SILC
Symbolic Constraints in Integer Linear Programming

Ernst Althaus   Alexander Bockmayr   Mathias Elf

Thomas Kasper   Michael Jünger   Kurt Mehlhorn

Structured Integer Linear Constraints (SILC)

- Integer Linear Programming (branch-and-cut, branch-and-price) is a powerful paradigm for solving hard combinatorial problems
- problem specification is low-level (equalities and inequalities) despite model languages
- reuse is limited
- constraint programming, the competitor, uses high-level constraints, e.g.

  $$\textit{alldiff}\,(x_1, \ldots, x_n) \text{ enforces } x_i \neq x_j \text{ for } i \neq j.$$
- separation and pricing require advanced algorithms
- SILC project
  - specification language with high-level constraints
  - knowns about graphs and geometric objects (inherited from LEDA)
  - uses ABACUS as the underlying branch-and-cut system
  - collect high-level constraints
  - develop the corresponding separation routines
  - put them in a library

Example: Traveling Salesman Problem

$\langle TSP.c \rangle \equiv$

```
graph              G;
edge_array<int>    cost(G);

// read G and the cost function from some file

ILP_problem        P(optsense_min);

var_map<edge>      EV(nil);
forall_edges(e,G) EV[e] = P.add_binary_variable(cost[e]);

P.add_constraint(TOUR(G, EV));

P.solve();
```

- *TOUR*-constraint states that edges with $x_e = 1$ form a tour in G
- it is part of the library, e.g., realized as a combination of
- *solve* uses branch-and-cut paradigm
- state-of-the-art algorithm results

Variable Maps

- the underlying LP-solver indexes variables by integers 0, 1, 2, ...
- want to index them by objects of the application, e.g.,

  by the edges of a graph

- variable maps make the connection

```
var_map<edge>      EV(nil);
forall_edges(e,G) EV[e] = P.add_binary_variable(cost[e]);
```

- connection is bi-directional
  - $EV.\mathit{defined}(e)$ returns true if e has a variable associated with it
  - $EV[e]$ is the variable connected with e,
  - $EV[v]$ is the edge connected with the (internal) variable v or nil.

- inequality maps are similar

Inequality Generators

ILPs frequently use degree constraints:

$$\text{for all nodes } v \text{ of } G\text{: } \sum_{e \in \delta(v)} x_e = 2$$

in SCIL one writes:

```
forall_nodes(v,G)
  P.add_inequality(new degree_equality(G,EV,v,2));
```

where EV maps the edges of G to variables and v is a node of G.

The class *degree_equality* is defined on the next slide.

⟨*degree_equality*⟩≡

```
class degree_equality : public Inequality{
  ⟨degree_equality - private part⟩
public:
  degree_equality(graph& G_, var_map<edge>& EV_, node v0_, int deg_);
    // stores the arguments in the private variables G, EV, v0, deg
  double right_hand_side() const { return deg; }
  SILC_sense  sense() const { return SILC_equal;}
  double coeff(var v) const
  { edge e = EV[v];
    if (e == EV.nil_object()) return 0;
    if ((source(e) == v0) || (target(e) == v0)) return 1; else return 0;
  }
  list<var> non_zero_entries() const
  { edge e; list<var> L;
    forall_inout_edges(e,v) L.append(EV[e]);
    return L;
  }
}
```

The Implementation of the Tour Constraint

cut_inequality(G, EV, C) generates the constraint $\sum_{e \in \delta(C)} x_e \geq 2$.

$\langle$*TOUR constraint*$\rangle \equiv$

```
class TOUR {
  graph& G; var_map<edge>& EV;
public:
  TOUR(graph& G_, var_map<edge>& EV_) G(G_), EV(EV_): {}
  void init(branch_and_cut_algorithm& BC)
  { forall_nodes(v, G) P.add_inequality(degree_inequality(G, EV, v, 2)); }
  Separation_Status separate(Subproblem& s)
  { edge_array<double> c(G);
    forall_edges(e,G) c[e] = s.xVal(EV[e]);
    node_array<bool> C(G);
    double cut_value = MIN_CUT(G, c, C);
    if ( cut_value < 2 ) { s.add_inequality(cut_inequality(G,EV,C));
                            return SCIL_success_in_separation; }
    else return SCIL_failure_in_separation;
```

```
  }

  // other member functions, e.g., feasibility test
}
```

Status of the Project

- have a 50 page preliminary report describing the system
- 5 pages are suggestions for improvements
- implementation conforms with report, at least to a large extent
- report and system are evolving
- system has only been used by insiders (= us)
- system has been used successfully by us for
  - curve reconstruction based on TSP
  - problems in bioinformatics: multiple sequence alignment and docking
- library of symbolic constraints and separation routines is still very small.
- project is open for additional members
  - to contribute to the library
  - to experiment with the system and to further develop it