


Declaratively solving tricky Google Code Jam problems with Prolog-based ECLiPSe CLP system

Sergii Dymchenko
Independent Researcher
sdymchenko@progopedia.com

Mariia Mykhailova
Independent Researcher
michaylova@gmail.com

Introduction

code jam Google Code Jam¹ (GCJ) is one of the biggest programming competitions in the world. An individual GCJ round is usually 2-4 hours long and poses 3 or more problems. A solution is considered correct if it produces correct answers for all given test cases within a certain time limit (**4 minutes** for the "small" input and **8 minutes** for the "large" one). GCJ competitors can use **any** freely available programming **language** or system.

 ECLiPSe² [1] is an open-source **Prolog**-based system that integrates various **logic programming** extensions, in particular constraint logic programming (**CLP**).

We claim that **declarative** programming with ECLiPSe is **better** suited for solving certain common kinds of problems offered in GCJ than **imperative** programming.

[1] J. Schimpf and K. Shen. ECLiPSe - from LP to CLP. *Theory Pract. Log. Program.*, 12(1-2):127-156, Jan. 2012.

Problem: Star Wars

The essence of the problem is: you are given a set of **N** 4-tuples of integers (x_i, y_i, z_i, p_i) . Find the minimal possible **Y** for which exists a triplet (x, y, z) such that for each original tuple

$$|x_i - x| + |y_i - y| + |z_i - z| \leq p_i Y$$

This problem was one of the harder problems from round 2 of GCJ 2008, yet it can be almost trivially modeled and solved as a **linear programming** problem.

Direct translation of the problem statement to a model results in **non-linear** constraints, but these can be easily **converted to linear** constraints using the fact that

$$|X| \leq Max \equiv (X \leq Max) \wedge (-X \leq Max)$$

```
:- lib(eplex).
model(Xs, Ys, Zs, Ps, X, Y, Z, P) :-
    P $>= 0,
    ( foreach(Xi, Xs), foreach(Yi, Ys),
      foreach(Zi, Zs), foreach(Pi, Ps),
        param(X, Y, Z, P) do
          +(Xi - X) + (Yi - Y) + (Zi - Z) $=<= Pi * P,
          +(Xi - X) + (Yi - Y) - (Zi - Z) $=<= Pi * P,
          +(Xi - X) - (Yi - Y) + (Zi - Z) $=<= Pi * P,
          +(Xi - X) - (Yi - Y) - (Zi - Z) $=<= Pi * P,
          -(Xi - X) + (Yi - Y) + (Zi - Z) $=<= Pi * P,
          -(Xi - X) + (Yi - Y) - (Zi - Z) $=<= Pi * P,
          -(Xi - X) - (Yi - Y) + (Zi - Z) $=<= Pi * P,
          -(Xi - X) - (Yi - Y) - (Zi - Z) $=<= Pi * P ).
find(P) :-
    eplex_solver_setup(min(P)),
    eplex_solve(_),
    eplex_var_get(P, typed_solution, P),
    eplex_cleanup.
```

Linear programming solution for "Star Wars"

Running times for small (4 minutes time limit) and large (8 minutes) inputs

Problem	Library	Small	Large
Triangle Areas	ic	0.2s	1.4s
Dancing With the Googlers	ic	0.2s	timeout
Dancing With the Googlers	eplex	0.3s	1.7s
Star Wars	eplex	0.2s	0.4s
Mine Layer	eplex	0.2s	2.9s

Results were obtained on a 64-bit Linux machine with Intel Core i7-4900MQ CPU @2.80GHz using ECLiPSe 6.1 #191. For linear (integer) programming free COIN-OR solvers bundled with ECLiPSe were used.

Problem: Triangle Areas

Given integer **N**, **M** and **A**, find a triangle with vertices in integer points with coordinates $X_i :: [0..N]$, $Y_i :: [0..M]$, with an area $S = A/2$, or say that it does not exist.

The problem is almost perfect for solving with **CLP**. Variables are **discrete**, constraints are **non-linear**, and we are looking for any **feasible** solution. One vertex of the triangle can be chosen arbitrarily. To calculate the triangle area, place one vertex in $(0, 0)$, then

$$2S = |x_2 y_3 - x_3 y_2|$$

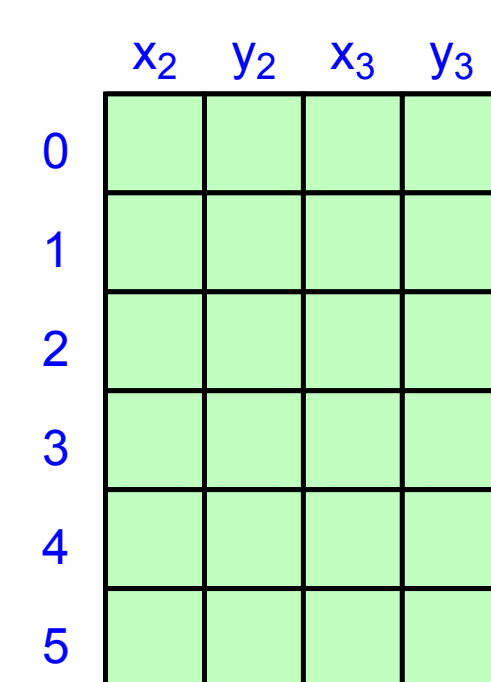
```
:- lib(ic).

model(N, M, A, [X2, Y2, X3, Y3]) :-
    [X2, X3] :: 0..N,
    [Y2, Y3] :: 0..M,
    A #= abs(X2 * Y3 - X3 * Y2).

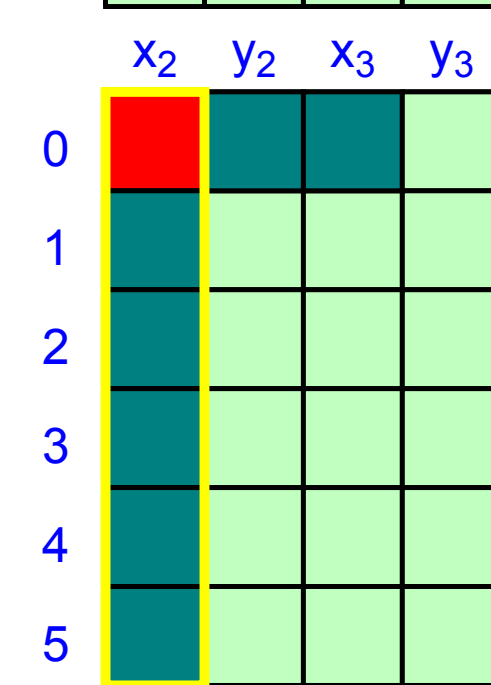
do_case(Case_num, N, M, A) :-
    printf("Case #%w: ", [Case_num]),
    ( model(N, M, A, Points), labeling(Points) ->
      printf("0 0 %w %w %w %w", Points)
    );
    write("IMPOSSIBLE")
),
nl.

main :-
    read([C]),
    ( for(Case_num, 1, C) do
      read([N, M, A]),
      do_case(Case_num, N, M, A) ).
```

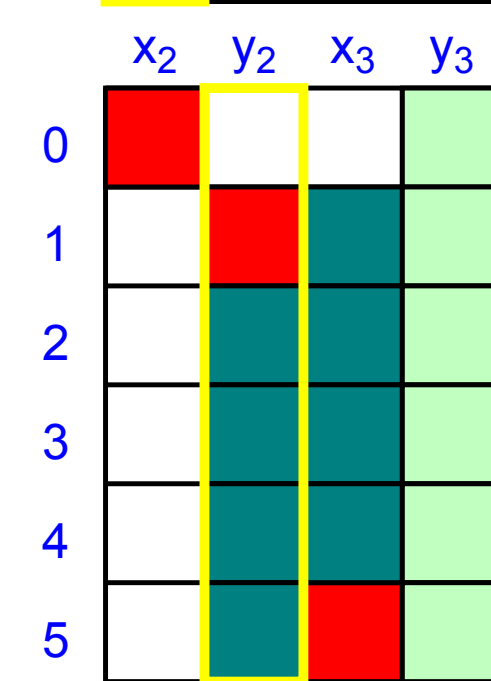
Complete ECLiPSe program for "Triangle Areas"



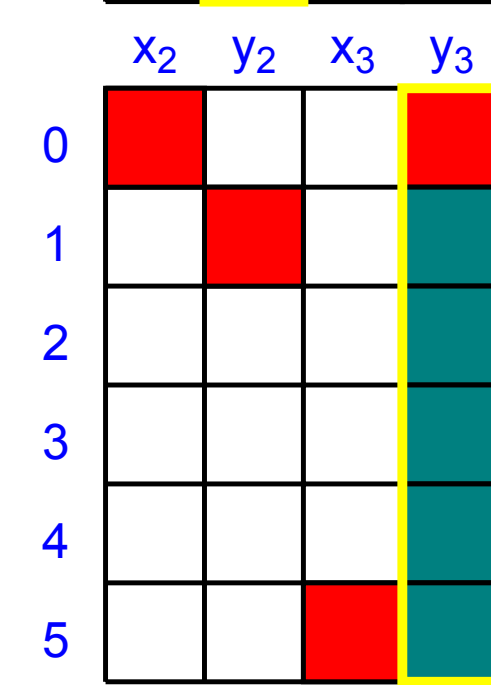
Initially all values from the initial domains are possible for all variables.



The default heuristic is "pick the first variable, pick the smallest value for it". So $x_2=0$ is picked. This rules out **all other values for x_2** , and this also rules out $y_2=0$ and $x_3=0$ — these values violate constraint $abs(x_3 \times y_2) = 5$.



Next value $y_2=1$ is picked. This rules out **all other values for y_2** , and from constraint $abs(x_3 \times 1) = 5$ follows that $x_3=5$.



The value of y_3 is not present in any constraints now (because $x_2=0$), any value can be chosen, the smallest value $y_3=0$ is picked.

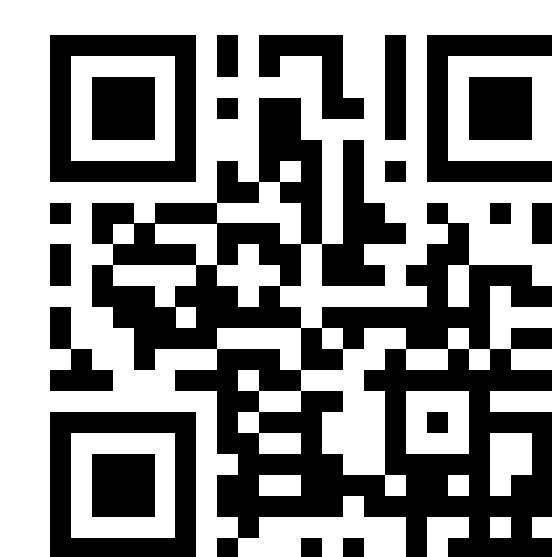
Visualization of constraint propagation and search for the case N = 5, M = 5, A = 5

Conclusions

Many GCJ problems that are hard to solve in time-restricted and stressful competition environment can be relatively easily modeled and solved in ECLiPSe.

Our **declarative** solutions for some problems require **simpler** and often **fewer** mental **steps** than possible imperative solutions in a language like C++ or Java.

Running times of our programs are several orders of magnitude smaller than the time limit imposed by GCJ rules.



<http://goo.gl/nYYnvs>

¹ <https://code.google.com/codejam>

² <http://www.eclipseclp.org/>