First-order MIP solving with SCIP and Mercury

James Cussens, University of York

StarAI@UAI15, 2015-07-16
Using a logic program to create a MIP instance

:- interface.
:- import_module mfoilp.

:- type atom.

:- pred variable(atom::out) is multi.
:- pred initial_constraint(lincons::out) is nondet.
:- pred delayed_constraint(lincons::out) is nondet.
:- func objective(atom) = float.
:- func lb(atom) = float.
:- func ub(atom) = float.
:- func vartype(atom) = vartype.

:- implementation

% problem definition here ...
delayed_constraint(lincons(finite(1.0),LinExpr,posinf)) :-
    cluster(Cluster), length(Cluster) > 1,
    Closure = (    
      pred(1.0 * fv(Child,Parents)::out) is nondet :-
        member(Child,Cluster), score(Child,Parents,_),
        not (member(X,Parents),member(X,Cluster))),
    solutions(Closure,LinExpr).
Naive cutting plane generation

:- pred consfail(sol::in,lincons::out) is nondet.

consfail(Sol,Cons) :-
    prob.delayed_constraint(Cons),
    Cons = lincons(Lb,LExp,Ub),
    activity(LExp,Sol,0.0,ConsVal),
    ((Ub=finite(Ubf),ConsVal > Ubf)
    ; (Lb=finite(Lbf),ConsVal < Lbf)).
Solving easy problems

> make solution ...
Making Mercury/cs/prob.c ...
Making Mercury/os/prob.o ...
-> linking mfoilp.linux.x86_64.gnu.opt.spx
presolved problem has 40 variables ... and 8 constraints
  1 constraints of type <folinear>
  7 constraints of type <setppc> ...

frac | cuts  | dualbound |
--- | --- | --- |
 0  |    0  |  2.041602e+02 |
 0  |    1  |  2.042993e+02 |
 9  |   23  |  2.269866e+02 |
Solving easy problems (ctd)

... 
frac | cuts | dualbound |
* 0 | 46 | 2.456443e+02 | ...

SCIP Status : problem is solved
objective value: 245.644264
fv(0, []) 1 (obj:71.123881)
fv(1, [0, 4]) 1 (obj:21.675646)
fv(2, [0]) 1 (obj:66.320026)
fv(3, [4]) 1 (obj:12.214946)
fv(4, []) 1 (obj:2.8762)
fv(5, [1, 4]) 1 (obj:2.247729)
fv(6, [1]) 1 (obj:16.935375)
fv(7, [2]) 1 (obj:52.250461)
Towards less naive cutting plan generation

:- pred consfail(sol::in,lincons::out) is nondet.

- Can simply demand that the user writes a custom definition of consfail/2.
- Or unfold definition of consfail/2 and reorder goals before compilation.
Next jobs

- Write code for a (naive) pricer.
- Look at symmetry detection.

):- pred price(dual_lpsol::in, variable::out) is nondet.

price(DualLPSol,Var) :-
    variable(Var),
    reduced_cost(DualLPSol,Var,RedCost),
    RedCost < 0.