New Directions in the Development of AMPL

Robert Fourer, David M. Gay***

AMPL Optimization LLC
www.ampl.com — +1 773-336-AMPL

* Industrial Eng & Management Sciences, Northwestern Univ
** Computer Science, University of New Mexico

INFORMS Annual Meeting
Austin, TX — November 7-10, 2010
Session MD18, *Modeling Languages II*

New Directions in Development of the AMPL Modeling Language and System

We summarize current and planned projects to extend and enhance AMPL, with the aim of facilitating both formulation and implementation of optimization models. Extensions to AMPL's language are designed to allow for more natural description of discrete models and stochastic data. New solver interfaces will make nontraditional solvers more accessible for practical modeling. A new callable interface to the AMPL system will facilitate business deployment.

AMPL

Algebraic modeling language: symbolic data

```
set SHIFTS;                  # shifts

param Nsched;                # number of schedules;
set SCHEDS = 1..Nsched;      # set of schedules

set SHIFT_LIST {SCHEDS} within SHIFTS;

param rate {SCHEDS} >= 0;        # pay rates
param required {SHIFTS} >= 0;    # staffing requirements

param least_assign >= 0;         # min workers on any schedule used
```

AMPL

Algebraic modeling language: symbolic model

```
var Work {SCHEDS} >= 0 integer;
var Use  {SCHEDS} >= 0 binary;

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use1 {j in SCHEDS}:
   least_assign * Use[j] <= Work[j];

subject to Least_Use2 {j in SCHEDS}:
   Work[j] <= (max {i in SHIFT_LIST[j]} required[i]) * Use[j];
```

AMPL

Explicit data independent of symbolic model

```
set SHIFTS := Mon1 Tue1 Wed1 Thu1 Fri1 Sat1
              Mon2 Tue2 Wed2 Thu2 Fri2 Sat2
              Mon3 Tue3 Wed3 Thu3 Fri3 ;

param Nsched := 126 ;

set SHIFT_LIST[1] := Mon1 Tue1 Wed1 Thu1 Fri1 ;
set SHIFT_LIST[2] := Mon1 Tue1 Wed1 Thu1 Fri2 ;
set SHIFT_LIST[3] := Mon1 Tue1 Wed1 Thu1 Fri3 ;
set SHIFT_LIST[4] := Mon1 Tue1 Wed1 Thu1 Sat1 ;
set SHIFT_LIST[5] := Mon1 Tue1 Wed1 Thu1 Sat2 ;  .......

param required :=  Mon1 100  Mon2 78  Mon3 52
                   Tue1 100  Tue2 78  Tue3 52
                   Wed1 100  Wed2 78  Wed3 52
                   Thu1 100  Thu2 78  Thu3 52
                   Fri1 100  Fri2 78  Fri3 52
                   Sat1 100  Sat2 78 ;
```

AMPL

Solver independent of model & data

```
ampl: model sched1.mod;
ampl: data sched.dat;

ampl: let least_assign := 7;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.1.0: optimal integer solution; objective 266
473 MIP simplex iterations
72 branch-and-bound nodes

ampl: option omit_zero_rows 1, display_1col 0;
ampl: display Work;

Work [*] :=
  2 12    16 14    29  7    53  7     91 17    112  9    122 29
  3  7    18  7    37 21    78 21    100 19    116 20    124  7
  6 10    20  7    41  8    82 21    109  7    118 16
;
```

AMPL

Language independent of solver

```
ampl: option solver gurobi;
ampl: solve;

Gurobi 3.0.0: optimal solution; objective 266
396 simplex iterations
3 branch-and-cut nodes

ampl: display Work;

Work [*] :=
  1 29    37 35    84 18    91 17   101 11   112 18   118 17
 21 36    71  7    89 18    95  7   109 10   116  7   124 36
;
```

Topics

The company

- People
- Business developments

The language

- Varied prospective enhancements
- More natural formulations

The solvers

- Conic programming
- Nontraditional alternatives

The system

- APIs & IDEs
- AMPL as a service (in the cloud)

The Company

Background

- AMPL at Bell Labs (1986)
- AMPL commercialization (1993)
- AMPL Optimization LLC (2002)

Developments

- People
- Business

People

Bob Fourer

- Founder & . . .

Dave Gay

- Founder & . . .

Bill Wells

- Director of business development

Business Developments

AMPL intellectual property

- ❖ Full rights acquired from Alcatel-Lucent USA
  - ∗ corporate parent of Bell Laboratories
- ❖ More flexible licensing terms available

CPLEX with AMPL

- ❖ Sales transferred from IBM to AMPL Optimization
- ❖ Full lineup of licensing arrangements available

AMPL distributors

- ❖ New for Japan: *October Sky Co., Ltd.* →
- ❖ Others continue active
  - ∗ Gurobi, Ziena
  - ∗ MOSEK, TOMLAB
  - ∗ OptiRisk

AMPL

AMPL

最強の最適化モデリング言語

究極のスケーラビリティー

AMPLは、スケーラビリティが要求される大規模な最適化問題を効率よくモデル化するための代数モデリング言語です。簡単な例題をモデル化できるだけのモデリング言語が氾濫する中、AMPLは、現実社会の実務問題を解決するためのプロフェショナル用のモデリング ツールと言えます。特に、集合を効率よく操作するための演算の豊富さと柔軟さは、他のモデリング言語の追従を許しません。また、ネットワークおよび区分的線形関数を入力するための特別な表記法も組み込んでおり、ユーザー数の多さ、そして、長年の開発の積み重ねによって、極めて使い易いインターフェイスをユーザに提供します。AMPLは、まさに最強のモデリング言語です。

AMPL MEANS BUISINESS

The Language

Background

- ❖ Power & convenience
  - ∗ Linear and nonlinear modeling
  - ∗ Extensive indexing and set expressions
- ❖ Prototyping & deployment
  - ∗ Integrated scripting language
- ❖ Business & research
  - ∗ Major installations worldwide
  - ∗ Hundreds of citations in scientific & engineering literature

Plans . . .

The Language

Plans

- Further set operations
  - ∗ arg min/arg max
  - ∗ sort set by parameter values
  - ∗ arbitrary selection from an unordered set
- Random parameters/variables
  - ∗ send as input to stochastic solvers
- Enhanced scripting
  - ∗ faster loops
  - ∗ functions defined by scripts
- ***More natural formulations . . .***

More Natural Formulations

Common Areas of Confusion

Examples from my e-mail . . .

- I have been trying to write a stepwise function in AMPL but I have not been able to do so:

  ```
  fc[wh] = 100 if x[wh] <=5
           300 if 6 <= x[wh] <=10
           400 if 11 <= x[wh]
  ```

  where `fc` and `x` are variables.

- *I have a set of nonlinear equations to be solved, and variables are binary. Even I have an xor operator in the equations. How can I implement it and which solver is suitable for it?*

- I'm a recent IE grad with just one grad level IE course under my belt. . . .

  ```
  minimize Moves: sum{emp in GROUPA}
     (if Sqrt((XEmpA[emp] - XGrpA)^2 +
              (YEmpA[emp] - YGrpA)^2) > Ra then 1 else 0)
  ```

  Is there some documentation on when you can and cannot use the if-then statements in AMPL (looked through the related forum posts but still a bit confused on this)?

Common Areas of Confusion

Examples from my e-mail (cont'd)

- I have a problem need to add a such kind of constraint:

  Max[sum(Pi * Hi)]; i is from 1 to 24;

  in which Pi are constant and Hi need to be optimized.
  Bound is −180 <= Hi <= 270. One of the constraints is

  sum(Ci) = 0; here Ci = Hi if Hi > 0 and Ci = Hi/1.38 if Hi < 0

  Is it possiable to solve this kind of problem with lp_solve?
  and how to setup the constraint?
- *. . . is there a way to write a simple "or" statement in AMPL like in Java or C++?*
- I need to solve the following optimization problem:

  Minimize − |x1| − |x2|

  subject to

  x1 − x2 = 3

  Do you know how to transform it to standard linear program?

Currently Implemented

Extension to mixed-integer solver

- CPLEX indicator constraints
  - `Use[j] = 1 ==> Work[j] >= least_assign;`

Translation to mixed-integer programs

- General variable domains

  -
    ```
    var Work {j in SCHEDS} integer,
          in {0} union interval[lo_assign, hi_assign];
    ```
- Separable piecewise-linear terms
  - `<<avail_min[t]; 0,time_penalty[t]>> Use[t]`

Translation to general nonlinear programs

- Complementarity conditions

  -
    ```
    0 <= ct[cr,u] complements
          ctcost[cr,u] + cv[cr] >= p["C",u];
    ```

Prospective Extensions

Existing operators allowed on variables

- ❖ Nonsmooth terms
- ❖ Conditional expressions

New forms

- ❖ Operators on constraints
- ❖ New aggregate operators
- ❖ Generalized indexing: variables in subscripts
- ❖ New types of variables: object-valued, set-valued

Solution strategies

- ❖ Transform to standard MIPs
- ❖ ***Send to alternative solvers*** *(will return to this)*

Piecewise-Linear Terms

Transportation (multiple rates)

```
minimize Total_Cost:  sum {i in ORIG, j in DEST}
  << limit1[i,j], limit2[i,j];
     rate1[i,j], rate2[i,j], rate3[i,j] >> Trans[i,j];
```

```
minimize Total_Cost:  sum {i in ORIG, j in DEST}
  << {p in 1..npiece[i,j]-1} limit[i,j,p];
     {p in 1..npiece[i,j]} rate[i,j,p] >> Trans[i,j];
```

Production (overtime)

```
maximize Total_Profit:  sum {p in PROD, t in 1..T}
  (rev[p,t]*Sell[p,t] - pcost[p]*Make[p,t] - icost[p]*Inv[p,t]) -
  sum {t in 1..T} << avail_min[t]; 0,time_penalty[t] >> Use[t];
```

General Variable Domains

Workforce Scheduling

```
param least_assign >= 0;

var Work {j in SCHEDS} integer, in {0} union
   interval [least_assign, (max {i in SHIFT_LIST[j]} required[i])];
```

Extensions

Logical Operators

Flow shop scheduling

```
subj to NoConflict {i1 in JOBS, i2 in JOBS: ord(i1) < ord(i2)}:

   Start[i2] >= Start[i1] + setTime[i1,i2] or
   Start[i1] >= Start[i2] + setTime[i2,i1];
```

Balanced assignment

```
subj to NoIso {(i1,i2) in TYPE, j in ROOM}:

  not (Assign[i1,i2,j] = 1 and
     sum {ii1 in ADJ[i1]: (ii1,i2) in TYPE} Assign[ii1,i2,j] = 0);
```

Location-transportation

```
subj to Capacity {i in WHSE}:

   if Build[i] = 1
      then sum {j in CUST} Ship[i,j] <= cap[i]
      else forall {j in CUST} Ship[i,j] = 0;
```

Implication Operator

Multicommodity flow with fixed costs

```
subject to DefineUsedA {i in ORIG, j in DEST}:
   Use[i,j] = 0 ==> sum {p in PROD} Trans[i,j,p] = 0;
```

```
subject to DefineUsedB {i in ORIG, j in DEST, p in PROD}:
   Use[i,j] = 0 ==> Trans[i,j,p] = 0;
```

Workforce planning

```
var NoShut {m in MONTHS} binary;
var LayoffCost {m in MONTHS} >=0;

subj to NoShutDefn1 {m in MONTHS}:
   NoShut[m] = 1 ==> LayoffCost[m] = 0;

subj to NoShutDefn2 {m in MONTHS}:
   NoShut[m] = 0 ==> LayoffCost[m] =
      snrLayoffWages * ShutdownDays[m] * maxNumberSnrEmpl;
```

Extensions

Counting Operators

Transportation

```
subj to MaxServe {i in ORIG}:
  card {j in DEST: sum {p in PRD} Trans[i,j,p] > 0} <= mxsrv;
```

```
subj to MaxServe {i in ORIG}:
  count {j in DEST} (sum {p in PRD} Trans[i,j,p] > 0) <= mxsrv;
```

```
subj to MaxServe {i in ORIG}:
  atmost mxsrv {j in DEST} (sum {p in PRD} Trans[i,j,p] > 0);
```

Extensions

"Structure" Operators

Assignment

```
subj to OneJobPerMachine:
   alldiff {j in JOBS} (MachineForJob[j]);
```

```
subj to CapacityOfMachine {k in MACHINES}:
   numberof k {j in JOBS} (MachineForJob[j]) <= cap[k];
```

. . . argument in ( ) may be a more general list

Extensions

Variables in Subscripts

Assignment

```
minimize TotalCost:
  sum {j in JOBS} cost[j,MachineForJob[j]];
```

Sequencing

```
minimize CostPlusPenalty:
  sum {k in 1..nSlots} setupCost[JobForSlot[k-1],JobForSlot[k]] +
  sum {j in 1..nJobs} duePen[j] * (dueTime[j] - ComplTime[j]);

subj to TimeNeeded {k in 0..nSlots-1}:
  ComplTime[JobForSlot[k]] =
    min( dueTime[JobForSlot[k]],
         ComplTime[JobForSlot[k+1]]
           - setupTime[JobForSlot[k],JobForSlot[k+1]]
           - procTime[JobForSlot[k+1]] );
```

Extensions

Object-Valued Variables

Location

```
set CLIENTS;
set WHSES;

param srvCost {CLIENTS, WHSES} > 0;
param bdgCost > 0;

var Serve {CLIENTS} in WHSES;
var Open {WHSES} binary;

minimize TotalCost:
   sum {i in CLIENTS} srvCost[i,Serve[i]] +
   bdgcost * sum {j in WHSES} Open[j];

subject to OpenDefn {i in CLIENTS}:
   Open[Serve[i]] = 1;
```

Set-Valued Variables

Crew scheduling

```
set SKILLset {SKILLS} within STAFF;

var CREWset {FLIGHTS} within STAFF;

. . . . . . .

subject to CrewSize {j in FLIGHTS}:

  card (CREWset[j]) = nbCrew[j];

subject to SkillReq {i in SKILLS, j in FLIGHTS}:

  card (SKILLset[i] inter CREWset[j]) >= nbSkills[i,j];

subject to NonConsecutive {j in FLIGHTS}:

  CREWset[j] inter CREWset[next(j)] = { };
```

The Solvers

Communication while solver is active

- ❖ Speed up multiple solves
- ❖ Support callbacks

Conic programming

- ❖ Barrier solvers available
- ❖ Stronger modeling support needed

Nontraditional alternatives

- ❖ Global optimization
- ❖ Constraint programming
- ❖ Varied hybrids

Conic Programming

Simple convex quadratic constraints

- Ball: $x_1^2 + \ldots + x_n^2 \le b$
- Cone: $x_1^2 + \ldots + x_n^2 \le y^2, \;\; y \ge 0$
- Cone: $x_1^2 + \ldots + x_n^2 \le yz, \;\; y \ge 0, \;\; z \ge 0$

. . . variables can be generalized to linear terms

Similarities

- Describe by lists of coefficients
- Solve by extensions of LP barrier methods; extend to MIP

Differences

- Quadratic part not positive semi-definite
- Nonnegativity is essential
- ***Many convex problems can be reduced to these . . .***

Equivalent Problems: Minimize

Sums of . . .

- norms & squared norms
- norms / linear terms

Max of . . .

- norms
- logarithmic Chebychev terms
  - $\max_i \left|\log(a_i x) - \log(b_i)\right|$

Product of . . .

- negative powers
  - $\prod_i (a_i x + b_i)^{-\alpha_i}$ for rational $\alpha > 0$
- minus positive powers

. . . ***and certain sum-max combinations***

Equivalent Problems: Subject to

Similar expressions involving

- norms & squared norms
- norms / linear terms
- negative & minus positive powers
- minus positive powers

. . . see Jared Erickson's talk in MC33.1 for details

Modeling SOCPs

Current situation

- each solver recognizes some elementary forms
- modeler must convert to these forms

Goal

- recognize many equivalent forms
- automatically convert to a canonical form
- further convert as necessary for each solver

Example: Sum of Norms

```
param p integer > 0;
param m {1..p} integer > 0;
param n integer > 0;

param F {i in 1..p, 1..m[i], 1..n};
param g {i in 1..p, 1..m[i]};
```

```
param p := 2 ;
param m := 1 5  2 4 ;
param n := 3 ;

param g (tr): 1   2 :=
          1  12   2
          2   7  11
          3   7   1
          4   8   0
          5   4   . ;

param F := ...
```

Example: Original Formulation

```
var x {1..n};

minimize SumOfNorms:
   sum {i in 1..p} sqrt(
      sum {k in 1..m[i]} (sum {j in 1..n} F[i,k,j] * x[j] + g[i,k])^2 );
```

```
3 variables, all nonlinear
0 constraints
1 nonlinear objective; 3 nonzeros.

CPLEX 12.2.0.0: at12228.nl contains a nonlinear objective.
```

Example: Converted to Quadratic

```
var x {1..n};
var Max {1..p};

minimize SumOfNorms: sum {i in 1..p} Max[i];

subj to MaxDefinition {i in 1..p}:
   Max[i]^2 >=
      sum {k in 1..m[i]} (sum {j in 1..n} F[i,k,j] * x[j] + g[i,k])^2;
```

```
5 variables, all nonlinear
2 constraints, all nonlinear; 8 nonzeros
1 linear objective; 2 nonzeros.

CPLEX 12.2.0.0: QP Hessian is not positive semi-definite.
```

Example: Simpler Quadratic

```
var x {1..n};
var Max {1..p} >= 0;
var Fxplusg {i in 1..p, 1..m[i]};

minimize SumOfNorms: sum {i in 1..p} Max[i];

subj to MaxDefinition {i in 1..p}:
   Max[i]^2 >= sum {k in 1..m[i]} Fxplusg[i,k]^2;

subj to FxplusgDefinition {i in 1..p, k in 1..m[i]}:
   Fxplusg[i,k] = sum {j in 1..n} F[i,k,j] * x[j] + g[i,k];
```

```
14 variables:
      11 nonlinear variables
      3 linear variables
11 constraints; 41 nonzeros
      2 nonlinear constraints
      9 linear constraints
1 linear objective; 2 nonzeros.

CPLEX 12.2.0.0: primal optimal; objective 11.03323293; 11 barrier iters
```

Nontraditional Solvers

Global nonlinear

- BARON *
- LINDO Global *
- LGO

Constraint programming

- IBM ILOG CP
- ECLiPSe
- SCIP *

** combined with mixed-integer*

Implementation Challenges

Requirements

- ❖ Full description of functions
- ❖ Hints to algorithm
  - ∗ convexity, search strategy

Variability

- ❖ Range of expressions recognized
  - ∗ hence range of conversions needed
- ❖ Design of interface

The System

APIs & IDEs

- Current options
- Alternatives under consideration

AMPL in the cloud

- AMPL & solver software as a service
- Issues to be resolved

APIs (Programming Interfaces)

Current options

- AMPL scripting language
- put/get C interface
- OptiRisk Systems COM objects

Alternatives under consideration

- multiplatform C interface
- object-oriented interfaces in C++, Java, Python, . . .

Scripting Language

Programming extensions of AMPL syntax

```
for {i in WIDTHS} {
   let nPAT := nPAT + 1;
   let nbr[i,nPAT] := floor (roll_width/i);
   let {i2 in WIDTHS: i2 <> i} nbr[i2,nPAT] := 0;
};

repeat {

   solve Cutting_Opt;
   let {i in WIDTHS} price[i] := Fill[i].dual;

   solve Pattern_Gen;
   printf "\n%7.2f%11.2e  ", Number, Reduced_Cost;

   if Reduced_Cost < -0.00001 then {
      let nPAT := nPAT + 1;
      let {i in WIDTHS} nbr[i,nPAT] := Use[i];
   }
   else break;

   for {i in WIDTHS} printf "%3i", Use[i];
};
```

put/get C Interface

Send AMPL commands & receive output

- `Ulong put(GetputInfo *g, char *s)`
- `int get(GetputInfo *g, char **kind, char **msg, Ulong *len)`

Limitations

- Low-level unstructured interface
- Communication via strings

OptiRisk COM Objects

Object-oriented API

- Model management
- Data handling
- Solving

Limitations

- Windows only
- Older technology
- Built on put/get interface

API Development Directions

Multiplatform C interface

- Native to AMPL code
- Similar scope to COM objects

Object-oriented interfaces

- Built on C interface

IDEs (Development Environments)

Previous & current options

- AMPL Plus
- AMPL Studio

Alternatives under consideration

- Multiplatform graphical interface
- Spreadsheet interface

AMPL Plus

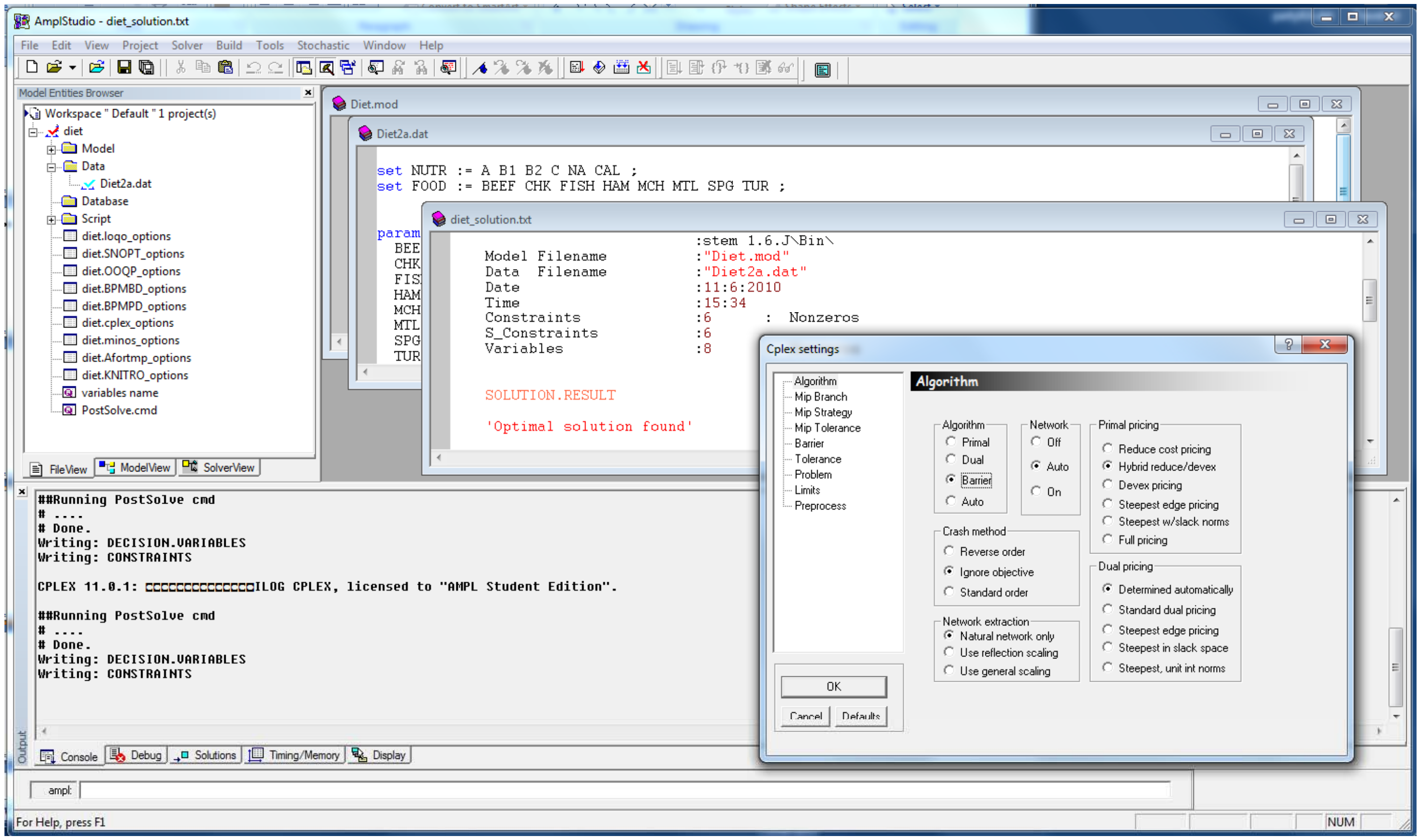
Menu-based GUI (1990s)

- Created by Compass Modeling Solutions
- Discontinued by ILOG

AMPL Studio

Menu-based GUI (2000s)

- Created by OptiRisk Systems
- Windows-based



Diet2a.dat:

```
set NUTR := A B1 B2 C NA CAL ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR ;

param
  BEE
  CHK
  FIS
  HAM
  MCH
  MTL
  SPG
  TUR
```

diet_solution.txt:

```
                             :stem 1.6.J\Bin\
    Model Filename           :"Diet.mod"
    Data  Filename           :"Diet2a.dat"
    Date                     :11:6:2010
    Time                     :15:34
    Constraints              :6       :  Nonzeros
    S_Constraints            :6
    Variables                :8


    SOLUTION.RESULT

    'Optimal solution found'
```

Console:

```
##Running PostSolve cmd
# ....
# Done.
Writing: DECISION.VARIABLES
Writing: CONSTRAINTS

CPLEX 11.0.1: □□□□□□□□□□□□□□ILOG CPLEX, licensed to "AMPL Student Edition".

##Running PostSolve cmd
# ....
# Done.
Writing: DECISION.VARIABLES
Writing: CONSTRAINTS
```

Cplex settings

Algorithm | Mip Branch | Mip Strategy | Mip Tolerance | Barrier | Tolerance | Problem | Limits | Preprocess

Algorithm

Algorithm: Primal [✓ Barrier] Dual Auto

Network: Off [✓ Auto] On

Primal pricing: Reduce cost pricing [✓ Hybrid reduce/devex] Devex pricing Steepest edge pricing Steepest w/slack norms Full pricing

Crash method: Reverse order [✓ Ignore objective] Standard order

Dual pricing: [✓ Determined automatically] Standard dual pricing Steepest edge pricing Steepest in slack space Steepest, unit int norms

Network extraction: [✓ Natural network only] Use reflection scaling Use general scaling

OK | Cancel | Defaults

IDE Development Directions

Multiplatform graphical interface

- ❖ Focused on command-line window
  - ∗ Same rationale as MATLAB
- ❖ Implemented using new API
- ❖ Tools for debugging, scripting, option selection . . .

Spreadsheet interface

- ❖ Data in spreadsheet tables (like Excel solver)
- ❖ AMPL model in embedded application

AMPL in the Cloud

AMPL as a service

- ❖ Solvers included
  - ∗ optional automated solver choice
- ❖ Charges per elapsed minute
- ❖ Latest versions available

Issues to be resolved

- ❖ Licensing arrangements with solvers
- ❖ Uploading & security of data
- ❖ Limitations of cloud services