

# **LPL : A Mathematical Modeling Language : Short Review**

Tony Hürlimann  
tony.huerlimann@unifr.ch  
Departement of Informatics  
University of Fribourg, Switzerland

December 8, 2011

# 1 Introduction

LPL is a computer-executable mathematical modeling language, which can be used to build, modify and document (linear and non-linear) mathematical and logical models. Initially, LPL was designed at the Department of Informatics at the University of Fribourg in the nineties to formulate large linear programming models with thousands of variables and constraints. For example, a general LP of the form  $\min\{cx \mid Ax \geq b\}$  with random data containing 2000 variables and 1000 constraints can be formulated in LPL as follows:

```
MODEL GENLP "The general LP";
SET m := /1:1000/;    n := /1:2000/;
PARAMETER
  A{m,n} := IF(Rnd(0,1)<0.02 , Rnd(0,60));
  c{n}    := IF(Rnd(0,1)<0.87 , Rnd(0,9));
  b{m}    := IF(Rnd(0,1)<0.87 , Rnd(10,70000));
VARIABLE x{n};
CONSTRAINT R{m}: SUM{n} A*x <= b;
MAXIMIZE ObjFunction: SUM{n} c*x;
WRITE 'Objective Function Value = %7.2f \n\n\
      n          x[n]\n%-8s %6.2f\n' :
      ObjFunction, ROW{n|x} (n,x);
END
```

This is a *complete* formulation and can be submitting to the LPL compiler; it takes 10 seconds to generate and solve the whole model Through the Internet or on a typical PC. This model can be executed through the Internet. Try it by clicking here: [lp2000\[?\]](http://diuflx71.unifr.ch/lpl/Solver.jsp?name=/lp2000)<sup>1</sup>.

Several large real-live models have been written and are maintained in LPL. A typical application is a model with more than 20'000 variables and 5'000 constraints which support the Swiss Federal security policy of food supply. The model is maintained at the Department of Informatics at the University of Fribourg and is used at the Federal Department of Agriculture of Switzerland. In addition, several institutions and companies use LPL as a tool in optimization.

---

<sup>1</sup><http://diuflx71.unifr.ch/lpl/Solver.jsp?name=/lp2000>

## 2 LPL for Non-linear Problems

The actual version of the LPL modeling language however, goes far beyond the paradigm of linear programming. It can formulate non-linear models which are submitted to a non-linear solver such as LOQO, MINOS, or ConOPT. As an example, let us formulate and solve the problem of finding the maximum diameter of equal non-overlapping  $n$  circles contained in a unit square [4]. This problem is equivalent to maximizing the minimum pairwise distance among  $n$  points in a unit square and can be formulated mathematically as follows :

$$\begin{aligned} & \max && t \\ & \text{subject to} && (x_i - x_j)^2 + (y_i - y_j)^2 \geq t \text{ with } i, j \in \{1, \dots, n\}, i < j \\ & && x_i, y_i \in [0 \dots 1] \end{aligned}$$

In LPL, the problem with  $n = 15$  can be stated as follows:

```
MODEL circles "Packing Circles in a Square";
  SetRandomSeed(55);
  PARAMETER n := 15 "The number of circles";
             nt := Trunc(SQRT(n)+0.5);
  SET i, j:= 1..n;
  PARAMETER -- initial center of circles
    x0{i}:=Rnd(((i-1)%nt)/nt,((i-1)%nt+1)/nt);
    y0{i}:=y0[i]:=Rnd(Trunc((i-1)/nt)/nt,Trunc((i-1)/nt+1)/nt),
                IF(y0>1,y0-1,y0));
  VARIABLE
    x{i} [0,1] := y0 "x-position";
    y{i} [0,1] := y0 "y-position";
    t "Diameter of the circles";
  CONSTRAINT
    R{i,j|i<j}: (x[i]-x[j])^2+(y[i]-y[j])^2 >= t
    "The circles must not overlap";
  MAXIMIZE obj: t "Make circles as large as possible";

  PARAMETER T := sqrt(t);
  WRITE 'Diameter of the circles is: %10.8f\n' : T;
END
```

LPL generates automatically the derivatives and then handed the problem over to a non-linear solver which finds the best solution known for this problem at the present time. The initial points  $(x_0, y_0)$  are given by (i) partitioning the unit square into equal smaller squares and then (ii) generate random points inside each smaller square. The model can be executed through the Internet. Try it by clicking here: [circles\[?\]](#)<sup>2</sup>.

### 3 LPL for Discrete Problems

LPL can also formulate discrete models. As an example, let us formulate a classical NP-hard scheduling problem: the single machine weighted tardiness problem. This problem can be stated as follows:  $n$  jobs are given, each consumes a certain time  $p_j$  to be processed on the single machine. The machine can only process one job at the same time. Processing of a job cannot be interrupted until the job has been completed. Furthermore, each job has a due date  $d_j$  and a weight  $w_j$ . The problem is to find a sequence of processing the jobs such that total tardiness over all jobs is minimized. If job  $j$  is completed at time  $C_j = \sum_{i \leq j} p_i$ , then the tardiness of job  $j$  is defined as:  $T_j = \max(0, C_j - d_j)$ . The objective is now to minimize their weighted sum:  $\sum_{j \in J} w_j T_j$  over all permutations of the jobs.

Mathematically, this problem can be stated as follows: Let  $\pi$  be a vector of the  $n$  distinct numbers in the range  $[1, n]$ . Then one needs to minimize the expression

$$\min \left( \sum_{i=1}^n w_{\pi_i} \cdot \max(0, \sum_{j=1}^i p_{\pi_j} - d_{\pi_j}) \right)$$

over all permutations of  $\pi$ .

A concrete problem instance with 40 jobs – again using a randomly generated data set is modelled below<sup>3</sup>.

The problem is automatically handed over to a heuristic tabu-search solver intergrated within the LPL package. This solver finds an optimal solution in 90% of all instances with 40 jobs in less than 1 minute. As far as the author is aware, no other modelling system – even commercial once – is capable to formulate such problems in this efficient and short way. The model can be executed through the Internet. Try it by clicking here: : [wtardy\[?\]](#)<sup>4</sup>.

<sup>2</sup><http://diuflx71.unifr.ch/lpl/Solver.jsp?name=/circles>

<sup>3</sup>The data set is chosen in a way that the model instances are especially hard to solve. For further information on how to generate such a data set, see [3]

<sup>4</sup><http://diuflx71.unifr.ch/lpl/Solver.jsp?name=/wtardy>

The model can be coded in LPL as follows:

```
MODEL wT "One machine weighted tardiness";
SET i, j "jobs" := /1:40/;
PARAMETER TF:=0.6; RDD:=0.2; C;
p{i}:=Trunc(Rnd(1,100)); PP:=SUM{i} p;
l:=Trunc(PP*(1-TF-RDD/2)); u:=Trunc(PP*(1-TF+RDD/2));
d{i}:=Trunc(Rnd(1,u)); d{i}:=IF(d<p,p,d);
w{i} := 1;
DISTINCT VARIABLE x{i} [1,#i] :=i ;
MINIMIZE swT: C:=0, SUM{i} (0 ?> (C:=C+p[x],C-d[x]));
END
```

LPL has many other unique features: a powerful report and input generator; formulation of logical constraints; unit checker; semi-automatically documentation generator; a database generator; an open and easy-to-use solver interface to many solvers. The whole functionality of LPL can also be linked as a dynamic link library into other applications. A restricted version of the LPL software together with a complete documentation and a short tutorial is available free of charge at the download page of the LPL-site:

<http://www.virtual-optima.com>

## References

- [1] Hürlimann T., Reference Manual for the LPL Modeling Language (at the LPL-site).
- [2] Hürlimann T., Mathematical Modeling and Optimization, An Essay for the Design of Coputer-Based Modeling Tools, Kluwer Academic Publ., (Applied Optimization 31), 1999.
- [3] Potts C.N. and van Wassenhove L.N., A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, Operations Research 33(2) (1985), p. 363-376.
- [4] Maranas C.D., Floudas C.A., Pardalos P.M., New Results in the Packing of Equal Circles in a Square, Working Paper, 1993.