Za iskanje kromatičnega števila grafa sva napisala celoštevilski linearni program.

Za spremenljivke sva vzela x\_i,j , ki zavzamejo vrednosti 0 ali 1 in sicer 1, če je vozlišče i pobarvano z barvo j in 0 sicer. Tako za vse i in j od 1 do n, ker imamo n vozlišč in najslabše možno barvanje grafa je tako, da ima vsako vozlišče svojo barvo. Poleg tega vzamemo za spremenljivke še y\_j za j od 1 do n, ki nam pove, če smo uporabili j-to barvo.

Pri ILP minimiziramo \sum\_(j=1)^n y\_j

Nato postavimo omejitve:

\sum\_(j=1)^n x\_i,j = 1 , za vsako vozlišče i od 1 do n, saj ima vozlišče i lahko samo 1 barvo

Za vsako povezavo (i,j) mora veljati:

x\_i,k + x\_j,k <= 1 , za vsako barvo k od 1 do n, saj sosednja vozlišče ne smeta biti oba iste barve

Za y-e pa postavimo omejitve:

x\_i,k – y\_k <= 0 , za vsak i od 1 do n, saj če smo na kateremkoli vozlišču uporabili barvo k mora biti y\_k enak 1.

Časovna zahtevnost algoritma:

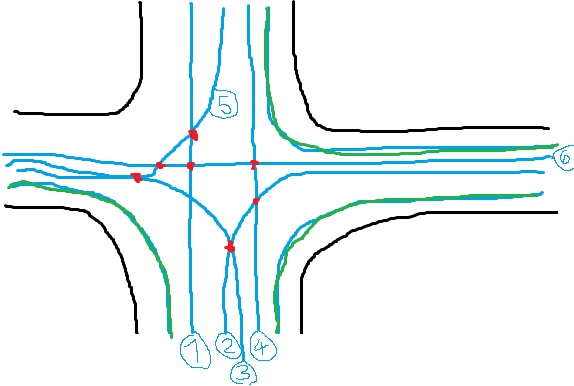
Ker so y spremenljivke odvisne od x, bi časovno zahtevnost ocenil na podlagi x spremenljivk.

Denimo, da imamo graf z n vozlišči. Ko skonstruiramo spremenljivke x\_i,j tako dobimo n\*n spremenljivk in ker so spremenljivke enake 1 ali 0 imamo za vsako 2 možnosti, torej je vseh možnih kombinacij x-ov 2^(n^2), če pa bi gledali vse možne kombinacije bi morali tudi za vsako preveriti O(n) omejitev. Torej če bi recimo sestavili celoštevilski program, ki preveri vse možnosti bi bila časovna zahtevnost O(n\*2^(n^2)), kar pomeni da bi se pri takem algoritmu izogibali grafov z velikim številom vozlišč. Sicer pa je časovna zahtevnost seveda odvisna od sestave celoštevilskega linearnega programa in je dosti manjša, vendar še vedno raste eksponentno.

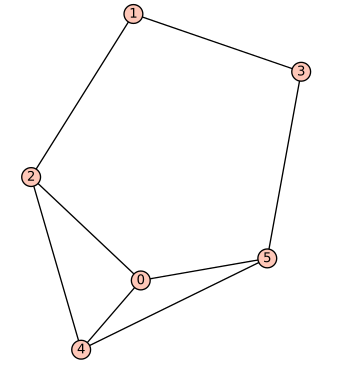
Uporaba algoritmov za preverjanje kromatičnega števila grafa v praksi:

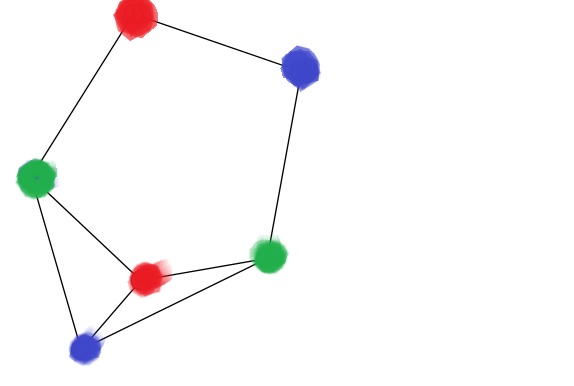
V resničnem življenju lahko veliko nalog prevedemo na iskanje kromatičnega števila grafa, kar nam pomaga, da lahko nalogo rešimo pravilno in optimalno z algoritmom.

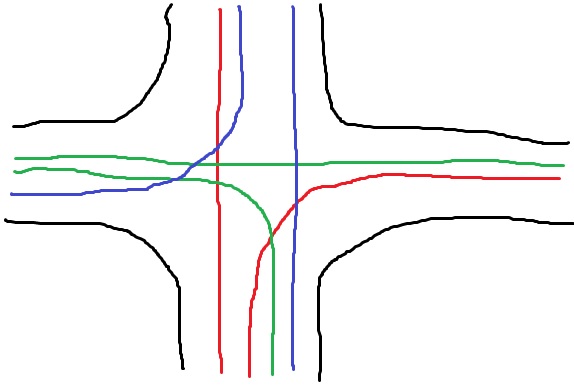
Primeri so:

* Denimo, da imamo več radijskih postaj, ki se med sabo sporazumevajo. Postaje imamo razporejene po zemljevidu in če sta 2 radijski postaji dovolj blizu ena drugi, lahko ena drugo moti. Torej za vozlišča uporabimo radijske postaje, povezave pa naredimo med tistimi postajami, ki so dovolj blizu ena drugi. Zdaj iščemo vsaj koliko različnih radijskih frekvenc potrebujemo, da se naprave med sabo ne bodo motile. Različne radijske frekvence pa so v temu primeru barve.
* Malce bolj nenavaden primer, za katerega nebi prvo pomislili, da se lahko rešuje z iskanjem kromatičnega števila grafa pa je na cesti. Recimo da imamo križišče na cesti in gledamo koliko najmanj intervalov za semaforje potrebujemo, da bodo vsi avti prevozili križišče brez trčenja. Narišemo krivulje, po katerih lahko avti v prometu potujejo. Označimo krivulje kot vozlišča. Če se 2 krivulji sekata naredimo med njima povezavo. Tiste krivulje, ki nobenemu ne sekajo poti lahko ignoriramo.

Sliko zapišemo kot graf:

Kromatično število grafa je 3, tako da ga lahko pobarvamo s 3-mi barvami:



Kar nam da vedeti, da imamo lahko 3 različne intervale semaforjov, da vsi avti lahko prevozijo križišče. Še kot slika: