja mujal nullid;
- **ones**(A) on ühtedest koosnev maatriks mille dimensioonid võrduvad maatriksi A omadega;
- **diag**(A) - kui A on etteantud maatriks, saame vektori, mille elementideks on maatriksi A peadiagonaali elemendid;
- **diag**(v) - kui v on etteantud vektor, saame maatriksi, mille peadiagonaalil on vektori v elemendid, ülejäänud nullid;
- **diag**(v,k) - kui v on etteantud vektor, saame maatriksi, mille peadiagonaalist k võrra üles (k > 0) või alla (k < 0) nihutatud kõrvaldiagonaalil on vektori v elemendid, ülejäänud nullid;
- A*B on maatriksite A ja B maatrikskorrutis;

- A.*B on maatriksite A ja B korrutis vastavate elementide kaupa;
- A+B on maatriksite A ja B summa;
- A-B on maatriksite A ja B vahe;
- 2*A on maatriks A, mille elementideks on maatriksi A kahekordsed;
- A + 3 liidab maatriksi A igale elemendile arvu 3;
- **sum**(A) annab maatriksi A elementide summa;
- **max**(A) annab maatriksi A maksimaalse elemendi;
- **sin**(A) leiab maatriksi A iga elemendi siinuse;
- **inv**(A) on maatriksi A pöördmaatriks;
- **norm**(A,p) annab maatriksi A p – normi;
- A/B on maatriks $A*B^{-1}$ (kui pöördmaatriks B^{-1} eksisteerib);
- B \A on maatriks B^{-1} *A (kui pöördmaatriks B^{-1} eksisteerib);
- A./B on maatriks, mille elementideks on vastavate elementide jagatised;
- B.\A on maatriks, elementideks vastavate elementide pöördjagatised;
- **expm**(A) arvutab maatriksi A eksponendi;
- **logm**(A) leiab maatriksi A naturaallogaritmi, A = **expm**(**logm**(A));
- **sqrtm**(A) – ruutjuur maatriksist, A = **sqrt**(A)***sqrt**(A);

Maatriksi elemendid sisestatakse ridade kaupa nurksulgudesse. Reaelemendid eraldatakse üksteisest komade või tühikutega ning read semikoolonitega.

```
>>A=[1 2,3;4,5 6;]
 A  =

    1.    2.    3.
    4.    5.    6.
```

Maatrikseid saab kasutada teiste maatriksite elementidena. Sel juhul peavad alammaatriksite ridade arvud ühtima:

```
>>C=[A,B]
```

```
C  =

    1.    2.    3.    6.     6.
    4.    5.    6.    15.    15.
```

Juhuslike arvudega maatriksi genereerimiseks on pakettides käsk rand(m,n), kus m on maatriksi ridade ja n veergude arv:

```
>>rand(5,2)
 ans  =

    0.5608486   0.2320748
    0.6623569   0.2312237
    0.7263507   0.2164633
    0.1985144   0.8833888
    0.5442573   0.6525135
```

Veel näiteid sisseehitatud funktsioonide kasutamise kohta.

```
>>Z = zeros(2,4)
Z =
0 0 0 0
0 0 0 0
>>F = 5*ones(3,3)
F =
5 5 5
5 5 5
5 5 5
>>N = fix(10*rand(1,10))
N =
9    2    6    4    8    7    4    0    8    4
```

```
>>B=[ones(2,3),zeros(2,3)]
 B  =

    1.    1.    1.    0.    0.    0.
    1.    1.    1.    0.    0.    0.
```

peep.*miidla****@ut.ee***

Sissejuhatus, pakett **MATLAB**

õpijuhis

Hädavajalikud teadmised. Peab teadma, kuidas

- käivitada paketti **MATLAB**, **MAT**rix **LAB**oratory;
- defineerida muutujaid ja nendega arvutusoperatsioone sooritada;
- kasutada käskude sisestamisel semikoolonit;
- esitada arve erinevates formaatides;
- kasutada sisseehitatud konstante (**ans**, **eps**, **pi** , **realmax**, **realmin**, **Inf**, **NaN**);
- kasutada paketi erinevaid aknaid;
- saada abi käsuakna kaudu (**help**, **doc**, **lookfor**);
- kasutada abiakent;
- toimetada tööpiirkonnas (**Workspace**, **clear**, **who**, **whos**);
- sisestada vektoreid ja maatrikseid, s.h. kooloni kasutamisega;
- sisestada avaldisi, ka mitut ühel real ning ühte mitmel real;
- eraldada massiividest elemente ja alammassiive indeksite ja kooloni abil;
- kasutada vektorite loomiseks käske **linspace** ja **logspace**;
- luua maatrikseid sisseehitatud käskude abil (**diag**, **eye**, **ones**, **rand**, **zeros**);
- kasutada transponeerimisoperaatorit;
- kasutada tavalisi (*, /, ^) ja elementidekaupa (.*, ./, .^) operatsioone;
- seadistada töökausta;
- lugeda sisse andmeid failist (**load**, **fopen**, **fscanf**);

- salvestada andmeid (**save**, **fclose**);

- kasutada sisseehitatud funktsioone;

- tippida programmidesse ja käskude osadena kommentaare (%);

- töötada polünoomidega (**conv**, **deconv**, **poly**, **polder**, **polyval**, **polyfit**, **roots**);

- teha andmete graafikuid (**plot**, **subplot**, **loglog**, **semilogx**, **semilogy**, **axis**, **grid**, **gtext**, **legend**, **text**, **xlabel**, **ylabel**, **title**);

- luua ja muuta m-faile (skripte ja funktsioone);

- kirjutada **for** ja **while** tsükleid;

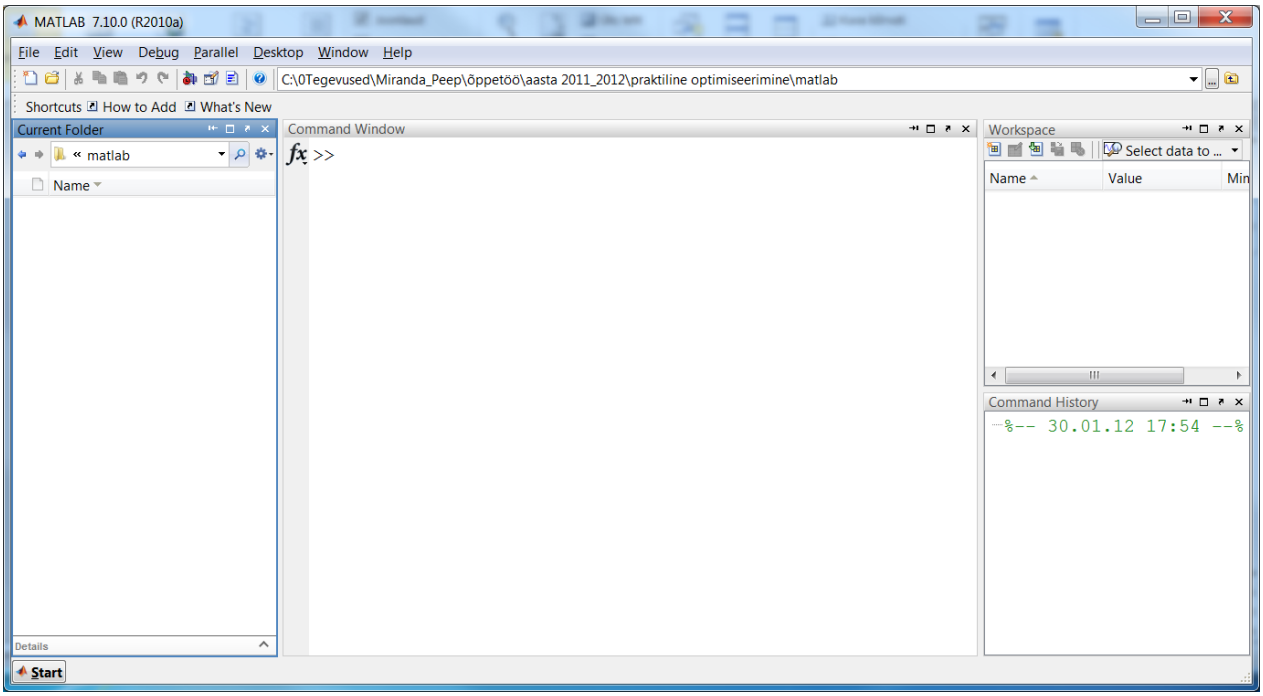
- kirjutada tingimuslikke konstruktsioone: **if**...**end**, **if**...**elseif**...**end**, **if**... **elseif**... **else**...**end**.

peep.miidla@ut.ee

Programmipakett MATLAB

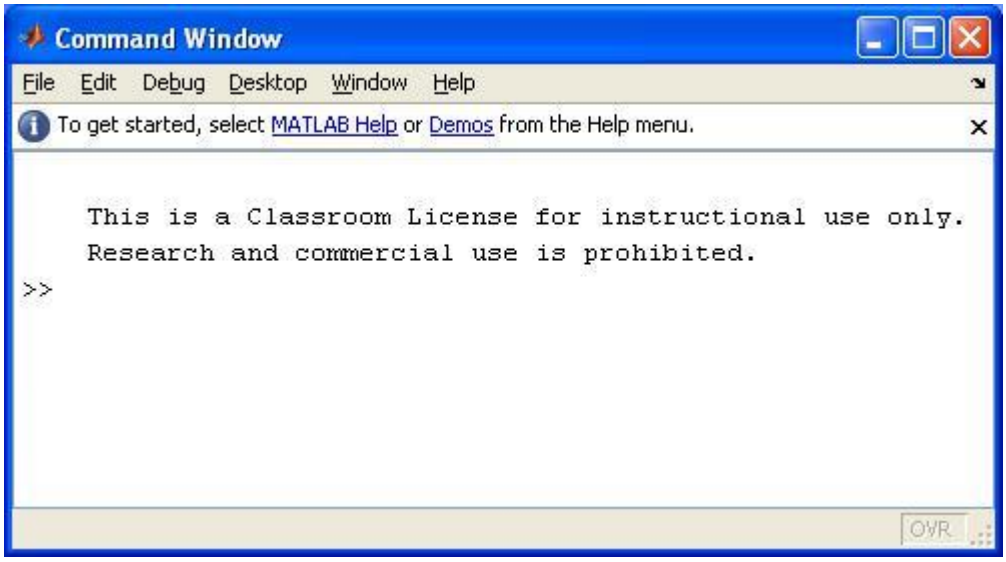
Esimene vaade

Töölaud, mis ilmub pärast programmi käivitamist:



Töölaual on vaikimisi käivitusrezhiimis näha neli akent: käsuaken (*command window*), käskude ajaloo aken (*command history*), töökausta aken (*current directory*) ja mälupiirkonna aken (*workspace*). Kõiki neid saab vastava akna päisriba parempoolses otsas asuvale noolele klõpsamise järel töölaualt lahti ankurdada, st. vastavalt kasutaja soovile eraldiseisvaiks muuta. Kõikide akende struktuur on sama, nagu teistelgi WINDOWS keskkonna omadel. Töölaua alumises vasakus nurgas on paketi **MATLAB** *start*-nupp, mis avab harjumuspärase rippmenüü.

Käsuaken, mille viibalt sisestatakse tegevused (korraldused):

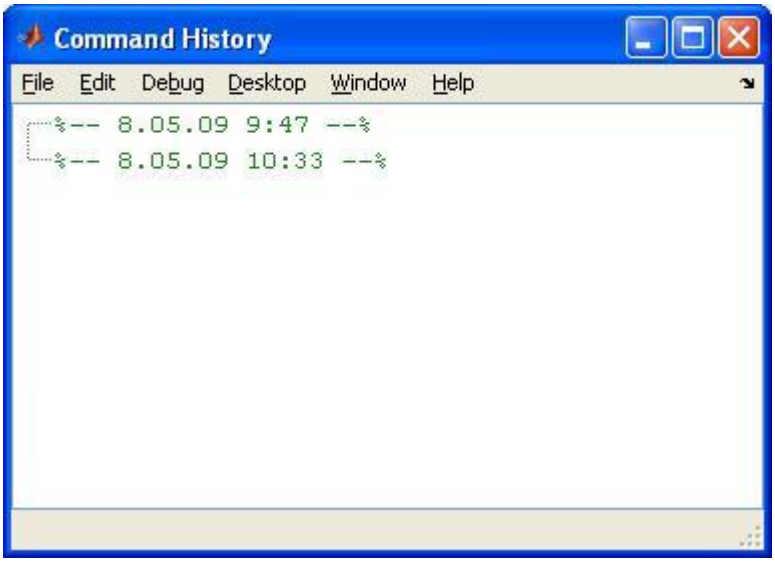


Muutujate nimed ehk identifikaatorid peavad algama tähega, ei tohi sisaldada tühikuid ja täpitähti, samuti mõningaid teisi sümboleid. Parem on nimedes kasutada ainult tähti ja numbreid.

Muutujate nimedes on tähed tõstutustundlikud, st. suured ja väikesed tähed eristatakse.

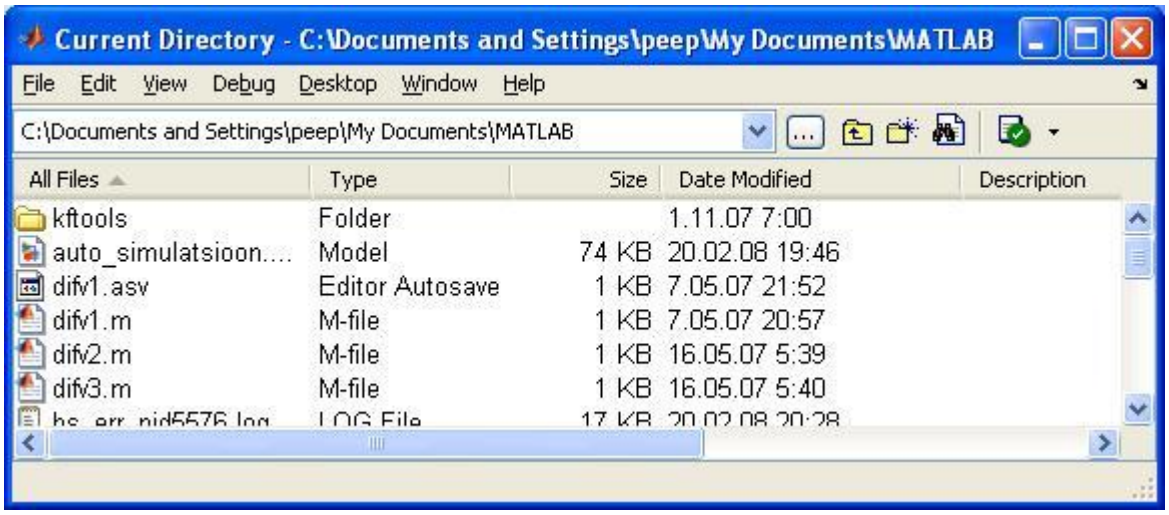
Tagasiankurdusnool on kuju muutnud ja asub menüüriba parempoolses otsas.

Käskude ajaloo aken:



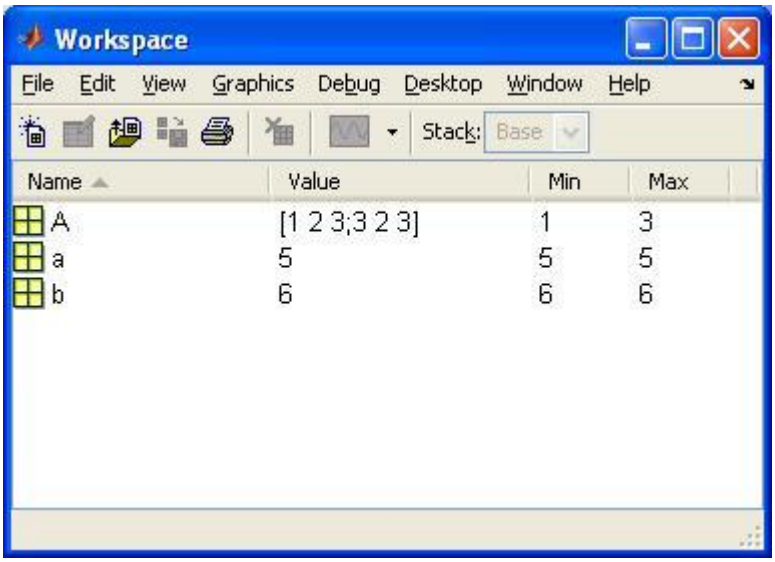
Siin on näha kasutajaprofiilis sisestatud tegevused käsuaknasse. Infot on mugav kasutada uuel seansil eelmiste tegevuste meeldetuletamiseks ja taastamiseks (*copy – paste*).

Töökausta aken:



Selles on näha töökausta sisu. Selles olevaid programme ja funktsioone (m-faile), mille on loonud kasutaja, saab käsuaknast käivitada. Töökausta salvestatakse ka toimeti autosalvestuse failid, mälupiirkonna (*workspace*) seis, joonised, algandmed jne.

Mälupiirkonna aken:



Mälupiirkonna (*workspace*) aknas on näha käimasoleval sessioonil kasutusele võetud globaalsete muutujate nimistu, iseloomustus ja väärtused (kui võimalik). Mälupiirkond on programmi **MATLAB** käivitamisel tühi ja see tühjendatakse paketi sulgemisel.

peep.miidla@ut.ee

Võrrandite lahendamine paketi **MATLAB** abi.

Olgu vaja leida järgmise võrrandi lahend (lahendid):

$$f(x) = 0 . \qquad (1)$$

Siin f on etteantud ühemuutuja funktsioon. Toodud ülesannet nimetatakse veel funktsiooni f nullkohtade (ka juurte) leidmise ülesandeks. Tähistame sümboliga x^* võrrandi (1) täpse lahendi.

Selle lahendamiseks pole üheselt määratud üldist võtet. Enne võrrandi (1) numbrilise lahendamise juurde asumist tuleks leida vastus järgmistele üldistele küsimustele:

- Kas funktsiooni f väärtused on kergesti leitavad?

- Millised on nõuded lähislahendi täpsusele?

- Kas on ette teadaolevaid nõudeid lahendusalgoritmile (kiirus, stabiilsus jne.)?

- Kas f on polünoom?

- Kas funktsioonil f on iseärasusi?

- Millised on veel teada olevad funktsiooni f kvalitatiivsed omadused?

Võrrandi (1) lahendamise üldine strateegia:

- Joonistada funktsiooni f graafik. Selle alusal saab lokaliseerida funktsiooni f nullkoha ja sageli valida iteratsioonimeetodeile vajaliku alglähendi.

- Valida või täpsustada alglähend.

- Täpsustada valitud iteratsioonimeetodiga alglähendit vajaliku täpsuse saavutamiseni.

Alglähendi täpsustamine lõigu jagamise meetodil.

Esmalt tuleb määrata (näiteks graafiku alusel, seda analüütiliselt kontrollides) lõik [a,b], mille otstes funktsioon f on erinevate märkidega, st. f(a)*f(b) < 0. Kui f on pidev

funktsioon, sisaldab lõik [a,b] vähemalt ühe funktsiooni f nullkoha (milline matemaatilise analüüsi tulemus selle garanteerib?). Järgnevalt lõigu jagamise meetodi pseudokood.

Antud: f, a, b, n, funktsiooni f väärtustamise algoritm (m-fail, otsedefinitsioon).

```
dx = (b − a)/n
xvasak = a
for i = 1:n
xparem = xvasak + dx
    if {f(x) muudab märki lõigus [xvasak, xparem]} % Erinevad kontrollivõimalused.
        {salvestada [xvasak, xparem] edasiseks lahendamiseks}
    end
xvasak = xparem
end
```

Funktsiooni f märgi muutumise kontrollimiseks kaks põhilist võimalust:

- **if** f(xvasak)*f(xparem) < 0

- **if sign**(f(vasak)) ~= **sign**(f(parem)) % Loogiline eitus: ~= .

Järgnevalt tuleb valida iteratsioonimeetod, kas harilik või Newtoni iteratsioonimeet|od ja koostada m-fail, mis lahendab ülesande, st. leiab lähendite jada {x(0), x(1),…,x(m)}, mis koonduks võrrandi (1) täpseks lahendiks x^* protsessis m → ∞. Tulemust kontrollida paketi MATLAB sisseehitatud vahenditega. Järgnevalt ülevaade nendest.

fzero leiab ühemuutuja funktsiooni nullkoha.

Näide.

```
>>g = inline('t^2-4')
g =
    Inline function:
    g(t) = t^2-4
>>fzero(g,1)
```

ans =

2

roots leiab polünoomi nullkohad (juured).

Näide:

>>p = [3 2 0 1]; % Sisestatakse polünoomi **p(x) = 3x^3 + 2x^2 – 1** kordajad**.**

>> r = **roots**(p) % Polünoomi nullkohad, mille kordajad on massiivis p.

r =

-1.0000

0.1667 + 0.5528i

0.1667 - 0.5528i

abs reaalarvu absoluutväärtuse või kompleksarvu mooduli leidmine.

inline luuakse objekt käsurealt (*inline object*)**.** Näide ülalpool.

feval leiab stringina defineeritud funktsiooni väärtuse.

Näide:

>> funktsioon=@(t)t^2-4

funktsioon =

@(t)t^2-4

>> **feval**(funktsioon, 3)

ans =

5

poly leiab juurte alusel polünoomi kordajad.

Näide:

>> **poly**([1,2])

ans =

1 -3 2

polyval leiab polünoomi väärtuse.

Näide:

```
>>polyval(p,6)  %  Vektor p on defineeritud ülalpool.
ans =
        33
```

Tegelikult piirprotsessi $m \to \infty$ muidugi ei realiseerita, vaid peatutakse lõpliku m korral ning loetakse $x(m) \approx x^*$. Iteratsioonimeetodi peatumistingimuse pidev kontroll, st. arvu m avastamine garanteerib, et ei sooritata üleliigseid arvutusi vajaliku täpsuse saavutamiseks ja selle võib formuleerida kahel viisil:

- Pärast lähendi x(k) arvutamist kontrollida, kas kaks viimast lähendit on üksteisele juba nii lähedal, et neid võib etteantud täpsuse δ piires samastada, st. kontrollida, kas

$$| x(k) - x(k-1) | < \delta \ .$$

Kui see võrratus on rahuldatud, võib võtta $x(m) = x(k)$, $x(m) \approx x^*$.

- Pärast lähendi x(k) arvutamist kontrollida, kas funktsiooni f väärtus f(x(k)) on nullile piisavalt lähedal, et lähendi x(k) võiks lähislahendiks võtta:

$$| f(x(k))| < \varepsilon \ ?$$

Kui see võrratus on rahuldatud, võib võtta $x(m) = x(k)$, $x(m) \approx x^*$. Viimast peatumise kontrolli meetodit nimetatakse ka peatumiseks hälbe järgi. Mõlemad esitatud võtted on kasutatavad nii hariliku kui ka Newtoni iteratsioonimeetodi korral.

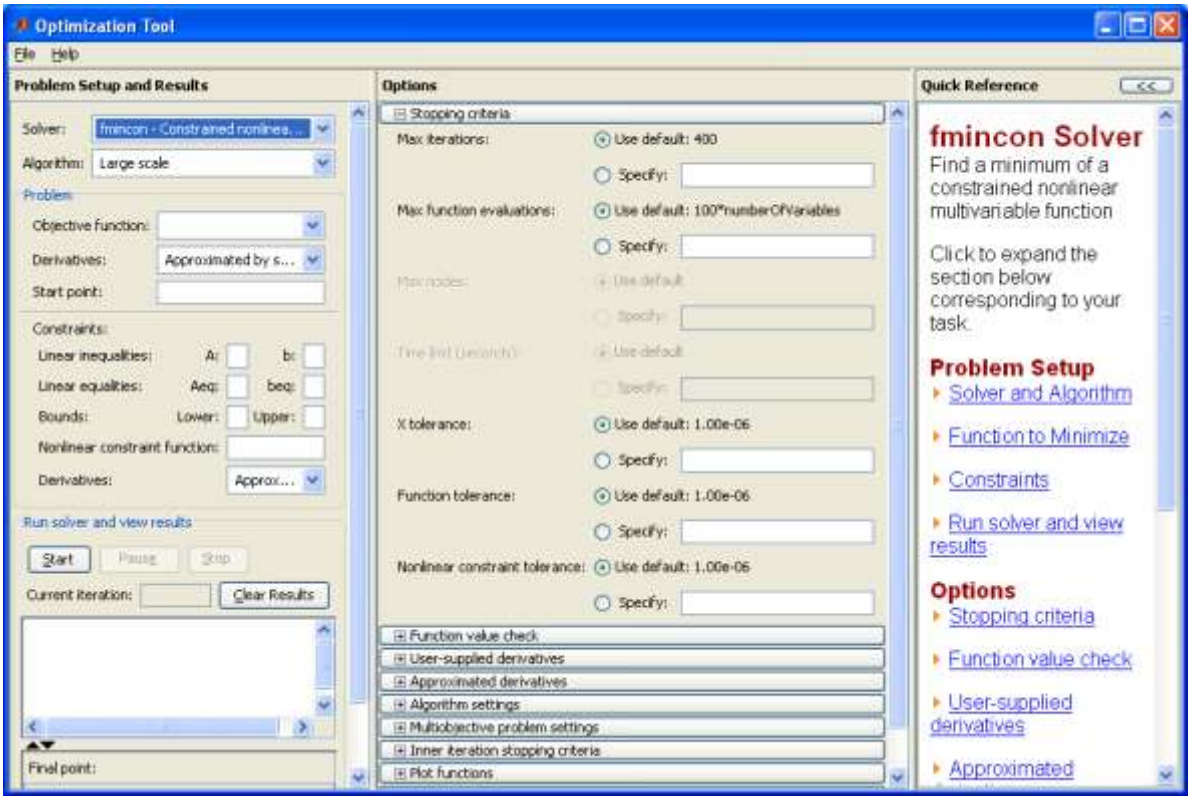
peep.miidla@ut.ee

MATLAB, optimiseerimise *Toolbox*

Optimiseerimisvahendi (*Optimization Toolbox*) on graafilise kasutajaliidese tüüpi abiline ja selle avamiseks tuleb esmalt käivitada programmipakett **MATLAB**, siis paketi töölaua *Start* nupust:

Start → Toolboxes → Optimization → Optimization Tool (optimtool) .

Aken, mis avaneb, on selline:



Kolmest veerust vasakpoolne on ülesande sisestamiseks (*Problem Setup and Results*), teine mitmesuguste lahendusparameetrite etteandmiseks (*Options*) ja kolmas – abiaken (*Quick Reference*). Olulisim on vasakpoolne, mis määrab ülesande tüübi ja kust sisestatakse algandmed. Optimiseerimismeetodid on paketis **MATLAB** realiseeritud sisseehitatud funktsioonidena ja neid saab valida dialoogiaknas „Solver" sisestusriba paremal äärel oleva noole abil. Need funktsioonid ja seega ka vastavad ülesanded, mida kõnesoleva vahendiga saab lahendada, on järgmised.

Lineaarse ja ruutplaneerimise ülesanded (*Linear and Quadratic Minimization problems*).

linprog - lineaarse planeerimise ülesanne (*Linear programming*).

quadprog - ruutplaneerimise ülesanne (*Quadratic programming*).

Võrrandite ja süsteemide lahendamine (*Nonlinear zero finding, equation solving*).

fzero – skalaarse funktsiooni nullkoha leidmine (*Scalar nonlinear zero finding*).
fsolve – mittelineaarse võrrandisüsteemi lahendamine (*Nonlinear system of equations solve, function solve*).

Lineaarne vähimruutude meetod (*Linear least squares*).

lsqlin - lineaarne vähimruutude meetod lineaarsete kitsendustega (*Linear least squares with linear constraints*).
lsqnonneg - lineaarne vähimruutude meetod lahendite mittenegatiivsuse nõudega (*Linear least squares with nonnegativity constraints*).

Funktsioonide minimiseerimine (*Nonlinear minimization of functions*).

fminbnd – skalaarse tõkestatud funktsiooni miinimumi leidmine (*Scalar bounded nonlinear function minimization*).
fmincon – mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained nonlinear minimization*).
fminsearch – mitmemõõtmeline kitsendusteta minimiseerimine Melder- Nead'i meetodil (*Multidimensional unconstrained nonlinear minimization, by Nelder-Mead direct search method*).
fminunc - mitmemõõtmeline kitsendusteta minimiseerimine (*Multidimensional unconstrained nonlinear minimization*).
fseminf - mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained minimization, semi-infinite constraints*).

Mittelineaarne vähimruutude meetod (*Nonlinear least squares*).

lsqcurvefit – mittelineaarsete funktsioonide lähendamine vähimruutude meetodil (*Nonlinear curvefitting via least squares*).
lsqnonlin - mittelineaarne vähimruutude meetod (Nonlinear least squares with upper and lower bounds).

Mitme sihifunktsiooniga miinimumi leidmine (*Nonlinear minimization of multi-objective functions*).

fgoalattain - mitme sihifunktsiooniga miinimumi leidmine (*Multidimensional goal attainment optimization*).
fminimax – mitmemõõtmelised minimax ülesanded (*Multidimensional minimax optimization*).

Abiaken (*Help*), demovahendid (*Demo*) ja paljud muud materjalid aitavad täita ülejäänud lahtreid. Peamised optimiseerimisülesanded leiavad käsitlemist auditoorse töö käigus.

peep.miidla@ut.ee

MATLAB, *Optimization Toolbox*

Lineaarse planeerimise ülesande lahendamine

Optimization Toolbox'i jaoks esitatakse lineaarse planeerimise ülesanne kujul:

Maksimiseerida $\mathbf{f}^T\mathbf{x}$ kitsendustel $\mathbf{Ax} \leq \mathbf{b}$; $\mathbf{Bx} = \mathbf{beq}$, $\mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}$.

Sel juhul on sihifunktsiooniks $\mathbf{f}^T\mathbf{x}$, vektorite **f** ja **x** skalaarkorrutis ja lubatavate lahendite hulgaks $\mathbf{\Omega} = \{\mathbf{x} \in \mathbf{X} = \mathbb{R}^n \mid \mathbf{Ax} \leq \mathbf{b}, \mathbf{Bx} = \mathbf{beq}, \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}\}$.

Algandmeteks on:

f – sihifunktsiooni kordajate vektor;

A - etteantud maatriks, m rida ja n veergu; määrab kitsendused võrratuste kujul, $\mathbf{Ax} \leq \mathbf{b}$;
b - etteantud veeruvektor pikkusega m; määrab kitsendused võrratuste kujul, $\mathbf{Ax} \leq \mathbf{b}$;

B - etteantud maatriks, k rida ja n veergu; määrab kitsendused võrduste kujul, $\mathbf{Bx} = \mathbf{beq}$;
beq - etteantud veeruvektor pikkusega k; määrab kitsendused võrduste kujul, $\mathbf{Bx} = \mathbf{beq}$;

Tõkked $\mathbf{x}_{min}$ ja $\mathbf{x}_{max}$, vastavalt *Lower* ja *Upper*, optimiseeritavale muutujale **x**.

Algandmed võivad olla sisestatud nii kasutajaliidese akendesse kui tööpiirkonda (*Workspace*). Viimasel juhul antakse kasutajaliideses ette vastavate massiivide identifikaatorid. Lineaarse planeerimise ülesande lahendab paketi funktsioon **linprog**, mille kasutamise lihtsaim süntaks on selline:

x = **linprog**(f,A,b).

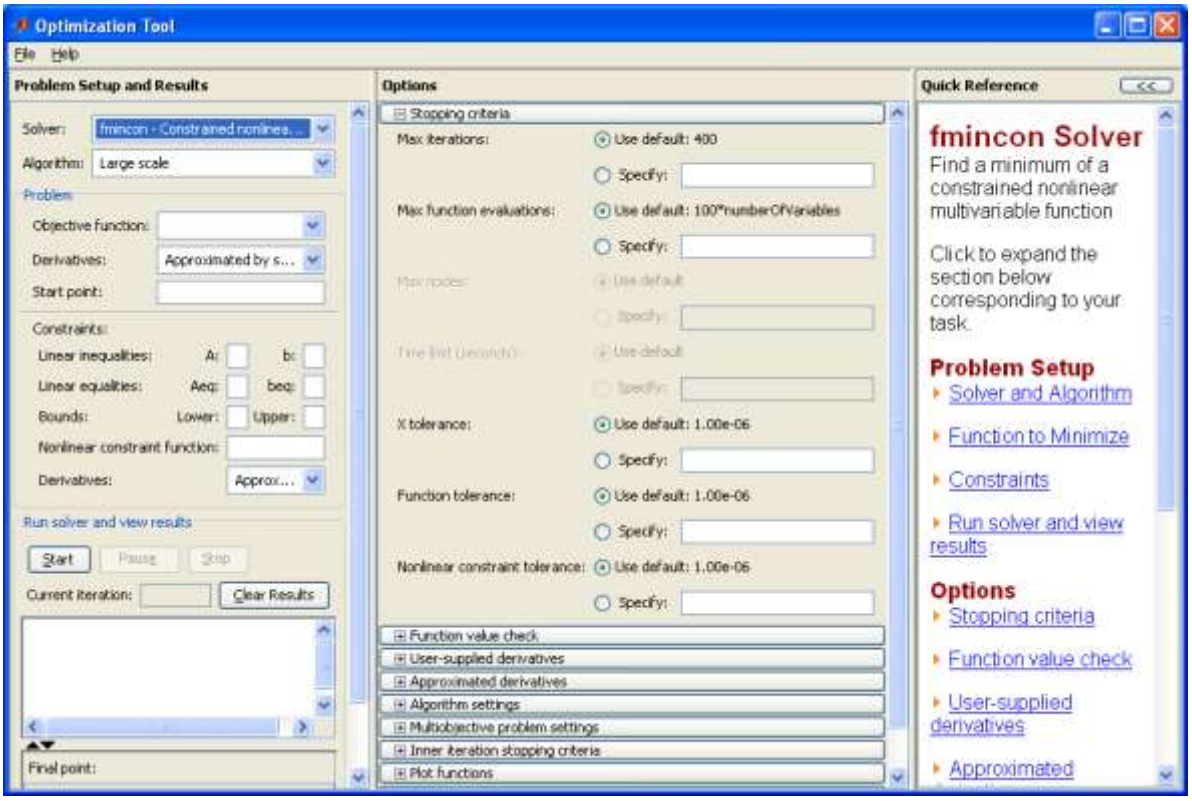
peep.miidla@ut.ee

MATLAB, optimiseerimise *Toolbox*

Optimiseerimisvahendi (*Optimization Toolbox*) on graafilise kasutajaliidese tüüpi abiline ja selle avamiseks tuleb esmalt käivitada programmipakett **MATLAB**, siis paketi töölaua *Start* nupust:

Start → Toolboxes → Optimization → Optimization Tool (optimtool) .

Aken, mis avaneb, on selline:



Kolmest veerust vasakpoolne on ülesande sisestamiseks (*Problem Setup and Results*), teine mitmesuguste lahendusparameetrite etteandmiseks (*Options*) ja kolmas – abiaken (*Quick Reference*). Olulisim on vasakpoolne, mis määrab ülesande tüübi ja kust sisestatakse algandmed. Optimiseerimismeetodid on paketis **MATLAB** realiseeritud sisseehitatud funktsioonidena ja neid saab valida dialoogiaknas „Solver" sisestusriba paremal äärel oleva noole abil. Need funktsioonid ja seega ka vastavad ülesanded, mida kõnesoleva vahendiga saab lahendada, on järgmised.

Lineaarse ja ruutplaneerimise ülesanded (*Linear and Quadratic Minimization problems*).

linprog - lineaarse planeerimise ülesanne (*Linear programming*).

quadprog - ruutplaneerimise ülesanne (*Quadratic programming*).

Võrrandite ja süsteemide lahendamine (*Nonlinear zero finding, equation solving*).

fzero – skalaarse funktsiooni nullkoha leidmine (*Scalar nonlinear zero finding*).
fsolve – mittelineaarse võrrandisüsteemi lahendamine (*Nonlinear system of equations solve, function solve*).

Lineaarne vähimruutude meetod (*Linear least squares*).

lsqlin - lineaarne vähimruutude meetod lineaarsete kitsendustega (*Linear least squares with linear constraints*).
lsqnonneg - lineaarne vähimruutude meetod lahendite mittenegatiivsuse nõudega (*Linear least squares with nonnegativity constraints*).

Funktsioonide minimiseerimine (*Nonlinear minimization of functions*).

fminbnd – skalaarse tõkestatud funktsiooni miinimumi leidmine (*Scalar bounded nonlinear function minimization*).
fmincon – mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained nonlinear minimization*).
fminsearch – mitmemõõtmeline kitsendusteta minimiseerimine Melder- Nead'i meetodil (*Multidimensional unconstrained nonlinear minimization, by Nelder-Mead direct search method*).
fminunc - mitmemõõtmeline kitsendusteta minimiseerimine (*Multidimensional unconstrained nonlinear minimization*).
fseminf - mitmemõõtmeline kitsendustega minimiseerimine (*Multidimensional constrained minimization, semi-infinite constraints*).

Mittelineaarne vähimruutude meetod (*Nonlinear least squares*).

lsqcurvefit – mittelineaarsete funktsioonide lähendamine vähimruutude meetodil (*Nonlinear curvefitting via least squares*).
lsqnonlin - mittelineaarne vähimruutude meetod (Nonlinear least squares with upper and lower bounds).

Mitme sihifunktsiooniga miinimumi leidmine (*Nonlinear minimization of multi-objective functions*).

fgoalattain - mitme sihifunktsiooniga miinimumi leidmine (*Multidimensional goal attainment optimization*).
fminimax – mitmemõõtmelised minimax ülesanded (*Multidimensional minimax optimization*).

Abiaken (*Help*), demovahendid (*Demo*) ja paljud muud materjalid aitavad täita ülejäänud lahtreid. Peamised optimiseerimisülesanded leiavad käsitlemist auditoorse töö käigus.

peep.miidla@ut.ee

Duaalne lineaarse planeerimise ülesanne

Iga LP ülesandega saab seostada ühe teise ülesande, mis kasutab samu andmeid, aga teist muutujate komplekti. See on (esialgsega) duaalne LP ülesanne (*dual Linear Programming Problem*).

(LP) $\mathbf{max\ p := c_1x_1 + c_2x_2 + ... + c_nx_n = c^Tx}$

Kitsendustel *subject to*

$$\mathbf{a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \le b_i\ ,\ i = 1,2, ...\ , m,}\quad \text{ehk } or \quad \mathbf{Ax \le b\ ;}$$

$$\mathbf{x_j \ge 0\ ,\ j = 1,2, ...\ , n\ .}$$

Siin $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2,\ldots,\mathbf{x}_n)^T$ on otsustusmuutujate vektor (*vector of decision variables whose values the decision maker can control, subject to constraints and bound restrictions on them*). Kitsendused määravad (LP) lubatavate lahendite hulga (*set of feasible solutions*) $\mathbf{\Omega = \{x \in X = \mathbb{R}^n \mid Ax \le b,\quad x \ge 0\ \}}$. Lubatavat lahendit $\mathbf{x}^*$, mis realiseerib sihifunktsiooni $\mathbf{c^Tx}$ maksimumi, nimetatakse (LP) optimaalseks lahendiks (*optimum solution*). Sihifunktsiooni kordajad $\mathbf{c}_1, \mathbf{c}_2,\ldots,\mathbf{c}_n$ näitavad, kui palju suureneb sihifunktsiooni väärtus, kui vastava otsustusmuutuja väärtus suureneb ühiku võrra.

(DLP) $\mathbf{min\ q := b_1y_1 + b_2y_2 + ... + b_my_m = b^Ty\ .}$

Kitsendustel *subject to* :

$$\mathbf{a_{1j}\ y_1 + a_{2j}y_2 + ... + a_{mj}y_m \ge c_j\ ,\ j = 1,2, ...\ , n\ ,}\quad \text{ehk } or \quad \mathbf{A^Ty \ge c\ ;}$$

$$\mathbf{y_j \ge 0}\ ,\ \mathrm{j} = \mathbf{1,2, ...\ , m\ .}$$

Siin (DLP) puhul on otsustusmuutujate vektoriks $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2,\ldots,\mathbf{y}_m)^T$ (*vector of decision variables*). Kitsendused määravad (DLP) lubatavate lahendite hulga (*set of feasible solutions*) $\mathbf{\Omega_D = \{y \in \mathbb{R}^m \mid A^Ty \ge c,\quad y \ge 0\ \}}$. Lubatavat lahendit $\mathbf{y}^*$, mis realiseerib sihifunktsiooni $\mathbf{b^Ty}$ maksimumi, nimetatakse (DLP) optimaalseks lahendiks (*optimum solution*).

Ülesandega (DLP) duaalseks ülesandeks on (LP), seega (LP) ja (DLP) on vastastikku duaalsed.

Kehtivad järgmised tulemused,

(i) Lahenduva LP optimaalsetest lahenditest vähemalt üks asub lubatavate lahendite hulga mingis tipus. Kuna (LP) puhul on lubatavate lahendite tippe lõplik hulk, siis meetod, mis vaatleb ainult lubatavate lahendite hulga tippe, on lõplik. Selline on simpleksmeetod.

(ii) Kui ühel duaalsetest ülesannetest (LP) või (DLP) on olemas optimaalne lahend, vastavalt kas $\mathbf{x}^*$ või $\mathbf{y}^*$, siis on see olemas ka teisel, kusjuures sihifunktsioonide ekstreemsed väärtused on võrdsed,

$$\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^* .$$

(iii) Selleks, et ülesandel (LP) oleks olemas optimaalne lahend, on tarvilik ja piisav, et nii vaadeldaval, kui ka temaga duaalsel ülesandel leiduks vähemalt üks lubatav lahend.

(iv) (LP) ja (DLP) mistahes lubatavate lahendite **x** ja **y** puhul kehtib võrratus

$$\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y} .$$

(v) (DLP) optimaalse lahendi $\mathbf{y}^*$ komponendi $\mathbf{y}_i^*$ väärtus näitab, kui palju muutub sihifunktsiooni väärtus lähteülesande (LP) vastava parema poole $\mathbf{b}_i$ muutmisel 1 võrra eeldusel, et optimaalne baas jääb samaks.

Seega näitavad duaalsete muutujate optimaalsed väärtused varjatud tulu, mida on võimalik saada vastava ressursi koguse suurendamisel 1 võrra.

Kaasik Ü., Kivistik L., Operatsioonianalüüs, Valgus, Tallinn, 1982.

peep.miidla@ut.ee

Jalgrattad.

Lineaarse planeerimiülesande näitena vaatleme jalgrattaid tootvat kompaniid **Acme**. See toodab kahte sorti sõiduvahendeid – mägi- ja maanteejalgrattaid.

Acme tahab määrata kindlaks, millises koguses kumbagi toota, et müügitulu oleks maksimaalne.

Acme eeldab, et suudab kõik jalgrattad maha müüa.

Andmed tootmisprotsessi kohta on järgmised. Erinevaid mudeleid tootvate meeskondade suutlikkus on erinev: päevane toodang võib olla kuni kaks mägijalgratast ja 3 maanteeratast päevas.
Toodangu pudelikaelaks on värvimisosakond. Seal kulub iga jalgratta viimistlemiseks sama aeg, sõltumata ratta tüübist ja see suudab läbi lasta maksimaalselt neli jalgratast päevas.
Arvepidamine näitab, et ühelt mägijalgrattalt saadakse müügitulu 15$ ja maanteejalgrattalt 10$.
Formuleerime selle probleemi lineaarse planeerimise ülesandena.

Esmalt määrame muutujad. Nendeks on suurused, mida muuta/juhtida tahetakse: päevas toodetavate mägijalgrataste arv x_1 ja maanteejalgrataste arv x_2. Kohe on selge nende mittenegatiivsuse nõue:

$$x_1 \geq 0 \text{ ja } x_2 \geq 0.$$

Fikseerime sihifunktsiooni (nõudeks oli maksimiseerida müügitulu):

$$\max z = 15x_1 + 10x_2.$$

Kitsendused tootmisprotsessi kohta on sellised:

$$x_1 \leq 2 \text{ (mägijalgrataste päevane väljalase);}$$
$$x_2 \leq 3 \text{ (maanteejalgrataste päevane väljalase);}$$
$$x_1 + x_2 \leq 4 \text{ (värviosakonna päevane suutlikkus.}$$

Vektor – maatrikskujul same selle ülesande kirja panna nii:

$$\max \mathbf{c}^T\mathbf{x} \quad \text{kitsendustel} \quad \mathbf{Ax} \leq \mathbf{b},$$

kus $\mathbf{c} = (15, 10)^T$, $\mathbf{b} = (2, 3, 4)^T$, A =[1, 0; 0, 1; 1, 1], $\mathbf{x} = (x_1, x_2)^T$.

Lahendame selle ülesande simpleksmeetodil.

SAMM 1. Teisendame LPÜ lineaarseks võrrandisüsteemiks abitundmatute s, t ja u sissetoomisega.

$$-15x_1 - 10x_2 + z = 0,$$
$$x_1 + s = 2,$$
$$x_2 + t = 3,$$
$$x_1 + x_2 + u = 4.$$

SAMM 2. Kirjutame välja esialgse lubatava simplekstabeli (baasitabeli). Selle viimases veerus on rea kõigi kordajate summa, kontrollsumma.

muutuja	x_1	x_2	s	t	u	z	vabaliige	summa	testsuhe
0. rida	**−15**	**-10**	**0**	**0**	**0**	**1**	**0**	**−24**	
1. rida	**1**	**0**	**1**	**0**	**0**	**0**	**2**	**4**	**2/1 = 2**
2. rida	**0**	**1**	**0**	**1**	**0**	**0**	**3**	**5**	**∞**
3. rida	**1**	**1**	**0**	**0**	**1**	**0**	**4**	**7**	**4/1 = 4**

Leiame lubatava, st kõiki kitsendusi rahuldava baasilahendi.

Baasilahend on selline lubatav lahend, milles nullist võivad erineda vaid muutujad, millele vastavad veerud moodustavad baasi **m**-mõõtmelises ruumis; **m** – kitsenduste arv, antud juhul **m** = 3.

s = 2, t =3, u =4 , aktiivsed muutujad.

x_1 = 0, x_2 = 0 , mitteaktiivsed muutujad.

SAMM 3. Juhtveeru valimine (*Pivoting column).*

Reegel: juhtveeruks saab muutujate (**x_1, x_2, s, t, u**) veerg, mille nullindas reas on vähim negatiivne arv. Kui negatiivseid elemente nullindas reas pole (välja arvatud muutujale **p** vastava, vabaliikmete ja summa veerud), on eelmisel sammul leitud baasilahend ka optimaalne, s.t. ülesande lahend.

Juhtveeruks valime muutujale **x_1** vastava veeru, s.t. esimese.

SAMM 4. Juhtelemendi valimine juhtveerus. Juhtelement peab alati olema positiivne (ei valita nullilisi ja negatiivseid elemente), juhtveeru iga positiivse elemendi **a** jaoks arvutame testsuhte **v**/**a**, kus **v** on elementi **a** sisaldava rea vabaliige ning juhtelemendiks valitakse see, mille puhul testsuhe on vähim. Rida, milles asub juhtelement, nimetatakse juhtreaks.

Esimese veeru puhul:

(1. rida), testsuhe : 2/1 = 2;

(2. rida), testsuhe on tõkestamata;

(3. rida): testsuhe : 4/1 = 4.

Juhtelemendiks on arv **1** muutujale x_1 vastavas veerus reas number 1, mis on üksiti ka juhtreaks.

SAMM 5. Juhtteisendus: elimineerida juhtelemendi abil juhtveerust mittenullilised elemendid.

muutuja	x_1	x_2	s	t	u	p	vabaliige	summa	testsuhe
0. rida	**0**	**-10**	**15**	**0**	**0**	**1**	**30**	**36**	
1. rida	**1**	**0**	**1**	**0**	**0**	**0**	**2**	**4**	∞
2. rida	**0**	**1**	**0**	**1**	**0**	**0**	**3**	**5**	3/1 = 3
3. rida	**0**	**1**	**-1**	**0**	**1**	**0**	**2**	**3**	2/1 = 2

Leiame lubatava baasilahendi.

$x_1 = 2$, **t =3, u =2 ,** aktiivsed muutujad.

$x_2 = 0$ **, s = 0 ,** mitteaktiivsed muutujad.

Uus juhtelement on märgitud, kordame elimineerimissammu.

muutuja	x_1	x_2	s	t	u	p	vabaliige	summa	testsuhe
0. rida	**0**	**0**	**5**	**0**	**10**	**1**	**50**	**66**	
1. rida	**1**	**0**	**1**	**0**	**0**	**0**	**2**	**4**	
2. rida	**0**	**0**	**1**	**1**	**-1**	**0**	**1**	**2**	
3. rida	**0**	**1**	**-1**	**0**	**1**	**0**	**2**	**3**	

Leiame lubatava baasilahendi.

$x_1 = 2$, $x_2 = 2$, **t =3,** aktiivsed muutujad.

s = 0, u =0 , mitteaktiivsed muutujad.

Reas number null ei ole enam negatiivseid elemente, pole võimalik vahetada baasimuutujaid nii, et sihifunktsiooni väärtus kasvaks, seega on leitud baasilahend ka optimaalne. Sihifunktsiooni maksimaalne väärtus **p = 50.**
Lineaarse planeerimisülesande lahend, kui see leidub,ehk kui lubatavate lahendite hulk on mittetühi ja tõkestatud, asub lubatavate lahendite hulga tipus. See on lineaarse planeerimise ülesande puhul alati nii.

Eelneva aluseks on Carleton'i Ülikooli professori (Ottawa, Canada) John W. Chinneck'i veebileht http://www.sce.carleton.ca/faculty/chinneck/po.html, mida on kasutatud autori loal. Soovitan kõigil tutvuda sealsete ulatuslike ja huvitavate materjalidega.

peep.miidla@ut.ee

Lineaarse planeerimisülesande üldine püstitus
The general formulation of Linear Programming problem

Sihifunktsioon *Objective Function*:

$$\max p := c_1x_1 + c_2x_2 + \ldots + c_nx_n = c^Tx .$$

Kitsendused muutujatele *Constraints* **:**

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i , i = 1,2, \ldots , m,$$

ehk *or*

$$Ax \leq b .$$

Kitsendused muutujate märkidele *Sign Restrictions*:

$$x_j \geq 0 , j = 1,2, \ldots , n .$$

Kui $b \geq 0$, siis sellist ülesande püstitust nimetatakse ka LPÜ standardkujuks *in the case* $b \geq 0$ *this is called Standard Form of LP*.

Eelmisega duaalse lineaarse planeerimisülesande püstitus
The Dual Linear Programming problem for previous

Sihifunktsioon *Objective Function*:

$$\min q := b_1y_1 + b_2y_2 + \ldots + b_my_m = b^Ty .$$

Kitsendused muutujatele *Constraints*:

$$a_{1j}\, y_1 + a_{2j}y_2 + \ldots + a_{mj}y_m \geq c_j , j = 1,2, \ldots , n ,$$

ehk *or*

$$A^Ty \geq c$$

Kitsendused muutujate märkidele *Sign Restrictions* :

$$y_j \geq 0 , j = 1,2, \ldots , m .$$

Märgime, et lineaarset planeerimisülesannet kujul

$$\max p \text{ kitsendustel } Ax = b , x \geq 0$$

nimetatakse mõnikord **LPÜ kanooniliseks kujuks** (*Canonical form*). Siin on lubatavate lahendite hulk antud ainult võrdustega. Sellisele kujule tuleb ülesanne viia simpleksmeetodi rakendamiseks. Kanoonilisele kujule saab teisendada iga lineaarse planeerimise ülesande. Märgime, et kanoonilises kujus puudub tingimus $b \geq 0$.

SIMPLEKSMEETOD

põhikujul antud lineaarse planeerimisülesande (LPÜ) lahendamiseks

Vaatleme näidet ja esitame simpleksalgoritmi selle põhjal

Maksimiseerida $\mathbf{p = 2x - 3y + 4z}$

kitsendustel

$$\mathbf{4x - 3y + z \le 3},$$
$$\mathbf{x + y + z \le 10},$$
$$\mathbf{2x + y - z \le 10},$$

kus x, y, ja z on mittenegatiivsed.

SAMM 1. Teisendada LPÜ lineaarseks võrrandisüsteemiks (vajadusel abitundmatute sissetoomisega).

$$\mathbf{-2x + 3y - 4z + p = 0},$$

$$\mathbf{4x - 3y + z + s = 3},$$

$$\mathbf{x + y + z + t = 10},$$

$$\mathbf{2x + y - z + u = 10},$$

Abitundmatuteks on **s**, **t** ja **u**.

SAMM 2. Kirjutada välja esialgne lubatav simplekstabel (baasitabel). Selle viimases veerus on rea kõigi kordajate summa, kontrollsumma.

muutuja	x	y	z	s	t	u	p	vabaliige	summa
0. rida	−2	3	−4	0	0	0	1	0	−2
1. rida	4	−3	1	1	0	0	0	3	6
2. rida	1	1	1	0	1	0	0	10	14
3. rida	2	1	−1	0	0	1	0	10	13

Leiame lubatava, st kõiki kitsendusi rahuldava baasilahendi. Baasilahend on selline lubatav lahend, milles nullist võivad erineda vaid muutujad, millele vastavad veerud moodustavad baasi **m**-mõõtmelises ruumis; **m** – kitsenduste arv, antud juhul **m** = 3.

s = 3, t =10, u =10 Need on aktiivsed muutujad.

x = 0, y = 0, z = 0 Mitteaktiivsed muutujad.

SAMM 3. Juhtveeru valimine (*Pivoting column).*

Reegel: juhtveeruks saab muutujate (**x, y, z, s, t, u**) veerg, mille nullindas reas on vähim **negatiivne** arv. Kui selliseid on rohkem kui üks, võib valida ükskõik millise neist. Kui negatiivseid elemente nullindas reas pole (välja arvatud muutujale **p** vastava, vabaliikmete ja summa veerud), on eelmisel sammul leitud baasilahend ka optimaalne, s.t. ülesande lahend.

Juhtveeruks valime muutujale **z** vastava veeru, s.t. kolmanda.

SAMM 4. Juhtelemendi valimine juhtveerus.

Reeglid:

- Juhtelement peab alati olema positiivne (ei valita nullilisi ja negatiivseid elemente);
- Juhtveeru iga positiivse elemendi **a** jaoks arvutame testsuhte **v**/**a**, kus **v** on elementi **a** sisaldava rea vabaliige.
- Juhtelemendiks valitakse see, mille puhul testsuhe on vähim. Rida, milles asub juhtelement, nimetatakse juhtreaks.

Meie näites kolmanda veeru puhul:

(1. rida): testsuhe = 3/1 = 3;

(2. rida): testsuhe = 10/1 = 10;

(3. rida): *vastav element on negatiivne, testsuhet ei arvutata.*

Juhtelemendiks on arv **1** muutujale **z** vastavas veerus reas number 1, mis on üksiti ka juhtreaks.

SAMM 5. Juhtteisendus: elimineerida juhtelemendi abil juhtveerust mittenullilised elemendid.

muutuja	x	y	z	s	t	u	p	vabaliige	summa	
0. rida	**14**	**-9**	**0**	**4**	**0**	**0**	**1**	**12**	**22**	
1. rida	**4**	**−3**	**1**	**1**	**0**	**0**	**0**	**3**	**6**	
2. rida	**-3**	**4**	**0**	**-1**	**1**	**0**	**0**	**7**	**8**	
3. rida	**6**	**-2**	**0**	**1**	**0**	**1**	**0**	**13**	**19**	

Leiame lubatava baasilahendi:

z = 3, t = 7, u = 13 Need on aktiivsed muutujad.

x = y = 0, s = 0.

SAMM 6. Korrata samme 3 – 5 kuni esimeses reas pole enam negatiivseid elemente (välja arvatud summade ja vabaliikmete veerud).

muutuja	x	y	z	s	t	u	p	vabaliige	summa
0. rida	7.25	0	0	1.75	2.25	0	1	27.75	40
1. rida	1.75	0	1	0.25	0.75	0	0	8.25	12
2. rida	-0.75	1	0	-0.25	0.25	0	0	1.75	2
3. rida	4.75	0	0	0.5	0.5	1	0	16.5	23

Leiame lubatava baasilahendi:

y = 1.75, z = 8.25, u = 16.5 .

x = 0, s = t = 0.

See on optimaalne lahend. Sihifunktsiooni väärtuseks on 27.75.

peep.miidla@ut.ee

Telekad, näide

Firma valmistab kaht sorti televiisoreid – väiksemaid (1. tüüp) ja suuremaid (2. tüüp). Väiksema teleka komponentide valmistamise peale kulub 6 tundi ja kokkupanemisele 2 tundi; suuremate puhul on toodete ettevalmistamise aeg 2 tundi ja kokkupanemise oma 4 tundi. Kokku on firma vaadeldava ajaperioodi jooksul suuteline osade ettevalmistamisele kulutama 1800 tundi ning kokkupanemisele 1600 tundi. Lisaks on firmal sama ajaperioodi jooksul võimalik paigaldada 350 suurema teleka kuvarit ning nendelt on ette tellitud 75 suuremat telerit, mis tuleb kindlasti valmistada. Suuremate telerite valmistamine maksab 242$ ja need müüakse 250$ eest; väiksemate puhul on vastavad arvud 147$ ja 150$. Määratleda otsustusmuutujad, panna kirja sihifunktsioon ja kitsendused ning lahendada LP ülesanne.

A production firm makes two types of television sets. It takes 6 hours to fabricate the components and 2 hours to assemble a smaller set and on the other hand, bigger sets require 2 hours to fabricate the components and 4 hours to assemble. Only 1800 fabrication hours and 1600 assembly hours are available during any time period. Additionally, the firm has only 350 bigger TV tubes available, and they have a prior order for 75 bigger televisions that they need to fill. Bigger televisions can be produced for $242 and sold for $250, while smaller sets can be produced for $147 and sold for $150. Define the decision varables, develop an objective function, and form the appropriate constraints. Solve the LP Problem.

Ülesande seade *Problem Setup.*

Otsustusmuutujad: $\mathbf{x}_1$ - toodetavate väiksemate telerite arv; $\mathbf{x}_2$ - suuremate telerite arv.

Sihifunktsiooni kordajad on 3 ja 8 vastavalt; sihifunktsioon: $\mathbf{max\ p} = 3\mathbf{x}_1 + 8\mathbf{x}_2$.

Kitsendused.

Tõke telekate kokkupanemise ajale: $2\mathbf{x}_1 + 4\mathbf{x}_2 \leq 1600$;

Tõke osade ettevalmistamisajale: $6\mathbf{x}_1 + 2\mathbf{x}_2 \leq 1800$;

Tõke suuremate telekate kokkupanemise võimalustele: $0\mathbf{x}_1 + 1\mathbf{x}_2 \leq 350$;

Suuremate telekate kokkupanemise miinimumarv: $0\mathbf{x}_1 + 1\mathbf{x}_2 \geq 75$.

Kitsendused muutujatele: $\mathbf{x}_1, \mathbf{x}_2 \geq \mathbf{0}$.

The Decision Variables are: $\mathbf{x}_1$ - number of smaller sets produced; $\mathbf{x}_2$ - number of bigger sets.

Objective Function: **max p** = $3\mathbf{x}_1 + 8\mathbf{x}_2$.

Number of Assembly Hours available: $2\mathbf{x}_1 + 4\mathbf{x}_2 \leq 1600$;

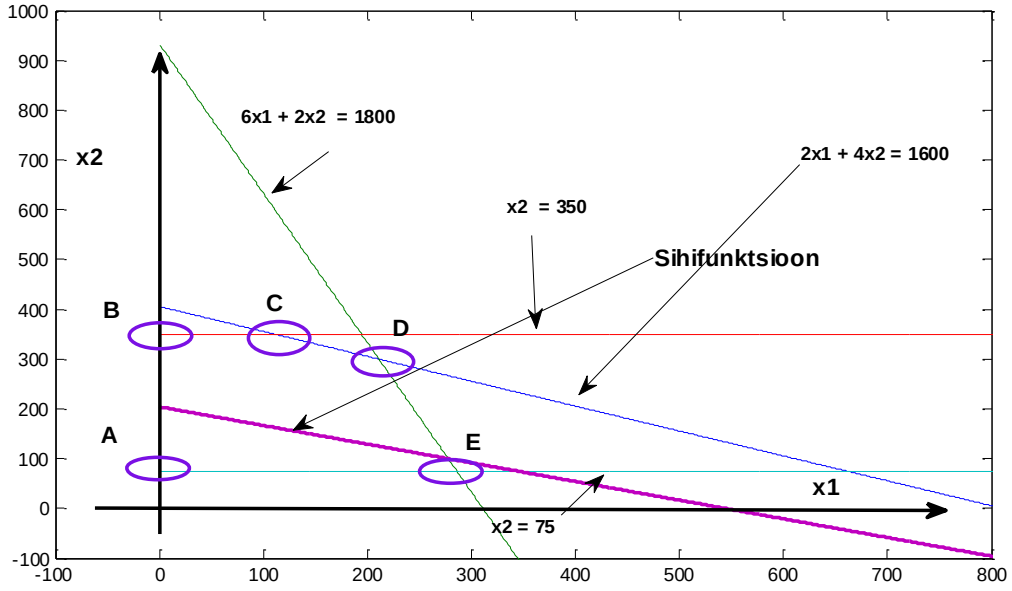
Number of Fabrication Hours available: $6\mathbf{x}_1 + 2\mathbf{x}_2 \leq 1800$;

Number of Fabrication Constraint: $0\mathbf{x}_1 + 1\mathbf{x}_2 \leq 350$;

Minimum number of bigger Sets Constraint: $0\mathbf{x}_1 + 1\mathbf{x}_2 \geq 75$.

Additionally, $\mathbf{x}_1, \mathbf{x}_2 \geq \mathbf{0}$.

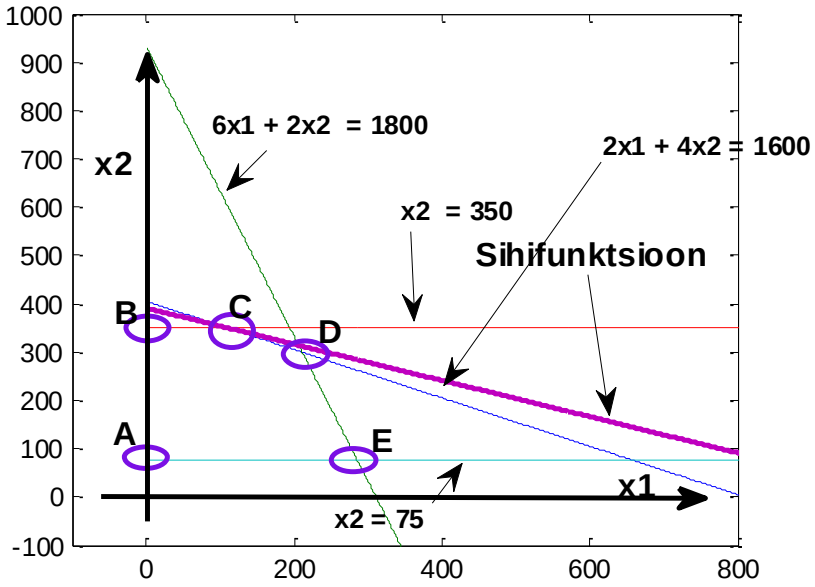
Lahendame ülesande graafiliselt ***Let's solve the TV problem graphically.***



Ellipsitega on tähistatud lubatavate lahendite hulga tipud. Nende koordinaadid ja vastavad sihifunktsiooni väärtused:

A (0, 75), p = 600; B (0, 350), p = 2800; C (100, 350), p = 3100; D (200 300), p = 3000; E (275, 75), p = 1425.

Neljas kitsendus välistab lubatavate lahendite hulgast koordinaatide alguspunkti. Näeme, et optimaalne lahend asub tipus C. Graafiline lahendamine on otstarbekas kahe otsustusmuutuja juhul.



peep.miidla@ut.ee

Täisarvuline lineaarse planeerimise ülesanne
(*Integer Programming Problem*).

TP korral loobutakse muutujate pidevuse nõudest, lisandub nõue, et optimaalne lahend või mõned selle komponendid oleksid täisarvulised.

Lahendusvõtted.
- Kõigi lubatavate täisarvuliste komponentidega lahendite läbivaatamine. Täisarvulisuse nõudega komponendid fikseeritakse ja ülejäänute suhtes lahendatakse lineaarse planeerimise ülesanne. Väga suur arvutustöö maht, suurte ülesannete korral ei tule kõne alla.

- Suunatud otsingud, harude ja tõkete meetod. Idee: lahendada algne ülesanne LP(0) kui üldine lineaarse planeerimise ülesanne, ilma täisarvulisuse nõueteta. Selle lahendist $x^*[LP(0)]$ valitakse komponent x^*_p , mis peaks olema täisarvuline, aga pole. Moodustatakse kaks uut ülesannet LP(1) ja LP(2), kus esialgse ülesande LP(0) kitsendustele on lisatud uued:

$$LP(1):\ x_p \le \lfloor x^*_p \rfloor ,$$
$$LP(2):\ x_p \ge \lceil x^*_p \rceil ,$$

väiksem lähim täisarv ja suurem lähim täisarv, vastavalt. Ülesanded LP(1) ja LP(2) lahendatakse (pideva) lineaarse planeerimise meetodiga (*relaxation*). Kui LP(k), k=1,2, lahend on halvem, kui seni teadaolev parim lubatav lahend ülesandele LP(0), jäetakse ülesanne LP(k) edasise vaatluse alt välja. Märgime, et ülesande LP(0) iga (optimaalne) lahend on lubatavaks lahendiks mõlemale ülesandele LP(1) ja LP(2). Seega, lahendades ülesanded LP(1) ja LP(2), saab kätte ka ülesande LP(0) lahendi. Hargnemist ja hülgamist korratakse kuni saadakse lahend, mille vajalikud komponendid on täisarvulised.

- Suunatud otsingud, lõikemeetodid täisarvulise planeerimise ülesande lahendamiseks. Idee: Esiteks lahendame vastava pideva ülesande LP(0) täisarvulisuse nõudeid arvestamata. Kui lahend ei ole täisarvuline, lõikame lubatavate lahendite hulgast välja leitud optimaalse lahendi koos selle mingi ümbrusega ja lahendame ülesande uuesti. Jätkame protsessi, kuni jõuame täisarvulise lahendini. Tuntud on Gomory algoritmid.

Paketis **MATLAB** lahendatakse binaarset täisarvulist ülesannet funktsiooniga **bintprog**

Binaarne täisarvuline lineaarse planeerimise ülesanne, määramisülesanne (*Binary Integer Programming Problem*). Selle püstitus kanoonilisel kujul:

$$\mathbf{max\ p := c_1x_1 + c_2x_2 + \ldots + c_nx_n = c^Tx}$$

kitsendustel

$$\mathbf{Ax = b\ ,}\ x_i \ge 0\ ,\ i = 1, 2, \ldots, n\ ,$$

x_i on täisarvuline kõikide või osade indeksi väärtuste i = 1, 2, …,n korral.

Tüüpülesandeks on selline: jagada n asja (ülesannet, töökohta, ajalõiku, …) m täitja vahel mingis mõttes optimaalsel viisil.

Optimaalsuse kriteeriumiks võib olla näiteks töö tegemiseks kuluv summaarne aeg, kulutatavate vahendite kogus, mingi kvaliteedi hinnang jne.

Konkreetsemalt, kulugu töötajal i ülesande j täitmiseks c_{ij} sekundit. Jaotada n ülesannet n täitja vahel nii, et igaüks täidaks parajasti üht ülesannet ja kõikide ülesannete täitmiseks kuluv summaarne aeg oleks minimaalne.
Toome sisse muutujad ja otsime nendele binaarseid (täisarvulisi) väärtusi järgmiselt:

$x_{ij} = 1$ kui töötaja i täidab ülesannet j,
$x_{ij} = 0$ kui töötaja i ei täida ülesannet j.

Tuleb minimiseerida kõikide ülesannete täitmiseks kuluv summaarne aeg

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j}$$

kitsendustel

$$\sum_{i=1}^{n} x_{i,j} = 1,\ j = 1,2,\ldots,n;$$

$$\sum_{j=1}^{n} x_{i,j} = 1, i = 1,2,\ldots,n.$$

peep.miidla@ut.ee

Täisarvuline LPÜ, näide
(*Binary Integer Programming Problem*)

Ümardamine ei aita. Vt. ülesannet:

Leida

$$\max z = x_1 + x_2$$

kitsendustel

$$x_1 + 10x_2 \leq 20,$$
$$x_1 \leq 2,$$
$$x_1, x_2 \geq 0, \text{ täisarvulised.}$$

Kui lahendada seda ülesannet täisarvulisuse nõuet eirates, st nagu tavalist lineaarse planeerimise ülesannet, on lahendiks $x_1 = 2$, $x_2 = 1.8$; $z = 11$. Kui ümardada lahend lähima täisarvuni, saame mittelubatava lahendi $x_1 = 2$, $x_2 = 2$; $z = 12$. Kui ümardada allapoole, saame lahendiks $x_1 = 2$, $x_2 = 1$; $z = 7$, mis pole optimaalne; seda on hoopis lahend $x_1 = 0$, $x_2 = 2$; $z = 10$. Viimane on üsna kaugel LPÜ lahendist.

Rändkaupleja ülesanne (*Salesperson Problem*). Samasugune, nagu määramisülesanne.

Antud n linna vahekaugustega $c_{i,j}$. Leida lühim tee, mis algab ja lõpeb samas linnas ning läbib kõiki linnu täpselt ühe korra.

Muutujad: $x_{i,j} = 1$, kui kaupleja tee läheb linnast i linna j, vastasel korral $x_{i,j} = 0$.

Sihifunktsioon:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j}.$$

Kitsendused:

$$\sum_{i=1}^{n} x_{i,j} = 1,\ j = 1,2,\ldots,n;$$

$$\sum_{j=1}^{n} x_{i,j} = 1, i = 1,2,\ldots,n.$$

peep.miidla@ut.ee

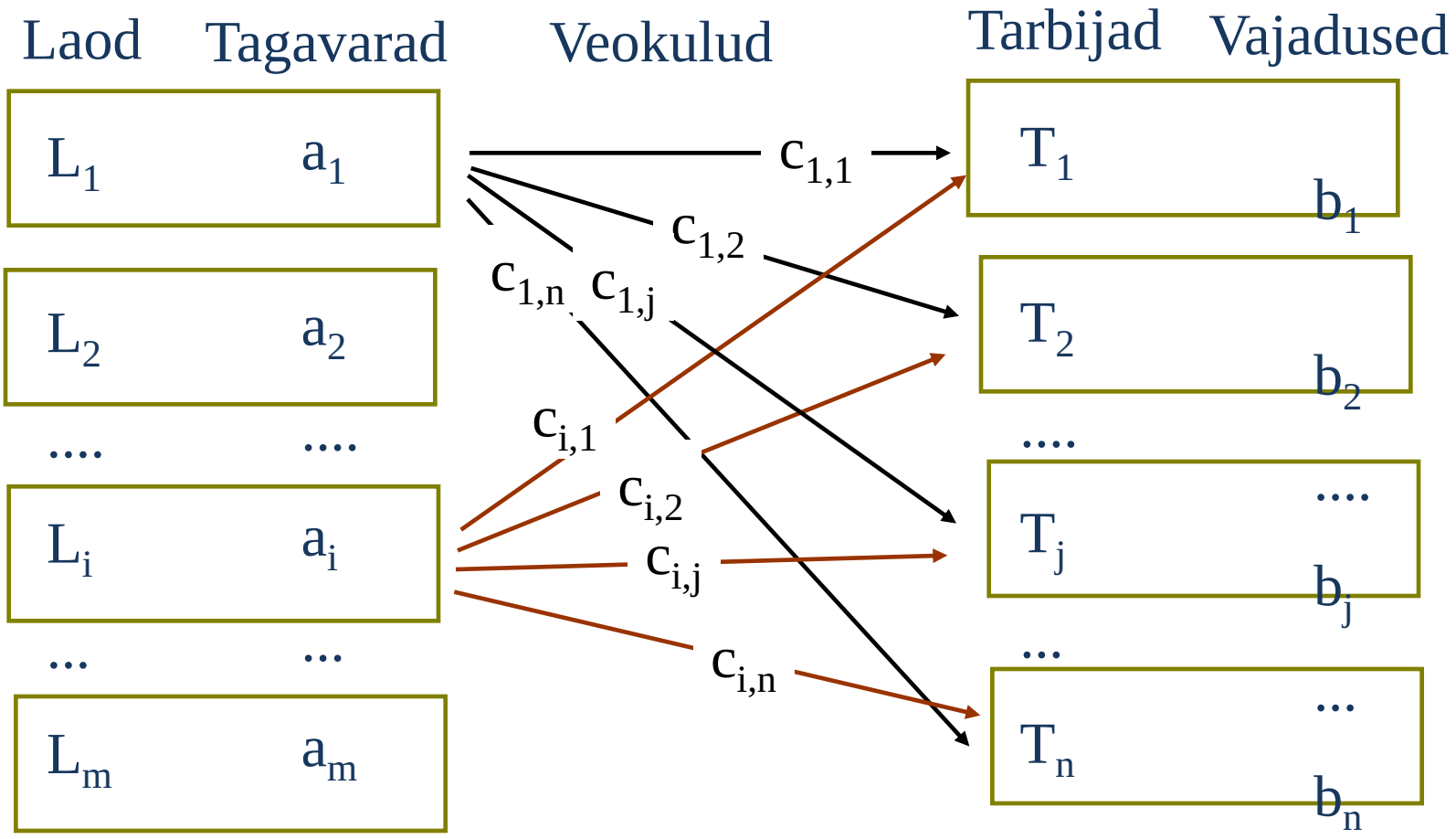
Transpordiülesanne

MTMM.00.306

Praktiline optimiseerimine *Practical Optimization*

peep.miidla@ut.ee

Ülesande püstitus



Ülesanne: koostada selline veoplaan, et kõigi tarbijate vajadused saaksid rahuldatud nii et veokulu oleks minimaalne.

Transpordiülesande esitamine LP ülesandena

Olgu $x_{ij} \geq 0$ - laost L_i tarbijale T_j veetava (ühesuguse) kauba kogus.

Ühestki laost ei saa rohkem kaupa välja vedada, kui seal on:

$$\sum_{j=1}^{n} x_{ij} \leq a_i, \quad i = 1, \ldots, m. \qquad (1)$$

Iga tarbija T_j vajadus peab saama rahuldatud:

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n. \qquad (2)$$

Vedude kogukulu, sihifunktsioon:

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}. \qquad (3)$$

Ülesanne: leida kitsendusi (1) ja (2) rahuldavad mittenegatiivsed suurused x_{ij} , mis annavad sihifunktsioonile (3) minimaalse väärtuse.

Transpordiülesande lahendi olemasolu

Kui vajadusi on võimalik rahuldada, st ülesandel leidub lahend, siis

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j . \qquad (4)$$

Kui valida : $x_{ij} = \frac{a_i b_j}{a}$, kus $a = \sum_{i=1}^{m} a_i$,

siis $x_{ij} \geq 0$ ja

$$\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} \frac{a_i b_j}{a} = \frac{b_j}{a} \sum_{i=1}^{m} a_i = \frac{b_j}{a} a = b_j ,$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \frac{a_i b_j}{a} = \frac{a_i}{a} \sum_{j=1}^{n} b_j = \frac{a_i}{a} b \leq a_i .$$

Kitsendused on seega täidetud ja pakutud lahend on lubatav.

Kuna sihifunktsiooni (3) väärtused saavad olla vaid mittenegatiivsed, on sihifunktsioon alt tõkestatud. Järelikult on ülesanne lahenduv.

Fiktiivne tarbija

Kui tingimus (4) on täidetud range võrratusena, $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, siis leidub ladudes kaupa rohkem kui tarbijad vajavad.

Tähistame tarned fiktiivsele tarbijale T_0 , st kauba jäägi laos L_i pärast reaalsete tarbijate vajaduste rahuldamist:

$$x_{i0} = a_i - \sum_{j=1}^{n} x_{ij}, \quad i = 1, \ldots, m.$$

Fiktiivse tarbija (fiktiivsed) vajadused ja veokulud on sellised:

$$b_0 = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j, \qquad c_{i0} = 0, \quad i = 1, \ldots m.$$

Fiktiivsele tarbijale T_0 omistatakse kauba ülejäägid ladudest. Nii võrdsustatakse ladudes olevate kaupade summa tarbijate vajadustega.

Transpordiülesande kanooniline kuju

Kui on täidetud tingimus $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j,$

saame transpordiülesande sõnastada nn. kanoonilisel kujul:

Leida võrrandisüsteemi

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

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

mittenegatiivsete lahendite hulgast see, mis muudab minimaalseks funktsiooni

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij}.$$

Transporditabel e. veokulude maatriks

	b_1	b_2	$\ldots$	b_n
a_1	$c_{1,1}$	$c_{1,2}$	$\ldots$	$c_{1,n}$
a_2	$c_{2,1}$	$c_{2,2}$	$\ldots$	$c_{2,n}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
a_m	$c_{m,1}$	$c_{m,2}$	$\ldots$	$c_{m,n}$

peep.miidla@ut.ee

Samm lubatava baasilahendi leidmiseks

Valime transporditabelis ruudu (p, q) milles veokulud on minimaalsed, st. $c_{pq} = \min c_{ij}$. Fikseerime veoplaani komponendi

$$x_{pq} = \min (a_{p,} b_q).$$

1. Kui $a_p < b_q$, siis $x_{pq} = a_p$ ja lao L_p tagavara on sellega „ammendatud“ ning jätame rea p edaspidi vaatluse alt välja. Tarbija T_q uueks (veel rahuldamata) vajaduseks saab $\bar{b}_q = b_q - a_p$.

2. Kui $b_q < a_p$, siis $x_{pq} = b_q$ ja tarbija T_q vajadus on rahuldatud. Veeru q jätame edaspidi vaatluse alt välja ning lao L_p uueks tagavaraks saab $\bar{a}_p = a_p - b_q$.

3. Kui $b_q = a_p$, siis jäetakse vaatluse alt välja kas rida p või veerg q. Kui järgi on jäänud vaid üks rida, siis vähendatakse veergude arvu, kui aga järgi on jäänud vaid üks veerg, siis vähendatakse ridade arvu. Baasilahend tuleb sel juhul kidunud (sisaldab null-komponenti).

Ruutplaneerimise ülesanne

Ruutplaneerimise ülesande püstitus. Lähtume üldisest optimiseerimisülesande püstitusest

Leida $\min \mathbf{F}(\mathbf{x})$ üle kõigi elementide $\mathbf{x} \in \mathbf{\Omega}$. (*)

Sellist ülesannet nimetatakse ruutplaneerimise ülesandeks (*Quadratic Programming Problem*), kui **sihifunktsioon**

$$\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}$$

(*Objective Function*) on ruutfunktsioon optimiseerimismuutuja $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ suhtes:

$$\mathbf{min\ 1/2*x'*H*x + f'*x}.$$

Siin $\mathbb{R}$ on reaalarvude hulk, **H** – etteantud kordajate maatriks ja **f** – etteantud kordajate vektor. Tähistustes on kinni peetud paketi **MATLAB** optimiseerimise abivahendi (*Optimization Toolbox*) skeemist.
Kitsendused on lineaarsed, st. hulka $\mathbf{\Omega} = \{x \in \mathbf{X} \mid \mathbf{g}_i(x) \leq 0,\ i = 1,2,\ldots,m\}$ määravad funktsioonid $\mathbf{g}_i \rightarrow \mathbb{R}$, $i = 1,2,\ldots,m$, niinimetatud **kitsendusfunktsioonid** (*Constraint functions*) olgu lineaarsed. Hulka $\mathbf{\Omega} = \{x \in \mathbf{X} \mid \mathbf{g}_i(x) \leq 0,\ i = 1,2,\ldots,m\}$ nimetatakse **lubatavate lahendite hulgaks** (*Feasible Set*). Kitsendused esitatakse kujul

$$\mathbf{Ax} \leq \mathbf{b};\ \mathbf{Bx} = \mathbf{beq},\ \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}\ .$$

peep.miidla@ut.ee

Mittelineaarse planeerimise ülesanne (*Nonlinear programming*)

Lähtume **optimiseerimisülesande** üldisest püstitusest.

Olgu antud funktsioon $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}$, nõndanimetatud **sihifunktsioon** (*Objective Function*); $\mathbb{R}$ on reaalarvude hulk. Veel olgu antud funktsioonid $\mathbf{g}_i \rightarrow \mathbb{R}$, i = 1,2,…,m, niinimetatud **kitsendusfunktsioonid** (*Constraint functions*). Hulka $\mathbf{\Omega} = \{x \in \mathbb{R}^n \mid \mathbf{g}_i(x) \leq 0,$ i = 1,2,…,m} nimetatakse **lubatavate lahendite hulgaks** (*Feasible Set*).

Leida min $\mathbf{f}(x)$ üle kõigi elementide $\mathbf{x} \in \mathbf{\Omega}$. (*)

Elementi $\mathbf{x}^* \in \mathbb{R}^n$, mis realiseerib funktsiooni **f** maksimumi lubatavate lahendite hulgal **Ω**, nimetatakse optimiseerimisülesande (*) **lahendiks**. Optimiseerimisülesannetel on sageli rohkem kui üks lahend. Mittelahenduvus tähendab seda, et lubatavate lahendite hulk **Ω** on tühi või sihifunktsioon (alt) tõkestamata.

Kui sihifunktsioon **f** ja kitsendused $\mathbf{g}_i$ (i = 1,2,…,m) on mittelineaarsed, nimetatakse ülesannet (*) mittelineaarseks planeerimisülesandeks. Optimiseerimisülesannete ja –meetodite valdkond on kiiresti arenev, paralleelselt modifitseerub ka vastav terminoloogia.

Mõnikord formuleeritakse maksimiseerimisülesanne, milles nõutakse sihifunktsiooni maksimumi leidmist. See on lihtsasti ja üldisust kahandamata viidav kujule (*): kui mingi element **x** minimiseerib funktsiooni **f** väärtuse hulgal **Ω**, siis maksimiseerib seesama element funktsiooni –**f** väärtuse samal hulgal.

Mõnikord $\mathbf{\Omega} = \mathbb{R}^n$, siis nimetatakse ülesannet kitsendusteta ülesandeks (*Unconstrained Optimization*) või kitsendusteta minimiseerimisülesandeks.

Eestikeelne raamat: Kaasik, Ü., Kivistik, L. Operatsioonianalüüs. Tallinn, Valgus. 1982.

peep.miidla@ut.ee

Rändkaupleja ülesanne.
(*The Travelling Salesperson Problem, TSP*)

Aluseks on binaarne täisarvuline lineaarse planeerimise ülesanne (*Binary Integer Programming Problem*), mille püstitus kanoonilisel kujul:

$$\mathbf{max\ p := c_1x_1 + c_2x_2 + ... + c_nx_n = c^Tx}$$

kitsendustel

$$\mathbf{Ax = b\ ,}\ x_i \geq 0\ ,\ i = 1, 2, \ldots, n\ ,$$

x_i on täisarvuline kõikide või osade indeksi väärtuste i = 1, 2, …,n korral.

Rändkaupleja ülesanne (*Travelling Salesperson Problem*).
On antud n linna vahekaugustega $c_{i,j}$. Leida lühim tee, mis algab ja lõpeb samas linnas ning läbib kõiki linnu täpselt ühe korra.

Binaarsed täisarvulised muutujad: $x_{i,j}$ = 1, kui kaupleja tee läheb linnast i linna j, vastasel korral $x_{i,j}$ = 0 .

Sihifunktsioon:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j}.$$

Kitsendused:

$$\sum_{i=1}^{n} x_{i,j} = 1,\ j = 1,2,...,n;$$

$$\sum_{j=1}^{n} x_{i,j} = 1, i = 1,2,...,n.$$

peep.miidla@ut.ee

TSP lahendamine simuleeritud lõõmutamise meetodiga.

(1) *Configuration*: Linnad nummerdatakse, i = 1...N. Koordinaadid: (x_i, y_i).
Konfiguratsiooniks on arvude 1...N, permutatsioonid, mida interpreteeritakse järjekorrana, milles linnu külastatakse.

(2) *Rearrangements*: Liikumised naaberseisundeisse.
Kaht tüüpi liikumised:

a) A section of path is removed and then replaced with the same cities running in opposite order.
b) A section of path is removed and then replaced in between two cities on another, randomly chosen, part of the path.

(3) *Objective function*: Sihifunktsiooniks on tee kogupikkus, mida minimiseeritakse:

$$C = \sum_{i=1}^{N} ((x_i - x_{i+1})^2 + (y_i - y_{i+1})^2)^{0.5}$$

Kokkuleppeliselt on teekonna punktiks N+1 alguspunkt.

peep.miidla@ut.ee

Geneetiline algoritm, GA, *genetic algorithm*.

Geneetilised algoritmid (GA) simuleerivad evolutsiooni looduses läbi generatsioonida järjest elujõulisemate isendite (paremate lahendite) poole. GA rakendamine võimaldab leida globaalse ekstreemumi.

Elementaartegevused GS puhul, nagu need on bioloogilistelt aktiviteetidelt võetud:

paaritumine (*mating*), paljunemine (*reproduction*), kloonimine, (*cloning*), ristamine (*crossover*), mutatsioon (*mutation*).

GA etapid.

Esimeseks sammuks on optimiseerimisparameetrite kodeerimise viisi määramine. Levinud on binaarkodeering või reaalarvuline kodeerimine, konkreetsetes rakendustes leidub ka teisi võimalusi. Üldiselt peab GA iga lahend olema kodeeritud stringina, ühesuguse pikkusega n kogu algoritmi rakendamise jooksul. Iga lahend esitub sellise nn organismi ehk kromosoomina (pikkusega n), iga kromosoomi positsiooni (muutujat selles) nimetatakse geeniks, geeni väärtust alleeliks.

Näiteks, kui n = 3, siis kromosoomi üldkuju on $x = (x_1, x_2, x_3)$, kus x_1, x_2, ja x_3 on geenid selles kromosoomis kolmel positsioonil. Kromosoomis (3, 8, 9) on teise geeni alleeliks 8.

TSP korral, kui linnad on nummerdatud, {1, 2,…,n}, on kromosoomideks selle stringi permutatsioonid, milles sümbolid 1, 2,…, n esinevad igaüks täpselt ühe korra.

Teiseks tuleb fikseerida sihifunktsioon. Räägime edaspidi selle miinimumi leidmisest ja oletame, et sihifunktsioon on positiivsete väärtustega kõigil lubatavail lahendeil. Viimase võib saavutada konstandi liitmisega sihifunktsioonile (lubatavaid lahendeid on lõplik hulk).

Kolmandaks minnakse üle kitsendusteta ülesandele. Selleks tuuakse sisse trahvifunktsioon (*penalty function*), mis on null lubatavel lahenditel (kromosoomidel) ja positiivne (suur) mittelubatavatel. Sobivuse mõõt (*fitness measure, evaluation function*), funktsioon f defineeritakse kui sihifunktsiooni ja trahvifunktsiooni summa.

Näide. TSP korral on sihifunktsiooniks

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j}.$$

Trahvifunktsiooni pole vaja, iga permutatsioon osutub lubatavaks lahendiks, kui igast linnast on võimalik liikuda igasse teise.

Algoritm.

1. Start: genereeritakse algpopulatsioon, millest algab areng täiuslikuma suunas. Populatsioon on sama suurusega kogu algoritmi rakendamise käigus. Teel täiuslikuni hüljatakse halvemad ja lubatakse paljuneda vaid parimatel lahenditel. Tavaliselt genereeritakse algpopulatsioon juhuslikult.

Näide: TSP puhul võib seda teha "lähima naabri" meetodil: igast linnast liigutakse lähima (naaber)linnani, kus veel käidud pole.

2. Leitakse populatsiooni iga kromosoomi x sobivus f(x), sobivusskoor.

3. Uue populatsiooni, nn uue generatsiooni konstrueerimine: paljundamine ja kloonimine. Teatud protsent (tüüpiliselt 10% - 40%) kromosoomidest kantakse muutumatul kujul üle järgmisse põlvkonda.

- Paljundamiseks valitakse vanemad tõenäosuslikult. Tõenäosus saada valitud ja paljunemise teel otse üle kantud järgmisse põlvkonda, on reeglina proportsionaalsed sobivusskooriga.

- Kloonimine. Deterministlik tegevus, sarnane paljundamisega. Vahe selles, et teatud kindel arv parima sobivusskooriga organisme kantakse otse üle järgmisse põlvkonda (paljundamisel valitakse need juhuslikkuse alusel).

4. Uue populatsiooni, nn uue generatsiooni konstrueerimine: paaritamine või ristamine ja mutatsioonid. Nii saadakse suurem osa uuest kromosoomide generatsioonis, millest omakorda suurem osa ristatakse väiksem muteeritakse. Valitakse kaks vanemat (*parent*) parima sobivusskooriga kromosoomide seast. Mida parem sobivusskoor, seda suurem tõenäosus saada valitud vanemaks. On välja töötatud mitmeid erinevaid võtteid, ristamise määr kõigub vahemikus 0.6 – 0.9.

- Ühepunkti-ristamine (*One-point crossover*). Genereeritakse kaks last e. järeltulijat. Olgu vanemateks $x = \{x_1, x_2,\ldots,x_n\}$ ja $y = \{y_1, y_2,\ldots,y_n\}$. Juhuslikkuse alusel valitakse ristamispunkt (*crossover position*) r, vahemikus 1,…,n ja lasteks on $u = \{x_1,\ldots,x_{r-1},y_r,\ldots,y_n\}$ ja $v = \{y_{1,\ldots,}y_{r-1},x_r,\ldots,x_n\}$. On kaks võimalust: kas kanda mõlemad lapsed uude generatsiooni või ainult parema sobivusskooriga laps, teine hüljatakse.

- Kahepunkti-ristamine (*Two-point crossover*). Olgu vanemateks $x = \{x_1, x_2,\ldots,x_n\}$ ja $y = \{y_1, y_2,\ldots,y_n\}$. Juhuslikkuse alusel valitakse kaks ristamispunkti (*crossover position*) r ja s, $r < s$, vahemikus 1,…,n ja järeltulijaiks on kromosoomid $u = \{x_1,\ldots,x_{r-1},y_r,\ldots,y_{s,}x_{s+1},\ldots,x_n\}$ ja $v = \{y_{1,\ldots,}y_{r-1},x_r,\ldots,x_s,y_{s+1}\ldots,y_n\}$. Jällegi on kaks võimalust: kas kanda mõlemad lapsed uude generatsiooni või ainult parema sobivusskooriga laps ja teine hüljatakse.

- Juhuslik ristamine. Antud vanemate puhul saadakse järeltulijad nii, et $u_i = x_i$ ja $v_j = y_j$ tõenäosusega α ning $u_i = y_i$ ja $v_j = x_j$ tõenäosusega $1 - \alpha$. Siin α on ette valitud arv 0 ja 1 vahel. Kantakse üle kas mõlemad või vaid üks järeltulija.

Näide: TSP, aga ka teiste ülesannete korral võib juhtuda, et kirjeldatud ristamisalgoritmid annavad mittelubatavaid järeltulijaid. Sel juhul kasutatakse osaliselt kattuvaid ristamisoperaatoreid (*partially matched crossover operator or PMX*).

- Osaliselt kattuv ristamisoperaator (*partially matched crossover operator*). Olgu vanemateks (permutatsioonideks) $x = \{x_1, x_2,\ldots,x_n\}$ ja $y = \{y_1, y_2,\ldots,y_n\}$. Juhuslikkuse alusel valitakse kaks ristamispunkti (*crossover position*) r ja s, $r < s$, vahemikus 1,…,n. Kahepunkti ristamise algoritm muutub järgmiselt. Esimese lapse u saamiseks indeksi muutumisel vahemikus t = r,…,s: kui $x_t \neq y_t$, vahetada x_t ja y_t omavahel vanema x permutatsioonis. Teise järeltulija saamiseks talitada sümmeetriliselt, vahetades x ja y osad. Saab näidata, et saadakse uued permutatsioonid ja seega lubatavad lahendid.

Näide. Olgu n = 6 ja vanemateks x = {4,⌋ 5, 6, 2, 1,⌋ 3}, y = {1,⌋ 2, 6, 4, 3,⌋ 5}. Ristamispunktid olgu näidatud sümboliga ⌋, st. r = 2 ja s = 5. Laps u saadakse nii, et vahetatakse 5 ja 2, 2 ja 4 ning 1 ja 3; selle tagajärjel kujuneb x nii: x → {4, 2, 6, 5, 1, 3} → { 2, 4, 6, 5, 1, 3 } → u = { 2, 4, 6, 5, 3, 1}. Analoogiliselt teiseneb y → v = { 3, 5, 6, 2, 1, 4}.

- Mutatsioonid. Nende tõenäosus on reeglina väike (0.001), st kokkuvõttes muudetakse uues generatsioonis juhuslikult väikest osa kromosoomidest.

Kui uus generatsioon on valmis, korratakse iteratsiooni (uue generatsiooni konstrueerimist) taas. Lõpetamise kriteeriumiks on reeglina asjaolu, et mitme uue generatsiooni jooksul pole lahendi paranemist ette tulnud või kui ettenähtud arvutusressursid on ammendatud.

Testülesanne: Rändkaupleja ülesanne (Travelling Salesperson Problem).

Antud n linna vahekaugustega $c_{i,j}$. Leida lühim tee, mis algab ja lõpeb samas linnas ning läbib kõiki linnu täpselt ühe korra.

Muutujad: $x_{i,j} = 1$, kui kaupleja tee läheb linnast i linna j, vastasel korral $x_{i,j} = 0$.

Sihifunktsioon:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j}.$$

Kitsendused:

$$\sum_{i=1}^{n} x_{i,j} = 1,\ j = 1,2,\ldots,n;$$

$$\sum_{j=1}^{n} x_{i,j} = 1, i = 1,2,...,n.$$

Viiteid:

http://www.lalena.com/AI/Tsp/

http://optlab-server.sce.carleton.ca/POAnimations2007/GeneticAlg.aspx

http://www.codeproject.com/Articles/1403/Genetic-Algorithms-and-the-Traveling-Salesman-Prob

http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php

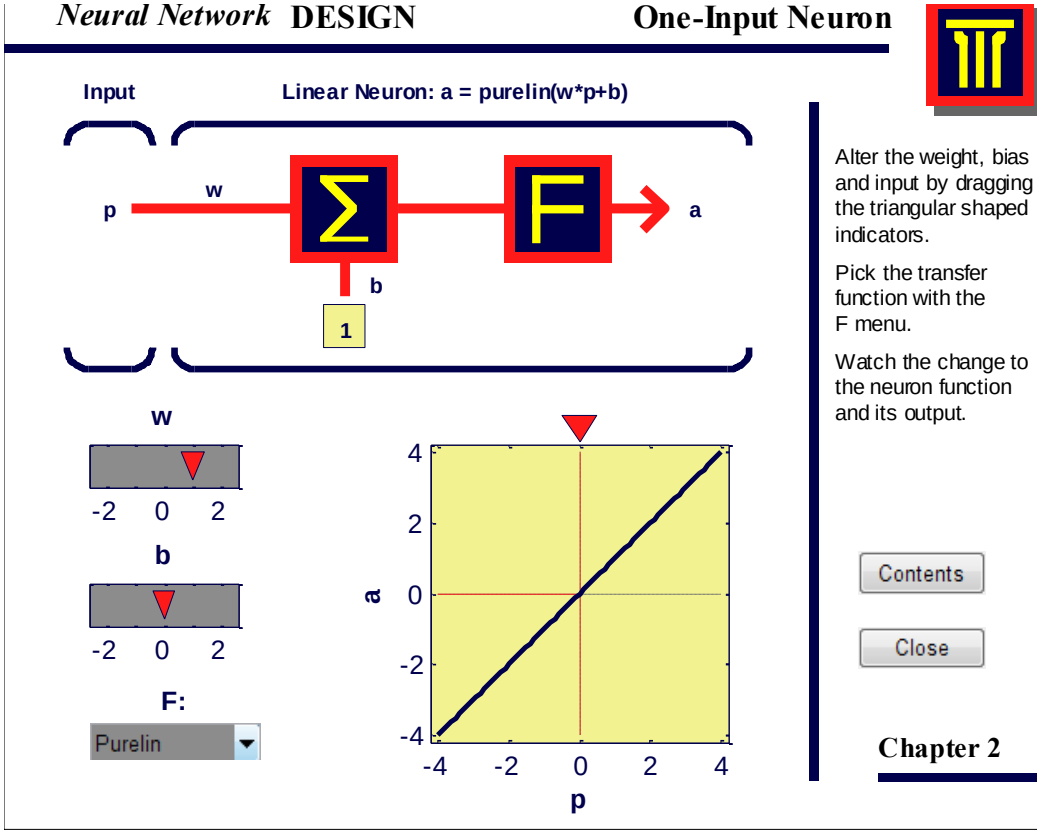
peep.miidla@ut.ee

Närvivõrgud *Neural Network*

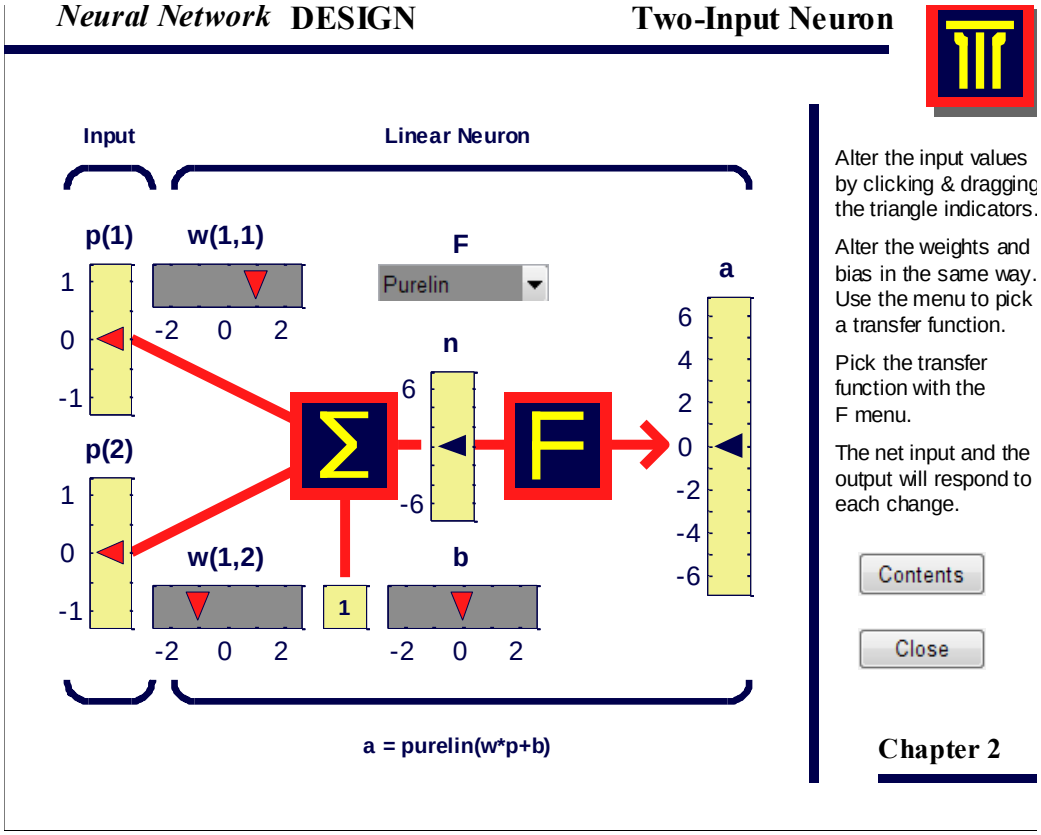
Lineaarne neuron, st ülekandefunktsioon on lineaarne: a = w*p + b.

Siin a – väljund; p – sisend; w – kaal.

Pilt paketi MATLAB vastavalt rakenduslehelt.



Kahe sisendiga lineaarne neuron, [a(1) a(2)] = [w(1)*p(1) + w(2)*p(2)]



Veebilehti

http://www.mathworks.se/products/neural-network/index.html

http://www.learnartificialneuralnetworks.com/

http://sydney.edu.au/engineering/it/~irena/ai01/nn/travtute.htm

Pattern recognition:
http://www.byclb.com/TR/Tutorials/neural_networks/Index.aspx

Küllalt vana juhend:
http://www.cse.msu.edu/~cse802/notes/ArtificialNeuralNetworks.pdf

MIT:
http://ocw.mit.edu/courses/brain-and-cognitive-sciences/9-641j-introduction-to-neural-networks-spring-2005/

peep.miidla@ut.ee

Simuleeritud lõõmutamine (*Simulated Annealing*)

Globaalse optimiseerimise üheks meetodiks on simuleeritud lõõmutamine, mis ühendab endas Monte Carlo ja lokaalse otsingu aspekte. See meetod on inspireeritud füüsikaliste süsteemide käitumisest. Statistiline mehaanika näitab, et paljude süsteemide olekuparameetrid kipuvad kõikuma juhuslikul, kuid siiski suunatud viisil. Lõõmutamine on metallide töötlemise meetod, milles tahkes olekus metall kuumutatakse kõrge temperatuurini ja seejärel jahutatakse aeglaselt vastavalt soovitud tulemusele ja sellest johtuvale spetsiaalsele jahutamiserežiimile. Meetodi motivatsiooniks on asjaolu, et kui kuumutamistemperatuur on piisavalt kõrge kindlustamaks metalliosakeste juhuslikku paiknemist ja jahutamine on küllalt aeglane soojustasakaalu säilitamiseks, siis paigutuvad metalliosakesed (aatomid) „iseenesest“ mustrisse, mis vastab globaalsele energiamiinimumile ning garanteerib ideaalse kristalli tekke.

Simuleeritud lõõmutamine on matemaatiline algoritm analoogiliste eesmärkide saavutamiseks ehk parima olekumustri leidmiseks.

Toome sisse dünaamilise süsteemi olekute ruumi (*State Space*), mis kujutab endast süsteemi olekumuutujate $x = (x_1, x_2, \ldots, x_n)$ kõiki võimalikke väärtusi, lubatavaid lahendeid. Olekuruumi kaht punkti A ja B käsitletakse naabritena kui on võimalik "lihtne ja vahetu" üleminek nende kahe vahel. Näiteks kui olekuid kirjeldavad nullidest ja ühtedest koosnevad ühesuguse pikkusega lõplikud vektorid, on naabriteks olekuvektorid, mis erinevad täpselt ühe positsiooni võrra (ühel on seal null, teisel üks ja ülejäänud komponendid langevad kokku).

Tähistagu E(A) seisundi A energiat; oletame, et on võimalik arvutada see energia iga olekupunkti kohta.

Simuleeritud lõõmutamise algoritm.

Samm 1. *Initialize,* algatamine. Startida juhuslikult valitud väga kõrge temperatuuriga olekust.

Samm 2. *Move,* liikumine naaberolekusse. Valida võimalik naaberolek.

Samm 3. *Calculate score,* energia arvutamine. Arvutada energiamuutus liikumisel valitud naabrusse.

Samm 4. *Choose,* valik. Langetada otsus oleku muutmise kohta.

Samm 5. *Update and repeat,* Uuendamine ja protsessi kordamine. Vastava otsuse puhul liikuda naaberseisundisse ja korrata protsessi sammust 2.

Iteratsioone korrata, kuni lõpetamistingimuste täidetuseni.

Algoritmi pseudokood.

Algorithm SIMULEERITUD_LÕÕMUTAMINE

Määrata: olekuruum X, energiafunktsioon (sihifunktsioon) f, naaberolekute tunnus.

begin

temp = ALGTEMPERATUUR; % Juhuslikult.

seisund = ALGSEISUND; % Juhuslikult.

while (*temp* > LÕPPTEMPERATUUR) **do**

while (*lõpetamise_tingimus* = **false**) **do**

uus_seisund = NAABEROLEK(*seisund*);

energiamuut = ENERGIA(*uus_seisund*) - ENERGIA (*seisund*);

if (*energiamuut* < 0) **then**

seisund = uus_ seisund;

else if (RANDOM(0,1) > $e^{-(\text{ energiamuut/temp})}$) **then**

place = new_place; %Lubatakse seisundimuutus ka juhuslikult.

end if

temp = UUSTEMPERATUUR(*temp*);

end while

end while

end

Valem üleminekutõenäosuse leidmiseks, millega süsteem läheb seisundist A seisundisse B, on selline:

$P(A \rightarrow B) = \exp(-[E(B)-E(A)] / kT)$,

kus k on Boltzmanni konstant.

Füüsikaliste mõistete ja optimiseerimisterminite võrdlus.

Termodünaamika	Simuleeritud lõõmutamine, kombinatoorne optimiseerimine
Süsteemi olekud	Lubatavad lahendid
Energia	Maksumus
Oleku muutumine	Naaberlahendid
Temperatuur	Kontrollparameeter
Tardunud olek	Heuristiline lahend

peep.miidla@ut.ee

Ülevaade sipelgaalgoritmist.

ACO (*Ant Colony Optimization*) ehk sipelgakoloonia optimiseerimine on suhteliselt uus optimeerimisülesannete lahendamise **metaheuristika**, mis sai alguse Milaano Polütehnikainstituudis aastal 1992 Marco Dorigo poolt kaitstud doktoriväitekirjast.

Sipelgaalgoritmi tööd saab üldisemal juhul kirjeldada järgmiselt: ühte kolooniasse kuuluvad m sipelgat liiguvad paralleelselt ja asünkroonselt läbi kõrvalolevate probleemi olekute, koostades niimoodi lahendeid, milleks on teed läbi graafi. Kõik probleemi võimalikud olekud on esitatud graafina, nn. **konstruktsioonigraafina**. Liikumine toimub graafis tõenäosusreegli põhjal, mis arvestab lokaalselt kättesaadavat heuristilist informatsiooni ja sipelgate poolt produtseeritud suuruste, nn. **feromoonijälgede** väärtusi. Kui konkreetne sipelgas on lahendi valmis saanud, lubatakse tal uuendada feromooni väärtused lahendile kuuluvatel graafi komponentidel, kusjuures väljastatava feromooni kogus sõltub leitud lahendi kvaliteedist. Lisaks võib feromooni väljastamist sipelgatel lubada igal üleminekul ühelt graafi komponendilt teisele lahendi konstrueerimise käigus. Igas ACO algoritmis on tähtsal kohal ka feromooni aurustumine ja lisategevused.

Igal sipelgal k on mälu M_k juba külastatud tippude nimekirja hoidmiseks. Samuti on mälu vajalik naabruse, st. teatud mõttes lähedaste tippude N_i^k koostamiseks tipus i. Feromooni kogus sipelga k lahendisse s_k kuuluvatel servadel muutub vastavalt reeglile

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{k}, \ \forall \text{ serva } l_{ij} \in s_k \text{ korral},$$

kus lisatav feromooni kogus $\Delta\tau_{ij}^{k}$ sõltub lahendi s_k kvaliteedist.

Lahendite koostamine sipelgasüsteemis toimub järgmisel viisil: igal sammul valib tipus i asuv sipelgas k järgmise tipu j oma naabrusest N_i^k tõenäosusega p_{ij}^{k}, mis sõltub nii feromoonijälgede väärtustest τ_{ij} tippude i ja j vahel kui ka nn. **heuristilisest** informatsioonist η_{ij}. Tähistame tee tipust i tippu l sümboliga c_{il}. Tõenäosuse arvutamise valem on selline:

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{c_{il} \in N_j^k} \tau_{il}^{\alpha}\left[\eta_{il}\right]^{\beta}}.$$

Tõenäosuste p_{ij}^{k} leidmisel figureerivad tähtsad sipelgaalgoritmi parameetrid: α on feromoonijälgede τ_{ij} ja β heuristilise informatsiooni η_{ij}

astendajad. Kui näiteks $\alpha = 0$, siis sipelgas teeb valiku ainult heuristilise informatsiooni põhjal; juhul, kui $\beta = 0$, siis vastavalt feromoonijälgede põhjal.

Lisaks sipelgate käitumise muutmisele feromoonide abil on algoritmis olulisel kohal feromooni **aurustumine** ning lisategevused (**daemon actions**). Feromooni aurustumine on protsess, mille käigus feromooni kogus τ_{ij} konstruktsioonigraafi igal serval väheneb automaatselt aja möödudes. Vähenemine on seotud kordajaga ρ, mis kuulub vahemikku [0, 1]. Seda kordajat nimetatakse feromooni **aurustumisteguriks** ja aurustumisvalem on järgmine:

$$\tau_{ij}(\text{järgmisel etapil}) = (1 - \rho) * \tau_{ij}(\text{eelmisel etapil}).$$

Feromooni aurustumine on vajalik otsingu kiire koondumise vältimiseks lokaalselt optimaalse lahendi ümber. Selle protsessi abil on realiseeritud **unustamine** (*forgetting*), mis aitab sipelgatel avastada uusi teid otsinguruumis.

Lisategevusteks nimetatakse kõrgemal tasemel sooritatavaid protseduure, millega püütakse sipelgate tegevust paremini koordineerida või juhtida. Üheks oluliseks näiteks võib tuua **lokaalse otsingu** (*local search*) leitud lahendi suunamiseks lokaalse ekstreemumini.

Et parandada sipelgaalgoritmi jõudlust, on välja pakutud mitmeid selle algoritmi täiendusi. Üks olulisimaid on nn. Max-Min sipelgasüsteem (MMAS). Loetleme üles selle eripärad.

Kõigepealt paneb see rõhu parimatele lahenditele: ainult vaadeldava iteratsiooni (lahendite koostamise tsükli) parim sipelgas või seni kogu otsingu parim sipelgas võib feromooni väljastada ning feromooni kogus sõltub leitud lahendi kvaliteedist. Kui feromooni väärtusi uuendab ainult seni parim sipelgas, läheb otsinguprotseduur kiiresti **ahneks** ning otsing hakkab toimuma seni leitud lahendi ümber. Kui feromooni väljastamist lubatakse ainult iteratsiooni parimal sipelgal, on otsing vähem ühele poolele suunatud ja rohkem servi saavad feromooni juurde. Selline strateegia võib mõnikord viia kiiresti otsingu **stagnatsiooni**, kus kõik sipelgad kasutavad üht ja sama teed, sest feromooni kogus parimatele lahenditele kuuluvatel servadel hakkab piiramatult kasvama.

Viimatinimetatud ohu vastu võitleb teine meetodisse MMAS sisse viidud muudatus, mis piirab feromooni väärtuse vahemikku $[\tau_{min}, \tau_{max}]$, kus väärtus τ_{max} kujutab endast ülemist tõket maksimaalsele võimalikule feromooni kogusele konstruktsioonigraafi servadel. Seega jääb ka suvalises tipus asuval sipelgal suvalise järgmise tipu valimise tõenäosus teatud vahemikku $[p_{min}, p_{max}]$.

Algoritmi kolmas eripära seisneb selles, et feromooni algväärtusteks igal serval võetakse maksimaalne väärtus τ_{max}. Koos väikese feromooni

aurustumisteguriga tagab see otsingu laialiminemist otsinguprotseduuri alguses ning suuremat arvu võimalikke teede avastamist algusest peale.

Viimane muudatus puudutab algoritmi töö haldamist: kui otsing näitab seisaku märke, näiteks pole õnnestunud seni parima kvaliteediga lahendit parandada teatud arvu iteratsioonide jooksul, **reinitsialiseeritakse** feromooni väärtused algväärtusteks. Mõnikord tehakse seda ka juhuslikult paremate lahendite saavutamise eesmärgil.

Võttes kõik MMAS algoritmi eripärad kokku, võib öelda, et algne sipelgasüsteem on tehtud oluliselt paindlikumaks. Seda tõestavad ka testid, mis näitavad, et MMAS on üks parima jõudlusega sipelgasüsteemi laiendusi.Tekib küsimus, et kas mõni metaheuristiline meetod on parem kui sipelgaalgoritm või mitte. Maailmakirjanduses on teaduslikult korrektselt kirjeldatud ühe autorite kollektiivi poolt läbi tehtud põhjalikku analüüsi ja viie meetodi võimekuse võrdlust tunniplaani koostamisel. Vaadeldud meetodeiks on *Evolutionary Algorithms*, *Ant Colony Optimization*, *Iterated Local Search*, *Simulated Annealing* ja *Tabu Search*. Põhitulemusena tuuakse selgelt välja, et ühelgi neist pole märgatavat eelist teiste ees üldjuhul. Tõsi, autorid näitavad ka seda, et erinevate erijuhtude korral võib mõnel meetodil eeliseid olla kuid seda ikkagi vaid erijuhtudel.

Allpool toodud tabelist leiame enim levinud ACO algoritmid koos autori ja väljatöötamise aastaga.

Algoritm	**Autor**	**Aasta**
Sipelgasüsteem (*Ant System*)	Dorigo, Maniezzo & Colorni	1991
Elitistlik sipelgasüsteem (*Elitist AS*)	Dorigo	1992
Sipelgakoloonia süsteem (*Ant Colony System - ACS*)	Dorigo & Gambardella	1996
Max-Min sipelgasüsteem MMAS	Stützle & Hoos	1996
ANTS (*Approximated Nondeterministic Tree Search*)	Maniezzo	1998
Rivipõhine sipelgasüsteem (*Rank-Based AS*)	Bullnheimer, Hartl & Strauss	1999
Parim-halvim sipelgasüsteem (*Best-Worst AS*)	Cordón, et al.	2000
Hüperkuubi sipelgasüsteem (*Hyper-cube ACO*)	Blum, Roli, Dorigo	2001

peep.miidla@ut.ee