NAME

lp_solve − a mixed integer linear programming (MILP) solver

SYNOPSIS

lp_solve [*options*] [*input file*]

DESCRIPTION

lp_solve solves mixed integer linear programming problems. The program accepts models of problems in various different formats (including custom formats through XLIs) and attempts to solve them. There are a large number of different options which can be used to alter the solving process to improve performance or get more accurate results.

lp file format

The lp file format is a simple, human readable file format for linear programming models. It is the default format used by **lp_solve**.

The format consists of series of statements, each ending in a semicolon. The first statement is always the objective function, followed by any number of constraints, followed by any number of declarations.

The objective function consists of an expression to be optimized, optionally prefixed by **max:** or **min:** to specify which direction to optimize. The default is to maximize.

Each constraint consists of a relational expression, optionally prefixed by a name followed by a colon. The expression must contain at least one relational operator (<, <=, =, =>, >). You can specify ranges two operators at once (e.g. 2 < x < 4) instead of two separate constraints.

Declarations can be one of:

int *var1* [**,** *var2*]... ;
: Marks the variables as integers.

bin *var1* [**,** *var2*]... ;
: Marks the variables as binary (can only take 0 or 1).

sec *var1* [**,** *var2*]... ;
: Marks the variables as semi-continuous. These variables can always take the value 0 even if it is outside the variable’s range.

free *var1* [**,** *var2*]... ;
: Marks the variables as free. These variables have no lower bound, unlike other variables which have a default lower bound of 0.

sos [*name***:**] *var1* [**,** *var2*]... [< *sostype*];
: Marks the variables as part of a special ordered set (SOS). Out of the variables listed, only one can be non-zero at once. If *sostype* is specified, then up to that number of variables can be non-zero as long as they are all adjacent to each other (e.g. for an *sostype* of 2, *var1* and *var2* could be non-zero, but not *var1* and *var3*).

Both C style comments (**/*** comment here ***/**) and C++ style comments (// comment until end of line) can be used anywhere in the file.

OPTIONS

General options

−wafter
: Writes the model after solving it (normally any model conversions are processed before solving). This is useful if presolving is used to simplify the model.

−parse_only
: Parse the input model, but stop before solving it. This can be used to convert models between formats without solving them.

−timeout *sec*
: Timeout if no solution has be found after *sec* seconds.

−nonames
: Ignore variable and constraint names in the input model (this is the same as combining both **−norownames −nocolnames**).

−norownames
: Ignore constraint names in the input model.

−nocolnames
: Ignore variable names in the input model.

−min Minimizes the LP problem, overriding the setting in the input model.

−max Maximizes the LP problem, overriding the setting in the input model.

Built-in model languages

−lp Read the model from a file in LP format. This the the default.

−mps Read the model from a file in ’fixed width’ MPS format.

−fmps Read the model from a file in ’free’ MPS format. This is the same as **−mps** except that fields are separated by a number of blanks instead of starting at fixed columns.

−wlp *filename*
: Converts the input model to LP format and writes it to the a file.

−wmps *filename*
: Converts the input model to ’fixed width’ MPS format and writes it to a file.

−wfmps *filename*
: Converts the input model to ’free’ MPS format and writes it to a file.

External language interface (XLI)

−rxli *xliname filename*
: Read the model using an external language interface (XLI). *xliname* specifies the path to the shared library containing the XLI, and is searched for according to the normal library search rules in **ld.so**(8) with the addition that the library filename is prefixed with *lib* and suffixed with *.so* if necessary.

−rxlidata *datafilename*
: Provides a file containing extra data used by the reading XLI library.

−rxliopt *options*
: Extra options to pass to the reading XLI library.

−wxli *xliname filename*
: Converts the input model to the format used by an XLI and writes it to a file. The format of *xliname* is described in **−rxli** above.

−wxliopt *options*
: Extra options to pass to the writing XLI library.

−wxlisol *xliname filename*
: Writes the model’s solution to a file using an XLI library. The format of *xliname* is described in **−rxli** above.

−wxlisolopt *options*
: Extra options to pass to the solution writing XLI library.

Printing, verbosity and debugging

−h Print a usage message and exit.

−S*level* Solution detail. Each *level* builds on top of the previous one and adds more detail to the solution.

0	print nothing
1	objective value only
2	variables (default level)
3	constraints
4	duals
5	lp model
6	scales
7	lp tableau

−time Print CPU time to parse input and to calculate solution.

−v[*level*]
Verbosity level. Controls the level of messages printed about the operation of the program.

0	neutral
1	critical
2	severe
3	important (default when **−v** is provided with no *level*)
4	normal (default when no **−v** option is given)
5	detailed
6	full

−t Trace pivot selection.

−d Debug mode. All intermediate results and branch-and-bound decisions are printed.

−R Report information while solving the model.

−Db *filename*
Create a dump of internal model variables before solving the model to given file.

−Da *filename*
Create a dump of internal model variables after solving the model, to given file.

−i[**a**] Print all intermediate valid solutions. If **a** is used, prints only non-zero values. Can give useful solutions even if the total run time is too long.

Parameter files

−rpar *filename*
Reads a list of parameters from a file. The parameters file can specify the same options as you can by passing arguments to **lp_solve** but can be more convenient if you need to change lots of settings. The format of the parameters file is in the INI format often used by Windows applications. All parameters are read from key-value pairs in a single section. The default section is **[Default]**.

−rparopt '**−h** *header*'
Sets options for reading the parameters file. Currently the only option is **−h** which changes the section within the file to read the parameters from.

−wpar *filename*
Writes a parameters file based on the arguments given to **lp_solve**.

−wparopt '**−h** *header*'
Sets options for writing the parameter file. Currently the only option is **−h** which changes the section within the file to write the parameters to.

Basis files

−rbas *filename*
: Reads a basis file which is used as the starting point for solving the LP model. Basis files must be in MPS basis format.

−gbas *filename*
: Computes a basis from a list of initial guesses for each variable. Each line of the given file is in the format: *variable***:***value* where *value* is the initial guess for *variable*.

−wbas *filename*
: After solving the LP model, writes a basis file in MPS format. This file can be read back using **−rbas**.

Available options:

0	no crash basis (default).
1	most feasible basis.
2	least degenerate basis.

Integer and branch and bound options

−noint Ignore integer restrictions on variables.

−f Stops the branch and bound algorithm immediately after finding its first solution.

−o *bound*
: Stops the branch and bound algorithm immediately after finding a solution whose objective function has a value greater than this bound.

−b *bound*
: Sets a lower bound for the objective function. The branch and bound algorithm will immediately reject all solutions lower than this value.

−depth *limit*
: Sets the depth limit for the branch and bound algorithm. A value of 0 disables depth limiting. A negative value sets a relative depth limit based on the number of variables in the input model. The default limit is −50.

−e *tolerance*
: Sets the tolerance which is used to determine whether a floating point number is an integer. A number has to be within this value of an integer to be considered one. The default value is 1e-7.

−g *tolerance*, **−ga** *tolerance*
: Sets the absolute MIP gap used by the branch and bound algorithm. This tolerance is the difference between the best-found solution yet and the current solution. If the difference is smaller than this tolerance then the solution (and all the sub-solutions) is rejected. This can result in faster solving times, but results in a solution which is not the perfect solution. The default value is 1e-11.

−gr *tolerance*
: Sets the relative MIP gap used by the branch and bound algorithm. This is similar to the absolute tolerance except that the difference is scaled by the best found solution before being compared to this tolerance. The default value is 1e-11.

−c, **−cc** During branch and bound, take the ceiling branch first.

−cf During branch and bound, take the floor branch first.

−ca During branch and bound, allow the algorithm to decide which branch to take. This is the default.

−n *solution*
: If the branch and bound algorithm produces multiple solutions with the same objective function value, prints the nth solution. Solution numbers start at 1.

−B*rule* Sets a branch and bound rule to use. The default rules are: **−B5 −Bg −Bd −Bc**

These rules are mutually exclusive:

0	select lowest indexed non-integer column.
1	selection based on distance from the current bounds.
2	selection based on the largest current bound.
3	selection based on largest fractional value.
4	simple, unweighted pseudo-cost of a variable.
5	extended pseudo-costing strategy based on minimizing the number of integer infeasibilities.
6	extended pseudo-costing strategy based on maximizing the normal pseudo-cost divided by the number of infeasibilities.

The above rules can be combined with any of the following:

w	WeightReverse - select by criterion minimum (worst), rather than criterion maximum (best).
b	BranchReverse - when **−ca** is selected, choose the opposite direction.
g	Greedy
p	PseudoCost - toggles between weighting based on pseudocost or objective function value.
f	DepthFirst - select the node that has already been selected before the most number of times.
r	Randomize - ddds a randomization factor to the score for any node candidate.
G	GubMode
d	Dynamic - when **−Bf** is selected, switch off this mode when a first solution is found.
s	RestartMode - enables regular restarts of pseudocost value calculations.
B	BreadthFirst - select the node that has been selected before the fewest number of times or not at all.
o	AutoOrder - order variables to improve branch-and-bound performance.
c	ReducedCostFixing - do bound tightening during B&B based of reduced cost info.
i	StringInit - initialize pseudo-costs by strong branching.

Simplex algorithm options

−prim, **−simplexpp**
Prefer the primal simplex method for both phase 1 and phase 2.

−dual, **−simplexdd**
Prefer the dual simplex method for both phase 1 and phase 2.

−simplexdp
Prefer the dual simplex method for phase 1 and primal method for phase 2. This is the default.

−simplexpd
Prefer the primal simplex method for phase 1 and dual method for phase 2.

−bfp *filename*
Sets the basis factorization package to use. *filename* refers to a shared library loaded in the same way **−rxli** loads libraries.

−o*value*
Sets whether the objective function is stored in the top row of the constraint matrix or in separate storage. **−o0** places the matrix in separate storage, and **−o1** (the default) places it in the constraint matrix.

−C*mode*
Sets the basis crash mode. When base crash is enabled, a heuristic ’crash procedure’ is executed before the first simplex iteration to quickly choose a basis matrix that has fewer artificial variables. This procedure tends to reduce the number of iterations to optimality since a number of iterations are skipped.

−r *value*
Sets the maximum number of pivots between a re-inversion of the matrix. For stability reasons, lp_solve re-inverts the matrix on regular times. The default is 250 for the LUSOL bfp and 42 for the other BFPs.

−trej *value*
Sets the value that is used as a tolerance pivot element to determine whether a value should be considered as 0. The default is 2e-7.

−epsd *value*
Sets the value that is used as a tolerance for reduced costs to determine whether a value should be considered as 0. The default is 1e-9.

−epsb *value*
Sets the value that is used as a tolerance for the Right Hand Side (RHS) to determine whether a value should be considered as 0. The default is 1e-10.

−epsel *value*
Sets the value that is used as a tolerance for rounding values to zero. The default is 1e-12.

−epsp *value*
Sets the value that is used as perturbation scalar for degenerative problems. The default is 1e-5.

−improve*level*
Sets the iterative improvement level. *level* is a number which can be created by combining the different levels below. The default is **−improve6**.

0	none.
1	running accuracy measurement of solved equations on Bx=r.
2	improve initial dual feasibility by bound flips.
4	low-cost accuracy monitoring in the dual.
8	check for primal/dual feasibility at the node level.

−piv*rule*
Sets a simplex pivot rule or mode to use. The default rules are: **−piv2 −piva**

These rules are mutually exclusive:

0	select first.
1	select according to Dantzig.
2	select Devex pricing from Paula Harris.
3	select steepest edge.

The above rules can be combined with any of the following modes:

f	PrimalFallback - when steepest edge (**−priv3**) is selected, fallback to Devex in primal.
m	Multiple - preliminary implementation of the multiple pricing scheme. This means that attractive candidate entering columns from one iteration may be used in the subsequent iteration, avoiding full updating of reduced costs. In the current implementation, lp_solve only reuses the 2nd best entering column alternative
a	Adaptive - temporarily use alternative strategy if cycling is detected.
r	Randomize - adds a small randomization effect to the selected pricer.
ll	LoopLeft - scan entering/leaving columns left rather than right.
la	LoopAlternate - scan entering/leaving columns alternating left/right.
h	HarrisTwoPass - use Harris' primal pivot logic rather than the default.
t	TrueNormInit - use true norms for Devex and Steepest Edge initializations.

−degen*option*
Specifies if special handling must be done to reduce degeneracy/cycling while solving. The default options are: **−degenf −degens**.

Available options:

c	ColumnCheck
d	Dynamic
f	FixedVars

s	Stalling
n	NumFailure
l	Lostfeas
i	Infeasible
b	DuringBB
r	RHSPerturb
p	BoundFlip

Other solving options

−presolve[*option*]

Enables a presolve option. These options attempt to speed up calculations by simplifying the model before solving it. Any of these options can be combined together. If *option* is not specified, row and column presolving is enabled (as if **−presolverow −presolvecol** had been used). The default not to presolve.

Available options:

row	Rows - presolve rows.
col	Cols - presolve columns.
l	LinDep - eliminate linearly dependent rows.
s	Sos - convert constraints to SOSes (only SOS type 1 handled).
r	ReduceMip - if the phase 1 solution process finds that a constraint is redundant then this constraint is deleted.
k	Knapsack - simplification of knapsack-type constraints through addition of an extra variable, which also helps bound the objective function.
q	ElimEQ2 - direct substitution of one variable in 2-element equality constraints; this requires changes to the constraint matrix.
m	MergeRows - merges neighboring >= or <= constraints when the vectors are otherwise relatively identical into a single ranged constraint.
fd	ColFixDual - variable fixing and removal based on considering signs of the associated dual constraint.
bnd	Bounds - does bound tightening based on full-row constraint information.
d	Duals - presolve duals.
f	ImpliedFree - identify implied free variables (releasing their explicit bounds).
slk	ImpliedSlk- converts qualifying equalities to inequalities by converting a column singleton variable to slack.
g	ReduceGCD - reduce (tighten) coefficients in integer models based on GCD argument.
b	ProbeFix - attempt to fix binary variables at one of their bounds.
c	ProbeReduce - attempt to reduce coefficients in binary models.
rowd	RowDominate - identify and delete qualifying constraints that are dominated by others, also fixes variables at a bound.
cold	ColDominate - deletes variables (mainly binary), that are dominated by others (only one can be non-zero).

−s[*mode*] [*scalelimit*]

Sets the scaling algorithm and/or scaling limit to use. *scalelimit* can only be specified when using the mutually exclusive integer modes. Omitting *mode* is the equivalent of using **−s4**. The default algorithm and limit is: **−s1 5 −si −se**

These rules are mutually exclusive:

0	no scaling.
1	geometric scaling.
2	Curtis-Reid scaling.
3	scale to convergence using largest absolute value.
4	numerical range-based scaling.

5	scale to convergence using logarithmic mean of all values.
6	scale based on the simple numerical range.
7	scale quadratic.

The above rules can be combined with any of the following:

p	also do power scaling.
i	also scale integer variables.
e	ensure no scaled number is outside the range -1..1.

EXAMPLE

Model solving

The following example shows a model being solved by **lp_solve**. The first line contains the objective function to be maximized and the last 3 lines contain the constraints.

```
$ cat example.lp
max: 143 x + 60 y;

120 x + 210 y <= 15000;
110 x + 30 y <= 4000;
x + y <= 75;

$ lp_solve -S3 example.lp
Value of objective function: 6315.63

Actual values of the variables:
x                          21.875
y                          53.125

Actual values of the constraints:
R1                         13781.2
R2                            4000
R3                              75
```

Format conversion

This example converts a file in lp format to mps format.

```
$ lp_solve -parse_only -lp example.lp -wmps example.mps
```

SEE ALSO

ld.so(8)