qsopt_ex-interface

An Interface to QSopt exact LP solver

1.0

03/28/2016

Jayant Apte

Jayant Apte
Email: jayant91089@gmail.com
Homepage: https://sites.google.com/site/jayantapteshomepage/
Address: Department of Electrical and Computer Engineering
Drexel University
Philadelphia, PA 19104

Contents

Chapter 1

Introduction

qsopt_ex-interface is a GAP package that provides an interface to *QSopt* exact rational linear program solver [ACDE09] by Applegate,Cook,Dash and Espinoza. This is a minimalist package exposing parts of qsopt to GAP. The particular version of QSopt-exact solver this package currently follows is 2.5.10-patch 3 of a fork of the original software maintained by Jon Lund Steffenson [Ste15], which removes certain dependencies and makes the software easier to build. qsopt_ex-interface provides a C wrapper qsinterface.c to the solver. It is currently available for Unix/Linux systems running GAP 4.5+.

Chapter 2

Installation

Assuming you already have GAP 4.5+ installed, you can follow the steps below to install the package:

- To get the newest version of `qsopt_ex-interface`, download the .zip archive from `https://github.com/jayant91089/qsopt_ex-interface` and unpack it using '`unzip qsopt_ex-interface-x.zip`' in the terminal. Do this preferably inside the *pkg* subdirectory of your GAP 4 installation. It creates a subdirectory called `qsopt_ex-interface`. If you do not know the whereabouts of the *pkg* subdirectory, invoke the following in GAP:

```
GAPInfo.("RootPaths");
```

Look for pkg directory inside any of the paths returned.

- Once unpacked, go to `qsopt_ex-interface` directory and run the install script `unix-install.sh` from the terminal as `sh unix-install.sh`. This locally installs qsopt exact and its dependencies (GMP [GtGdt15],libz and libbz2) in lib and include folders. Alternatively, if you have qsopt-exact and GMP already installed on your system, you can edit the Makefile inside `qsopt_ex-interface` directory so that gcc finds the .so libraries. In latter case, you must manually '`make all`' from the terminal inside `qsopt_ex-interface` directory.
- Above step creates an executable \texttt{qsi} inside the `qsopt_ex-interface` directory, which serves as the interface. Note that before using the package in GAP, one must edit either the environment variable `LD_LIBRARY_PATH` or the so that \texttt{qsi} finds the locally installed libraries.
- One can now start using `qsopt_ex-interface` by invoking

```
LoadPackage( "qsopt_ex-interface");
```

from within GAP. To expose more `QSopt exact` functionality to GAP, one can extend the C part of the interface i.e. `qsinterface.c`. The relevent details of how the interface works are in `qsinterface.c` itself.

Chapter 3

Usage

3.1 Available functions

In this section we shall look at the functions provided by qsopt_ex-interface. `qsopt_ex-interface` allows GAP to communicate with external LP solver process via a stream object of category IsInputOutputStream(). This stream serves as a handle via which one can load/solve/modify linear programs. Note that it is possible to maintain several such steams (and hence LPs) at any given time. However, the gap commands to solve/modify these LPs that are currently available in this package are blocking functions.

3.1.1 LoadQSLP

▷ `LoadQSLP(`*`obj, A, b, linrows, qs_exec, optargs`*`)` (function)

Returns: A list

This function loads an LP by invoking external qsopt-exact LP solver process. It accepts following arguments:

- obj - Objective function coefficients, provided as a list
- A - A list of lists corresponding to constraints
- b - Right hand side of constraints
- $linrows$ - A list of indices of members of A that are equalities
- *qs_exec* - A string describing complete path to 'qsi' executable (including 'qsi')

Returns a list $[s, rval]$ where 's' is a gap object of category IsInputOutputStream() and 'rval' $= 1/-1$ indicates success/failure. If 'rval=1', 's' is ready to be be used to solve linear programs.

3.1.2 LoadQSLPobj

▷ `LoadQSLPobj(`*`s, obj`*`)` (function)

Returns: An integer

This function loads a new objective. It accepts following arguments:

- s - gap object of category IsInputOutputStream(), handle to an already loaded LP

- obj - Objective function coefficients, provided as a list

Returns an integer 'rval' $= 1/-1$ that indicate success/failure. If 'rval=1', the LP associated with 's' is successfully modified.

3.1.3 SolveQSLP

▷ `SolveQSLP(s, optargs)` (function)

Returns: An integer

This function solves an LP by invoking external qsopt-exact LP solver process. It accepts following arguments:

- s - gap object of category IsInputOutputStream(), handle to an already loaded LP
- $optargs$ - A list of optional arguments. Currently supports only one optional argument, which is an integer specifying simplex variant to use: $optargs = [1]$ for primal simplex, $optargs = [2]$ for dual simplex and $optargs = [3]$ for either

Returns an integer $status$ that is the integer returned by `mpq_QSget_status`() function.

3.1.4 FlushQSLP

▷ `FlushQSLP(s)` (function)

Returns:

This function terminates the external processes associated with given LP handle. It accepts following arguments:

- s - gap object of category IsInputOutputStream(), handle to an already loaded LP

Returns Nothing

3.1.5 GetQSLPsol_primal

▷ `GetQSLPsol_primal(s)` (function)

Returns: A list

This function obtains the primal solution along with the associated vertex vertex, for the most recently solved LP. It accepts following arguments:

- s - gap object of category IsInputOutputStream(), handle to an already loaded LP

Returns A list $[status, val\_rval, val, x\_rval, x]$ if optimal solution exists and a list $[status]$ otherwise. If $status = 1$, val_rval and x_rval indicate validity of val and x (valid if 1 and invalid if -1) which are optimal solution and (primal) vertex achieving optimal solution respectively. Other status values correspond to the integer returned by `mpq_QSget_status`() function.

3.1.6 GetQSLPsol_dual

▷ `GetQSLPsol_dual(s)` (function)

Returns: A list

This function obtains the primal solution along with the associated vertex vertex, for the most recently solved LP. It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP

Returns A list $[status, val\_rval, val, y\_rval, y]$ if optimal solution exists and a list $[status]$ otherwise. If $status = 1$, val_rval and x_rval indicate validity of val and x (valid if 1 and invalid if -1) which are optimal solution and (dual) vertex achieving optimal solution respectively. Other status values correspond to the integer returned by `mpq_QSget_status()` function.

3.1.7 ChangeQSrhs

▷ `ChangeQSrhs(s, row, coef)` (function)

Returns: An integer

This function changes the value of single rhs coefficient in specified row. It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP
- *row* - row index of the inequility whose rhs is to be changed
- *coef* - new rhs coefficient

Returns A an integer which is itself returned by QSopt_ex function `mpq_QSchange_rhscoef`

3.1.8 DelQSrow

▷ `DelQSrow(s, row)` (function)

Returns: An integer

This function deletes the specified row. (Note that for repeated use, one must relabel rows as QSopt_ex would treat eg. the second row as first row if we delete the first row) It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP
- *row* - row index of the inequility whose rhs is to be changed

Returns A an integer which is itself returned by QSopt_ex function `mpq_QSchange_rhscoef`

3.1.9 ChangeQSsense

▷ `ChangeQSsense(s, row, coef)` (function)

Returns: An integer

This function changes the sense (equality or inequality) of a particular row. It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP
- *row* - row index of the inequility whose sense is to be changed
- *newsense* - A single character string describing the new sense, "L" for $\leq$ and "E" for $=$

Returns An integer which is itself returned by QSopt_ex function `mpq_QSchange_sense`

3.1.10 ChangeQScoef

▷ ChangeQScoef(*s, row, coef*) (function)

Returns: An integer

This function changes a particular coefficient in the constraint matrix. It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP
- *row* - row index of the inequility to which the coefficient to be changed belongs
- *col* - column index of the inequility whose sense is to be changed
- *coef* - A rational number or an integer

Returns A an integer which is itself returned by QSopt_ex function `mpq_QSchange_sense`

3.1.11 DisplayLPQS

▷ DisplayLPQS(*s*) (function)

Returns: Nothing

This function displays an already loaded LP. It accepts following arguments:

- *s* - gap object of category IsInputOutputStream(), handle to an already loaded LP

Returns Nothing

3.2 Example

Following example explains the standard workflow with qsopt `qsopt_ex-interface`. We show how to load, solve, display and modify a linear program.

Example

```
gap> #  absolute path to the interface executable
> qs_exec:="/home/aspitrg3-users/jayant/qsopt_interface/dummy";;
gap> # Construt a 3-D cube
> A:=[[1,0,0],[0,1,0],[0,0,1],[-1,0,0],[0,-1,0],[0,0,-1]];;
gap> b:=[1,1,1,0,0,0];;
gap> rlist:=LoadQSLP([1,1,1],A,b,[],qs_exec);;
gap> rlist[1]; # stdin/stdout handle to the loaded LP
< input/output stream to dummy >
gap> s:=rlist[1];;
gap> DisplayLPQS(s);
Problem
 prob
Maximize
 obj:   c0 +  c1 +  c2
Subject To
 r0:   c0 <= 1
 r1:   c1 <= 1
 r2:   c2 <= 1
 r3:  -  c0 <= 0
 r4:  -  c1 <= 0
```

```
 r5:  -  c2 <= 0
Bounds
 c0 free
 c1 free
 c2 free
End
gap> SolveQSLP(s,[]); # returns status, 1 for success
1
gap> rlist:=GetQSLPsol_primal(s);; # get primal solution
gap> rlist[1]; # return status
1
gap> rlist[2]; # val_rval, 0 means sane
0
gap> rlist[3]; # val, LP solution
3
gap> rlist[4]; # x_rval, 0 means sane
0
gap> rlist[5]; # x, optimum vertex
[ 1, 1, 1 ]
gap> rlist:=GetQSLPsol_dual(s);;  #  get dual solution
gap> rlist[1]; # status
1
gap> rlist[2]; # val_rval
0
gap> rlist[3]; # val
3
gap> rlist[4]; # y_rval
0
gap> rlist[5]; # y
[ 1, 1, 1, 0, 0, 0 ]
gap> LoadQSLPobj(s,[-1,-1,-1]); # to minimize, negate the objective
1
gap> SolveQSLP(s,[]); # returns status, 1 for success
1
gap> rlist:=GetQSLPsol_primal(s);  #  get primal solution
[ 1, 0, 0, 0, [ 0, 0, 0 ] ]
gap> ChangeQSsense(s,1,"E"); # tighten first inequality (r0)
0
gap> DisplayLPQS(s);
Problem
prob
Maximize
 obj:  -  c0 -  c1 -  c2
Subject To
 r0:   c0 = 1
 r1:   c1 <= 1
 r2:   c2 <= 1
 r3:  -  c0 <= 0
 r4:  -  c1 <= 0
 r5:  -  c2 <= 0
Bounds
 c0 free
 c1 free
```

```
 c2 free
End
gap> ChangeQSrhs(s,1,3/2); # change first row r0's rhs to 3/2
0
gap> DisplayLPQS(s);
Problem
prob
Maximize
obj:  -  c0 -  c1 -  c2
Subject To
 r0:   c0 = 3/2
 r1:   c1 <= 1
 r2:   c2 <= 1
 r3:  -  c0 <= 0
 r4:  -  c1 <= 0
 r5:  -  c2 <= 0
Bounds
 c0 free
 c1 free
 c2 free
End
gap> SolveQSLP(s,[]); # returns status, 1 for success
1
gap> rlist:=GetQSLPsol_primal(s);  #  get primal solution
[ 1, 0, -3/2, 0, [ 3/2, 0, 0 ] ]
gap> DelQSrow(s,1); # delete the first row
0
gap> DisplayLPQS(s);
Problem
prob
Maximize
 obj:  -  c0 -  c1 -  c2
Subject To
 r1:   c1 <= 1
 r2:   c2 <= 1
 r3:  -  c0 <= 0
 r4:  -  c1 <= 0
 r5:  -  c2 <= 0
Bounds
 c0 free
 c1 free
 c2 free
End
```

References

[ACDE09] David Applegate, William Cook, Sanjeeb Dash, and Daniel Espinoza. QSopt-ex 2.6 — A computer algebra system for polynomial computations, 2009. 3

[GtGdt15] Torbörn Granlund and the GMP development team. GNU MP: The GNU Multiple Precision Arithmetic Library 6.0.0, 2015. 4

[Ste15] Jon Lund Steffensen. QSopt-ex 2.5.10 patch 3 - a fork adding improvements to the build system, library and a python interface, 2015. 3

Index