Williams' Models

Tony Hürlimann
info@matmod.ch

March 13, 2025
(First version: Sep 04, 1994)

Abstract

This paper contains all 29 case studies of the textbook of P. Williams (see [3]). For each model only a short description of the problem is given followed by the formulation of the model in LPL format. The name of the model is **will??** where **??** means the number (01–29) of the case study in the book. Several small models spread over the book are also implemented. Their name is **willi???** and **???** means the page of the book. All models can directly be executed from this paper (on the Internet).

Contents

1 Introduction

The text contains the LPL source code from all model projects and some other models from the book [3]. Each model can be executed directly through the Internet by clicking the red link at the top title of each model. From the model explanation you will need the book itself.

2 Food Manufacture I (will01)

—- Run LPL Code , HTML Document –

Problem: A company manufactures (over the next six months) one food by refining various raw oils, which come in vegetable and non-vegetable oil – each with a given "hardness". The company can buy oils in the different months. The storage capacity is not a limitation but costs 5ct/ton. The refining capacity, however, is limited at 200 tons of veg oil and 250 tons of non-veg oils. The "hardness" of the food must be in certain given ranges. How many tons of each oil should the company buy, store, and use in each period in order to maximize profit by selling the food at a given price ?

Model: The Model is as follows:The problem is from [3] Chapter 13.1.

Listing 1: The Complete Model implemented in LPL [6]

```
model Will01 "Food Manufacture I";
set p := [ January February March April May June ];
    r    := [ VEG1 VEG2 OIL1 OIL2 OIL3 ] "all oils";
    v{r} := [ VEG1 VEG2 ]                "veg oils";
parameter pr{p,r} := [
     110 120 130 110 115, 130 130 110  90 115
     110 140 130 100  95, 120 110 120 120 125
     100 120 150 110 105,  90 100 140  80 135 ];
  hardness{r} := [ 8.8 6.1 2.0 4.2 5.0 ];
  Pr:=150; Sto:=500;  vegCapa:=200; oilCapa:=250;
variable
  food{p}    "Quantity of food produced";
  buy{p,r}   "Quantity bought in each period";
  use{p,r}   "Quantity used in each period";
  sto{p,r}   "Quantity stored in each period";
constraint
  Bal{r,p}: if(p=1,Sto,sto[p-1,r]) + buy = use + sto;
  Endstore{r}: sto[#p,r] = Sto;
  Capacity1{p}: sum{v} use <= vegCapa;
  Capacity2{p}: sum{r|~v} use <= oilCapa;
  Hardness{p}: 3*food <= sum{r} hardness*use <= 6*food;
  Conserve{p}: sum{r} use = food;
maximize profit: sum{p}Pr*food - sum{p,r}(pr*buy+5*sto);
Writep(profit,buy,use,sto,food);
end
```

3 Food Manufacture II (will02)

—- Run LPL Code , HTML Document –

Problem: The model is the same as will01, with exception of the additional restrictions:

1. The food may never be made up of more than three different oils.
2. If an oil is used at least 20 tons must be used.
3. If either vegetable oil `Veg1` or `Veg2` is used, then non-vegetable oil `Oil3` must also be used.

Model: The Model is as follows:The problem is from [3] Chapter 12.2. To impose the additional conditions we introduce a 0-1 variable d which is 1 only if the qunatity used (use) of an oil is not zero, that is we must impose:

$$use > 0 \leftrightarrow d = 1$$

Listing 2: The Complete Model implemented in LPL [6]

```
model Will02 "Food Manufacture II";
set p  := [ January February March April May June ];
    r    := [ VEG1 VEG2 OIL1 OIL2 OIL3 ] "all oils";
    v{r} := [ VEG1 VEG2 ]                "veg oils";
parameter pr{p,r} := [
    110 120 130 110 115, 130 130 110  90 115
    110 140 130 100  95, 120 110 120 120 125
    100 120 150 110 105,  90 100 140  80 135 ];
  hardness{r} := [ 8.8 6.1  2.0 4.2 5.0 ];
  Pr:=150; Sto:=500; vegCapa:=200; oilCapa:=250;
variable
  food{p}    "quantity of food produced";
  buy{p,r}   "quantity bought in each period";
  use{p,r}   "quantity used in each period";
  sto{p,r}   "quantity stored in each period";
  binary d{p,r};  // is 1 if use>0
constraint
  Bal{r,p}: if(p=1,Sto,sto[p-1,r])+buy = use+sto;
  endstore{r}: sto[#p,r]=Sto;
  Capacity1{p}: sum{v} use <= 200;
  Capacity2{p}: sum{r|~v} use <= 250;
  Hardness{p}: 3*food <= sum{r} hardness*use <= 6*food;
  Conserve{p}: sum{r} use = food;
  Cond1{p}: sum{r} d <= 3;
  Cond2{p,r} :use-20*d >=0;            -- use>0 -> d=1
  Cond2a{p,r}:use-if(v,200,250)*d <=0; -- d=1 -> use>0
  Cond3{p}: d[p,'VEG1']+d[p,'VEG2']-2*d[p,'OIL3']<=0;
maximize profit: sum{p}Pr*food - sum{p,r}(pr*buy+5*sto);
Writep(profit,buy,use,sto,food);
end
```

4 Factory planning I (will03)

—- Run LPL Code , HTML Document –

Problem: Seven products are manufactured using different machines over the next 6 months. Each product yields a certain contribution to profit. Several machines will be down in different months for maintenance. We also want a given endstock for each product at the end of our time horizont. What must be manufactured, stored and sold each month in order to maximize profit, if the storage is limited and the market also has limitations on each product?

Model: The Model is as follows:The problem is from [3] Chapter 13.3.

Listing 3: The Complete Model implemented in LPL [6]

```
model Will03 "Factory planning I";
set
  p:=[Prod1 Prod2 Prod3 Prod4 Prod5 Prod6 Prod7] "prod";
  m:=[grinder vDrill hDrill borer planer]  "mach";
  t:=[Jan Feb Mar Apr May Jun]   "period";
parameter  profit{p} := [10 6 8 4 11 9 3];
  time{m,p} "machine time table" :=[.5 .7 . . .3 .2 .5,
    .1 .2 . .3 . .6 . , .2 . .8 . . . .6 ,
    .05 .03 . .07 .1 . .08 , . . .01 . .05 . .05];
  down{t,m} "number of machines down" := [
   (Jan,grinder) 1  (Feb,hDrill)  2  (Mar,borer)  1
   (Apr,vDrill)  1  (May,grinder) 1  (May,vDrill) 1
   (Jun,planer)  1  (Jun,hDrill)  1];
  qMach{m} "nr. of machines available" := [4 2 3 1 1];
  upper{t,p} "market limitation of sells" := [500 1000
   300 300 800 200 100, 600 500 200 0 400 300 150, 300
   600 0 0 500 400 100, 200 300 400 500 200 0 100, 0
   100 500 100 1000 300 0, 500 500 100 300 1100 500 60];
  storeCost := 0.5;   storeCapacity := 100;
  endStock  := 50;    hoursMonth    := 2*8*24;
variable
  manu{t,p}            "quantity manufactured";
  held{t,p}            "quantity stored";
  sell{t,p} [0..upper] "quantity sold";
constraint
  Balance{t,p}: held[t-1,p] + manu = sell + held;
  endstore{p}: held[#t,p] = endStock;
  storecapa{t,p|t<#t}: held <= storeCapacity;
  capa{t,m}: sum{p} time*manu <= hoursMonth*(qMach-down);
maximize Profit: sum{t,p} (profit*sell - storeCost*held);
Write('          Manufacture  Sell     Store\n');
Write{t,p}('%5s %5s %8d %8d %8d\n',if(p=1,t&'',''),p,manu,sell,held);
end
```

5 Factory planning II (will04)

—- Run LPL Code , HTML Document –

Problem: Seven products are manufactured using different machines over the next month. Each product yields a certain contribution to profit. Several machines will be down in different months for maintenance. We also want a given endstock for each product at the end of our time horizont. What must be manufactured, stored and sold each month in order to maximize profit, if the storage is limited and the market also has limitations on each product? Instead of stipulating when each machine is down, it is desired to find the best month for each machine to be down. Each machine must be down once in the six months except two grinders. Note that this is the same model as will03 except that `down` is now a variable.

Listing 4: The Complete Model implemented in LPL [6]

```
model Will04 "Factory planning II";
set
  p:=[Prod1 Prod2 Prod3 Prod4 Prod5 Prod6 Prod7] "prod";
  m:=[grinder vertDrill horiDrill borer planer]  "mach";
  t:=[January February March April May June]   "period";
parameter
  profit{p} "profit per p" := [ 10 6 8 4 11 9 3 ];
  time{m,p} "machine time table" := [
          0.5   0.7    .     .    0.3   0.2   0.5
          0.1   0.2    .   0.3    .    0.6    .
          0.2    .    0.8    .     .     .    0.6
          0.05  0.03   .    0.07  0.1    .    0.08
           .     .    0.01   .    0.05   .    0.05 ];
  qMach{m} "nr of machines" := [ 4  2  3  1  1];
  notDown{m}  "nr not-down" := [ 2  .  .  .  .];
  upper{t,p} "market limitation of sells" := [
         500  1000   300   300   800   200   100
         600   500   200     0   400   300   150
         300   600     0     0   500   400   100
         200   300   400   500   200     0   100
           0   100   500   100  1000   300     0
         500   500   100   300  1100   500    60 ];
  storeCost := 0.5;   storeCapacity := 100;
  endStock  := 50;    hoursMonth    := 2*8*24;
variable
  manu{t,p}               "quantity manufactured";
  held{t,p}               "quantity stored";
  sell{t,p}  [0..upper]   "quantity sold";
  integer down{m,t} [0..qMach-notDown]
           "number of machines down in a period";
constraint
  Balance{t,p}: held[t-1,p] + manu = sell + held;
  endstore{p}: held[#t,p]=endStock;
  storecapa{t,p|t<>#t}: held <= storeCapacity;
  capa{t,m}: sum{p} time*manu <= hoursMonth*(qMach-down);
  mustBeDown{m}: sum{t} down = qMach - notDown;
maximize Profit: sum{t,p} (profit*sell - storeCost*held);
Writep(Profit,manu,held,sell,down);
end
```

6 Manpower Planning (will05)

—- Run LPL Code , HTML Document –

Problem: A company is undergoing a number of changes over the next 3 years which will affect its manpower requirements in future years. “Less unskilled” and “more skilled” men are needed. It also has “semi-skilled”. The company must decide its policy with regard the following:

1. Recruitment from outside the company which is limited
2. Retraining from less skilled to more skilled
3. Redundancy and overmanning which causes extra costs and
4. Short-time working

The company declared to minimize the overall redundancy and costs.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.5.

Listing 5: The Complete Model implemented in LPL [6]

```
model Will05 "Manpower Planning";
  set year,y    := [1..3];
      skill,s,t := [Unskilled,SemiSkilled,Skilled];
  parameter
    CurrentStrength{s}     := [2000,1500,1000];
    Requirement{y,s}       := [1000, 1400, 1000,
              500, 2000, 1500,    0, 2500, 2000];
    LeaveFirstYear{s}      := [0.25, 0.20, 0.10];
    LeaveEachYear{s}       := [0.10, 0.05, 0.05];
    ContinueFirstYear{s}   := 1-LeaveFirstYear;
    ContinueEachYear{s}    := 1-LeaveEachYear;
    LeaveDownGraded        := 0.50;
    ContinueDownGraded     := 1-LeaveDownGraded;
    MaxRecruit{s}          := [500, 800, 500];
    MaxRetrainUnskilled    := 200;
    MaxOverManning         := 150;
    MaxShortTimeWorking    := 50;
    RetrainSemiSkilled     := 0.25;
    ShortTimeUsage         := 0.50;
    RetrainCost{s}         := [400,   500,    0];
    RedundantCost{s}       := [200,   500,  500];
    ShortTimeCost{s}       := [500,   400,  400];
    OverManningCost{s}     := [1500, 2000, 3000];
  variable
    LaborForce,L{s,y};
    Recruite,  R{s,y}   [0..MaxRecruit];
    Retrain,   T{s,t,y};
    Redundant, D{s,y};
    ShortTime, S{s,y}   [0..MaxShortTimeWorking];
    OverManned,O{s,y};
  expression
    TotalRetrainCost   : sum{y,s} RetrainCost*T[s,s+1,y];
    TotalRedundantCost  : sum{s,y} RedundantCost*D;
    TotalShortTimeCost  : sum{s,y} ShortTimeCost*S;
```

```
    TotalOverManningCost: sum{s,y} OverManningCost*O;
    TotalManpowerCost:TotalRetrainCost+TotalRedundantCost
      +TotalOverManningCost + TotalShortTimeCost;
    TotalRedundantMen    : sum{s,y} D;
  constraint
    Continuity{s,y}: L =
       ContinueEachYear*if(y=1,CurrentStrength,L[s,y-1])
       + ContinueFirstYear*R + 0.95*T[s-1,s,y]
       + sum{t} ContinueDownGraded*T[s+t,s,y]
       - sum{t} T[s,s-t,y] - T[s,s+1,y] - D;
  RetainMaxUnskilled{y}:
     T['Unskilled','SemiSkilled',y]<=200;
  RetrainingSemiSkilled{y}: T['SemiSkilled','Skilled',y]
     <= RetrainSemiSkilled*L['Skilled',y];
  OverManning{y}: sum{s} O  <= MaxOverManning;
  Requirements{s,y}: L = Requirement+O+ShortTimeUsage*S;
  --minimize  Redundancy: TotalRedundantMen;
  minimize  Cost: TotalManpowerCost;
  Writep(TotalManpowerCost,TotalRedundantMen,
         Recruite,Retrain,ShortTime,OverManned);
end
```

7 Refinery Optimization I (will06)

—- Run LPL Code , HTML Document –

Problem: An oil refinery purchases two crude oil, which are subject to four processes:

1. Distillation which yields light, medium, and heavy naphtha as well as light oil, heavy oil, and residuum
2. Reforming which yields reformed gasoline using the naphthas.
3. Cracking which yields cracked oil and gasoline using the oils
4. Blending which yields the final products: regular and premium petrol, jet fuel, fuel oil, and lube oil. The objective is to maximize profit.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.6.

Listing 6: The Complete Model implemented in LPL [6]

```
model Will06 "Refinery Optimization I";
  set c  := [Crude1, Crude2];
    f    := [LightNaphtha MediumNaphtha HeavyNaphtha
             LightOil HeavyOil Residuum];
    n{f} := [LightNaphtha MediumNaphtha HeavyNaphtha];
    o{f} := [LightOil HeavyOil];
    a    := [CrackedOil CrackedGasoline];
    p:=[PremiumFuel RegularFuel JetFuel FuelOil LubeOil];
    m{p} := [PremiumFuel, RegularFuel];
  parameter
    DistillYld{c,f}:=[0.10 0.20 0.20 0.12 0.20 0.13,
                      0.15 0.25 0.18 0.08 0.19 0.12];
    ReformingYield{n}  := [0.60, 0.52, 0.45];
    CrackingYield{o,a} := [0.68, 0.28,  0.75, 0.20];
    ResiduumLubeOilYield     := 0.50;
    VaporPressure{o}         := [1.0, 0.6];
    VaporPressureCrackedOil  := 1.5;
    VaporPressureResiduum    := 0.05;
    FuelOilRatio{o}          := [.55555, .16666];
    FuelOilRatioCrackedOil   := 4/18;
    FuelOilRatioResiduum     := 1/18;
    OctaneNumberNaphtha{n}   := [90, 80, 70];
    OctaneNumberReformedGas  := 115;
    OctaneNumberCrackedGas   := 105;
    MinOctaneLevel{m}        := [94, 84];
    CrudeDailyAvail{c}       := [20000, 30000];
    DistillCapacity          := 45000;
    ReformingCapacity        := 10000;
    CrackingCapacity         :=  8000;
    MinLubeOil               :=   500;
    MaxLubeOil               :=  1000;
    MinPremiumProdPct        :=   0.40;
    Profit{p}           := [7.00 6.00 4.00 3.50 1.50];
  variable
    FinalProduct{p};     PurchaseCrude{c};
```

```
  DistillOutput{f};    NaphthaForReformGas{n};
  ReformedGasoline;    OilForCracking{o};
  CrackedOutput{a};    NaphthaForMotorFuel{n,m};
  ReformedGasForMotorFuel{m};
  CrackedGasForMotorFuel{m};
  OilForJetFuel{o};      ResiduumForJetFuel;
  CrackOilForJetFuel;    ResiduumForLubeOil;
maximize TotalProfit: sum{p} Profit*FinalProduct;
constraint
  DistillLimit "At most 45000 barrels of crude can be distilled per
      day":
       sum{c} PurchaseCrude  <= DistillCapacity;
  ReformingLimit "At most 10000 barrels of naphtha can be reformed
      per day":
       sum{n} NaphthaForReformGas  <= ReformingCapacity;
  CrackingLimit "At most 8000 barrels of oil can be cracked per day":
       sum{o} OilForCracking  <= CrackingCapacity;
  DistillationBalance{f} "Balance crude inputs to distillation
      outputs":
       sum{c} DistillYld*PurchaseCrude = DistillOutput;
  ReformedGasBalance "Balance naphtha inputs to reformed gas output":
       sum{n} ReformingYield*NaphthaForReformGas  = ReformedGasoline;
  CrackBalance{a} "Balance oil inputs to cracked output":
       sum{o} CrackingYield*OilForCracking = CrackedOutput;
  ResiduumLubeOil "Each barrel of residuum yields 0.5 barrels of lube
       oil":
     ResiduumLubeOilYield*ResiduumForLubeOil = FinalProduct['LubeOil'
        ];
  DivideReformGas "Divide reformed gas to motor fuels":
       ReformedGasoline = sum{m}ReformedGasForMotorFuel;
  DivideNaphtha{n} "Divide naphthas to reformed gas and motor fuels":
       DistillOutput = NaphthaForReformGas + sum{m}
          NaphthaForMotorFuel;
  DivideOil{o} "Divide light/heavy oils to cracking, jet fuel, and
      fuel oil":
       DistillOutput =  OilForCracking + OilForJetFuel + FuelOilRatio*
          FinalProduct['FuelOil'];
  DivideCrackGas "Divide cracked gasoline to motor fuels":
       CrackedOutput['CrackedGasoline'] = sum{m}
          CrackedGasForMotorFuel;
  DivideCrackedOil "Divide cracked oil to jet fuel and fuel oil":
       CrackedOutput['CrackedOil'] =  CrackOilForJetFuel
       + FuelOilRatioCrackedOil*FinalProduct['FuelOil'];
  DivideResiduum "Divide Residuum to jet fuel, fuel oil, and lube oil
      ":
       DistillOutput['Residuum'] =  ResiduumForJetFuel
       + FuelOilRatioResiduum*FinalProduct['FuelOil'] +
          ResiduumForLubeOil;
  BlendMotorFuel{m} "Blending motor fuels from naphtha, reformed gas,
       and cracked gas":
      FinalProduct = sum{n} NaphthaForMotorFuel +
        ReformedGasForMotorFuel + CrackedGasForMotorFuel;
  BlendJetFuel "Blending jet fuel from light/heavy oils, cracked oil
      and residuum":
      FinalProduct['JetFuel'] = sum{o} OilForJetFuel +
        CrackOilForJetFuel + ResiduumForJetFuel;
  MaxVaporPressure "Jet fuel maximum vapor pressure":
      sum{o} VaporPressure*OilForJetFuel + VaporPressureCrackedOil*
```

```
            CrackOilForJetFuel
        + VaporPressureResiduum*ResiduumForJetFuel <= FinalProduct['
            JetFuel'];
    MinPremFuel "Premium fuel at least 40% of Regular":
         FinalProduct['PremiumFuel'] >= MinPremiumProdPct*FinalProduct['
             RegularFuel'];
    MinOctFuel{m} "Motor fuels minimum octane levels":
        sum{n} OctaneNumberNaphtha*NaphthaForMotorFuel
        + OctaneNumberReformedGas*ReformedGasForMotorFuel
        + OctaneNumberCrackedGas*CrackedGasForMotorFuel
        >= MinOctaneLevel*FinalProduct;
   C{c}: PurchaseCrude <= CrudeDailyAvail;
   D: MinLubeOil<= FinalProduct['LubeOil'] <=MaxLubeOil;
  Writep(TotalProfit);
  SetFocus();  // write all variables
  while NextFocus(3) do
    while NextPosition() do
      Write('%-47s␣%d\n', GetName(0), GetValue(0));
    end;
  end
end
```

8 Refinery Optimization II (will06a)

—- Run LPL Code , HTML Document –

Problem: An oil refinery purchases two crude oil, which are subject to four processes:

1. Distillation which yields light, medium, and heavy naphtha as well as light oil, heavy oil, and residuum
2. Reforming which yields reformed gasoline using the naphthas.
3. Cracking which yields cracked oil and gasoline using the oils
4. Blending which yields the final products: regular and premium petrol, jet fuel, fuel oil, and lube oil. The objective is to maximize profit.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.6.

Listing 7: The Complete Model implemented in LPL [6]

```
model Will06a "Refinery Optimization II";
variable
  cr1; cr2; ln; mn; hn; lo; ho; r; lnrg; mnrg; hnrg; rg;
  locgo; hocgo; cg; co; lnpmf; lnrmf; mnpmf; mnrmf;hnpmf;
  hnrmf; rgpmf; rgrmf; cgpmf; cgrmf; lojf; hojf; rjf;
  cojf; lofo; hofo; rfo; cofo;   rlbo; pmf; rmf;
  jf; fo; lbo;
constraint
  r1: cr1 <= 20000;              --availabilities
  r2: cr2 <= 30000;
  r3: cr1+cr2 <= 45000;          --capacity
  r4: lnrg+mnrg+hnrg <= 10000;   --reforming capacity
  r5: locgo+hocgo <= 8000;       --cracking capacity
  r6: 500 <= lbo <= 1000;        --bounds
  r8: 0.1 *cr1 +0.15*cr2 >= ln;  --split under destillation
  r9:  +0.2 *cr1 +0.25*cr2 >= mn;
  r10: +0.2 *cr1 +0.18*cr2 >= hn;
  r11: +0.12*cr1 +0.08*cr2 >= lo;
  r12: +0.2 *cr1 +0.19*cr2 >= ho;
  r13: +0.13*cr1 +0.12*cr2 >= r;
  r14: +0.6*lnrg+0.52*mnrg+0.45*hnrg >= rg; --splitting under reforming
  r15: 0.68*locgo+0.75*hocgo = co; --splitting under cracking
  r16: 0.28*locgo+0.2 *hocgo = cg;
  r17: lnrg+lnpmf+lnrmf = ln;      --continuities
  r18: mnrg+mnpmf+mnrmf = mn;
  r19: hnrg+hnpmf+hnrmf = hn;
  r20: locgo+lojf+0.55*fo = lo; --fixed proportions on fuel oil in
     blending
  r21: hocgo+hojf+0.17*fo = ho;
  r22: cojf+0.22*fo = co;
  r23: rlbo+rjf+0.055*fo = r;
  r24: cgpmf+cgrmf = cg;         --further continuities
  r25: rgrmf+rgpmf = rg;
  r26: +lnpmf+mnpmf+hnpmf+rgpmf+cgpmf = pmf;
  r27: +lnrmf+mnrmf+hnrmf+rgrmf+cgrmf = rmf;
```

```
  r28: +lojf+hojf+cojf+rjf = jf;
  r29: +0.5*rlbo = lbo;  --lube-oil is 0.5 of residuum used
  r30: pmf >= 0.4*rmf;   --pmf/rmf must be 40%
  r31: +90*lnpmf+80*mnpmf+70*hnpmf+115*rgpmf+105*cgpmf >= 94*pmf; --
     octane number
  r32: +90*lnrmf+80*mnrmf+70*hnrmf+115*rgrmf+105*cgrmf >= 84*rmf;
  r33: +lojf+0.6*hojf+1.5*cojf+0.05*rjf <= jf;     --vapour pessure
     constraints
maximize profit: 7*pmf+6*rmf+4*jf+3.5*fo+1.5*lbo;
  Writep(profit);
  while NextFocus(3) do
    while NextPosition() do
      Write('%-10s␣%d\n', GetName(0), GetValue(0));
    end;
  end
end
```

9 Mining (will07)

—- Run LPL Code , HTML Document –

Problem: A mining company extracts ore from several mines over the next five years. The ore from different mines is of varying quality and the extracted amount of ore is limited. Furthermore, royalities are payable on each mine kept open. The company can operate on three of the four available mines at a time. A mine can be closed and no royalities are payable. A closed mine, however, must be closed down permanently. Which mines should be operated each period and how much should they produce? Future revenue and expentiture must be discounted.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.7.

Listing 8: The Complete Model implemented in LPL [6]

```
model Will07 "Mining";
set  m := [Mine1 Mine2 Mine3 Mine4];
  t,t0 := [Year1 Year2 Year3 Year4 Year5];
parameter
  Royalities{m}  := [5  4  4  5];
  ExtractLimit{m}:= [2  2.5 1.3 3];
  OreQuality{m}  := [1 .7 1.5 .5];
  BlendedQuality{t} := [0.9  0.8  1.2  0.6  1.0];
  rate := 10/100;
  sellPrice := 10;
  discountFactor{t} := 1/(prod{t0|t0<t} (1+rate));
variable
  x{m,t}[0..ExtractLimit] "output";
  q{t}                    "quantity of blended ore";
  binary d{m,t}           "mine is working (if x>0)";
  binary g{m,t}           "mine is open";
constraint
  WorkingMines{t}: atmost(3){m} d;
  NoWorkOnClosedMines{m,t}: d -> g;
  SubsequentYearsClosed{m,t|t<#t}: g[m,t+1] -> g;
  Blending{t}: sum{m} OreQuality*x = BlendedQuality*q;
  Combine{t}: sum{m} x = q;
  Working{m,t}: d <- x>0    "mine is working if x>0";
maximize profit: - (sum{m,t} discountFactor*Royalities*g)
                 + sum{t} discountFactor*sellPrice*q;
Writep(profit,x,q,discountFactor,d,g);
end
```

10 Farm Planning (will08)

—- Run LPL Code , HTML Document –

Problem: A farmer wishes to plan production on his farm for the next five years. He has heifers and milk-producing cows. They consume grain and sugar beet which can be grown both on the farm or bought (or sell) on the market. Heifers produce calves which can be sold (the bollocks are sold immediately) or which will become milk-producing cows in the third year. Labour can be recruited from outside and extra capital is also available to some extent. How should the farmer operate to maximize profit over the five years? How much grain and sugar beet should he plant, sell or buy; how much extra labour and capital should he use; and how many cows does he have in each year and of each age?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.8.

Listing 9: The Complete Model implemented in LPL [6]

```
model Will08 "Farm Planning";
set  t,k := [ Year1 Year2 Yera3 Year4 Year5 ];
  i := [ 1 2 3 4 ];   /*land groups*/
  j := [ 1 2 3 4 5 6 7 8 9 10 11 12 ]; --cow age
  j1{j} := [ 2 3 4 5 6 7 8 9 10 11 ];
parameter
  GrainYld{i}  "grain growing"  := [1.1  0.9  0.8  0.65];
  GrainArea{i} "area in arces"  := [ 20   30   20    10];
variable
  GR{i,t}      "grain grown in tons";
  SB{t}        "sugar beet grown in tons";
  GRBUY{t}     "bought grain in tons";
  GRSELL{t}    "sold grain in tons";
  SBBUY{t}     "bought sugar beet in tons";
  SBSELL{t}    "sold sugar beet in tons";
  EXLAB{t}     "extra labour recruited";
  EXCAP{t}     "capital outlay";
  HFSELL{t}    "number of heifers sold at birth";
  COWS{j,t}    "number of cows of age j in year t";
  NEWCOW{t}    "number of cows of age 0 at year t";
  PROF{t}      "profit in year t";
constraint
  ACCOMM{t} "Accommodation (housing capacity)":
    NEWCOW + sum{j|j<#j} COWS <= 130 + sum{k|k<=t} EXCAP;
  GRCONS{t} "Grain consumption":
    sum{j1} 0.6*COWS[j1,t] <= sum{i} GR + GRBUY - GRSELL;
  SBCONS{t} "Sugar beet consumption":
    sum{j1} 0.7*COWS[j1,t] <= SB + SBBUY - SBSELL;
  BOUNDS{i,t} "Grain growing 'capacity'":
    GR <= GrainYld*GrainArea;
  ACREAS{t} "Acreage limitation":
    sum{i} 1/GrainYld*GR + 2/3*SB + 2/3*NEWCOW
    + 2/3*COWS[1,t] + sum{j1} COWS[j1,t] <= 200;
  LABOUR{t} "Labour in 100 hours":
    0.1*NEWCOW + 0.1*COWS[1,t] + sum{j1} 0.42*COWS[j1,t]
    + sum{i} 0.04/GrainYld*GR + 0.14/1.5*SB <= 55+EXLAB;
  -- continuity constraints
  CTA{t|t<#t}: COWS[1,t+1] = 0.95*NEWCOW[t];
  CTB{t|t<#t}: COWS[2,t+1] = 0.95*COWS[1,t];
```

```
  CT{j1,t|t<#t}:COWS[j1+1,t+1] = 0.98*COWS[j1,t];
  CTC{t}: NEWCOW - sum{j1} 0.55*COWS[j1,t] + HFSELL = 0;
  ENDT "Cows at the end": 50<= sum{j1} COWS[j1,5] <=175;
  PROFIT{t}:
    sum{j1} 16.5*COWS[j1,t] + 40*HFSELL + 120*COWS[12,t]
    + sum{j1} 370*COWS[j1,t] + 75*GRSELL + 58*SBSELL
    - 90*GRBUY - 70*SBBUY - 120*EXLAB - 50*NEWCOW
    - 50*COWS[1,t] - sum{j1} 100*COWS[j1,t]
    - sum{i}15/GrainYld*GR - 20/3*SB - sum{t}39.71*EXCAP
    - PROF = 4000;
  -- initial conditions, fixing number of cows of each age at first
     period
  BOUNDS1{j}: COWS[j,1] = if(j<3 , 9.5 , 9.8);
-- maximize profit minus discounted capital expenditures
maximize TProf: sum{t} (PROF-158.85*EXCAP-39.71*t*EXCAP);
Writep(TProf,GR,SB,GRBUY,GRSELL,SBBUY,SBSELL,EXLAB,
       EXCAP,HFSELL,COWS,NEWCOW,PROF);
end
```

11 Economic Planning (will09)

—- Run LPL Code , HTML Document –

Problem: An economy consists of the three industries coal, steel, and transport. Each industry requires inputs from its own, inputs from the other industries, and manpower. Output may also be used to build productive capacity, which become effective two periods later. Manpower capacity is limited and an initial stock and productive capacity is given. We wanted to know the growth patterns for the economy pursuing the following objectives:

1. Maximizing the productive capacity at the end of the considered time while meeting an exogenous demand
2. Maximizing total production the the last two periods ignoring the exogenous demand (set to zero)
3. Maximizing the total manpower requirement ignoring the manpower capacity limitation while meeting the exogenous demand.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.9.

Listing 10: The Complete Model implemented in LPL [6]

```
model Will09 "Economic Planning";
  set  o,o1 := [Coal Steel Transport] "activities";
       t,t1 := [1..6] "periods";
  parameter
    req{o1,o} := [.1 .5 .4 , .1 .1 .2 , .2 .1 .2];
    cap{o1,o} := [.0 .7 .9 , .1 .1 .2 , .2 .1 .2];
    Mreq{o}   := [.6  .3  .2];
    Mcap{o}   := [.4  .2  .1];
    Stocks{o} := [ 150   80  100 ] "initial stock" ;
    Capa  {o} := [ 300  350  280 ] "capa. in period 0";
    consum{o} := [  60   60   30 ] "exo. demand";
    consum1{o}:= consum "make a copy of consum";
    lastx {o} := [166.4  105.7  92.3] "min. end output";
    manCapa   := 470 "labour capacity";
  variable
    x{o,t} [if(t=#t,lastx,0)..99999]   "outputs in t";
    s{o,t}             "stock level at beginning of t";
    y{o,t|t<#t-1} "extra productive capacity for t+1";
  constraint
    TotOutput{o1,t}: sum{o} (req*x+cap*y) + s <=
       x[o1,t-1] + s[o1,t-1] + if(t=1,Stocks,-consum);
    Manpower{t}: sum{o} (Mreq*x+Mcap*y) <= manCapa;
    prodCapa{o,t|t>1}: x - sum{t1|t1<t} y <= Capa;
  maximize obj: sum{o} Capa + sum{o,t} y[o,t];
  Write('-- (1) maximize the productive capacity\n');
  Writep(obj,y,x,s);
  consum{o}:=0;  /*ignore the exogenous demand*/
  maximize obj1: sum{o,t|t>=#t-1} x[o,t];
  Write('-- (2) maximize the final output\n');
  Writep(obj1,y,x,s);
  consum{o}:=consum1; /*recopy the original consum*/
```

```
  maximize obj2: sum{t,o} (Mreq*x[o,t+1] + Mcap*y[o,t+2])
    subject_to (Will09, ~Manpower);
  Write('-- (3) maximize the manpower requirement\n');
  Writep(obj2,y,x,s);
end
```

12 Decentralization (will10)

—- Run LPL Code , HTML Document –

Problem: A large company wants to move 5 of its departments out of London (to Bristol or Brighton). There are benefits (cheaper housing etc.) and costs (communication) to be derived from doing this. Where should each department be located so as to minimize overall yearly costs?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.10.

Listing 11: The Complete Model implemented in LPL [6]

```
model Will10 "Decentralization";
set D,i,k := [A B C D E] "Departements";
    C,j,l := [Bristol Brighton London] "Cities";
parameter
  Benefit{D,C}:=[10 10 0 15 20 0 10 15 0 20 15 0 5 15 0];
  Cost{C,C}:= [5 14 13, 14 5 9, 13 9 10] "Comm. costs";
  Comm{D,D}:=[(A,C) 1.0, (A,D) 1.5, (B,C) 1.4, (B,D) 1.2,
     (C,E) 2.0, (D,E) 0.7] "quant. of communications";
set CO{i,j,k,l} := k>i and Comm[i,k];
binary variable
  d{D,C}  "departement d is in city c";
  g{CO}; -- g[CO[i,j,k,l] <-> d[i,j] and d[k,l];
constraint
  -- each department must be located in exactly one city
  Dept{D}: sum{C} d = 1;
  -- at most 3 departments be located in the same city
  City{C}: atmost(3){D} d;
  La{CO[i,j,k,l]}: (g[CO]->d[i,j]) and (g[CO]->d[k,l]);
  Lc{CO[i,j,k,l]}: d[i,j] and d[k,l] -> g[CO];
minimize TCOST: -(sum{D,C|C<>'London'} Benefit*d)
   + sum{CO[i,j,k,l]} Comm[i,k]*Cost[j,l]*g[CO];
Write('Total communication cost %d\n\n', -1000*TCOST);
Write{C|exist{D} d}('Locate departements %2s in %s\n',
      {D|d} D , C);
end
```

This problem is a quadratic assignment problem and objective function in fact would be:

```
minimize TCOST: -(sum(D,C|C<>'London') Benefit*d
 +sum([i,j,k,l]=CO) Comm[i,k]*Cost[j,l]*d[i,k]*d[j,l];
```

To solve this with a standard linear solver, it is possible to reformulate the model as a linear integer program by the following steps

1. replace xy by a new 0-1 variable z (x and y are also 0-1 variables)
2. impose the logical condition: $z = 1 \leftrightarrow x = 1 \wedge y = 1$ by means of the three constraints: $z - x \leq 0$, $z - y \leq 0$, $x + y - z \leq 1$.

An alternative is the objective in the model code together with the two constraints `La` and `Lb` .

13 Curve Fitting (linear) (will11)

—- Run LPL Code , HTML Document –

Problem: Two quantities x and y are known to be correlated by a straight line $y = ax + b$. A number of corresponding values are known.

1. Find a straight line, such that the sum of absolute deviation of all corresponding values are minimized.

2. Find a straight line, such that the maximum deviation is minimized.

(Note that the standard way to model this is by regression and we minimize the sum of the quadratic deviations. In this approach different measures of "deviations" are used.)

Modeling Steps: The problem is from [3]. For a problem description see Chapter 13.11.

Define the data: A set I of points (x, y) is given. Model this be two vectors x_i and y_i, for each point $i \in I$.

1. The goal is to find a linear function: $y = Ax + B$ where A and B are unknown.

2. For each given point, we introduce a unknown positive P and negative N deviation from the line $y = Ax + B$. Hence, the following condition must hold:

$$y_i = Ax_i + B + P_i - N_i \quad \text{forall } i \in I$$

3. In a first round we minimize the sum of all deviations, that is:

$$\min \sum_i (P_i + N_i)$$

4. To minimize the maximal deviation, we introduce a new variable $maxdev$ and add the conditions $maxdev \geq P_i$ and $maxdev \geq N_i$ for all points i. Now $maxdev$ is larger than any P_i or N_i, that means it defines the maximal deviation. Finally, we minimize $maxdev$.

Listing 12: The Complete Model implemented in LPL [6]

```
model Will11 "Curve Fitting (linear)";
  set i := 1..19; -- 19 points are known
  parameter
    x{i}:=[0.0 0.5 1.0 1.5 1.9 2.5 3.0 3.5 4.0 4.5
           5.0 5.5 6.0 6.6 7.0 7.6 8.5 9.0 10.0];
    y{i}:=[1.0 0.9 0.7 1.5 2.0 2.4 3.2 2.0 2.7 3.5,
           1.0 4.0 3.6 2.7 5.7 4.6 6.0 6.8 7.3];
  variable
    A [-10..10]; B [-10..10]; maxdev;
    P{i}; N{i};
  constraint
    DEV{i}: y = A*x + B + P - N;
    ULA{i}: maxdev >= P;
    ULB{i}: maxdev >= N;
  minimize deviationSum: sum{i} (P+N);
  Write('Min. deviation sum: y=%7.4f*x + %7.4f\n', A,B);
  parameter a:=A; b:=B;
```

```
  minimize minmax: maxdev;
  Write('Min. max deviation: y=%7.4f*x + %7.4f\n', A,B);
  Draw.Scale(40,-40);
  x{i}:=x+1; y{i}:=y+1;
  {i} Draw.Circle(x,y,.07);
  Draw.Line(0,b,13,a*13+b,5);
  Draw.Line(0,B,13,A*13+B,11);
end
```

Solution: The solution is given as follows:

```
Minimize deviation sum:      y =  0.64*x +  0.58
Minimize maximum deviation: y =  0.63*x + -0.40
```

The two functions are depicted in Figure 1

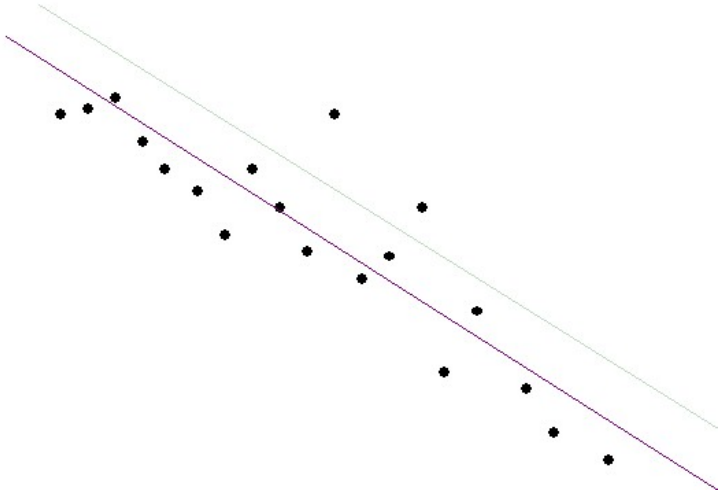


Figure 1: Linear Solution

Questions

1. Suppose now that a quadratic (not a linear) function $y = Ax^2 + Bx + C$ has to be found. How should A, B, and C be chosen such that (1) the sum of absolute deviation, (2) the maximum deviation is minimized?

Answers

1. The only part in the model that changes is the constraint `DEV`. It must be replaced by the following constraint:

   ```
   DEV{i}: y = A*x^2 + B*x + C + P - N;
   ```

   The model can be found and ran at: will11a.

The solution is:

```
Minimize deviation sum: y =  0.03*x^2 + 0.29*x + 0.98
Minimize max deviation: y =  0.13*x^2 - 0.62*x + 2.47
```

The two function are depicted in Figure 2

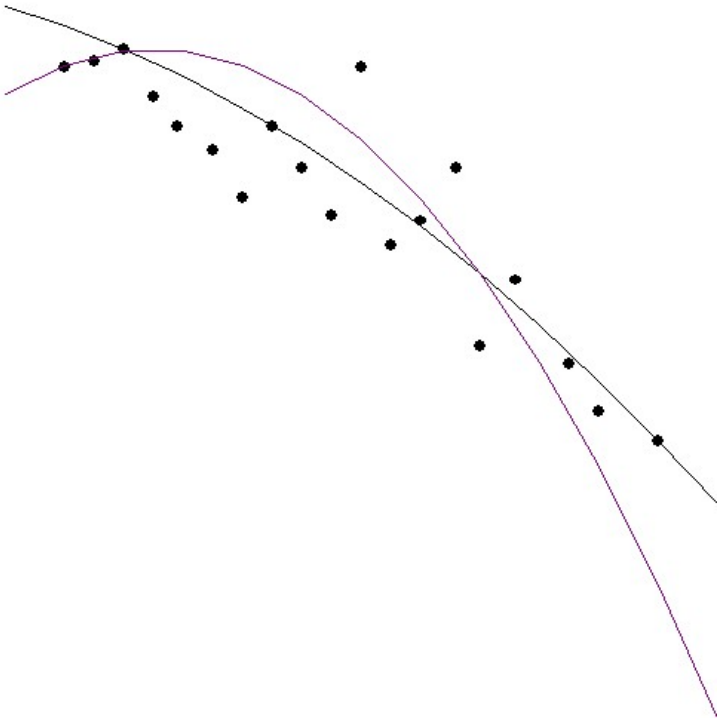


Figure 2: Quadratic Solution

14 Curve Fitting (quadratic) (will11a)

—- Run LPL Code , HTML Document –

Problem: This model calculates the best quadratic curve fitting. For a problem description see model will11.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.11.

Listing 13: The Complete Model implemented in LPL [6]

```
model Will11a "Curve Fitting (quadratic)";
  set i := 1..19;  -- 19 points are given
  parameter
    x{i}:=[0.0 0.5 1.0 1.5 1.9 2.5 3.0 3.5 4.0 4.5
           5.0 5.5 6.0 6.6 7.0 7.6 8.5 9.0 10.0];
    y{i}:=[1.0 0.9 0.7 1.5 2.0 2.4 3.2 2.0 2.7 3.5,
           1.0 4.0 3.6 2.7 5.7 4.6 6.0 6.8 7.3];
  variable
    A [-10..10]; B [-10..10]; C [-10..10]; maxdev;
    P{i}; N{i};
  constraint
    DEV{i}: y = A*x^2 + B*x + C + P - N;
    ULA{i}: maxdev - P >= 0;
    ULB{i}: maxdev - N >= 0;
  minimize deviationSum: sum{i} (P+N);
  Write('Min dev sum: y = %7.4f*x*x + %7.4f*x + %7.4f\n', A,B,C);
  parameter a:=A; b:=B; c:=C;
  minimize minmax: maxdev;
  Write('Min max dev: y = %7.4f*x*x + %7.4f*x + %7.4f\n', A,B,C);
  x{i}:=x+1; y{i}:=y+1;
  Draw.Scale(20,-20);
  {i} Draw.Circle(x,y,.07);
  {i in 1..12}
    Draw.Line(i-1,a*(i-1)^2+b*(i-1)+c,i,a*i^2+b*i+c);
  {i in 1..12}
    Draw.Line(i-1,A*(i-1)^2+B*(i-1)+C,i,A*i^2+B*i+C,5);
end
```

15 Logical Design (will12)

—- Run LPL Code , HTML Document –

Problem: A logical circuit has to be designed on the base of **nor**-gates such that it can perform a certain logical function. We may suppose that 7 gates will do it. The gates are connected as specified by a *InOut*-table. How many gates and which ones (of the seven) are needed to produce a given final output – here the **xor**-gate ?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.12. This problem also appeared in Williams H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

Listing 14: The Complete Model implemented in LPL [6]

```
model Will12 "Logical Design";
set
  i,j := [Gate1 Gate2 Gate3 Gate4 Gate5 Gate6 Gate7];
  J:= [1 2];       --maximal number of inputs on a gate
  K:= [1 2 3 4];  --4 combi. are given by a true table
  InOut{i,j} :=
    [Gate1 Gate2 , Gate1 Gate3
     Gate2 Gate4 , Gate2 Gate5
     Gate3 Gate6 , Gate3 Gate7];
parameter
  ALPHA{K,J}:= [ 0,0  0,1  1,0  1,1 ];
  Output{K} := [ 0 1 1 0 ];    -- this is the XOR gate
binary variable
  s{i};       -- =1 if the gate is needed, else 0
  t{i,J};     -- left and right input signal at a gate
  OUT{i,K};  -- output signal on the four combinations
constraint
  ENOR{i,J}: s-t >= 0;
  INNOR{i|exist{j} InOut[i,j]}:
     sum{j|InOut[i,j]} s[j] + sum{J} t[i,J] <= 2;
  LO{K,i,j|InOut[i,j]}: OUT[i,K] + OUT[j,K] <= 1;
  LC{K,i,J}: ALPHA*t+OUT <= 1;
  LOGC{K,i}: sum{j|InOut[i,j]} OUT[j,K]
                    +sum{J} ALPHA*t + OUT - s >= 0;
  POUT{i,K}: s-OUT >= 0;
  BOUNDS{K}: OUT[1,K] = Output[K];
  BOUNDS1: s[1] >= 1;  --force a gate to exist to avoid a trivial
     solution
minimize NumberOfGates: sum{i} s;
Writep(s,t);
end
```

Explanation: The **nor**-gate is a device that has exactly two entries a and b with value 0 or 1, and exactly one output c with value 0 or 1. The device simulates the Boolean **nor** (not or) function: $c = \neg(a \vee b)$. We use also the operator | for the nor operation, hence, it can also be written as $c = a|b$ (Figure 3).

Every Boolean operation can be expressed by a combination of **nor**-operations. For example, we have for the **and**-operation as:

$$a \wedge b \equiv (a|0)|(b|0)$$

a	b	a|b
1	1	0
1	0	0
0	1	0
0	0	1

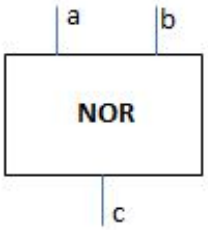
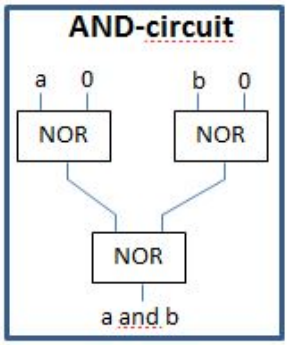


Figure 3: The **nor**-operation, and the **nor** gate

The **xor**-operation can be modeled as:

$$a \dot{\vee} b \equiv ((a|0)|(b|0)) \mid (a|b)$$

a	b	a and b	a xor b
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0



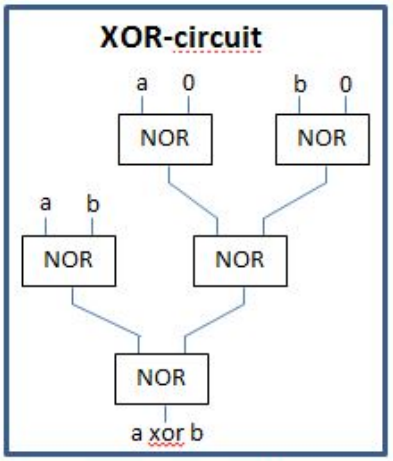


Figure 4: The **and/xor** circuit

Note that the solution of the model exposes exactly the circuit given in Figure 4.

16 Market Sharing I (will13)

—- Run LPL Code , HTML Document –

Problem: A distribution company with two division A and B has to allocate market share between its divisions. The company supplies 23 retailers with oil and spirit. Division A should control 40 (plus or minus 5%) of the company's operation with respect of the following criteria:

1. number of delivery points
2. spirit market
3. oil market in region 1
4. oil market in region 2
5. oil market in region 3
6. number of retailer with good/bad prospects

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.13. This problem also appeared in Williams H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

Listing 15: The Complete Model implemented in LPL [6]

```
model Will13 "Market Sharing I";
set  r :=  1..23;                          --all retailers
  r1{r}:= [1 2 3 4 5 6 7 8];               --in 1st region
  r2{r}:= [9 10 11 12 13 14 15 16 17 18];--in 2nd region
  r3{r}:= [19 20 21 22 23];                --in 3rd region
  k    := [1 2 3 4 5 6];                   --six criteria
  l    := [delivery spirit oil prospects];
parameter T{l,r} :=
  [11,47 ,44,25 ,10,26 ,26,54 ,18 ,51,20,105, 7,16,34,
     100,50 ,21,11,19,14,10,11  --number of deliveries
   34,411,82,157,5 ,183,14,215,102,21,54, 0 , 6,96,118,
     112,535,8 ,53,28,69,65,27  --size of spirit market
    9, 13,14, 17,18, 19,23, 21,  9,11,17, 18,18,17, 22,
      24, 36,43, 6,15,15,25,39  --size of oil market
    1, 1,  1,  0, 1,  1, 0,  0,  0, 1, 0,  0, 0, 0,  1,
       0,  0, 0, 0, 1, 0, 0, 0]; --good/bad prospects
  Total{k} := [730, 2395, 134, 215, 100, 8];
variable
  P{k};       -- positive deviation
  N{k};       -- negative deviation
  MAXDEV [0..120];
  binary M{r}; -- 1 if retailer is assigned to division A
constraint
  DELIVER:sum{r}  T[1,r] *M[r] +P[1]-N[1] = 0.4*Total[1];
  SPIRITS:sum{r}  T[2,r] *M[r] +P[2]-N[2] = 0.4*Total[2];
  OILA:   sum{r1} T[3,r1]*M[r1]+P[3]-N[3] = 0.4*Total[3];
  OILB:   sum{r2} T[3,r2]*M[r2]+P[4]-N[4] = 0.4*Total[4];
  OILC:   sum{r3} T[3,r3]*M[r3]+P[5]-N[5] = 0.4*Total[5];
  GROWTH: sum{r}  T[4,r] *M[r] +P[6]-N[6] = 0.4*Total[6];
  DEVA{k}: MAXDEV-P >= 0;
```

```
  DEVB{k}: MAXDEV-N  >= 0;
  BOUND1{k}: P <= 0.05*Total >= N;
--minimize SUMDEV: sum{k} Total*(P+N) + 8*(P[6]+N[6]);
--Writep(SUMDEV,MAXDEV,P,N,M);
minimize minmax: MAXDEV;
Writep(MAXDEV,P,N,M);
end
```

17 Market Sharing II (will13a)

—- Run LPL Code , HTML Document –

Problem: A distribution company with two division A and B has to allocate market share between its divisions. The company supplies 23 retailers with oil and spirit. Division A should control 40 (plus or minus 5%) of the company's operation with respect of the following criteria

1. number of delivery points
2. spirit market
3. oil market in region 1
4. oil market in region 2
5. oil market in region 3
6. number of retailer with good prospects
7. number of retailer with good/bad prospects

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 12.13. This problem also appeared in WILLIAMS H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

Extra tighter forms of constraints OILA, OILB, OILC have been added. These have been derived by LP using roofs and ceilings of the original constraints.

Listing 16: The Complete Model implemented in LPL [6]

```
model Will13a "Market Sharing II";
set  r :=  1..23;                              --all retailers
  r1{r}:= [1 2 3 4 5 6 7 8];                   --in 1st region
  r2{r}:= [9 10 11 12 13 14 15 16 17 18];--in 2nd region
  r3{r}:= [19 20 21 22 23];                    --in 3rd region
  k    := [1 2 3 4 5 6];                       --six criteria
  l    := [delivery spirit oil prospects];
parameter  T{l,r} :=
  [11,47 ,44,25 ,10,26 ,26,54 ,18 ,51,20,105, 7,16,34,
     100,50 ,21,11,19,14,10,11  --number of deliveries
   34,411,82,157,5 ,183,14,215,102,21,54, 0 , 6,96,118,
     112,535,8 ,53,28,69,65,27  --size of spirit market
    9, 13,14, 17,18, 19,23, 21,  9,11,17, 18,18,17, 22,
      24, 36,43, 6,15,15,25,39  --size of oil market
    1, 1,  1,  0, 1,  1, 0,  0,  0, 1, 0,  0, 0, 0,  1,
       0,  0, 0, 0, 1, 0, 0, 0]; --good/bad prospects
  Total{k} := [730, 2395, 134, 215, 100, 8];
  -- tighter lower and upper bounds
  UOA{r}:=[5,7,8,9,10,10,13,12];
  LOA{r}:=[2,3,3,4, 4, 4, 5, 5];
  UOB{r}:=[. . . . . 8,8,13,14,14,13,17,19,28,34];
  LOB{r}:=[. . . . . 8,9,14,15,15,14,18,20,30,36];
  UOC{r}:=[. . . . . 1,2,2,3,4];
  LOC{r}:=[. . . . . 1,2,2,3,5];
variable
  P{k};      -- positive deviation
```

```
  N{k};      -- negative deviation
  MAXDEV [0..120];
  binary M{r};  -- 1 if retailer is assigned to division A
constraint
  DELIVER: sum{r}  T[1,r] *M[r] +P[1]-N[1]=0.4*Total[1];
  SPIRITS: sum{r}  T[2,r] *M[r] +P[2]-N[2]=0.4*Total[2];
  OILA:    sum{r1} T[3,r1]*M[r1]+P[3]-N[3]=0.4*Total[3];
  OILB:    sum{r2} T[3,r2]*M[r2]+P[4]-N[4]=0.4*Total[4];
  OILC:    sum{r3} T[3,r3]*M[r3]+P[5]-N[5]=0.4*Total[5];
  GROWTH:  sum{r}  T[4,r] *M[r] +P[6]-N[6]=0.4*Total[6];
  NUOILA: sum{r1} UOA[r1]*M[r1] <= 33;
  NLOILA: sum{r1} LOA[r1]*M[r1] >= 11;
  NUOILB: sum{r2} UOB[r2]*M[r2] <= 75;
  NLOILB: sum{r2} LOB[r2]*M[r2] >= 63;
  NUOILC: sum{r3} UOC[r3]*M[r3] <= 5;
  NLOILC: sum{r3} LOC[r3]*M[r3] >= 5;
  DEVA{k}: MAXDEV-P >= 0;
  DEVB{k}: MAXDEV-N  >= 0;
  BOUND1{k}: P <= 0.05*Total >= N;
--minimize SUMDEV: sum{k} Total*(P+N) + 8*(P[6]+N[6]);
--Writep(SUMDEV,MAXDEV,P,N,M);
minimize MINMAX: MAXDEV;
Writep(MAXDEV,P,N,M);
end
```

18 Opencast Mining I (will14)

—- Run LPL Code , HTML Document –

Problem:

```
 |   |   |   |   |  16 blocks (B15-B30)
-------------------
  |   |   |   |     9 blocks (B6-B14)
 -----------------
    |   |   |       4 blocks (B2-B5)
  ---------------
      |   |         1 block  (B1)
```

The problem is to opencast mine within a limited area. Ore body and overburden are divided into blocks (e.g. 50 foot cubes). Each block has a certain revenue and a cost of extracting it. There is a permissible angle of slip (45 degree), so that a block can only be extracted if the (four) blocks above it have also been removed. Net income must be maximized.

Model: The Model is as follows:he problem is from [3]. For a problem description see Chapter 13.14. This problem appeared also in Williams H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

We define a binary variable D_i for each block i which is 1 if the block is extracted. The only restriction is that every block i laying on the top of a specific block j must also be removed. The condition $D_i = 1 \leftrightarrow D_j = 1$ where i is a block above j means, that block i must be removed if and only if block j is removed. This can be modeled as $D_i - D_j >= 0$.

Listing 17: The Complete Model implemented in LPL [6]

```
model Will14 "Opencast Mining I";
set i,j := [B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13
     B14 B15 B16 B17 B18 B19 B20 B21 B22 B23 B24 B25
     B26 B27 B28 B29 B30] "30 Blocks on 4 levels";
  LayingOn{i,j} :=
   [ (B1 ,*) B2 B3 B4 B5        (B2 ,*) B6 B7 B9 B10
     (B3 ,*) B7 B8 B10 B11      (B4 ,*) B9 B10 B12 B13
     (B5 ,*) B10 B11 B13 B14    (B6 ,*) B15 B16 B19 B20
     (B7 ,*) B16 B17 B20 B22    (B8 ,*) B17 B18 B21 B22
     (B9 ,*) B19 B20 B23 B24    (B10,*) B20 B21 B24 B25
     (B11,*) B21 B22 B25 B26    (B12,*) B23 B24 B27 B28
     (B13,*) B24 B25 B28 B29    (B14,*) B25 B26 B29 B30 ];
parameter
   -- Revenue of a block if excavated
  Rev{i}:=[12 ,  10 8 24 12,   4 4 1 6 6 2 8 8 4,
    1.5,1.5,1,0.5,2,2,1.5,1,3,4,3,1.5,3,3,3,1.5];
   -- Cost of a block if excavated
  Cost{i}:=[10,   8 8  8  8,   6 6 6 6 6 6 6 6 6,
      3   3   3   3,3,3,  3,3,3,3,3,  3,3,3,3, 3];
binary variable D{i} "1 if the block is extracted";
constraint CC{LayingOn[i,j]}: D[i]-D[j] <= 0;
maximize pr: sum{i} (Rev-Cost)*D;
Write('Profit=%4.1f\nExcavated blocks:%4s\n',pr,{i|D} i);
end
```

19 Opencast Mining II (will14a)

—- Run LPL Code , HTML Document –

Problem: In an opencast mining 30 blocks have to be excavated potential. Excavating one block cost an amount of money but returns also an amount of money. The problem is to select the subset of blocks such that total profit is maximal. The restriction is that if a block is excavated then all blocks laying on it must also be excavated.

Modeling Steps: This is the same problem as defined in will14. There the problem is formulated as a integer program, that is, as a mathematical model using integer variables. It has been shown that the problem can be translated into a **max flow problem** in a digraph. See [1] p. 731ff, and [2] p. 49, where the problem is to find the *optimal closure in a digraph*. The problem is to choose an optimal subset of "projects" where each project has a (positive or negative) benefit. There are restrictions of the form: if a certain project is chosen then some other projects must also be chosen. The problem is to choose the subset with the maximal total benefit.

The problem of open mining can be formulated by creating the following digraph $G(V, A)$. Let the vertices be the set of the blocks (V). Each block has a benefit (profit) b_i with $i \in V$ (return-cost). There is an arc (i, j), if choosing block i implies choosing block j, that is, block j laying on block i. The arc set is $(i, j) \in A$.

To formulate this problem as an max-flow problem, we build another graph $G'(V', A')$ as follows: Add two vertices s (*source*) and t (*sink*): $(V' = V \cup \{s, t\})$. For each vertex $j \in V | b_j > 0$, add an arc (s, j) to G' with capacity b_j. For each vertex $i \in V | b_i < 0$, add an arc (i, t) to G' with capacity $-b_i$. For each arc $(i, j) \in E$ add an arc to G' with capacity ∞. Hence:

$$E' = E \cup \{(s, j) | j \in V, b_j > 0\} \cup \{(i, t) | j \in V, b_i < 0\}$$

Now calculate the max-flow from source s to sink t.

Listing 18: The Complete Model implemented in LPL [6]

```
model Will14a "Opencast Mining II";
set i,j := [B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13
     B14 B15 B16 B17 B18 B19 B20 B21 B22 B23 B24 B25
     B26 B27 B28 B29 B30] "30 Blocks on 4 levels";
  LayingOn{i,j} :=
   [ (B1 ,*) B2 B3 B4 B5        (B2 ,*) B6 B7 B9 B10
     (B3 ,*) B7 B8 B10 B11      (B4 ,*) B9 B10 B12 B13
     (B5 ,*) B10 B11 B13 B14    (B6 ,*) B15 B16 B19 B20
     (B7 ,*) B16 B17 B20 B22    (B8 ,*) B17 B18 B21 B22
     (B9 ,*) B19 B20 B23 B24    (B10,*) B20 B21 B24 B25
     (B11,*) B21 B22 B25 B26    (B12,*) B23 B24 B27 B28
     (B13,*) B24 B25 B28 B29    (B14,*) B25 B26 B29 B30];
parameter
  Rev{i}:=[12 ,  10 8 24 12,    4 4 1 6 6 2 8 8 4,
     1.5,1.5,1,0.5,2,2,1.5,1,3,4,3,1.5,3,3,3,1.5];
  Cost{i}:=[10,    8 8  8  8,    6 6 6 6 6 6 6 6 6,
      3   3   3   3,3,3,   3,3,3,3,3,   3,3,3,3, 3];
  Profit{i}:=Rev-Cost;
  Addm(i,'s'); Addm(i,'t');  //add source/sink
parameter e{i,j}:=
    if(LayingOn,99,
       j=#j and i<#i-1 and Profit[i]<0,-Profit[i],
       i=#i-1 and j<#j-1 and Profit[j]>=0,Profit[j]);;
```

```
  e[#i,#i-1]:=100;
variable x{i,j|e} [0..e];
constraint A{i}: sum{j} x[i,j] - sum{j} x[j,i] = 0;
maximize obj: x[#i,#i-1];
parameter dd{i,j}:=GetValue(x,3);
set Dual{i,j|i<#i-1 and j<#j-1}:=GetValue(x,3);
set S{i}:= exist{j} Dual[j,i];
  {i,j} if(S and LayingOn,S[j]:=1);
  // draw the result as a graph G'
  Draw.Scale(20,20);
parameter X{i}; Y{i};;   --coordinates for the drawing
  X[31]:=0; Y[31]:=20;                          --source s
  X[1]:=10;  Y[1]:=15;                         --1st layer
  {i|i>1 and i<=5}(X[i]:=20, Y[i]:=3*(i-2)+10);    --2nd
  {i|i>5 and i<=14}(X[i]:=30, Y[i]:=3*(i-6)+5);    --3rd
  {i|i>14 and i<=30}(X[i]:=40, Y[i]:=2.5*(i-15)); --4th
  X[32]:=50; Y[32]:=20;                          --sink t
  {e[i,j]} Draw.Arrow(if(e=99,'',''&Round(e,-1)),
                      X[i],Y[i],X[j],Y[j],1);
  {e[i,j]|S and j<#j-1} Draw.Arrow(if(e=99,'',''&
           Round(e,-1)),X[i],Y[i],X[j],Y[j],1,3,3);
  {i} Draw.Circle(''&i,X,Y,1,if(S,3,1),0);
  Draw.Text('Red␣Blocks␣are␣excavated',0,3,30);
  Draw.Text('Profit='&sum{S[i]} Profit,0,5,30);
end
```

Solution: The total benefit is 17.5, the same result as in will14. To find the blocks to excavate, we inspect the dual variables from which we can reconstruct the set of the blocks.

20 Opencast Mining III (will14b)

—- Run LPL Code , HTML Document –

Problem:

```
 |   |   |   |   |  16 blocks (B15-B30)
-------------------
  |   |   |   |     9 blocks (B6-B14)
 -----------------
    |   |   |       4 blocks (B2-B5)
  ---------------
      |   |         1 block  (B1)
```

The problem is to opencast mine within a limited area. Ore body and overburden are divided into blocks (e.g. 50 foot cubes). Each block has a certain revenue and a cost of extracting it. There is a permissible angle of slip (45 degree), so that a block can only be extracted if the (four) blocks above it have also been removed. We define a binary variable D for each block which is 1 if the block is extracted. The return on investment must be maximized.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.14. This problem is the same as will14 except that not the net income but the return on investment must be maximized. This problem appeared also in Williams H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

Return on investment can be expressed as the ratio given by:

$$\frac{\sum_i Rev_i D_i}{\sum_i Cost_i D_i}$$

This function is not linear. In order to linearize it, we need to transform this function by the following steps:

1. Introduce a new continuous variable y.
2. Equating y to the return of investment and transform it gives:

$$\sum_i Cost_i D_i y - \sum_i Rev_i D_i = 0$$

3. Replace the expression $D_i y$ by a new variable z_i (for each block i).
4. Impose the condition $z \leftrightarrow D_i y$ by the following three constraints:

$$z_i \leq 10 D_i \quad , \quad z_i \leq y \quad , \quad -z_i + y + 10 D_i \leq 10 \quad , \quad \text{forall } i$$

To avoid of getting a 0/0 objective value by excavating nothing, we impose that at least one block must be excavated.

Listing 19: The Complete Model implemented in LPL [6]

```
model Will14b "Opencast Mining III";
set  i,j "30 Blocks on 4 levels";
  LayingOn{i,j} :=
```

```
    [(B1 ,*) B2 B3 B4 B5        (B2 ,*) B6 B7 B9 B10
     (B3 ,*) B7 B8 B10 B11      (B4 ,*) B9 B10 B12 B13
     (B5 ,*) B10 B11 B13 B14    (B6 ,*) B15 B16 B19 B20
     (B7 ,*) B16 B17 B20 B22    (B8 ,*) B17 B18 B21 B22
     (B9 ,*) B19 B20 B23 B24    (B10,*) B20 B21 B24 B25
     (B11,*) B21 B22 B25 B26    (B12,*) B23 B24 B27 B28
     (B13,*) B24 B25 B28 B29    (B14,*) B25 B26 B29 B30];
parameter
  Rev{i}:=[12 ,  10 8 24 12,    4 4 1 6 6 2 8 8 4,
     1.5,1.5,1,0.5,2,2,1.5,1,3,4,3,1.5,3,3,3,1.5];
  Cost{i}:=[10,    8 8  8  8,    6 6 6 6 6 6 6 6 6,
      3   3   3   3,3,3,  3,3,3,3,3,  3,3,3,3, 3];
variable y; z{i};  binary D{i};
constraint
  R1: sum{i} Cost*z - sum{i} Rev*D = 0;
  R2{i}: z - 10*D <= 0;
  R3{i}: z - y <= 0;
  R4{i}: -z + y + 10*D <= 10;
  R5: sum{i} D >= 1;
  CC{LayingOn[i,j]}:  D[i] - D[j] <= 0;
maximize return_on_investment: y;
Writep(return_on_investment,D);
end
```

21 Tariff Rates (Power Generation) (will15)

—- Run LPL Code , HTML Document –

Problem: This model concerns the scheduling of generators to meet different demands for electricity throughout the day. There are three generators that can be started and shut down at the beginning of five time intervalls per day. These generators have different running and start up costs and differ also in minimal and maximal electricity output. Minimize the cost while fullfilling the electricity demand during a day.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.15. The model appeared also in DAY R.E., Williams H.P., MAGIC: The Design and Use of an Interactive Modelling Language for Mathematical Programming, IMA Journal of Mathematics in Management (1986) 1, pp.53-65.

Listing 20: The Complete Model implemented in LPL [6]

```
model Will15 "Tariff Rates (Power Generation)";
  set i:=[G1 G2 G3] "generators";
    t:=[pm12_am6 am6_am9 am9_pm3 pm_pm6 pm6_pm12] "periods";
  parameter
    minl{i}   := [0.85 1.25 1.50];
    maxl{i}   := [2.00 1.75 4.00];
    costph{i} := [1.00 2.60 3.00];
    excost{i} := [2.00 1.30 3.00];
    kctminl{i}:= [1.70 1.625 4.5];
    startcost{i}:= [2 1 .5];
    number{i} := [12 10 5];
    demand{t} := [15 30 25 40 27];
    nhours{t} := [ 6 3 6 3 6];
  variable
    out{i,t} "Output of a generator typ in each period";
    integer num{i,t} "Number of generators working";
    integer nst{i,t} "Number of generators starting";
  constraint
    dem{t} "Demand must be meet at each period"
      : sum{i} out >= demand;
    guar{t} "a 15% extra output margin must be imposed"
      : sum{i} maxl*num >= 1.15*demand;
    output{i,t} "Output level must lie between min/max"
      : minl*num <= out <= maxl*num;
    st{i,t} "The number of generators starting"
      : nst >= num - num[i,(#t+t-2)%#t+1];
    gennum{i,t} "the number of generators are limited"
      : num <= number >= nst;
  minimize cost: sum{i,t} (excost*nhours*out
    + startcost*nst + (costph-kctminl)*nhours*num);
  Writep(cost,out,num,nst);
end
```

22 Hydro power (will16)

—- Run LPL Code , HTML Document –

Problem: This model concerns the scheduling of generators to meet different demands for electricity throughout the day. There are three thermal generator types and two two hydro generators that can be started and shut down at the beginning of five time intervalls per day. These generators have different running and start up costs and they differ also in minimal and maximal electricity output. The thermal generators can pump water back to the reservoir. The water level of the reservoir must be 16 at the beginning of day and must be between 15 and 20 during the day. Minimize the cost while fullfilling the electricity demand during a day.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.16. This is an extended model of will15. The extension concerns the hydro generators and revervoir.

Listing 21: The Complete Model implemented in LPL [6]

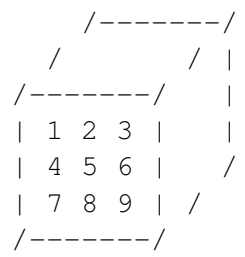
```
model Will16 "Hydro power";
  set i:=[G1 G2 G3] "generators";
      t:=[pm12_am6 am6_am9 am9_pm3 pm_pm6 pm6_pm12] "periods";
      h:= 1..2;
parameter
    minl{i}       := [0.85 1.25 1.50];
    maxl{i}       := [2.00 1.75 4.00];
    costph{i}     := [1.00 2.60 3.00];
    excost{i}     := [2.00 1.30 3.00];
    kctminl{i}    := [1.70 1.625 4.5];
    startcost{i}  := [2 1 .5];
    number{i}     := [12 10 5];
    demand{t}     := [15 30 25 40 27];
    nhours{t}     := [ 6 3 6 3 6];
    hydrolevel{h} := [0.9,1.4];
    startcosth{h} := [1.5,1.2];
    costphh{h}    := [0.09,0.15];
    numhyd{h}     := [1,1];
    redheight{h}  := [0.31,0.47];
variable
   out{i,t} "Output of a generator";
   heit{t}  "reservoir depth level";
   pump{t}  "pumped quantity to reservoir";
integer
   num{i,t}  [0..number];
   nst{i,t}  [0..number];
   numh{h,t} [0..numhyd];
   nsth{h,t} [0..numhyd];
constraint
   Dem{t}: sum{i} out +sum{h} hydrolevel*numh -pump >= demand;
   Guar{t}: sum{i} maxl*num >= 1.15*demand-sum{h} hydrolevel;
   output{i,t}:  minl*num <= out[i,t] <= maxl*num;
   St{i,t}: nst >= num - num[i,if(t=1,#t,t-1)];
   Sth{h,t}: nsth >= numh - numh[h,if(t=1,#t,t-1)];
   High{t}: heit[if(t<#t,t+1,1)] - heit - (nhours/3)*pump
      +sum{h} nhours*redheight*numh = 0;
   Bounds1: heit[1] = 16;
   Bounds2{t|t>1}: 15 <= heit <= 20;
minimize cost: sum{i,t}(excost*nhours*out
```

```
    + startcost*nst + (costph-kctminl)*nhours*num)
    + sum{h,t} (costphh*nhours*numh + startcosth*nsth);
Writep(cost,num,nst,out);
end
```

23 T3-dimensional Noughts and Crosses (will17)

—- Run LPL Code , HTML Document –

Problem: Given a 3x3x3 cube consisting of 27 cells, the problem is to fill each cell with either a black or a white ball so as to minimize the number of 'straight lines' in the cube containing balls of identical colour. A straight line is a row of 3 cells including all possible diagonals. (There are 49 such lines in all). The cells are numbered as indicated.



Model: The Model is as follows:

There exists 2^{27} feasible integer solution, clearly too many to be explored systematically. The symmetry could be exploit to restrict the number. We introduce a binary variable D, which is 1(0), if the cell contains a black(white) ball. Furthermore we introduce a 0-1 variable G for each possible straight line, which is 1, if the line contains only balls of the same color. The sum of these variables G must be minimized. For each of the 49 lines (for example cells 1, 2, and 3) 2 constraints are introduced as follows:

```
D[1]+D[2]+D[3]-G <= 2 "at most 2 of the ball are black"
D[1]+D[2]+D[3]+G >= 1 "at least 1 ball is black"
```

to model the condition

```
((D[1]+D[2]+D[3]=3) or (D[1]+D[2]+D[3]=0)) <--> G=1
```

which means: G is 1, if and only if all three cells of the line are black (sum is 3) or all three cells are white (sum is 0).

The problem is from [3]. For a problem description see Chapter 13.17. This problem appeared first in Williams H.P., Experiments in the formulation of integer programming problems, Math. Prog. Study, 2, pp.180-197.

Listing 22: The Complete Model implemented in LPL [6]

```
model Will17 "T3-dimensional Noughts and Crosses";
set  p "49 possible straight lines";
     c "27 cells";
  Lines{p,c} :=  -- 49 lines containing 3 cells each
  [(1,*)  1  2  3   (2,*)  4  5  6   (3,*)  7  8  9  --x
   (4,*) 10 11 12   (5,*) 13 14 15   (6,*) 16 17 18
   (7,*) 19 20 21   (8,*) 22 23 24   (9,*) 25 26 27
  (10,*)  1  4  7  (11,*)  2  5  8  (12,*)  3  6  9  --y
  (13,*) 10 13 16  (14,*) 11 14 17  (15,*) 12 15 18
  (16,*) 19 22 25  (17,*) 20 23 26  (18,*) 21 24 27
  (19,*)  1 10 19  (20,*)  2 11 20  (21,*)  3 12 21  --z
  (22,*)  4 13 22  (23,*)  5 14 23  (24,*)  6 15 24
  (25,*)  7 16 25  (26,*)  8 17 26  (27,*)  9 18 27
  (28,*)  1  5  9  (29,*)  3  5  7      --xy-diagonals
  (30,*) 10 14 18  (31,*) 12 14 16
```

```
  (32,*) 19 23 27  (33,*) 21 23 25
  (34,*)  1 13 25  (35,*)  7 13 19      --yz-diagolals
  (36,*)  2 14 26  (37,*)  8 14 20
  (38,*)  3 15 27  (39,*)  9 15 21
  (40,*)  1 11 21  (41,*)  3 11 19      --zx-diagonals
  (42,*)  4 14 24  (43,*)  6 14 22
  (44,*)  7 17 27  (45,*)  9 17 25
  (46,*)  1 14 27  (47,*)  3 14 25 --inner xyz diagonals
  (48,*)  7 14 21  (49,*)  9 14 19 ];
binary variable
  D{c};  --1 if cell has a black ball
  G{p};  --1 if all balls in a line have the same color
constraint
  LineA{p}: sum{c|Lines} D[c] - G <= 2;
  LineB{p}: sum{c|Lines} D[c] + G >= 1;
  Numb: sum{c} D = 14;   -- 14 balls are black
minimize OBJ: sum{p}  G;
Write(' %2d lines have the same color\n', sum {p}G);
Writep(D,G);
end
```

Further Comments: Note that the problem is the same as the 3d-Tic-Tac-Toe problem. a two- and three-dimensional version have been implemented in 2d-tic-tac-toe and 3d-tic-tac-toe.

24 Optimizing a Constraint (will18)

—- Run LPL Code , HTML Document –

Problem: In a integer programming problem with only 0-1 variables the following constraints occurs:

$$9x_1 + 13x_2 - 14x_3 + 17x_4 + 13x_5 - 19x_6 + 23x_7 + 21x_8 \leq 37$$

We want to find an equivalent formulation of this constraints, but with a smaller right-hand-side value. Furthermore, we want to find an equivalent constraint, where the sum of the absolute values of all coefficients is minimized.

Example: It can be easily checked, that the two following constraints are equivalent, if all variables are 0-1 variables:

$$93x_1 + 49x_2 + 37x_3 + 29x_4 \leq 111$$

is equivalent to

$$2x_1 + x_2 + x_3 + x_4 \leq 2$$

Formulate the problem as a linear mathematical model (LP).

Modeling Steps: The problem can be formulated by a LP using the following transformation:

1. Translate the constraints in a standard form with only positive coefficients in descending order by substituting:

   $$y_1 = x_7\ ,\ y_2 = x_8\ ,\ y_3 = 1 - x_6\ ,\ y_4 = x_4\ ,\ y_5 = 1 - x_3\ ,$$
   $$y_6 = x_5\ ,\ y_7 = x_2\ ,\ y_8 = x_1$$

   this gives the standard form:

   $$23y_1 + 21y_2 + 19y_3 + 17y_4 + 14y_5 + 13y_6 + 13y_7 + 9y_8 \leq 70$$

2. We want to find another equivalent constraints of the form:

   $$a_1y_1 + a_2y_2 + a_3y_3 + a_4y_4 + a_5y_5 + a_6y_6 + a_7y_7 + a_8y_8 \leq a_0$$

   The coefficients a_i become the variables of our problems.

3. We search for all subsets of the indices known as minimal “roof” and maximal “ceiling”. A minimal roof is a subset of indices for which the sum of the corresponding coefficients exceeds the right-hand side coefficient and no proper subset exceeds it. A maximal ceiling is a subset of indices for which the sum of the corresponding coefficients does not exceed the right-hand side and no subset properly containing it can also be a ceiling.

4. All ’roofs’ and all ’ceiling’ are a constraint.

5. Furthermore we add the constraint: $a_1 \geq a_2 \geq \ldots \geq a_8$.

The problem is from [3]. For a problem description see Chapter 13.18.

Listing 23: The Complete Model implemented in LPL [6]

```
model Will18 "Optimizing a Constraint";
  set
    j := [1 2 3 4 5 6 7 8]; --number of coefficients
    c := [1 2 3 4 5 6];     --number of 'roofs'
    r := [1 2 3 4 5 6];     --number of 'ceiling'
  parameter
    Ceiling{c,j}:=[1,2,3,0,0,0,0,0,
                   1,2,4,8,0,0,0,0,
                   1,2,6,7,0,0,0,0,
                   1,3,5,6,0,0,0,0,
                   2,3,4,6,0,0,0,0,
                   2,5,6,7,8,0,0,0];
    Roofing{r,j}:=[1,2,3,8,0,0,0,0,
                  1,2,5,7,0,0,0,0,
                  1,3,4,7,0,0,0,0,
                  1,5,6,7,8,0,0,0,
                  2,3,4,5,0,0,0,0,
                  3,4,6,7,8,0,0,0];
  variable  a{j};  b;
  constraint
    CEiling{c}: sum{j} a[Ceiling] <= b;
    ROofing{r}: sum{j} a[Roofing] >= b+1;
    Order{j|j<#j}: a[j] >= a[j+1];
  minimize Obja: b-a[3]-a[5];
  Writep(a,b);
  minimize Objb: sum{j} a;
  Writep(a,b);
end
```

25 Distribution I (will19)

—- Run LPL Code , HTML Document –

Problem: A company has two factories with a limited capacity and four depots with a maximum throughtput capacity. It sells its products to six customers with minimum requirements. The customers have some preferences to be supplied from a depot or a factory. What would the distribution pattern from factories to depots and customer be, if the overall cost are to be minimized? Is it possible to meet the customers preferences?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.19.

Listing 24: The Complete Model implemented in LPL [6]

```
model Will19 "Distribution I";
  set  f := [Liverpool,Brighton];
       d := [Newcastle, Birmingham, London, Exeter];
       c := [c1, c2, c3, c4, c5, c6];
  parameter
    FactoryCap{f}:= /Liverpool 150000, Brighton 200000/;
    DepotCap{d}:= /Newcastle 70000, Birmingham 50000, London 100000,
       Exeter 40000/;
    CustomerReq{c} := /c1 50000, c2 10000, c3 40000,
                       c4 35000, c5 60000, c6 20000/;
    FactoryDepotCost{f,d} :=
       /Liverpool Newcastle   0.5,
        Liverpool Birmingham  0.5,
        Liverpool London      1.0,
        Liverpool Exeter      0.2,
        Brighton  Birmingham  0.3,
        Brighton  London      0.5,
        Brighton  Exeter      0.2/;
    FactoryCustCost{f,c} :=
       /Liverpool c1 1.0,
        Liverpool c3 1.5,
        Liverpool c4 2.0,
        Liverpool c6 1.0,
        Brighton  c1 2.0/;
    DepotCustCost{d,c} :=
       /Newcastle  c2 1.5,
        Newcastle  c3 0.5,
        Newcastle  c4 1.5,
        Newcastle  c6 1.0,
        Birmingham c1 1.0,
        Birmingham c2 0.5,
        Birmingham c3 0.5,
        Birmingham c4 1.0,
        Birmingham c5 0.5,
        London     c2 1.5,
        London     c3 2.0,
        London     c5 0.5,
        London     c6 1.5,
        Exeter     c3 0.2,
        Exeter     c4 1.5,
        Exeter     c5 0.5,
        Exeter     c6 1.5/;
```

```
    --client preferences to be delivered from
    PrefFtoC{f,c} := [0,1,1,1,1,1,
                      1,1,1,1,1,1];
    --preferred depot of a customer
    PrefDtoC{d,c} := [1,0,1,1,1,1,
                      1,1,1,1,0,1,
                      1,1,1,1,1,0,
                      1,1,1,1,1,0];
  variable
    FactoryDepotQty,FD{f,d|FactoryDepotCost};
    FactoryCustQty,FC{f,c|FactoryCustCost};
    DepotCustQty,DC{d,c|DepotCustCost};
  constraint
    FactoryCapa{f}: sum{d} FD + sum{c} FC <= FactoryCap;
    DepotCapa{d}: sum{f} FD <= DepotCap;
    DepotBalance{d}: sum{c} DC = sum{f} FD;
    CustRequirement{c}: sum{f} FC + sum{d} DC = CustomerReq;
  minimize cost: sum{f,d} FactoryDepotCost*FD
    + sum{f,c}FactoryCustCost*FC + sum{d,c}DepotCustCost*DC;
  Writep(cost,FD,FC,DC);
  minimize Pref: sum{f,c}PrefFtoC*FC + sum{d,c}PrefDtoC*DC;
  Writep(cost,FD,FC,DC);
end
```

Solution: Transportation optimum is: 198500.

26 Distribution II (will19a)

—- Run LPL Code , HTML Document –

Problem: A company has two factories with a limited capacity and four depots with a maximum throughtput capacity. It sells its products to six customers with minimum requirements. The customers have some preferences to be supplied from a depot or a factory. What would the distribution pattern from factories to depots and customer be, if the overall cost are to be minimized? Is it possible to meet the customers preferences?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.19. This model is exactly the same as model will19:

Listing 25: The Complete Model implemented in LPL [6]

```
model Will19a "Distribution II";
  set
    F "factories" := [Liverpool Brighton];
    D "depots"    := [Newcastle Birmingham London Exeter];
    C "customers" := [c1 c2 c3 c4 c5 c6];
  parameter                -- distribution costs
    FactCap{F} "factory capacity" := [150,200];
    DepCap{D} "max. depot throughput" := [70,50,100,40];
    CustReq{C}"customer requirements" := [50,10,40,35,60,20];
    CostFtoD{F,D} := [0.5,0.5,1,0.2,
                      0,0.3,0.5,0.2];
    CostFtoC{F,C} := [1,0,1.5,2,0,1,
                      2,0,0  ,0,0,0];
    CostDtoC{D,C} := [0,1.5,0.5,1.5,0,1,
                      1,0.5,0.5,1,0.5,0,
                      0,1.5,2,0,0.5,1.5,
                      0,0,0.2,1.5,0.5,1.5];
    PrefFtoC{F,C} := [0,1,1,1,1,1,
                      1,1,1,1,1,1];
    PrefDtoC{D,C} := [1,0,1,1,1,1,
                      1,1,1,1,0,1,
                      1,1,1,1,1,0,
                      1,1,1,1,1,0];
  variable
    FtoD{F,D|CostFtoD} "quantity from factor to depot";
    FtoC{F,C|CostFtoC} "quantity from factory to costomer";
    DtoC{D,C|CostDtoC} "quantity from depot to customer";
  constraint
    FCAP{F}: sum{D} FtoD +sum{C} FtoC <= FactCap;
    DCAP{D}: sum{F} FtoD <= DepCap;
    DCONT{D}: sum{C} DtoC  = sum{F} FtoD;
    CREQ{C}: sum{F} FtoC +sum{D} DtoC = CustReq;
  minimize Cost: sum{F,D} CostFtoD*FtoD
      +sum{F,C} CostFtoC*FtoC + sum{D,C} CostDtoC*DtoC;
  Write('--model1: minimizing the costs\n');
  Writep(Cost,FtoD,FtoC,DtoC);
  minimize Pref: sum{F,C} PrefFtoC*FtoC +sum{D,C} PrefDtoC*DtoC;
  Write('--model2: meet customers preferences\n');
  Writep(Pref,FtoD,FtoC,DtoC);
  Write('%7.2f\n', sum {F,D}CostFtoD*FtoD+ sum {F,C}CostFtoC*FtoC
    + sum {D,C}CostDtoC*DtoC);
end
```

27 Depot Location (Distribution 2) (will20)

—- Run LPL Code , HTML Document –

Problem: A company has two factories with a limited capacity and four depots (Newcastle, Birmingham, London, and Exeter) with a maximum throughtput capacity. It sells its products to six customers with minimum requirements. The company has to decide to retain or close Newcastle and Exeter, to expand Birmingham, or to build new ones at Bristol and Northampton. The depot in London must be inchanged. In any case, there must be four depots. There are savings as well as costs of doing so. What would the distribution pattern from factories to depots and customer be, if the overall cost are to be minimized, and which depots must be opened, closed or expand?

Model: The Model is as follows:The problem is from [3] Chapter 13.20.

Listing 26: The Complete Model implemented in LPL [6]

```
model Will20 "Depot Location (Distribution 2)";
set  F "factories" := [Liverpool Brighton];
     D "depots" := [Newcastle Birmingham London Exeter
                    Bristol Northampton];
     C "customers" := [C1 C2 C3 C4 C5 C6];
parameter                              -- distribution costs
  FactCap{F} "factory capacity" := [150,200];
  DepCap{D} "max. throughput" :=  [70,20,100,40,30,25];
  CustReq{C}"customer require." := [50,10,40,35,60,20];
  CostFtoD{F,D} := [0.5,0.5,1,0.2,0.6,0.4
                    0,0.3,0.5,0.2,0.4,0.3];
  CostFtoC{F,C} := [1,0,1.5,2,0,1,  2,0,0  ,0,0,0];
  CostDtoC{D,C} := [
    0,1.5,0.5,1.5,0,1, 1,0.5,0.5,1,0.5,0,
    0,1.5,2,0,0.5,1.5, 0,0,0.2,1.5,0.5,1.5
    1.2,0.6,0.4,0,0.3,0.8, 0,0.4,0,0.5,0.6,0.9];
  FixCost{D} "cost of an action d" := [10,3,0,5,12,4];
variable
  FtoD{F,D|CostFtoD} "quantity from factor to depot";
  FtoC{F,C|CostFtoC} "quantity from factory to customer";
  DtoC{D,C|CostDtoC} "quantity from depot to customer";
  binary d{D|D<>3}; --1, if retained (Newcastle, Exeter)
           --expand (Birmingham), build (Bristol, Nort.)
constraint
  FCAP{F}: sum{D} FtoD +sum{C} FtoC <= FactCap;
  DCAP{D}: sum{F} FtoD <= DepCap*d+if(D=2,50,D=3,DepCap);
  DCONT{D}: sum{C} DtoC = sum{F} FtoD;
  CREQ{C}: sum{F} FtoC +sum{D} DtoC = CustReq;
  NUMB: sum{D|D<>4} d <= 2;
minimize Cost:sum{F,D}CostFtoD*FtoD+sum{F,C}CostFtoC*FtoC
               +sum{D,C} CostDtoC*DtoC+sum{D} FixCost*d;
Writep(Cost,FtoD,FtoC,DtoC,d);
end
```

28 Agricultural Pricing I (will21)

—- Run LPL Code , HTML Document –

Problem: The government of a country wants to fix the prices of its dairy products milk, butter, and cheese. All these products comes from the country's raw milk production. Parts of this production is exported or directly comsumed. The rest is available in the form of its components: fat, dry matter, and water to produce the mentioned products. Comsumtion, prices of previous years are known as well as the prices elasticities. The objective is to determine what prices and resultant demand will maximize total revenue. Politically, however, the prices should not rise to much: the cost of last year's consumtion should not increase. As an important result, the economic cost of this political limitation should be quantified (through the shadow prices on the indexing constraint). The data are as follows: Availability of fat and dry matter are 600′000 and 750′000 tons.

	Milk	Butter	Cheese 1	Cheese 2
Fat composition (%)	4	80	35	25
Dry M. (%)	9	2	30	40
previous year comsumption (1000t)	4820	320	210	70
previous year price (£/t)	297	720	1050	815
Price elasticity	0.4	2.7	1.1	0.4

Some cross elasticities are also given:

	Cheese 1	Cheese 2
Cheese 1		0.1
Cheese 2	0.4	

Recall that the elasticity of a product is defines as:

$$E = \frac{\text{Percentage decrease in demand}}{\text{Percentage increase in price}}$$

And the cross elasticity of two products A and B is given as:

$$E_{AB} = \frac{\text{Percentage increase in demand for } A}{\text{Percentage increase in price for } B}$$

The problem is to fix prices in a way to maximize total revenue of the agricultural sector. For a complete problem and model description see [3], Chapter 13.21.

Modeling Steps: The products are defined as a set $i, j \in$ Products. The data are the percentage content of fat (fat_i), the percentage content of dry matter (dry_i), previous year comsumption (con_i), previous year price (pri_i), the elastisities (E_i), and the cross elastisities of cheese ($E0_{i,j}$).

1. We introduce two variables which defines the qunatity consumed x_i and the price to be fixed p_i.

2. The total fat and dry matter produced limit the output:

$$\sum_i fat_i \cdot x_i \leq 600000 \quad , \quad \sum_i dry_i \cdot x_i \leq 750000$$

3. The price index should not be higher than the previous year:

$$\sum_i con_i \cdot p_i \leq \sum_i con_i \cdot pri_i$$

4. The comsumtion x_i is related to the prices p_i through the elasticities:

$$\frac{dx_i}{x_i} = -E_i \frac{dp_i}{p_i} + \sum_{j|E0_{i,j}} \frac{dp_j}{p_j}, \qquad \text{forall } i$$

One can approximate the differntial equation (where $\bar{z}$ is the known previous year quantity – here consumption or price):

$$\frac{dz}{z} \quad \text{by} \quad \frac{z - \bar{z}}{\bar{z}}$$

Hence we have the linear equations:

$$\frac{x_i - \bar{x}_i}{\bar{x}_i} = -E_i \frac{p_i - \bar{p}_i}{\bar{p}_i} + \sum_{j|E0_{i,j}} \frac{p_j - \bar{p}_i}{\bar{p}_j}, \qquad \text{forall } i$$

5. We would like to maximize the revenue for the agricultural sector:

$$\max \sum_i p_i x_i$$

Listing 27: The Complete Model implemented in LPL [6]

```
model Will21 "Agricultural Pricing I";
  set i,j:=[milk butter cheese1 cheese2];
  parameter
   fat{i} :=[4 80 35 25] "percentage composition";
   dry{i} :=[9  2 30 40] "dry matter composition";
   con{i} :=[4820 320 210 70] "past comsumpt. in 1000t";
   pri{i} :=[297 720 1050 815] "past price of 100t";
   E{i} :=[0.4 2.7 1.1 0.4] "price elastisitty";
   F{i,j}:=[(cheese1,cheese2) 0.1, (cheese2,cheese1) 0.4];
  variable x{i} [1..10000] "quantity comsumed";
  variable p{i} [0..10000] "price";
  constraint
    Fat: sum{i} fat*x <= 600000;
    Dry: sum{i} dry*x <= 750000;
    Pri: sum{i} con*p <= sum{i} con*pri;
    Ela{i}: (x-con)/con = - E*(p-pri)/pri
            + sum{j|F} F*(p[j]-pri[j])/pri[j];
  maximize revenu: sum{i} p*x;
  Writep(revenu,p,x);
  Write('%12.6f␣\n', {i}x);
end
```

Further Comments: The model define so far is a non-linear model. Using the elasticity relationship, we may substitute all x_i variables, the remaining model only contains variables p_i. The resulting model can be formulated as a convex quadrtaic model as follows – where Q is semi-definite positive matrix:

$$
\begin{array}{ll}
\max & \displaystyle\sum_{i,j} p_i Q_{i,j} p_j + A_i p_i \\
\text{subject to} & \text{three linear constraints} \\
& \text{linear constraints that make sure that comsumption is positive}
\end{array}
$$

The explicit model is given in WILL21a.

29 Agricultural Pricing II (will21a)

—- Run LPL Code , HTML Document –

Problem: This model is a QP formulation of the model WILL21

Model: The Model is as follows:

Listing 28: The Complete Model implemented in LPL [6]

```
model Will21a "Agricultural Pricing II";
variable MILK; BUTT; CHA; CHB; //prices
constraint
  QUALA: 260*MILK +960*BUTT +70.25*CHA -0.6*CHB >= 782;
  QUALB: 584*MILK +24*BUTT +55.2*CHA +5.8*CHB >= 35;
  INDEX: 4.82*MILK +0.32*BUTT +0.21*CHA +0.07*CHB <= 1.939;
  NONNEG1: 220*CHA -26*CHB <= 420;
  NONNEG2: -27*CHA +34*CHB <= 70;
  NONNEG3: MILK<=1.039 and BUTT<=0.987;
maximize REVENUE: -6492*MILK*MILK -1200*BUTT*BUTT -220*CHA*CHA
  -34*CHB*CHB +53*CHA*CHB +6748*MILK +1184*BUTT +420*CHA +70*CHB;
Write('Revenue=%d\n\nMilk = %d\nButter = %d\nCheese 1 = %d\nChesse 2 =
   %d\n',
   REVENUE,1000*MILK,1000*BUTT,1000*CHA,1000*CHB);
end
```

30 Efficiency Analysis (EA) (will22)

—- Run LPL Code , HTML Document –

Problem: This model shows the data envelopment analysis (DEA) approach for measuring efficiency. A car manufacturer wants to evaluate the efficiency of different garages, who have received a franchise to sell its cars. It does this by weighing input against output, comparing "cost" with "returns". Input and output can be measured by different criteria. In our case input is measure by using 6 coefficients: (1) number of staff, (2) Show room space, (3) catchment population in category I, (4) catchment population in category II, (5) enquiries Alpha model (6) enquiries in Beta model. Output is measured by three criteria: (1) Alpha model sales, (2) Beta model sales, (3) profit. We assume proportionality, that is, a "doubling" of its inputs result in a "doubling" of its output. Calculate which garages are the most efficient once.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.22. The approach used here is *data envelopment analysis* (DEA). For a more in-depth theoretical analysis of DEA, the intersted reader is referred to [5]. The idea is to maximize a weighted output while nomalizing the input to 1. The unknown weights must be found for each decision making unit (in our case a garage). This can be done by formulating a LP (see also model dea. The model must be solve for each garage.

Listing 29: The Complete Model implemented in LPL [6]

```
model Will22 "Efficiency Analysis (EA)";
 set j,j1 := 1..28 "garages";
     i    := 1..6  "inputs";
     k    := 1..3  "outputs";
 parameter
   Input{j,i} :=
     [7,8,10,12,8.5,4,     6,6,20,30,9,4.5,
      2,3,40,40,2,1.5,     14,9,20,25,10,6,
      10,9,10,10,11,5,     24,15,15,13,25,19,
      6,7,50,40,8.5,3,     8,7.5,5,8,9,4,
      5,5,10,10,5,2.5,     8,10,30,35,9.5,4.5,
      7,8,7,8,3,2,         5,6.5,9,12,8,4.5,
      6,7.5,10,10,7.5,4,   11,8,8,10,10,6,
      4,5,10,10,7.5,3.5,   3,3.5,3,2,2,1.5,
      5,5.5,8,10,7,3.5,    21,12,6,8,15,8,
      6,5.5,2,2,8,5,       3,3.6,3,3,2.5,1.5,
      30,29,120,80,35,20,  25,16,110,80,27,12,
      19,10,90,12,25,13,   7,6,5,7,8.5,4.5,
      12,8,7,10,12,7,      4,6,1,1,7.5,3.5,
      2,2.5,1,1,2.5,1,     2,3.5,2,2,1.9,1.2];
   Output{j,k} :=
     [2,0.6,1.5,      2.3,0.7,1.6,
      0.8,0.25,0.5,   2.6,0.86,1.9,
      2.4,1,2,        8,2.6,4.5,
      2.5,0.9,1.6,    2.1,0.85,2,
      2,0.65,0.9,     2.05,0.75,1.7,
      1.9,0.7,0.5,    1.8,0.63,1.4,
      1.5,0.45,1.45,  2.2,0.65,2.2,
      1.8,0.62,1.6,   0.9,0.35,0.5,
      1.2,0.45,1.3,   6,0.25,2.9,
      1.5,0.55,1.55,  0.8,0.2,0.45,
      7,2.5,8,        6.5,3.5,5.4,
      5.5,3.1,4.5,    1.2,0.48,2,
```

```
      4.5,2,2.3,      1.1,0.48,1.7,
      0.4,0.1,0.55,   0.3,0.09,0.4];
  J;
 variable x{j};  w;
 constraint
  ilevel{i}: sum{j} Input[j,i]*x[j]  <= Input[J,i];
  olevel{k}: sum{j} Output[j,k]*x[j] >= Output[J,k]*w;
 for{j1} do
  J:=j1;
  maximize weight: w;
  Write('J=%2d  1/w = %5.3f %s \n',
        J,1/w,if(w=1,'efficient',''));
 end
end
```

31 Milk Collection (will23)

—- Run LPL Code , HTML Document –

Problem: A small milk processing company is committed to collecting milk from 20 farms and taking it back to the depot for processing. The company has one tanker lorry with a capacity for carrying 80000 litres of milk. Eleven of the farms are small and need a collection only every other day. The other nine farms need a collection every day. Find the optimal route for the tanker lorry on each day, bearing in mind that it has to (i) visit all the "every day" farms, (ii) visit some of the "every other day" farms and work within its capacity. On alternate days, it must again visit the "every day" farms and also visit the "every other day" farms not visited on the previous day.

(Note: I have added the minimal necessary subtours that are needed to find a correct feasible solution.)

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.23.

Listing 30: The Complete Model implemented in LPL [6]

```
model Will23 "Milk Collection";
set f,g,h := [1..21]; --farms including depot as farm 1
    a{f}  := f<=10;   --1 to 10 visited every day
    o{g}  := g>10;    --11 to 21 visited every other day
    d     := [1..2];  --alternating days
parameter X{f}:=[0 -3 1 4 -5 -5 -4 6 3 -1 0 6 2 -2 6 1 -3 -6 2 -6 5];
  Y{f}:=[0 3 11 7 9 -2 -7 0 -6 -3 -6 4 5 8 10 8 1 5 9 -5 -4];
  Cap{f}:=[. 5 4 3 6 7 3 4 6 5 4 7 3 4 5 6 8 5 7 6 6];
  Dist{f,g|f<g}:=Round(Sqrt((X[f]-X[g])^2+(Y[f]-Y[g])^2),-1);
  Tcap:=80 "Tanker capacity";
binary variable
   x{f,g,d}  "1 if go beween farms on a day";
   y{f,d}    "1 if farm visited on a day";
constraint
   limit{d}: sum{o} Cap*y <= Tcap - sum{a} Cap "tanker capacity";
   dayvis{o[g]}: sum{d} y[g,d] = 1  "visit every other day";
   mat1{a,d}: sum{h|h>a} x[a,h,d]+sum{h|h<a} x[h,a,d] = 2 "matching for
       every day";
   mat2{o,d}: sum{h|h>o} x[o,h,d]+sum{h|h<o} x[h,o,d] = 2*y[o,d] "match
       every 2nd day";
   bound: y[11,1]=1 and y[11,2]=0;
   --or: bound: y[11,1] and ~y[11,2];
   va{o,h,d|f>o}: x[o,h,d] <= y[o,d]  "tighten cuts";
   vb{o,h,d|f<o}: x[h,o,d] <= y[o,d]  "tighten cuts";
set s:=[1..5]; // 5 subtours elimination
    S{s,f}:=[(1,*) 6 7 20, (2,*) 2 5 18, (3,*) 3 4 13 16
       19   (4,*) 8 9 21, (5,*) 1 2 6 7 10 17 ];
constraint
  Subtours{d,s}: sum{f,g|S[s,f] and S[s,g]} x[f,g,d] <= sum{f|S}1 - 1;
minimize cost: sum{f,g,d|f<g} Dist[f,g]*x[f,g,d];
Draw.Scale(40,-40);
{f,g|x[f,g,1]} Draw.CLine(X[f],Y[f],X[g],Y[g],1,3,3);
{f,g|x[f,g,2]} Draw.CLine(X[f],Y[f],X[g],Y[g],-1,5);
{f} Draw.Circle(f&'',X,Y,.3,1,0);
Draw.Line(-6,11,-5,11,3,3); Draw.Text('first␣␣day␣tour',-5,11);
Draw.Line(-6,10.5,-5,10.5,5,2); Draw.Text('second␣␣day␣tour',-5,10.5);
```

```
end
```

32 Yield Management (will24)

—- Run LPL Code , HTML Document –

Problem: An airline is selling tickets for flights to a particular destination. The flight will depart in three weeks' time. It can use up to six planes each costing $50 000 to hire. Each plane has 37 First Class seats, 38 Business Class seats, 47 Economy Class seats. Up to 10% of seats in any one category can be transferred to an adjacent category. It wishes to decide a price for each of these seats.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.24.

Listing 31: The Complete Model implemented in LPL [6]

```
model Will24 "Yield Management";
set   c       := 1..3; --Classes: First,Business and Economy
      s,j,k   := 1..3; --Scenes: Scenarios
      p       := 1..3; --Weeks from take-off
      o       := 1..3; --Price levels possible
parameter
   Cap{c}:=[37,38,47];    -- Capacities per plane
   Prob{s}:=[0.1,0.7,0.2];-- Probabilities of scenarios
   Price{p,c,o} :=  -- Prices by Period, Class and option
    [1200,1000, 950,   900, 800, 600,  500, 300, 200,
     1400,1300, 1150, 1100, 900, 750,  700, 400, 350,
     1500, 900, 850,   820, 800, 500,  480, 470, 450];
   Dem{p,s,c,o}:= --Demand by period,Scenario,Class and Option
    [10 15 20 , 20 25 35 , 45 55 60
     20 25 35 , 40 42 45 , 50 52 63
     45 50 60 , 45 46 47 , 55 56 64
     20 25 35 , 42 45 46 , 50 52 60
     10 40 50 , 50 60 80 , 60 65 90
     50 55 80 , 20 30 50 , 10 40 60
     30 35 40 , 40 50 55 , 50 60 80
     30 40 60 , 10 40 45 , 50 60 70
     50 70 80 , 40 45 60 , 60 65 70];
   Cost:=50000;           -- Cost per plane
binary variable
   pone{c,o};     -- set/or not price option period 1
   ptwo{s,c,o};   -- set/or not price option period 2
   pthr{s,j,c,o}; -- set/or not price option period 3
variable
   sone{s,c,o};     -- Tickets sold in period 1
   stwo{s,j,c,o};   -- Tickets sold in period 2
   sthr{s,j,k,c,o}; -- Tickets sold in period 3
   rone{s,c,o};     -- Revenue in period 1
   rtwo{s,j,c,o};   -- Revenue in period 2
   rthr{s,j,k,c,o}; -- Revenue in period 3
   xess{s,j,k,c} [0.. 0.1*Cap]; -- Excess capa. allowed
   xund{s,j,k,c} [0.. 0.1*Cap]; -- Capa. reduction allowed
integer variable number [0..6]; -- Nr. of planes to send
constraint
  -- Constraints forcing revenue equal to number sold time price
  r1a{s,c,o}: rone[s,c,o] <= Price[1,c,o]*sone[s,c,o];
  r1b{s,c,o}: Price[1,c,o]*sone[s,c,o]-rone[s,c,o] <=
       Price[1,c,o]*Dem[1,s,c,o]*(1-pone[c,o]);
```

```
  r2a{s,j,c,o}: rtwo[s,j,c,o] <= Price[2,c,o]*stwo[s,j,c,o];
  r2b{s,j,c,o}: Price[2,c,o]*stwo[s,j,c,o]-rtwo[s,j,c,o]
    <= Price[2,c,o]*Dem[2,j,c,o]*(1-ptwo[s,c,o]);
  r3a{s,j,k,c,o}: rthr[s,j,k,c,o] <= Price[3,c,o]*sthr[s,j,k,c,o];
  r3b{s,j,k,c,o}: Price[3,c,o]*sthr[s,j,k,c,o]-rthr[s,j,k,c,o]
    <= Price[3,c,o]*Dem[3,k,c,o]*(1-pthr[s,j,c,o]);
  -- Capacity limits in each class for all scenarios
   Kap{s,j,k,c}: sum{o} sone[s,c,o] +sum{o} stwo[s,j,c,o]
      +sum{o} sthr[s,j,k,c,o] <= (Cap[c]*number + xess[s,j,k,c] - xund[
         s,j,k,c]);
--            (Cap[c]*number + xess[c] - xund[c]);
   Adjust{s,j,k}: sum{c} xess[s,j,k,c] -sum{c} xund[s,j,k,c] = 0;
--                  sum{c} xess[c] -sum{c} xund[c] = 0;
  -- Exactly one price option in each class
   Lone{c}: sum{o} pone[c,o] = 1;
   Ltwo{s,c}: sum{o} ptwo[s,c,o] = 1;
   Lthr{s,j,c}: sum{o}  pthr[s,j,c,o] = 1;
   -- Numbers sold cannot exceed demand
   l1s{s,c,o}: sone[s,c,o] <= Dem[1,s,c,o]*pone[c,o];
   l2s{s,j,c,o}: stwo[s,j,c,o] <= Dem[2,j,c,o]*ptwo[s,c,o];
   l3s{s,j,k,c,o}: sthr[s,j,k,c,o] <= Dem[3,k,c,o]*pthr[s,j,c,o];
maximize Eyield: sum{s,c,o} Prob[s]*rone[s,c,o]
  +sum{s,j,c,o} Prob[s]*Prob[j]*rtwo[s,j,c,o]
  +sum{s,j,k,c,o} Prob[s]*Prob[j]*Prob[k]*rthr[s,j,k,c,o]
  -Cost*number;
Writep(Eyield);
end
```

33 Car Rental I (will25)

—- Run LPL Code , HTML Document –

Problem: A small (“cut price”) car rental company, renting one type of car, has depots in Glasgow, Manchester, Birmingham and Plymouth. There is an estimated demand for each day of the week except Sunday when the company is closed. These estimates are given in Table 12.20. It is not necessary to meet all demand. Cars can be rented for one, two or three days and returned to either the depot from which rented or another depot at the start of the next morning. How many cars should the company own and where should they be located at the start of each day while maximizing the profit?

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.25.

Listing 32: The Complete Model implemented in LPL [6]

```
model Will25 "Car Rental I";
set  i,j := 1..4;  --depots: Glasgow,Manchester,Birmingham,Plymouth
     t   := 1..6;  --days: Monday,...,Saturday
     k   := 1..3;  --hire: 1,2 or 3 day hire possible
parameter
   demand{i,t} := [
     100 150 135 83 120 230, 250 143 80 225 210 98,
      95 195 242 111 70 124, 160  99 55  96 115 80];
   --Proportion sent from each depot to other depots
   perret{i,j} := [
     0.6  0.2 0.1  0.1 , 0.15 0.55 0.25 0.05,
     0.15 0.2 0.54 0.11, 0.08 0.12 0.27 0.53];
   --Proportion cars hired for k days
   hireper{k} := [0.55 0.20 0.25];
   --Cost of transfer from depot to depot
   costtran{i,j} := [0 20 30 50, 20 0 15 35,
                     30 15 0 25, 50 35 25 0];
   repcap{i} := [0 12 20 0]; --Daily repair capacity
   rcosta{k} := [50 70 120]; --Rental price return to same depot
   rcostb{k} := [70 100 150]; --return to another depot
   rcostcomp{k} := [20 25 30]; --Marginal cost
variable
   n;         -- Total number of cars
   nu{i,t};   -- Number of undamaged cars
   nd{i,t};   -- Number of damaged cars
   tr{i,t};   -- Number of cars rented out
   eu{i,t};   -- Excess of undamaged cars (not rented out)
   ed{i,t};   -- Excess of damaged cars
              -- Not to be transferred or repaired
   tu{i,j,t}; -- Number of undamaged cars
               -- to be transferred to depot j
   td{i,j,t}; -- Number of damaged cars
               -- to be tranferred to depot j
   rp{i,t}; -- Number repaired at depot i during day t
constraint
   inudam{i,t|t>1}: sum{j,k|k<t} (0.9*perret[j,i]*hireper[k]*(tr[j,t-k
      ]))
     +sum{j,k|k>=t} (0.9*perret[j,i]*hireper[k]*(tr[j,t-k+6]))
     +sum{j} (tu[j,i,t-1])+rp[i,t-1]+eu[i,t-1] = nu[i,t];
   -- Net flow of undamaged cars into each depot on each day>Monday
```

```
   inudama{i}: sum{j,k}(0.9*perret[j,i]*hireper[k]*(tr[j,7-k]))
     +sum{j} (tu[j,i,6])+rp[i,6]+eu[i,6] = nu[i,1];
   -- Net flow of undamaged cars into each depot on Monday
   indam{i,t|t>1}: sum{j,k|k<t} (0.1*perret[j,i]*hireper[k]*(tr[j,t-k])
       )
     +sum{j,k|k>=t} (0.1*perret[j,i]*hireper[k]*(tr[j,t-k+6]))
     +sum{j} (td[j,i,t-1])+ed[i,t-1] = nd[i,t];
   -- Net flow of damaged cars into each depot on each day>Monday
   indama{i}: sum{j,k}(0.1*perret[j,i]*hireper[k]*(tr[j,7-k]))
     +sum{j} (td[j,i,6]) +ed[i,6] = nd[i,1];
   -- Net flow of damaged cars into each depot on Monday
   outudam{i,t}: sum{j} (tu[i,j,t]) +tr[i,t]+eu[i,t]=nu[i,t];
   -- Net flow of undamaged cars out of depot i on day t
   outdam{i,t|t>1}: sum{j} td[i,j,t] +rp[i,t-1]+ed[i,t] = nd[i,t];
   -- Net flow of damaged cars out of depot i on day t>Monday
   outdama{i}: sum{j} (td[i,j,1]) +rp[i,6]+ed[i,1] = nd[i,1];
   -- Net flow of damaged cars out of depot i on Monday
   repc{i,t}: rp[i,t] <= repcap[i]; -- Repair capacity
   demd{i,t}: tr[i,t] <= demand[i,t]; -- Renting out within demand
   tot: sum{i} (0.25*tr[i,1]+0.45*tr[i,2]+nu[i,3]+nd[i,3]) = n;
   -- Total number of cars owned equals those rented out on Monday for
       3 days
   -- plus those rented out Tuesday more than 1 day
   -- plus those in depots at beginning of Wednesday
maximize Profit: sum{i,t,k|t<>6}
        -- Hired out,not Saturday, and returned to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-rcostcomp[k]+10)*(tr[i,t]))
        -- Hired out,not Saturday, and returned to another depot
          +sum{i,j,t,k|i<>j and t<> 6}
        (perret[i,j]*hireper[k]*(rcostb[k]-rcostcomp[k]+10)*(tr[i,t]))
          +sum{i,t,k|t=6 and k=1}
        -- Hired out on Saturday for 1 day with return to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-20-rcostcomp[k]+10)*(tr[i,t
           ]))
          +sum{i,j,t,k|i<>j and t=6 and k=1}
        -- Hired out on Saturday for 1 day with return to another depot
        (perret[i,j]*hireper[k]*(rcostb[k]-20-rcostcomp[k]+10)*(tr[i,t
           ]))
          +sum{i,t,k|t=6 and k<>1}   -- Hired out on Saturday for more
              than 1 day with return to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-rcostcomp[k]+10)*(tr[i,t]))
          +sum{i,j,k,t|i<>j and t=6}
        -- Hired out on Saturday for more than 1 day with return to
           another depot
        (perret[i,j]*hireper[k]*(rcostb[k]-rcostcomp[k]+10)*(tr[i,t]))
          -sum{i,j,t}
        -- Cost of transfer to another depot
          (costtran[i,j]*(tu[i,j,t]+td[i,j,t]))
          -15*n;  --Marginal cost to company of cars owned
Writep(Profit,n);
end
```

34 Car Rental II (will26)

—- Run LPL Code , HTML Document –

Problem: This model extends the model Will25. The company wants to consider where it might be most worthwhile to expand repair capacity given the cost structure of the repair centers.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.26.

Listing 33: The Complete Model implemented in LPL [6]

```
model Will26 "Car Rental II";
set  i,j := 1..4; --depots: Glasgow,Manchester,Birmingham,Plymouth
     t   := 1..6; --days: Monday,...,Saturday
     k   := 1..3; --hire: 1,2 or 3 day hire possible
parameter
   demand{i,t} := [
     100 150 135 83 120 230, 250 143 80 225 210 98,
     95 195 242 111 70 124,  160  99 55  96 115 80];
   perret{i,j} := [
     0.6  0.2 0.1  0.1 ,  0.15 0.55 0.25 0.05,
     0.15 0.2 0.54 0.11,  0.08 0.12 0.27 0.53];
   hireper{k} := [0.55 0.20 0.25];
   costtran{i,j} := [       --Cost of transfer
     0 20 30 50,  20 0 15 35,
     30 15 0 25,  50 35 25 0];
   repcap{i} := [0 12 20 0]; --Daily repair capacity
   rcosta{k} := [50 70 120]; --Rental price return to same depot
   rcostb{k} := [70 100 150]; --return to another depot
   rcostcomp{k} := [20 25 30]; --Marginal cost
variable
   n;  -- Total number of cars
   nu{i,t};  --Number of undamaged cars at depot i at beginning of day
      t
   nd{i,t};  --Number of damaged cars at depot i at beginning of day t
   tr{i,t};  --Number of cars rented out at depot i on day t
   eu{i,t};  --Excess of undamaged cars at depot i at beginning of day
      t, Not rented out that day
   ed{i,t};  --Excess of damaged cars at depot i at beginning of day t,
       Not to be transferred or repaired that day
   tu{i,j,t}; --Number of undamaged cars at depot i at beginning of day
       t to be transferred to depot j
   td{i,j,t}; --Number of damaged cars at depot i at beginning of day t
       to be tranferred to depot j
   rp{i,t}; --Number repaired at depot i during day t
binary variable
   db1; -- First Birmingham expansion
   db2; -- Second Birmingham expansion
   dm1; -- First Manchester expansion
   dm2; -- Second Manchester expansion
   dp;  -- Plymouth expansion
constraint
   inudam{i,t|t>1}: sum{j,k|k<t} (0.9*perret[j,i]*hireper[k]*(tr[j,t-k
      ]))
     +sum{j,k|k>=t} (0.9*perret[j,i]*hireper[k]*(tr[j,t-k+6]))
     +sum{j} (tu[j,i,t-1])+rp[i,t-1]+eu[i,t-1] = nu[i,t];
   -- Net flow of undamaged cars into each depot on each day>Monday
```

```
   inudama{i}: sum{j,k}(0.9*perret[j,i]*hireper[k]*(tr[j,7-k]))
     +sum{j} (tu[j,i,6])+rp[i,6]+eu[i,6] = nu[i,1];
   -- Net flow of undamaged cars into each depot on Monday
   indam{i,t|t>1}: sum{j,k|k<t} (0.1*perret[j,i]*hireper[k]*(tr[j,t-k])
      )
     +sum{j,k|k>=t} (0.1*perret[j,i]*hireper[k]*(tr[j,t-k+6]))
     +sum{j} (td[j,i,t-1])+ed[i,t-1] = nd[i,t];
   -- Net flow of damaged cars into each depot on each day>Monday
   indama{i}: sum{j,k}(0.1*perret[j,i]*hireper[k]*(tr[j,7-k]))
     +sum{j} (td[j,i,6]) +ed[i,6] = nd[i,1];
   -- Net flow of damaged cars into each depot on Monday
   outudam{i,t}: sum{j} (tu[i,j,t]) +tr[i,t]+eu[i,t] = nu[i,t];
   -- Net flow of undamaged cars out of depot i on day t
   outdam{i,t|t>1}: sum{j} (td[i,j,t]) +rp[i,t-1]+ed[i,t] = nd[i,t];
   -- Net flow of damaged cars out of depot i on day t>Monday
   outdama{i}: sum{j} (td[i,j,1]) +rp[i,6]+ed[i,1] = nd[i,1];
   -- Net flow of damaged cars out of depot i on Monday
   repc1{i,t|i=1}: rp[i,t] <= repcap[i];              -- Repair capacity
       at Glasgow
   repc2{i,t|i=2}: rp[i,t] <= repcap[i]+5*db1+5*db2; -- Repair capacity
       at Birmingham
   repc3{i,t|i=3}: rp[i,t] <= repcap[i]+5*dm1+5*dm2; -- Repair capacity
       at Manchester
   repc4{i,t|i=4}: rp[i,t] <= repcap[i]+5*dp;        -- Repair capacity
       at Plymouth
   LI: db1>=db2 and dm1>=dm2 and db1+db2+dm1+dm2+dp <= 3; --Limits on
      expansion
   demd{i,t}: tr[i,t] <= demand[i,t]; -- Renting out within demand
   tot: sum{i} (0.25*tr[i,1]+0.45*tr[i,2]+nu[i,3]+nd[i,3]) = n;
   -- Total number of cars owned equals those rented out on Monday for
      3 days
   -- plus those rented out Tuesday more than 1 day
   -- plus those in depots at beginning of Wednesday
maximize Profit: sum{i,t,k|t<>6} -- Hired out,not Saturday, and
    returned to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-rcostcomp[k]+10)*(tr[i,t]))
        +sum{i,j,t,k|i<>j and t<>6} -- Hired out,not Saturday, and
           returned to another depot
        (perret[i,j]*hireper[k]*(rcostb[k]-rcostcomp[k]+10)*(tr[i,t]))
        +sum{i,t,k|t=6 and k=1} -- Hired out on Saturday for 1 day with
            return to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-20-rcostcomp[k]+10)*(tr[i,t
           ]))
        +sum{i,j,t,k|i<>j and t=6 and k=1} -- Hired out on Saturday for
            1 day with return to another depot
        (perret[i,j]*hireper[k]*(rcostb[k]-20-rcostcomp[k]+10)*(tr[i,t
           ]))
        +sum{i,t,k|t=6 and k<>1} -- Hired out on Saturday for more than
            1 day with return to same depot
        (perret[i,i]*hireper[k]*(rcosta[k]-rcostcomp[k]+10)*(tr[i,t]))
        +sum{i,j,k,t|i<>j and t=6} -- Hired out on Saturday for more
           than 1 day with return to another depot
        (perret[i,j]*hireper[k]*(rcostb[k]-rcostcomp[k]+10)*(tr[i,t]))
        -sum{i,j,t} -- Cost of transfer to another depot
        (costtran[i,j]*(tu[i,j,t]+td[i,j,t])) -15*n  --Marginal cost to
            company of cars owned
        -18000*db1-8000*db2-20000*dm1-5000*dm2-19000*dp; --Expansion
           costs
```

```
Writep(Profit,n);
end
```

35 Lost Baggage Distribution (will27)

—- Run LPL Code , HTML Document –

Problem: A small company with six vans has a contract with a number of airlines to pick up lost or delayed baggage, belonging to customers in the London area, from Heathrow airport at 6 p.m. each evening. The contract stipulates that each customer must have their baggage delivered by 8 p.m. The company requires a model, which they can solve quickly each evening, to advise them what is the minimum number of vans they need to use and to which customers each van should deliver and in what order. There is no practical capacity limitation on each van. All baggage that needs to be delivered in a two-hour period can be accommodated in a van.

Formulate optimisation models that will minimise the number of vans that need to be used, and within this minimum, minimise the time taken for the longest time delivery.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.27.

Listing 34: The Complete Model implemented in LPL [6]

```
model Will27 "Lost Baggage Distribution";
set  i,j := [1..15]; --locations
     k  := [1..6];   --vans
parameter
  X{i}:=[0 1 2 5 4 11 8 7 9 3 2 2 4 6 8.5];
  Y{i}:=[5 0 4 3 10 5 7 4 1 4 8 6 5 2 4];
  time{i,j}:=[0,20,25,35,65,90,85,80,86,25,35,20,44,35,82,
              0,0,15,35,60,55,57,85,90,25,35,30,37,20,40,
              0,0,0,30,50,70,55,50,65,10,25,15,24,20,90,
              0,0,0,0,45,60,53,55,47,12,22,20,12,10,21,
              0,0,0,0,0,46,15,45,75,25,11,19,15,25,25,
              0,0,0,0,0,0,15,15,25,45,65,53,43,63,70,
              0,0,0,0,0,0,0,17,25,41,25,33,27,45,30,
              0,0,0,0,0,0,0,0,25,40,34,32,20,30,10,
              0,0,0,0,0,0,0,0,0,65,70,72,61,45,13,
              0,0,0,0,0,0,0,0,0,0,20,8,7,15,25,
              0,0,0,0,0,0,0,0,0,0,0,5,12,45,65,
              0,0,0,0,0,0,0,0,0,0,0,0,14,34,56,
              0,0,0,0,0,0,0,0,0,0,0,0,0,30,40,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0,27,
              0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
binary variable
   x{i,j,k}; -- =1 iff van k goes i to j
   y{i,k};   -- =1 iff van k goes to i
   d{k};     -- =1 iff van k used
constraint
  ind{i,k|i<>1}: y[i,k] <= d[k]; --Is van k used?
  all{k}: y[1,k] >= d[k]; --All vans visit 1 (Heathrow)
  limit{k}: sum{i,j|i<>j and j<>1} ((time[i,j]+time[j,i])*x[i,j,k]) +
     sum{i,j|i<>j and j=1} (time[i,j]*x[i,j,k]) <=120; -- Time limit
     on each van
  num{i|i<>1}: sum{k} y[i,k] = 1; -- Exactly one van to each location (
     except all used to 1)
  into{i,k}: sum{j|j<>i} x[j,i,k]=y[i,k]; -- If van k visits i then
     exactly 1 link in
  out{i,k}: sum{j|j<>i} x[i,j,k]=y[i,k]; -- If van k visits i then
     exactly 1 link out
```

```
  D2{i,j,k}: x[i,j,k] + x[j,i,k] <= 1; --add all subtours of length 2
set s,t := [1..30];  --additional subtours (not needed)
  S{s,i}; //:=[(1,*) 4 14, (2,*) 5 7,
          // (3,*) 6 8 9 15, (4,*) 1 10
    //(5,*) 1 2 3 12 13 11 (6,*) 8 15 (7,*) 1 3 4];
constraint
  Subtours{k,s}: sum{i,j|S[s,i] and S[s,j]} x[i,j,k] <= sum{i|S}1 - 1;
minimize Number: sum{k} d[k]; -- Minimise number of vans
--constraint Num: sum{k} d[k]=2;
--minimize Time: max{k} ...
Writep(Number);
Draw.Scale(40,40);
{i,j,k|x} Draw.CLine(X[i],Y[i],X[j],Y[j],1,k+1,3);
{i} Draw.Circle(i&'',X,Y,.3,1,0);
end
```

36 Protein Folding (will28)

—- Run LPL Code , HTML Document –

Problem: We take a protein as consisting of a chain of amino acids. For the purpose of this problem, the amino acids come in two forms: hydrophilic (waterloving) and hydrophobic (water hating). Such a chain naturally folds so as to bring as many hydrophobic acids, as possible, close together.

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.28.

Listing 35: The Complete Model implemented in LPL [6]

```
model Will28 "Protein Folding";
set  A:=[1..50];    -- Number of acids in Protein
     H{A}:=         -- Hydrophobic acids
     [2,4,5,6,11,12,17,20,21,25,27,28,30,31,33,37,44,46];
parameter size:=8; -- Maximum linear dimension of protein
binary variable
   m{H,H}; -- = 1 iff acids i and j are matched
   f{A};   -- = 1 iff fold immediately after acid i
   d{A,A}; -- = 1 iff fold j immediately after acid i
variable
   l{A};  -- location (preceding acid number) of fold j
   p{A};  -- position (right or left of acid 1) of fold j
   s;     -- leftmost fold
   t;     -- rightmost fold
constraint
   mt{H[i],H[j], k in A| i+2<j and j<=#A and (j-i)%2=1 and i<=k and k<j
       and k<>(i+j-1)/2}: f[k]+m[i,j]<=1;
     -- Non contiguous matching requires only one fold between and even
        number of acids between
   mta{H[i],H[j]| i+2<j and j<=#A and (j-i)%2=1}: f[(i+j-1)/2]=m[i,j];
      -- Matching requires mid point fold  i
   bd{H[i],H[j]| i+1=j}: m[i,j]=0; -- Contiguous matches not counted
      again
   bda{H[i],H[j]| (j-i)%2=0}: m[i,j]=0; -- Cannot match acids with an
      odd number between
   c{i in A, j in A}: sum{k in A| k<i} f[k] +(i-j)*d[i,j]<=i;
   ca{i in A, j in A}: sum{k in A| k<i} f[k] - j*d[i,j] >=0; -- forces
      correct value oj d[i,j]
   loc{i in A, j in A}: l[j]+(#A-i)*d[i,j]<=#A;
   loca{i in A, j in A}: l[j]-i*d[i,j]>=0; -- forces correct location
      of folds
   pos0{j in A| j=1}: p[j]=l[j];
   pos1{j in A| j>1 and j%2=1}: p[j]-p[j-1]+l[j-1]-l[j]=-1;
   pos2{j in A| j>1 and j%2=0}: p[j]-p[j-1]-l[j-1]+l[j]=1; -- positions
       of folds
   mn{j in A}: s<=p[j]; -- leftmost fold
   mx{j in A}: t>=p[j]; -- rightmost fold
   depth: sum{i in A} f[i]<=size;
   breadth: t-s<=size;
   exclude: m[33,44]=0; --to get the same folding as Williams
maximize Number: sum{H[i],H[j]| i<j} m[i,j];
  -- Maximise number of matched hyprophobic acids, not counting those
     already matched by virtue of being contiguous in chain
```

```
Writep(Number);
Write{H[i],H[j]|m[i,j]}('Acid␣%2s␣matches␣acid␣%2s\n', i,j);
parameter YY; R; XX;  X{A}; Y{A};
{i in A} ( X[i]:=XX, Y[i]:=YY ,
           if(f,(YY:=YY+1, R:=1-R)),
           if(~f[i],  (if(R,(XX:=XX-1),(XX:=XX+1)))) );
Draw.Scale(50,50);
{i in A|i<#A} Draw.Line(X,Y,X[i+1],Y[i+1]);
{i in A,j in A|m[i,j]} Draw.Line(X[i],Y[i],X[j],Y[j],3,3);
{i in A} Draw.Circle(i&'',X,Y,.25,if(H,2,1),0);
end
```

Solution:

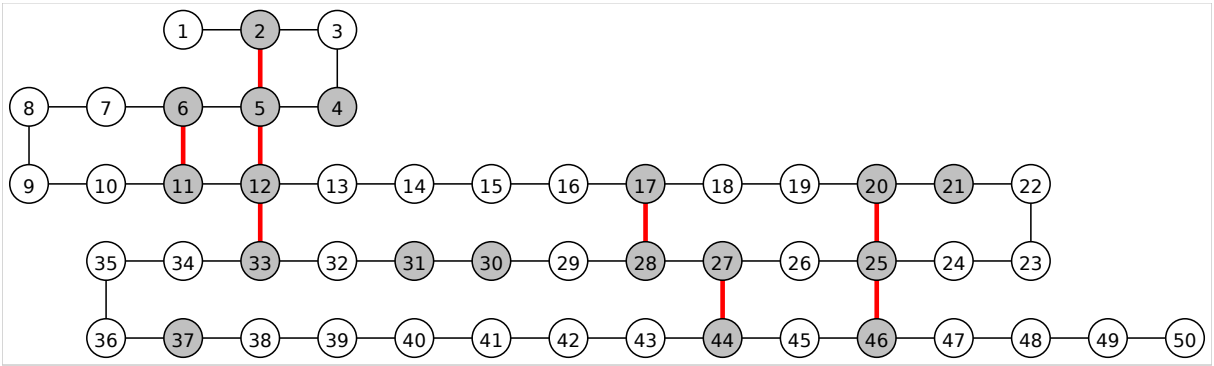


Figure 5: The Folding Structure

37 Protein Comparison (will29)

—- Run LPL Code , HTML Document –

Problem:

Model: The Model is as follows:The problem is from [3]. For a problem description see Chapter 13.29.

Listing 36: The Complete Model implemented in LPL [6]

```
model Will29 "Protein Comparison";
set  a,i := [1..9];
     b,j := [1..11];
parameter
   A{i,a} :=  //edges of the first protein
    [0,1,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,1,
     0,0,0,1,1,0,0,0,0,  0,0,0,0,0,0,0,0,0,
     0,0,0,0,0,1,0,0,0,  0,0,0,0,0,0,1,0,0,
     0,0,0,0,0,0,0,0,1,  0,0,0,0,0,0,0,0,1,
     0,0,0,0,0,0,0,0,0];
   B{j,b} :=   //edges of the second protein
     [0,0,0,1,0,0,0,0,0,0,0,  0,0,1,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,1,1,0,0,0,0,
      0,0,0,0,0,1,0,0,0,0,0,  0,0,0,0,0,0,0,1,0,0,0,
      0,0,0,0,0,0,0,1,0,1,0,  0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,1,0,  0,0,0,0,0,0,0,0,0,0,1,
      0,0,0,0,0,0,0,0,0,0,0,0];
binary variable  x{a,b}; --  =1 iff a corresponds to b
   w{i,j,a,b}; /*FREE;*/ -- =1 iff x[i,k]=x[j,l]=1
constraint
  Assa{a}: sum{b} x[a,b] <=1; -- At most 1 node from G2 assigned to
     each in G1
  Assb{b}: sum{a} x[a,b] <=1; -- At most 1 node from G1 assigned to
     each in G2
  Linka{i,j,a,b|A[i,a]=1 and B[j,b]=1 and i<a and j<b}:  w[i,j,a,b]<=x[
     i,j];
  Linkb{i,j,a,b|A[i,a]=1 and B[j,b]=1 and i<a and j<b}:  w[i,j,a,b]<=x[
     a,b];
  --Linkc{i,j,a,b|A[i,a]=1 and B[j,b]=1 and i<a and j<b}: w[i,j,a,b]>=x
     [i,j]+x[a,b]-1;
  Ncrs{i,j,a,b|i<a and b<j}: x[i,j]+x[a,b]<=1; -- No cross overs
maximize size: sum{i,j,a,b|A[i,a]=1 and B[j,b]=1 and i<a and j<b} w[i,j
    ,a,b];
  -- Maximise number of corresponding edges
Writep(size);
end
```

38 Product Mix I (willi005)

—- Run LPL Code , HTML Document –

Problem: A factory produces five type of products by using two processes and manpower. How much should be produced, when the capacities of the processes and the manpower is limited and the profit is to be maximized?

Model: The Model is as follows:The problem is from [3], page 6 and page 107.

Listing 37: The Complete Model implemented in LPL [6]

```
model Willi005 "Product Mix I";
  variable x1; x2; x3; x4; x5;
  constraint
    Grinding: 12*x1+20*x2 +25*x4+15*x5 <= 288;
    Drilling: 10*x1+8*x2+16*x3 <= 192;
    Manpower: 20*x1+20*x2+20*x3+20*x4+20*x5 <= 384;
  maximize profit: 550*x1+600*x2+350*x3+400*x4+200*x5;
  Writep(profit,x1,x2,x3,x4,x5);
end
```

39 Product Mix II (willi005a)

—- Run LPL Code , HTML Document –

Problem: A factory produces five type of products by using two processes and manpower. How much should be produced, when the capacities of the processes and the manpower is limited and the profit is to be maximized?

Model: The Model is as follows:The problem is from [3], page 6 and page 107.

Listing 38: The Complete Model implemented in LPL [6]

```
model Willi005a "Product Mix II";
  set  product,p := [1..5];
       process,r := [Grinding Drilling];
  parameter
    Profit{p}          := [550 600 350 400 200];
    Hours{r,p}     := [12 20  0 25 15, 10  8 16  0  0];
    AssemblyTime   := 20;
    DaysPerWeek    :=  6;
    HoursPerShift  :=  8;
    HoursPerDay    :=  2*HoursPerShift;
    Workers        :=  8;
    Machines{r}    :=  [3 2];
    WorkHoursAvail:=  Workers*DaysPerWeek*HoursPerShift;
    MachHours{r}   := Machines*DaysPerWeek*HoursPerDay;
  variable
    Produce,P{p};
  constraint
    ProcessLimit{r}: sum{p} Hours*P <= MachHours;
    WorkLimit: sum{p} AssemblyTime*P <= WorkHoursAvail;
  maximize TotalProfit: sum{p} Profit*P;
  Writep(TotalProfit,P);
end
```

40 Vegetable Oil Blending (willi008)

—- Run LPL Code , HTML Document –

Problem: A food is manufactured by refining raw oils and blending them together. How much should be taken from each raw oil in order to maximize profit?

Model: The Model is as follows:The problem is from [3], page 8.

Listing 39: The Complete Model implemented in LPL [6]

```
model Willi008 "Vegetable Oil Blending";
  set  oils,o          := [Veg1, Veg2, Oil1, Oil2, Oil3];
       VegOils,v{oils} := [Veg1, Veg2];
       NonVegOils,n{o} := [Oil1, Oil2, Oil3];
  parameter
    MaxVegRefine    := 200;
    MaxNonVegRefine := 250;
    Price           := 150;
    Cost{o}         := [110, 120, 130, 110, 115];
    Hardness{o}     := [8.8, 6.1, 2.0, 4.2, 5.0];
    MinHardness     := 3;
    MaxHardness     := 6;
  variable
    RawOils{o};
    Produce;
  constraint
    ProductWeight:     Produce = sum{o} RawOils;
    MaxVegRefining:    sum{v} RawOils <= MaxVegRefine;
    MaxNonVegRefining: sum{n} RawOils <= MaxNonVegRefine;
    HardnessC: MinHardness*Produce <=
      sum{o} Hardness*RawOils <= MaxHardness*Produce;
  maximize Profit: Price*Produce - sum{o} Cost*RawOils;
  Writep(Profit,Produce,RawOils);
end
```

41 Multi-Plant Planning (willi055)

—- Run LPL Code , HTML Document –

Problem: A company has two factories and manufactures two products. Two processes are used to make them with a limiting capacity. Calculate the maximal profit.

Model: The Model is as follows:The problem is from [3], page 55.

Listing 40: The Complete Model implemented in LPL [6]

```
model Willi055 "Multi-Plant Planning";
  set  factory,f := [A, B];
       product,p := [Standard, Deluxe];
       process,pr := [Grinding, Polishing];
  parameter
    Profit{p}       := [10, 15];
    Capacity{f,pr} := [80, 60, 60, 75];
    Time{pr,f,p}    := [4, 2, 5, 3,  2, 5, 5, 6];
    RawUse          :=   4;
    RawAvail        := 120;
  variable Produce,P{factory,product};
  constraint
    RawLimit: sum{f, p} RawUse*P <= RawAvail;
    ProcessLimit{f,pr}: sum{p} Time*P <= Capacity;
  maximize TotalProfit: sum{f,p} Profit*P;
  Writep(TotalProfit,P);
end
```

42 Transportation (willi082)

—- Run LPL Code , HTML Document –

Problem: A number of supplier and costumer are given. The transportation problem is to meet each costomer's requirement, while not exceeding the capacity of any supplier and at minimum cost.

Model: The Model is as follows:The problem is from [3], page 82.

Listing 41: The Complete Model implemented in LPL [6]

```
model Willi082 "Transportation";
  set  supplier,s :=  [S1 S2 S3];
       customer,c :=  [T1 T2 T3 T4];
  parameter
   Capacity{s}    := [135 56 93];
   Requirement{c} := [62 83 39 91];
   ShippingCost{s,c} :=
     [132  .   97 103, 85  91   .   ., 106  89 100  98];
  variable Ship{s,c|ShippingCost};
  constraint
    MaxCapacity{s}: sum{c} Ship <= Capacity;
    MinRequirement{c}: sum{s} Ship >= Requirement;
  minimize TotalCost: sum{s,c} ShippingCost*Ship;
  Writep(TotalCost,Ship);
end
```

43 Production Planning (willi085)

—- Run LPL Code , HTML Document –

Problem: A company produces a commodity in two shifts to meet known demands. The problem is to satisfy the demand at minimal cost.

Model: The Model is as follows:The problem is from [3], page 85.

Listing 42: The Complete Model implemented in LPL [6]

```
model Willi085 "Production Planning";
  set  s    := [Regular, Overtime];
       m,m1 := [January, February, March, April];
  parameter
    Capacity{s,m}:=[100 150 140 160, 50 75 70 80];
    ProdCost{s} := [1.00, 1.50];
    Demand{m}   := [80, 200, 300, 200];
    StorCost := 0.30;
    CombinedCost{m,s,m1} :=
        if(m1>=m,ProdCost) + if(m1>m,m1-m)*StorCost;
  variable
    Produce,P{m,s}  [0..Capacity];
    Inventory,I{m};
  constraint Balance{m}: I = I[m-1] + sum{s} P - Demand;
  minimize Cost: sum{s,m} ProdCost*P + sum{m} StorCost*I;
  Writep(Cost,P,I);
end
```

44 Minimum Cost Flow (willi090)

—- Run LPL Code , HTML Document –

Problem: Find the minimum flow through a network

Model: The Model is as follows:The problem is from [3], page 90.

Listing 43: The Complete Model implemented in LPL [6]

```
model Willi090 "Minimum Cost Flow";
  set node,s,d   := [n0 n1 n2 n3 n4 n5 n6 n7];
  parameter
    FlowCost{s,d}:=/n0 n2 5, n1 n3 4, n2 n3 2, n2 n4 6,
      n2 n5 5, n3 n4 1, n3 n7 2, n4 n2 4, n4 n5 6,
      n4 n6 3, n7 n6 4/;
    Availability{node} := /n0 10, n1 15/;
    Requirement{node}  := /n5 9, n6 10, n7 6/;
  variable Flow,F{s,d|FlowCost};
  constraint FlowBalance{node}:
        sum{s} F[s,node] + Availability
        = sum{d} F[node,d] + Requirement;
  minimize TotalCost: sum{s,d} FlowCost*F;
  Writep(TotalCost,F);
end
```

45 Shortest Path Problem (willi093)

—- Run LPL Code , HTML Document –

Problem: Finding the shortest path through a network

Model: The Model is as follows:The problem is from [3], page 93.

Listing 44: The Complete Model implemented in LPL [6]

```
model Willi093 "Shortest Path Problem";
  set node,s,d := [n0 n1 n2 n3 n4 n5 n6 n7 n8];
  parameter
    PathDist{s,d} :=
      / n0 n1  1, n0 n3  1, n1 n2  1, n1 n3  2, n2 n3  1,
        n2 n4  1, n2 n5  3, n3 n4  2, n3 n6  4, n4 n5  3,
        n4 n6  2, n4 n7  4, n5 n7  1, n6 n7  2, n6 n8  1, n7 n8  1/;
    StartPath{node} := /n0 1/;
    EndPath{node}   := /n8 1/;
  variable Path{s,d|PathDist};
  constraint
    FlowBalance{node}: sum{s} Path[s,node] + StartPath
          = sum{d} Path[node,d] + EndPath;
  minimize TotalDistance: sum{s,d} PathDist*Path;
  Writep(TotalDistance,Path);
end
```

46 Maximum Flow (willi094)

—- Run LPL Code , HTML Document –

Problem: Maximizing the flow through a network

Model: The Model is as follows:The problem is from [3], page 94.

Listing 45: The Complete Model implemented in LPL [6]

```
model Willi094 "Maximum Flow";
  set
    node ,s,d := [n0 n1 n2 n3 n4 n5 n6 n7];
    SourceNodes{node} := [n0, n1];
    SinkNodes{node}   := [n5, n6, n7];
  parameter
    FlowCapacity{s,d} :=
     / n0 n2  12, n1 n3  20, n2 n3   6, n2 n4  3,
       n2 n5   6, n3 n4   7, n3 n7   9, n4 n2  2,
       n4 n5   5, n4 n6   8, n7 n6   4/;
  variable
    Flow{s,d|FlowCapacity} [0..FlowCapacity];
    Sources{SourceNodes};
    Sinks{SinkNodes};
  constraint
    FlowBalance{node}: sum{s} Flow[s,node] + Sources
                     = sum{d} Flow[node,d] + Sinks;
  maximize TotalFlow: sum{SourceNodes} Sources;
  Writep(TotalFlow,Flow);
end
```

47 Critical Path (willi095)

—- Run LPL Code , HTML Document –

Problem: Finding the critical path in a network

Model: The Model is as follows:The problem is from [3], page 95.

Listing 46: The Complete Model implemented in LPL [6]

```
model Willi095 "Critical Path";
  set  t,s := [n0 n1 n2 n3 n4 n5 n6]  "Tasks";
     E{t,s}:= [n0 n1, n0 n2, n0 n3, n1 n3, n2 n5, n3 n4,
       n4 n2, n4 n5, n5 n6]   "A precedence list";
  parameter
    Duration{E} := /n0 n1 4, n0 n2 12, n0 n3 7, n1 n3 2,
          n2 n5 5, n3 n4 10, n4 n2 0, n4 n5 3, n5 n6 4/;
  string activities{E}:=['DigFoundations', 'ObtainTiles',
    'ObtainBricks', 'LayFoundations', 'Walls', 'Dummy24',
    'Roofing', 'Wiring', 'Painting'];
  variable S{t} "Starting time";
  constraint SeqRelation{E[t,s]}: S[s]-S[t] >= Duration;
  minimize FinishTime: S[#t];
  Writep(FinishTime,S);
end
```

48 A Small QP (willi142)

—- Run LPL Code , HTML Document –

Problem: A small QP model from [3] p.142. Geometrically, the maodel is shown in Figure 6.

Listing 47: The Complete Model implemented in LPL [6]

```
model Willi142 "A Small QP";
  variable x; y;
  constraint
    A: x + y <= 4;
    B: 2*x + y <= 5;
    C: -x + 4*y >= 2;
  minimize obj: x^2 - 4*x - 2*y;
  Writep(obj,x,y);
end
```

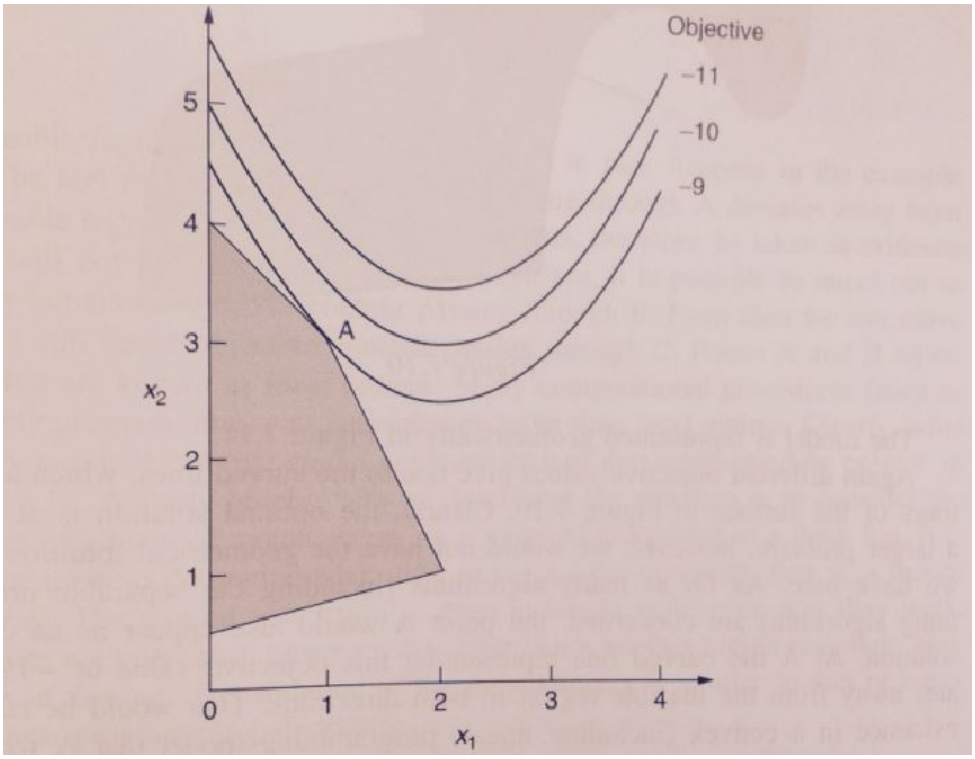


Figure 6: The Feasible Space and the Objective

References

[1] Orlin J.B. Ahuja R.K., Magnanti T.L. *Network Flows, Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Pulleyblank W.R. Schrijver A. Cook W.J., Cunningham W.H. *Combinatorial Optimization*. John Wiley and Sons, Inc., 1998.

[3] Williams H.P. *Model Building in Mathematical Programming*. John Wiley, Fifth Edition, 2013.

[4] MatMod. Homepage for Learning Mathematical Modeling : https://matmod.ch.

[5] Thall R.M. Seiford L.M. Recent developments in DEA: the mathematical programming approach to frontier analysis. *Journal of Econometrics, Vol 46, p. 7–38*, 1990.

[6] Hürlimann T. Reference Manual for the LPL Modeling Language, most recent version. https://matmod.ch/lpl/doc/manual.pdf.