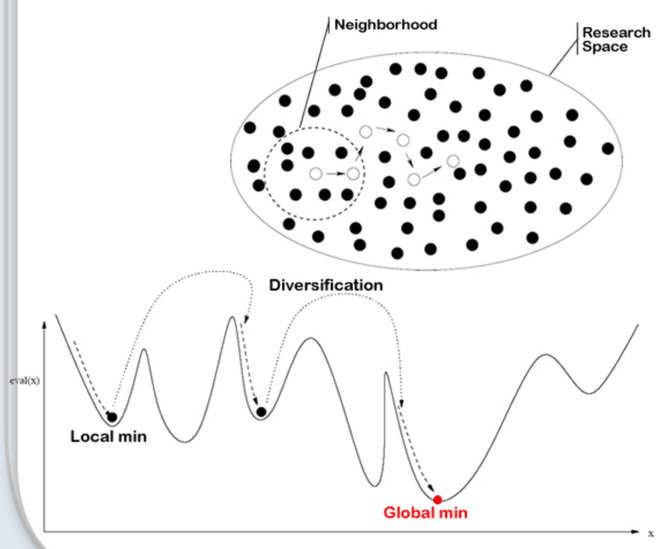
Towards Constraint-Based Local Search for Automatic Test Data Generation

Nadjib Lazaar*
Arnaud Gotlieb* Yahia Lebbah**
* IRISA-INRIA, Campus Beaulieu, Rennes, France
** Université d'Oran Es-Senia, Oran, Algeria

Local Search

- LS is a metaheuristic for solving computationally hard optimization problems.
- LS algorithm moves from solution to solution in the search space until a near optimal solution is found or a time bound is elapsed.



Constraint-Based Local Search

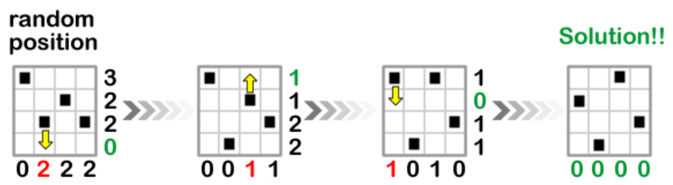
CBLS, as introduced by Michel and Van Hentenryck [MvH MIT Press 2005]], aims at introducing local search algo. within constraint-based models.

It consists of:

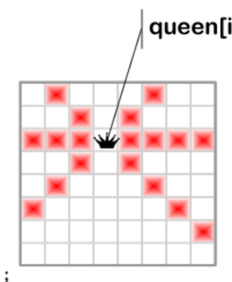
A modeling component offering several concepts like invariants and differentiable objects.

```
Abstract class Constraint {                    interface
bool      isTrue();                              for
name[]    getVariables();                  differenciable
int       violations() ;                      objects
int       violations(name x) ;
int       getAssignDelta(name x, int v);
int       getSwapDelta(name x, name y);
}
```

A search component aiming at simplifying the implementation of heuristics and meta-heuristics.



The Queens Problem in Comet




```
1.  range Size = 1..1024;
2.  LocalSolver ls();
3.  UniformDistribution distr(Size);
4.  var{int} queen[i in Size](ls,Size) := distr.get();

5.  ConstraintSystem S(ls);
6.  S.post(alldifferent(queen));
7.  S.post(alldifferent(all(i in Size) queen[i] - i));
8.  S.post(alldifferent(all(i in Size) queen[i] + i));
9.  ls.close();

10. Counter it(ls);
11. while (!S.isTrue()) {
12.   selectMax(q in Size)(S.violations(queen[q]))
13.     selectMin(v in Size)(S.getAssignDelta(queen[q],v))
14.       queen[q] := v;
15.   it++;
16. }
```

Our Approach

In our previous work, we built a Constraint -Based Testing environment that uses **static single assignment** as the constraint generation step and **domain reduction** as the constraint solving step to the structural testing for C program [Gotlieb et al. ISSTA'98].

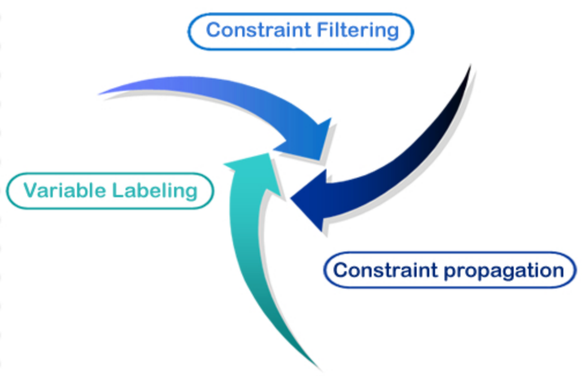
SSA Form

Each use of a variable refers to a single definition

$x := x + y;$ $\Rightarrow$ $x_1 := x_0 + y_0;$
$y := x - y;$ $\Rightarrow$ $y_1 := x_1 - y_0;$
$y := x - y;$ $\Rightarrow$ $y_2 := x_1 - y_1;$

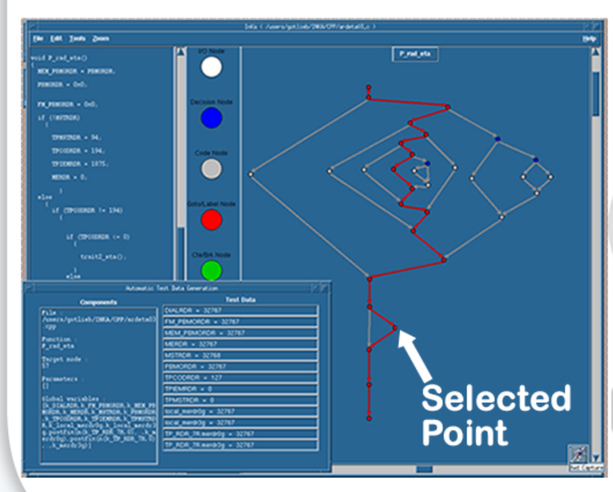
ϕ_function

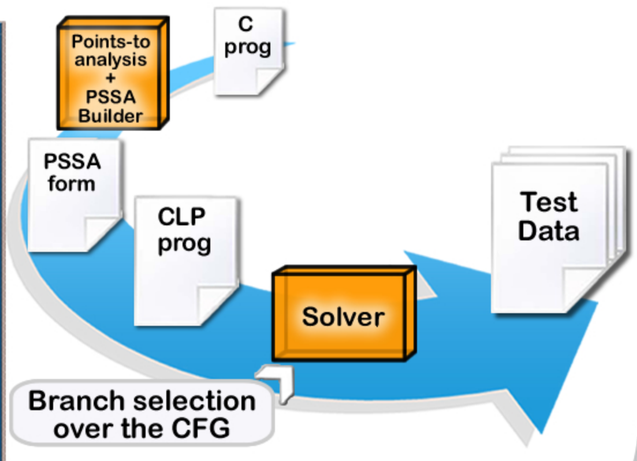
Constraint Solving



Euclide

- Implemented in Sicstus Prolog in the **Euclide** tool
- SSA For Structured Abstract Syntax trees: algo. From [Brandis & Mossenbock TOPLAS'94]
- Points-to analysis based on set-based data structures
- Architecture:





Current Work

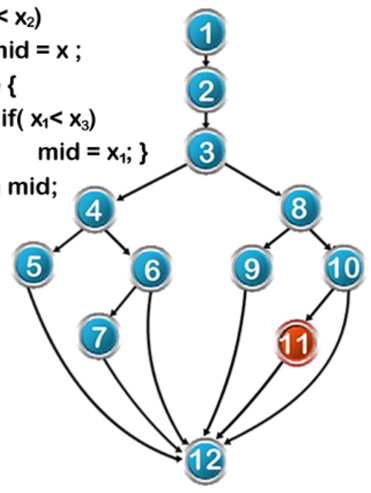
Our CBT framework implements arithmetic constraints, logical constraints and special combinators to modelize arrays, pointers and control structures like ITE(C, THEN, ELSE).
[Gotlieb et al. IST'06]

- For each of these constraints, the reflexive interface shown at side must be implemented.
- Thanks to this implementation, exploiting and experimenting various local search algorithms will become easier.

- Research works remain to decide how to implement the cost function that minimize the number of conflicts in the presence of complex control structure.

An Example

```
uch getmid(uch x1, uch x2, uch x3){
1.  ush mid;
2.  mid = x3;
3.  if(x2< x3)
4.    if(x1< x2)
5.      mid = x2;
6.    else{
          if(x1< x3)
7.          mid = x1; }
8.  else
      if(x1< x2)
9.        mid = x ;
10.   else {
          if( x1< x3)
11.         mid = x1; }
12. return mid;
}
```



$X = \{x_1, x_2, x_3\}$

$D(x_1) = D(x_2) = D(x_3) = 0..255$

$C \begin{cases} c_1 : x_3 \leq x_2 \\ c_2 : x_1 \leq x_2 \\ c_3 : x_3 < x_1 \end{cases}$ CSP

Solution!!
$x_1 = 10$
$x_2 = 15$
$x_3 = 5$

querying the constraints

```
Abstract class Constraint {                    interface
bool      isTrue();                              for
name[]    getVariables();                  differenciable
int       violations() ;                      objects
int       violations(name x) ;
int       getAssignDelta(name x, int v);
int       getSwapDelta(name x, name y);
}
```

CBLS