LANGFORD'S PROBLEM

ERWIN KALVELAGEN

Abstract. This documents describes how small instances of Langford's Problem can bs solved using Mixed Integer Programming.

1. Introduction

Langford's Problem [1] is an interesting theoretical problem. It originated from a father, being a Scottish mathematician, watching his children playing with colored blocks. Consider the sequences depicted in figures 1 and 2.

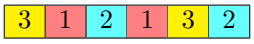

Figure 1. Langford's problem instance $L(2,3)$

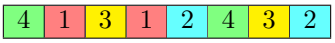

Figure 2. Langford's problem instance $L(2,4)$

Each number occurs two times (hence the 2 in names of the sequences). Between equal numbers k there are exactly k other numbers. The sequences are named $L(m,n)$ with m the multiplicity, and n the order of the sequence. Not all sequences $L(m,n)$ have solutions: there are no instances of $L(2,5)$ and $L(2,6)$. Both $L(2,3)$ and $L(2,4)$ have exactly one instance (apart from the obvious symmetry by reversing the sequence). Some sequences have a large number of solutions. For instance $L(2,7)$ has 26 solutions, and $L(2,8)$ has 150 solutions.

An example of a sequence with triples is $L(3,9)$. A possible solution for this sequence is:

$$L(3,9) = (3,4,7,9,3,6,4,8,3,5,7,4,6,9,2,5,8,2,7,6,2,5,1,9,1,8,1) \tag{1}$$

2. A first model

In this section we will try to develop an integer programming model that finds a sequence $L(3,9)$.

Date: September 20, 2002.

2.0.1. *Model langford.gms.* [1]

```
$ontext

   Langford's Problem L(3,9)

   Erwin Kalvelagen, July 2002

   References:
      http://www.lclark.edu/~miller/langford.html


$offtext


set n 'number (0..9)' /n1*n9/;
set m 'each number occurs three times' /m1*m3/;

alias (n,nn);
alias (m,mm);

set notfirst(m);
notfirst(m)$(ord(m)>1)=yes;

variable x(n,m) 'locations (1..27)';
x.lo(n,m) = 1;
x.up(n,m) = card(n)*card(m);

binary variable P(n,m,nn,mm) 'Permutation matrix';

free variable dummy    'objective variable';

equations
   unique(n,m)         'x should be unique'
   permutation1(n,m)   'Permutation matrix'
   permutation2(n,m)   'Permutation matrix'
   triplets(n,m)       'make sure spacing is correct'
   objective           'dummy objective function'
   summation           'extra constraint'
;

*
* Permutation matrix makes sure x's are unique
*
parameter v(n,m);
v(n,m) = ord(n)+card(n)*(ord(m)-1);
display v;
unique(n,m).. x(n,m) =e= sum((nn,mm),P(n,m,nn,mm)*v(nn,mm));
permutation1(n,m).. sum((nn,mm), P(n,m,nn,mm)) =e= 1;
permutation2(n,m).. sum((nn,mm), P(nn,mm,n,m)) =e= 1;


*
* place triples with correct amount of space between them
*
triplets(n,notfirst(m)).. x(n,m) =e= x(n,m-1) + ord(n) + 1;


*
* extra constraint sum x = 1 + 2 + ... + 27
*
summation.. sum((n,m), x(n,m)) =e= 0.5*(card(n)*card(m))*(card(n)*card(m)+1);

*
* dummy objective
*
objective.. dummy =e= 0;

$ontext
```

[1] http://amsterdamoptimization.com/models/langford.gms

```
*
* this can be used to test the formulation.
* test solution 34793648357469258276251918 1
*
parameter solution(n,m) /
   n3.m1 1
   n4.m1 2
   n7.m1 3
   n9.m1 4
   n3.m2 5
   n6.m1 6
   n4.m2 7
   n8.m1 8
   n3.m3 9
   n5.m1 10
   n7.m2 11
   n4.m3 12
   n6.m2 13
   n9.m2 14
   n2.m1 15
   n5.m2 16
   n8.m2 17
   n2.m2 18
   n7.m3 19
   n6.m3 20
   n2.m3 21
   n5.m3 22
   n1.m1 23
   n9.m3 24
   n1.m2 25
   n8.m3 26
   n1.m3 27
/;

x.fx(n,m) = solution(n,m);
$offtext


*
* priorities
*
P.prior(n,m,nn,mm) = 2*card(n)-ord(n)-ord(nn);

*
* write Cplex option file
*
file opt /cplex.opt/;
putclose opt 'mipemphasis 1'/;

model langford /all/;

*
* turn on option file reading and usage of priorities
*
langford.prioropt=1;
langford.optfile=1;

*
* select Cplex as MIP solver
*
option mip=cplex;
option iterlim=100000000;
option reslim=100000;

solve langford minimizing dummy using mip;

option x:0:1:1;
display x.l;
```

There are some remarks we can make about this model. First note that the permutation is now a four dimensional structure as the decision variable that must

be unique, is two dimensional. We can add the constraint $\sum x_i = 1 + 2 + \cdots + 27$. This constraint is automatically valid for an integer solution, but it helps the optimization algorithm when there are still fractional solutions. We have added priorities that tell the solver to place first the high numbered triples, as they are widely spaced, and are difficult to "squeeze in." Priorities are especially important when the solver does not receive any good pricing signals, as this model does not have a proper objective function that can help to distinguish solutions. When solving this model with Cplex we can add the option `mipemphasis 1` (emphasis on finding a feasible integer solution) as the first integer solution is immediately (proven) optimal, given the dummy objective function. Note that this also implies that we don't have to set an `OPTCR` value. A simple way to check the solution given in equation (1) is also provided. It is commented out, but a few simple edits will enable it.

The solution found with Cplex is as follows:

```
----    132 VARIABLE x.L  locations (1..27)

            m1          m2          m3

n1          23          25          27
n2          18          21          24
n3           3           7          11
n4          10          15          20
n5           2           8          14
n6           5          12          19
n7           1           9          17
n8           4          13          22
n9           6          16          26
```

This is different than the solution mentioned in equation (1). Indeed, there are three different solutions for $L(3, 9)$.

3. A second model

This is not the only possible formulation (which is a familiar theme). For instance we can calculate for all combinations (i, j), (i', j'):

$$d(i, j, i', j') = x(i, j) - x(i', j') \tag{2}$$

and then require that $|d(i, j, i', j')| \geq 1$ for all $(i, j) \neq (i', j')$. The minimum distance requirement can be linearized by saying $d(i, j, i', j') \leq -1 \vee d(i, j, i', j') \geq 1$. This "or"-constraint can be modeled easily using binary variables $\delta_{i,j,i',j'}$:

$$\begin{aligned} d(i, j, i', j') &\geq 1 - \delta_{i,j,i',j'} M \\ d(i, j, i', j') &\leq -1 + (1 - \delta_{i,j,i',j'}) M \end{aligned} \tag{3}$$

We need to choose an appropriate value for M. The following model implements this.

3.0.2. *Model langford2.gms.* [2]

```
$ontext

   Langford's Problem L(3,9),
   alternative formulation

   Erwin Kalvelagen, July 2002

   References:
```

[2] `http://amsterdaomoptimization.com/models/langford2.gms`

```
      http://www.lclark.edu/~miller/langford.html



$offtext


set n 'number (0..9)' /n1*n9/;
set m 'each number occurs three times' /m1*m3/;

alias (n,nn);
alias (m,mm);

set notfirst(m);
notfirst(m)$(ord(m)>1)=yes;

integer variable x(n,m) 'locations';
free variable dist(n,m,nn,mm) 'distance between two locations';
binary variable delta(n,m,nn,mm) 'used in |distance| >= 1';

scalar mn 'm times n';
mn = card(m)*card(n);

x.lo(n,m)=1;
x.up(n,m)=mn;


free variable dummy    'objective variable';

equations
   triplets(n,m)       'make sure spacing is correct'
   objective           'dummy objective function'
   summation           'extra constraint'
   distance            'calculate distances between x(i,j) and x(p,q)'
   mindist1            'abs(distance) >= 1'
   mindist2            'abs(distance) >= 1'
;

*
* distance measure approach
*
set comb(n,m,nn,mm);
comb(n,m,nn,mm)$((ord(n)<>ord(nn)) or (ord(m)<>ord(mm))) = yes;
distance(comb(n,m,nn,mm)).. dist(comb) =e= x(n,m) - x(nn,mm);
mindist1(comb(n,m,nn,mm)).. dist(comb) =g= 1 - delta(comb)*mn;
mindist2(comb(n,m,nn,mm)).. dist(comb) =l= -1 + (1-delta(comb))*mn;

dist.lo(comb) = -mn+1;
dist.up(comb) = mn-1;


*
* place triples
*
triplets(n,notfirst(m)).. x(n,m) =e= x(n,m-1) + ord(n) + 1;


*
* extra constraint
*
summation.. sum((n,m), x(n,m)) =e= 0.5*mn*(mn+1);

*
* dummy objective
*
objective.. dummy =e= 0;

$ontext
*
* this can be used to test the solution
* test solution 347936483574692582762519181
*
```

```
parameter solution(n,m) /
   n3.m1 1
   n4.m1 2
   n7.m1 3
   n9.m1 4
   n3.m2 5
   n6.m1 6
   n4.m2 7
   n8.m1 8
   n3.m3 9
   n5.m1 10
   n7.m2 11
   n4.m3 12
   n6.m2 13
   n9.m2 14
   n2.m1 15
   n5.m2 16
   n8.m2 17
   n2.m2 18
   n7.m3 19
   n6.m3 20
   n2.m3 21
   n5.m3 22
   n1.m1 23
   n9.m3 24
   n1.m2 25
   n8.m3 26
   n1.m3 27
/;

x.fx(n,m) = solution(n,m);
$offtext

x.prior(n,m) = 2*card(n) - 2*ord(n);
delta.prior(n,m,nn,mm) = 2*card(n)-ord(n)-ord(nn);

*
* write Cplex option file
*
file opt /cplex.opt/;
putclose opt 'mipemphasis 1'/;



model langford /all/;

*
* turn on option file reading and usage of priorities
*
langford.prioropt=1;
langford.optfile=1;

*
* select Cplex as MIP solver
*
*option mip=cplex;

option iterlim=1000000;

solve langford minimizing dummy using mip;

display x.l;
```

The results with Cplex 7.5 are displayed in table 1. In this case the model with more variables and equations performs better than the smaller model. This is not unusual in integer programming: it is difficult to predict performance based on the size of the model.

In [2] a constraint programming approach to solve this problem is discussed.

	langford.gms	**langford2.gms**
equations	101	2126
variables	757	1432
discrete variables	729	729
resource usage	370	80
iterations	3182053	258489
nodes	47520	11339

TABLE 1. Results for the Langford problems

REFERENCES

1. C. Dudley Langford, *Problem*, Mathematical Gazette **42** (1958), 228.
2. Barbara M. Smith, *Modelling a permutation problem*, Tech. Report 2000.18, University of Leeds, School of Computer Studies, June 2000.

AMSTERDAM OPTIMIZATION MODELING GROUP LLC, WASHINGTON DC
E-mail address: erwin@amsterdamoptimization.com