

LPL : Eine mathematische Modellersprache

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

14. Oktober 2008

1 Einleitung

LPL ist eine computer-ausführbare, mathematische Modelliersprache, die dazu verwendet werden kann, lineare und nicht-lineare mathematische Modelle zu formulieren. Die erste Version von LPL wurde in den Neuziger Jahren am Institut für Informatik der Universität Freiburg entworfen, und war ursprünglich dafür konzipiert worden, grosse lineare Optimierungsmodelle mit Tausenden von Variablen und Beschränkungen zu formulieren. Beispielsweise, ein generisches Modell der Form $\min\{cx \mid Ax \geq b\}$ mit zufallsgenerierten Daten, welches aus 2000 Variablen und 1000 linearen Beschränkungen besteht, kann in LPL wie folgt kodiert werden:

```
MODEL GENLP "The general LP";
  OPTION randomSeed := 1;
  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
```

Dies ist eine vollständige Formulierung des Problems. Ein LPL-Compiler zusammen mit einem externen Lösungsalgorithmus braucht auf einem typischen PC oder per Internet gerade mal zehn Sekunden, um das gesamte Modell zu generieren und die optimale Lösung auszugeben. Das Modell kann per Internet an folgender Adresse getestet werden: [lp2000¹](http://diuflx71.unifr.ch/lpl/GetModel?name=lp2000).

LPL wird in der Praxis in verschiedenen Unternehmungen eingesetzt, die grosse LP lsen mssen. Probleme mit 20'000 Variablen und 10'000 linearen Beschrnkungen sind keine Seltenheit.

Die Funktionalitt von LPL geht allerdings weit über das Paradigma der linearen Optimierung hinaus. Mit ihr können nicht-lineare Modelle formu-

¹<http://diuflx71.unifr.ch/lpl/GetModel?name=lp2000>

liert werden, die automatisch einem kommerziellen Lösungsalgorithmus wie LOQO, MINOS oder ConOPT übergeben werden können. Sie kann auch gewisse diskrete Modelle behandeln. Es sollen dafür je noch ein Beispiel gegeben werden.

2 Ein nicht-lineares Problem

Das Problem besteht darin, für alle n sich nicht überlappenden Kreise den größten Kreisradius zu finden, sodass sie alle noch innerhalb eines Einheitsquadrats Platz finden. Dieses Problem ist äquivalent damit, paarweise Distanzen zwischen n Punkten im Einheitsquadrat zu maximieren und kann mathematisch folgendermaßen formuliert werden (siehe [?]):

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

In LPL kann das Problem mit $n = 15$ folgendermaßen formuliert werden:

```
MODEL circles "Packing Circles in a Square";
OPTION randomSeed := 55;
PARAMETER n := 15 "The number of circles";
          nt := TRUNC(SQRT(n)+0.5);
SET i ALIAS j:= 1..n;
PARAMETER -- initial center of circles
  x0{i}:=RND(((i-1)%nt)/nt,((i-1)%nt+1)/nt);
  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 generiert automatisch die Ableitungen, wie sie für einen nicht-linearen Solver benötigt werden, dann wird es dem nicht-linearen Solver übergeben und gelöst. Die Initiallösung (x_0, y_0) wird generiert, indem (i) das Einheitsquadrat in n gleich grosse kleinere Quadrate zerlegt wird, und (ii) ein Punkt innerhalb jedem kleineren Quadrat zufallsartig generiert wird. Das Modell kann über Internet an folgender Adresse getestet werden: [circles²](#).

3 Ein diskretes Reihenfolge-Problem

Als Beispiel soll das klassische one-machine tardiness Problem herangezogen werden. Dieses Problem kann wie folgt geschrieben werden: Gegeben seien n Aufgaben (jobs), die von einer Maschine sequentiell in je p_j Zeiteinheiten bearbeitet werden können. Jede Aufgabe sollte zu einem bestimmten Zeitpunkt d_j abgeschlossen sein und besitzt zudem noch eine Bearbeitungspriorität w_j . Das Problem ist, eine bestimmte Abarbeitungssequenz der Aufgaben auf der Maschine zu finden, sodass die Gesamtzeit der zu spät vollendeten Aufgaben minimal wird. Wenn eine Aufgabe j zur Zeit $C_j = \sum_{i \leq j} p_i$ vollendet ist, dann kann man die "Verspätung der Aufgabe j " als $T_j = \max(0, C_j - d_j)$ definieren. Das Ziel ist also, den Betrag $\sum_{j \in J} w_j T_j$ über alle möglichen Sequenzen zu minimieren. Ein konkretes Problem mit 40 Aufgaben und zufallsgenerierten Daten kann in LPL folgendermassen formuliert werden (siehe [?] zur Daten-Generierung):

```
MODEL wT "One machine weighted tardiness";
OPTION randomSeed := 1;
SET i ALIAS 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 ;
```

²<http://diuflx71.unifr.ch/lpl/GetModel?name=circles>

```
MINIMIZE swT: C:=0, SUM{i} (0 ?> (C:=C+p[x],C-d[x]));  
END
```

Auch dies ist eine vollständige Formulierung des Problems. In diesem Fall, wird das Problem von LPL automatisch einem heuristischen TABU-Suchverfahren zur Lösung übergeben. Dieses findet die optimale Lösung innerhalb einer Minute in 90% der Fälle. Kein anderes Modellersystem – ob kommerziell oder nicht – (soweit dem Author bekannt) kann ein solches Problem in dieser effizienten und konzisen Weise formulieren und lösen.

LPL besitzt viele weitere Vorteile gegenüber anderen verwandten Systemen: Ein mächtiger Eingabe- und Rapportgenerator ist integriert, logische Beschränkungen können formuliert werden, Masseinheiten überprüfen die Konsistenz von Modellen, die Modelldokumentation kann halb-automatisch generiert werden, ein Datenbankgenerator ist eingebunden, LPL besitzt eine offene und einfach zu programmierende Schnittstelle zu vielen Lösungsalgorithmen. Die Funktionalität von LPL kann auch als dynamisch-linkbare Bibliothek in andere Applikationen eingebunden werden. Eine beschränkte Version der LPL-Software, zusammen mit einer vollständigen Dokumentation und einem Tutorial ist frei verfügbar und kann vom folgenden LPL-Server heruntergeladen werden :

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

Literatur

- [1] Hürlimann T., Reference Manual for the LPL Modeling Language, (LPL-site).
- [2] Hürlimann T., [1999], Mathematical Modeling and Optimization, An Essay for the Design of Computer-Based Modeling Tools, Kluwer Academic Publ., (Applied Optimization 31).
- [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.