A principled approach to automatically recommend optimization methods

Sofiane Tanji, François Glineur

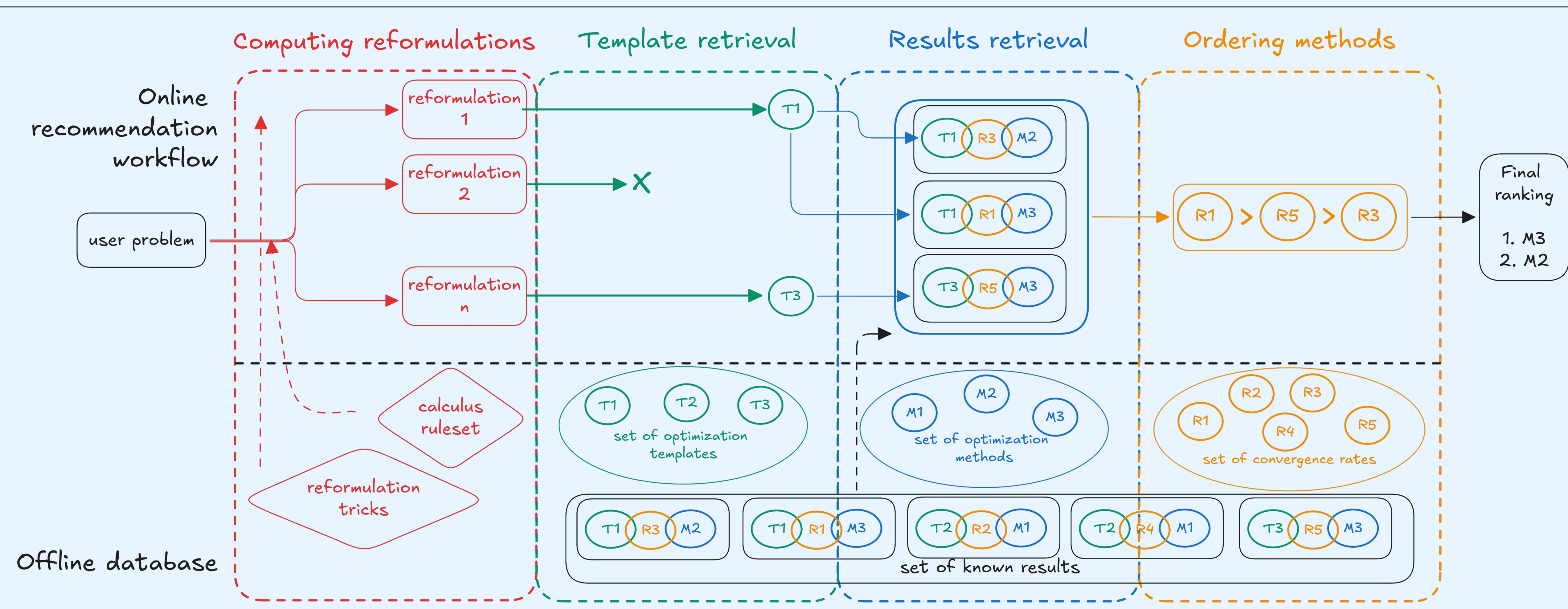
UCLouvain

sofiane.tanji@uclouvain.be

POSTER

SLIDES

Our approach to recommend optimization methods



Context

Find the best methods to efficiently solve the general problem

$$\min_{x \in \mathcal{X}} f(x) \tag{OPT}$$

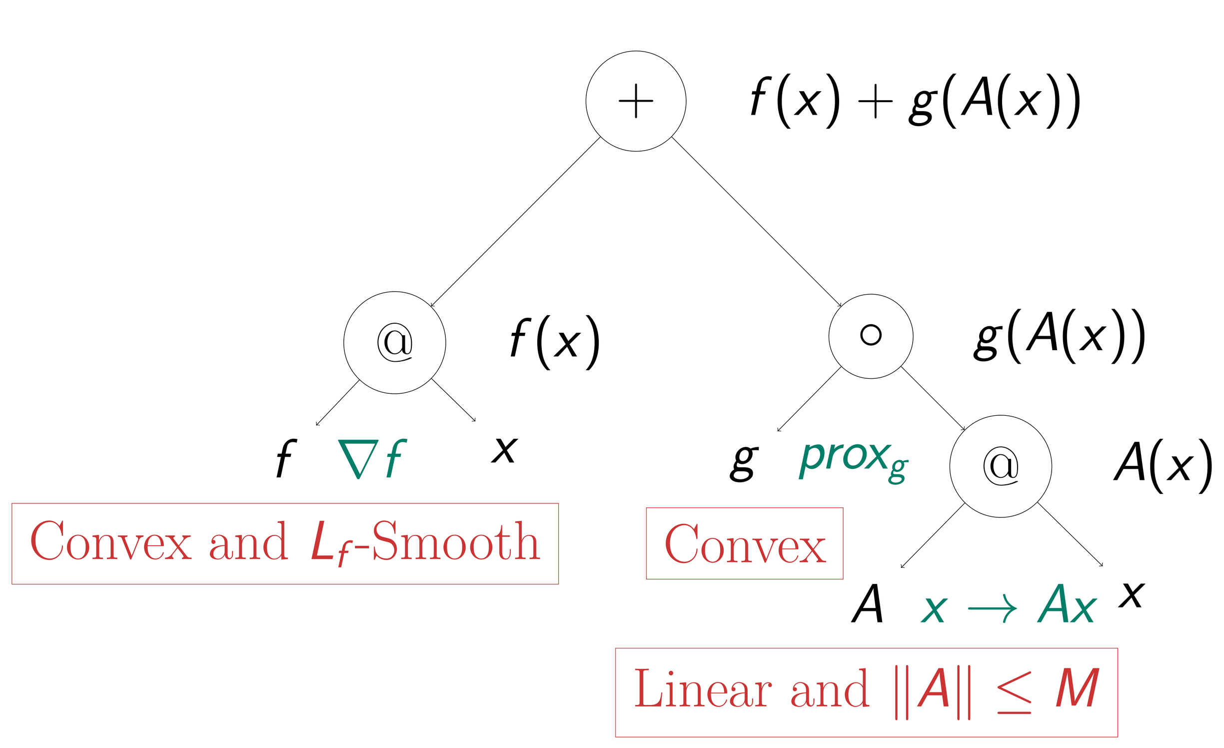
where f is composed of multiple subfunctions f_i, linked by common operators and involving multiple variables.

- Each f_i satisfy at least one assumption (L-Lipschitz gradient, convexity etc.)
- We access each f_i through black-box oracles (gradient, proximal operator etc.)

Problem representation

$$\min_{x \in \mathbb{R}^n} f(x) + g(A(x)) \tag{1}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ is convex, L_f-smooth. Access to ∇f.
- $g : \mathbb{R}^m \to \mathbb{R}$ is convex. Access to $prox_g$.
- $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear mapping with $\|A\| \leq M$. Access to $x \to Ax$.



Code example

```
pb = Problem()
f = pb.declare_function("f", Rn, R)
g = pb.declare_function("g", Rn, R)
A = pb.declare_function("A", Rn, Rn)
x = pb.declare_variable("x", Rn)
f.add_property(Convex())
f.add_property(Smooth(0, 10.))
g.add_property(Convex())
A.add_property(Linear(0, 5.))
pb.declare_oracle(Derivative(f))
pb.declare_oracle(Proximal(g))
pb.declare_oracle(LinearMap(A))
pb.set_objective(f(x) + g(A(x)))
```

Finding new formulations

Ingredients used to compute new reformulations

1. **Elementary reformulation operations:** commutativity of operators, reparametrization, transfer of conditioning, losing structure etc.
2. **Calculus ruleset:** gradient/subgradient/prox/Fenchel calculus etc.

Compute (many) reformulations automatically by applying above tools to the original problem recursively

Database of known results

Claim: All convergence theorems have the following form

Theorem (informal): Worst-case convergence rate of Algorithm 1

Suppose assumptions $\mathcal{A} = \{A1, A2, A3\}$ hold. Consider initial conditions $\mathcal{I}$.

Then, Algorithm 1 with parameters $\mathcal{P}$ applied to Problem (1) satisfies for all $k \geq 1$

$$F(x_k) - F(x^*) \leq \varphi(k, \mathcal{A}, \mathcal{I}, \mathcal{P}) \tag{2}$$

meaning we have to encode 3 elements : **Template problem**, **Method parameters**, **Convergence rate**.

Template problem: a "common" optimization problem for which researchers provided convergence guarantees

Method parameters: any parameter appearing in the rate function, potentially based on an unknown quantity $(L, \mu \dots)$

Convergence rate: a performance measure and a function upper bounding it as tightly as possible.

A ranking assistant

We rank optimization methods with a four-step approach:

1. **Automatic computation of reformulations**
2. **Template retrieval:** Match reformulations to known templates
3. **Results retrieval:** Query the database for all results associated to matched templates
4. **Ordering methods:** Compare retrieved convergence rates using heuristics
   - Drop asymptotically worse methods if iteration budget is high
   - Compare leading coefficients
   - Sample parameter values (if unknown) in provided range, allowing rate estimation