SEP # :

Title: An interface to CLP – a powerful replacement for linpro and readmps
Version: 1.0
Author: Y. Collette
Review:
Commented:
State: Draft
Scilab-Version: 5.0
Vote:"No: discussions and suggestions"
Created: Monday, 22 september 2008

Abstract

This SEP propose a new interface to the CLP library as a replacement for linpro and readmps

Rationale

Linpro was the tool from the optimization toolbox dedicated to linear programming. Due to licensing problem, this tool has been removed from scilab-5.0.

It is maybe time to consider a new solution to replace linpro and readmps /mps2linpro (a tool to read mps files which is an exchange format for linear programming problems).

Today, there are 3 (and maybe more) available linear programming libraries:

- glpk: http://www.gnu.org/software/glpk/;
- lpsolve: http://lpsolve.sourceforge.net/5.5/;
- CLP: http://www.coin-or.org/projects/Clp.xml;

These 3 tools deal with sparse matrices. This is a good thing because this obviously allows to deal with huge linear programming problems. Linpro was not able to deal with sparse matrices and was of limited utility on big problems. Linpro was also buggy and not easy to maintain (a big piece of fortran code).

These 3 tools deal with integer and / or binary variables. This is also a good thing because this extends the applicability of linear programming to a great diversity of problems. Linpro was not able to deal with integer and / or binary variables.

Why do we have to choose CLP ?

A first interface between GLPK and Scilab was designed, but soon a big problem shows up: it was not possible to short-circuit some asserts which where disseminated everywhere in the GLPK code. So, when you try to access an index of a matrix which was not defined, if your constraint matrix has not a good shape, an assert was raised and GLPK + Scilab crashed. The GLPK developper (M. Makorin) is not willing to modify these asserts and define hooks to allow scilab to deal with these asserts.

Then interfaces between LPSOLVE, CLP and Scilab were designed. Soon CLP appears to be the best solution (look at the benchmark here:http://plato.asu.edu/bench.html).

This benchmark underlines that CLP was a better free tool for linear programming than both GLPK and LPSOLVE. Better than this, CLP is a commercial tool from IBM (OSL) which has been turned into opensource projects in 2004. If you want more informations concerning CLP, see https://projects.coin-or.org/Clp/wiki/FAQ.

CLP is a valuable tool. See the success stories: https://projects.coin-or.org/Clp/wiki/SuccessStories

CLP is able to deal with sparse matrices, CLP has a built in support for reading and writing mps files. CLP is actively supported by a community and IBM. CLP is open source and CLP is a C++ library (easier to maintain than the fortran linpro).

The interface to scilab is currently in C++ and is now stable enough to load and solve tests from the lplib (http://wpweb2.tepper.cmu.edu/fmargot/mps.html) and miplib (http://miplib.zib.de/).

Be careful, these sets of instances are sometimes infeasible (we need to know what is the behavior of a solver faced to an infeasible instance) and sometimes extremely difficult to solve (and thus they require other solving strategy like branch and cut – see the CBC library on the coin-or website[1]).

This preliminary interface can be found here: http://code.google.com/p/scilab-mip/

With this interface will come another interface to readmps. This function has been splitted into 2 parts: a first function called read_mps_file which returns only the informations related to the size of the problem. If the problem is loadable into scilab (not too big), then we can call read_mps_file_MP which returns all the data related to the problem. The sparse matrix of constraints, the vector of coefficients of the objective function, etc ...

Today, the readmps functions are based on the glpk interface but will be switched to the CLP interface to have the support for read / write mps files.

Example Usage

```
// A LP example which defines A as a sparse matrix


printf('Solve problem with sparse matrix\n');
// c the objective function coefficients
// a the constraint matrix coefficients
// b the bounds of the constraints
c = [0 0 0 -1 -1]';
a = sparse([-2 0 0 1 0;...
             0 1 0 0 2;...
             0 0 1 3 2]);
b       = [4 12 18]';
// lb the vector of lower bounds
// ub the vector of upper bounds
lb      = [0,0,0,0,0]';
ub      = 100*[1,1,1,1,1]';
// vartype the kind of variables. I stands for integer.
vartype = 'IIIII'; // All the variables are integer
```

1 The CBC library is used in mathematica. See https://projects.coin-or.org/Cbc/wiki/SuccessStories

```
// in the param structure, parameters related to the behavior of clp are set
param = init_param();
param = add_param(param,'maxnumiterations',10000);
param = add_param(param,'maxnumseconds',10000);
param = add_param(param,'primaltolerance',1e-7);
param = add_param(param,'dualtolerance',1e-7);
param = add_param(param,'verbose',1);
param = add_param(param,'solver',1); // 3 interior - other simplex
param = add_param(param,'optim_dir', 1); // optimisation direction:
// 1 - minimize, -1 - maximize, 0 - ignore

[xmin,lambda,status] = clp([],c,a,b,[],[],lb,ub,vartype,param);
// The first argument of clp is a sparse quadratic matrix.
// CLP can solved quadratic problems.
// xmin is the solution found by clp
// lambda is the vector of Lagrangian coefficients
// status is an integer indicating how has ended the optimization

printf('solution found: \n');disp(xmin);
printf('status = %d\n',status); disp(lambda);
```

Changelog

1.0 — Initial version

Copyright