Mathematical Programming: Modelling and Applications

Sonia Cafieri

LIX, École Polytechnique

cafieri@lix.polytechnique.fr

November 2009

AMPL script - RandomWalk algorithm

```
param m, integer; # number of clauses
param n, integer; # number of variables (2n is the number of literals)
set M := 1..m;
set N := 1..n;
# phi[i,j] = 1  if x_j appears in clause i
#           = -1 if bar{x}_j appears in clause i
#           = 0  otherwise
param phi{M,N} integer, default 0;

param k >=0 integer;  # number of clauses you aim to satisfy
param x{N} binary;
param y{M} binary, default 0;

data max2sat_m9_n3.dat;

param fstar integer, >= 0, default 0;
param xstar{N} binary, default 0;
param ii integer;
param jj integer;
param l integer, default 1;
param ll integer, default 1;
param satisf integer, default 0;
param changedv{M,N} binary, default 0;
param flag binary, default 0;
param termination binary, default 0;
param iter integer, default 0;
param maxiter integer, default 100;

let k := m-1;
```

AMPL script - RandomWalk algorithm

```
# start with a random generated assignments
for {i in 1..n} {
   if (Uniform01() <= 0.5 ) then {
      let x[i]:= 0;
   } else {
      let x[i] := 1;
   }
}

repeat while (termination = 0) {

  # check if the current assignment satisfies the clauses
  for {i in 1..m} {
     let y[i] := 0;
     let flag := 0;
     let jj := 1;
     repeat while (jj <= n and flag = 0) {
         if (phi[i,jj] <> 0) then {
            if ((phi[i,jj] = 1 and x[jj] = 1) or
                (phi[i,jj] = -1 and x[jj] = 0) ) then {
               let y[i] := 1;
               let flag := 1;
            }
         }
         let jj := jj+1;
     }
  }
```

AMPL script - RandomWalk algorithm

```
# count the number of satisfied clauses
let satisf := 0;
for {i in 1..m} {
   if (y[i] = 1) then {
      let satisf := satisf + 1;
   }
}

if (satisf < k ) then {

     # choose the unsatisfied clause to be changed
     let ii := 1;
     let flag := 0;
     repeat while (ii <= m  and flag = 0) {
       if (y[ii] = 0) then {
            let l := ii;
            let flag := 1;
       }
       let ii := ii+1;
     }

     # change the clause l
     let flag := 0;
     let jj := 1;
     repeat while (jj <= n  and flag = 0) {
       if (phi[l,jj] <> 0 and changedv[l,jj] = 0) then {
           let ll := jj;
           let flag := 1;
           let changedv[l,jj] := 1;
       }
       let jj := jj+1;
     }
```

AMPL script - RandomWalk algorithm

```
       # change the variable ll
       if(x[ll] = 0) then {
           let x[ll]:= 1;
       } else {
           let x[ll]:= 0;
       }


  } else {
      print "problem solved";
      let termination := 1;
  }

  let iter := iter +1;
  if(iter > maxiter) then {
    print "iteration limit reached";
    let termination := 1;
  }
} #endwhile


 for {i in 1..n} {
    let xstar[i] := x[i];
 }
 printf "iterations = %d\n", iter-1;
 printf "satisfied clauses = %d\n", satisf;
 display xstar;
```

AMPL script - RandomWalk algorithm

```
C:\01_Fabio\SR2PI\2018\INF580-2018\Cafieri>ampl < maxsat.run
problemsolved
iterations= 6
satisfied clauses =8
xstar [*] :=
1  1
2  1
3  0
;
```