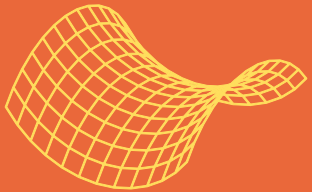


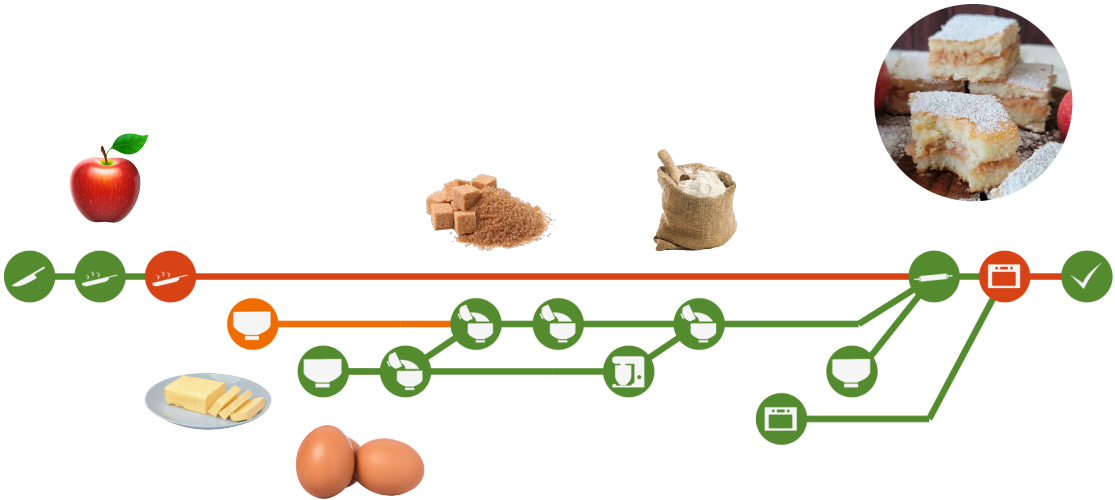
UNO

UnoSolver.jl, a unified SQP/barrier solver for nonlinearly constrained optimization

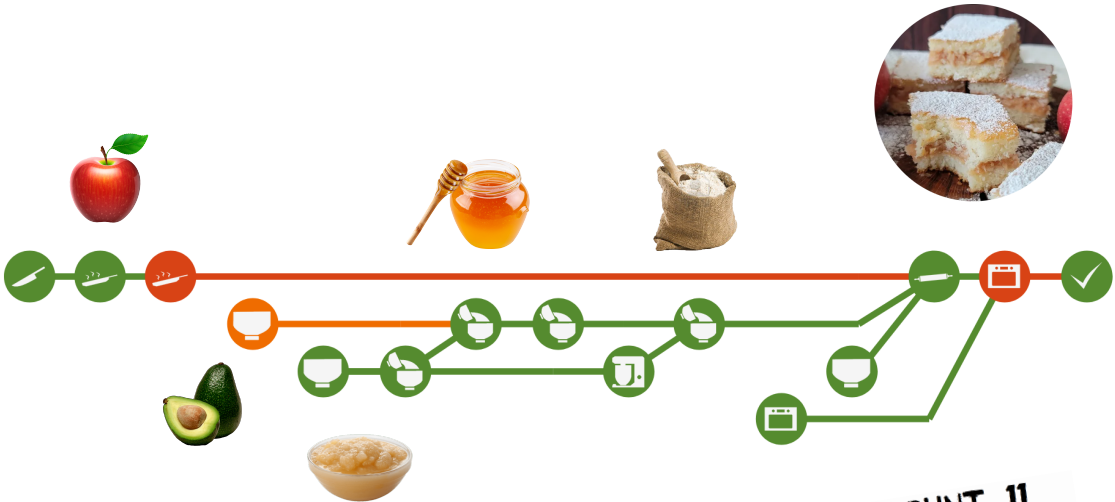
Charlie Vanaret (ZIB) Alexis Montoison (ANL / AMD)

May 2026

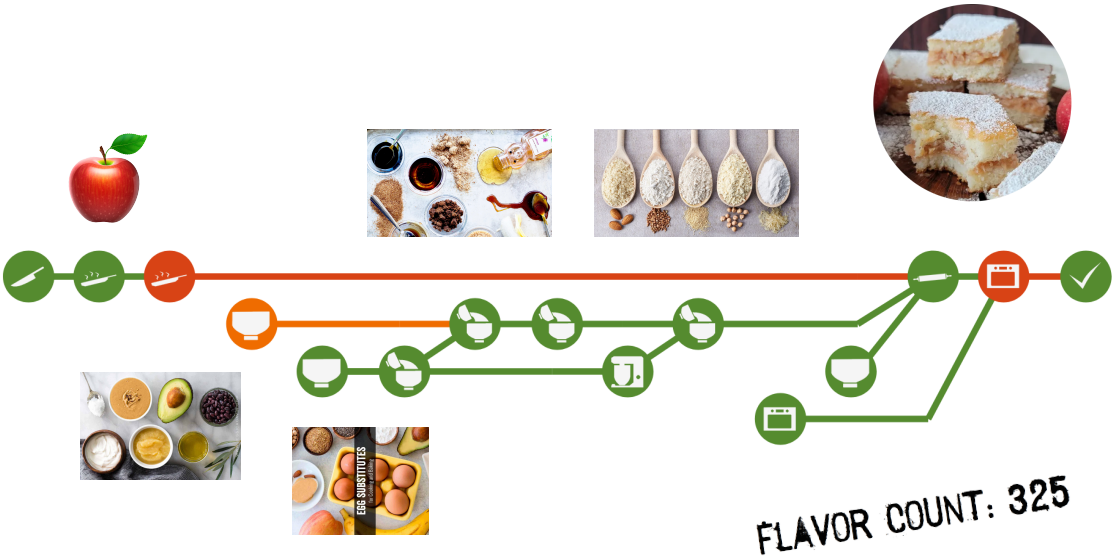
Unified cake recipe: *lenja pita* – the original recipe



Unified cake recipe: different flavors of *lenja pita*



Unified cake recipe: a generic recipe for *lenja pita*



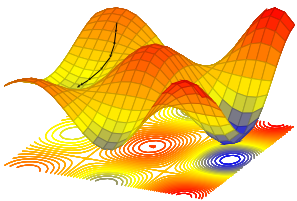
Lagrange-Newton methods for nonlinearly constrained optimization

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & f(x) & \\ \text{s.t.} & c(x) = 0 & (\text{dual } y) \\ & x \geq 0 & (\text{dual } z) \end{cases} \qquad \text{(NLP)}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ twice continuously differentiable

- Optimality (John) conditions at (x^*, y^*, z^*, ρ^*)

$$\begin{aligned} \text{(stationarity)} \quad & \rho^* \nabla f(x^*) - \nabla c(x^*) y^* - z^* = 0 \\ \text{(primal feasibility)} \quad & c(x^*) = 0, \quad x^* \geq 0 \\ \text{(dual feasibility)} \quad & z_i^* \geq 0, \quad \rho^* \geq 0, \quad (y^*, z^*, \rho^*) \neq (0, 0, 0) \\ \text{(complementarity)} \quad & x_i^* z_i^* = 0, \quad \forall i \in \{1, \dots, n\} \quad \text{combinatorial!} \end{aligned}$$



Uno: a modern, modular C++ solver[1]

- organize constrained optimization strategies into coherent hierarchy
- modular architecture: strategy combinations assembled on the fly
- reduce cost of development and maintenance of multiple solvers
- facilitate comparison of methods on given instance
- accelerate development of novel research ideas
- presets mimic existing solvers: `ipopt` (IPM), `filtersqp` (SQP)

[1] Vanaret and Leyffer 2026. "Implementing a unified solver for nonlinearly constrained optimization". Accepted in Mathematical Programming Computation.

An abstract unification framework

Most methods can be expressed as a combination of:

- reformulation layer
  - **constraint relaxation strategy** relaxes $c(x) = 0$
  - **inequality handling method** handles combinatorics of $x \geq 0$

  } reformulation (RNLP)

- Lagrange-Newton subproblem about (x_k, y_k, z_k)
  - **Hessian model** approximates Lagrangian Hessian of (NLP)
  - **inertia correction strategy** corrects inertia of Hessian or KKT matrix of (RNLP)
- subproblem solver layer
  - **subproblem solver** approximately solves subproblem (curvature? inequalities? convex?)
- globalization layer
  - **globalization strategy** acceptance/rejection of trial iterate
  - **globalization mechanism** action taken upon rejection

The unified double-loop framework

Given: initial point $(x^{(0)}, y^{(0)}, z^{(0)})$
Set $k \leftarrow 0$
while (x_k, y_k, z_k) **not optimal do**

repeat

Reformulate the problem wrt $c(x) = 0$ and $x \geq 0$

Solve (sequence of) subproblem (s) about (x_k, y_k, z_k)

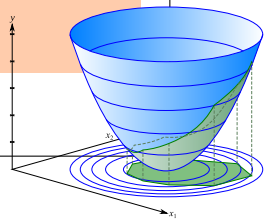
$$\begin{cases} \min\limits_{d \in \mathbb{R}^n} & \frac{1}{2} d^T H_k d + g_k^T d \\ \text{s.t.} & c_k + J_k d = 0 \\ & (x_k + d \geq 0) \\ & (\|d\| \leq \Delta_{k,l}) \end{cases}$$

Assemble trial iterate $(\hat{x}_k, \hat{y}_k, \hat{z}_k)$

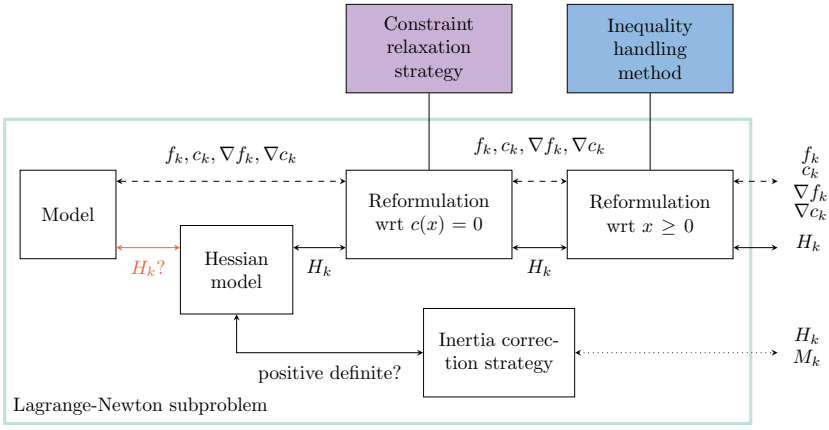
until $(\hat{x}_k, \hat{y}_k, \hat{z}_k)$ **is** acceptable

Update $(x_{k+1}, y_{k+1}, z_{k+1}) \leftarrow (\hat{x}_k, \hat{y}_k, \hat{z}_k)$
Increment k

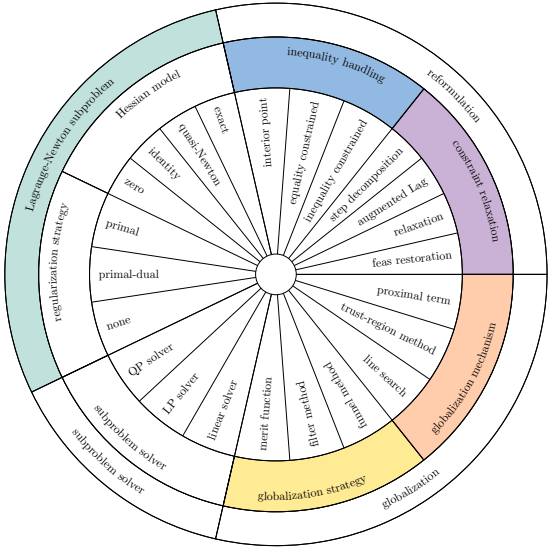
end



The Lagrange-Newton subproblem



A coherent hierarchy of NLP strategies



How SQP and barrier solvers fit within the framework

(a) filterSQP

(b) IPOPT

UnoSolver.jl

UnoSolver.jl is a registered Julia package

- thin wrapper around complete C API
- interface to MathOptInterface.jl for handling JuMP models
- interface to NLPModels.jl (e.g., ExaModels.jl and CUTEst.jl models)

```
pkg> add UnoSolver
julia> using UnoSolver
```

Features of UnoSolver.jl

UnoSolver.jl ships with:

- BQPD: a nullspace active-set nonconvex QP solver (BQPD_jll)
- the HiGHS LP solver (HiGHS_jll)
- MUMPS (MUMPS_seq_jll)
- SSIDS (SPRAL_jll)
- an option to use libHSL by hot-swapping HSL_jll

Uno_jll.jl compiled with demuxer libblastrampoline_jll

- switch between BLAS and LAPACK backends at runtime (OpenBLAS, Intel MKL, Apple Accelerate)

An efficient MOI interface

- functions to set single component of x_0 or bounds
- relevant when the problem barely changes between two solves

Solving with UnoSolver.jl

```
using UnoSolver, JuMP

model = Model(() -> UnoSolver.Optimizer(preset="filtersqp"))

@variable(model, x <= 0.5, start = -2);
@variable(model, y, start = 1);

@objective(model, Min, 100 * (y - x^2)^2 + (1 - x)^2);

@constraint(model, x*y >= 1);
@constraint(model, x + y^2 >= 0);

optimize!(model)
```

hs015: experimenting with various methods

```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp"))
```

```
Objective evaluations:   8
```

hs015: experimenting with various methods

```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp"))
```

```
Objective evaluations:   8
```

```
model = Model(() -> UnoSolver.Optimizer(preset="ipopt"))
```

```
Objective evaluations:   22
```

hs015: experimenting with various methods

```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp"))
```

```
Objective evaluations:   8
```

```
model = Model(() -> UnoSolver.Optimizer(preset="ipopt"))
```

```
Objective evaluations:   22
```

```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp", hessian_model="LBFGS"))
```

```
Objective evaluations:   5
```

hs015: experimenting with various methods

```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp"))
```

```
Objective evaluations:   8
```

```
model = Model(() -> UnoSolver.Optimizer(preset="ipopt"))
```

```
Objective evaluations:   22
```

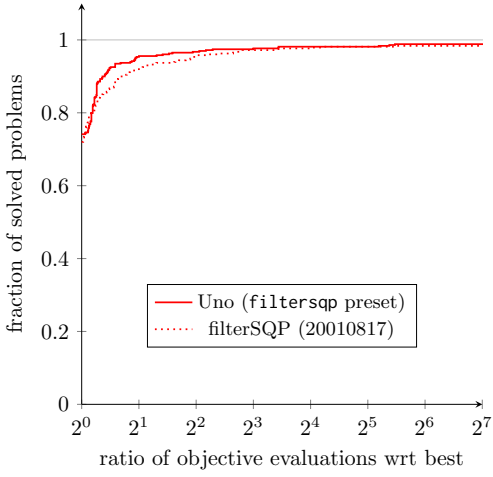
```
model = Model(() -> UnoSolver.Optimizer(preset="filtersqp", hessian_model="LBFGS"))
```

```
Objective evaluations:   5
```

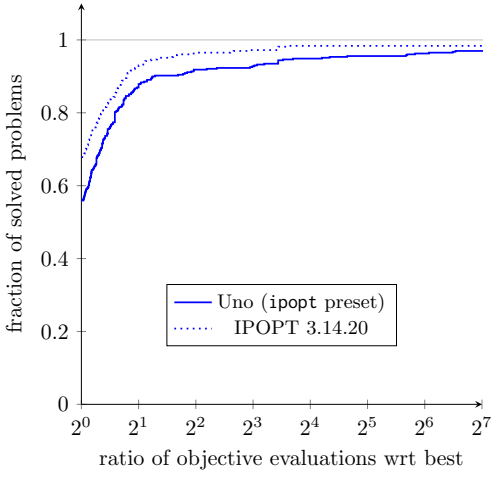
```
model = Model(() -> UnoSolver.Optimizer(preset="ipopt", globalization_strategy="
    funnel_method"))
```

```
Objective evaluations:   21
```

Performance profile vs filterSQP and IPOPT on 429 small CUTE instances

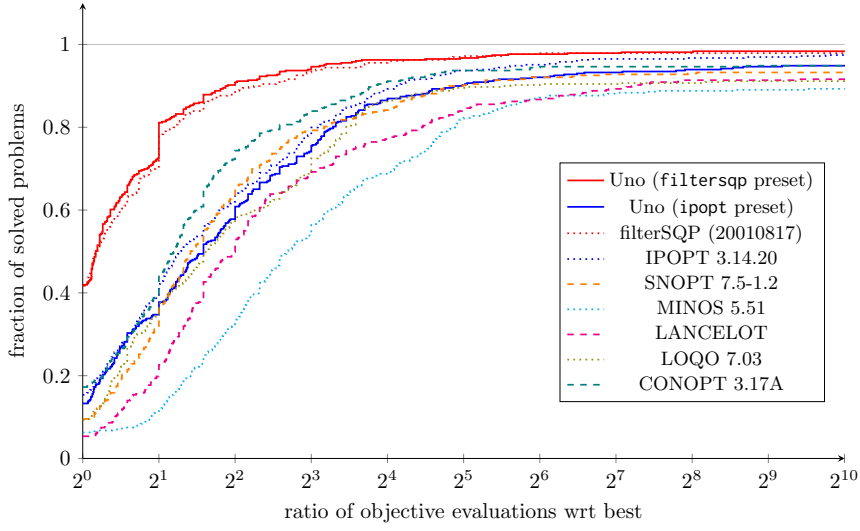


(a) Uno `filtersqp` vs filterSQP



(b) Uno `ipopt` vs IPOPT

Performance profile of Uno vs SNOPT, MINOS, LANCELOT, LOQO, CONOPT



Success stories

franckgaga 2 Nov 2025

I just tested UnoSolver.jl with `filtersqp`, a multiple shooting transcription, and exact Hessian computed by sparse ForwardDiff.jl on the inverted pendulum case study (unstable system + highly nonlinear/nonconvex). It's about 2.1 times faster than Ipopt.jl, all other things being equal.

baggepinnen Nov 2025

I just tried Uno with the filtersqp present as the inner optimizer for a mixed-integer MPC problem, it was 6x faster despite using tighter tolerances than Ipopt. What a nice addition to the julia optimizer ecosystem 🙂

cgeoga Dec 2025

That algorithm 4 in KNITRO really was a workhorse for me, but `uno` with the `filtersqp` preset has worked so well for me that I have very happily given up my KNITRO license.

Current and future developments (1/2)

- TRON-like solver for bound-constrained QP
  - Gauss-Newton feasibility problem
  - BCL
- interface Alexis' `Krylov.jl` (shared library + C interface)
- KNITRO-IPM implementation
  - Byrd-Omojokun (trust-region step decomposition) framework
  - exploit negative curvature
- exploit optimal control structure (with D. Kießling)
- adapt for reoptimization by recycling internal workspaces
  - MINLP
  - `InfiniteOpt.jl`

Current and future developments (2/2)

Alternative to log barrier: exponential(-multiplier) function[2] for some multiplier estimates $w \geq 0$:

$$\min_{x \in \mathbb{R}^n} \quad \psi(x, w, \mu) := f(x) + \mu \sum_{j=1}^{m} w_j \exp(-c_j(x)/\mu),$$

and build sequence $\{(\mu_k, w_k)\}$ with decreasing μ. No step truncation required

Notice:

- $y(x, w, \mu) := w \exp(-c(x)/\mu)$ converge to Lagrange multipliers
- if we pick w to approach $y > 0$, $\exp(-c(x)/\mu) \to 1$ therefore $c(x) \to 0$
- $y_j(x, w, \mu)$ converges exponentially fast to 0 for inactive constraints $c_j(x) > 0$ if $\mu \to 0^+$

[2] Kort and Bertsekas 1973. "Multiplier methods for convex programming".

Takeaway

- Uno abstracts the workflows of (most) Lagrange-Newton methods
  - coherent hierarchy of optimization strategies
  - powerful way to combine strategies on the fly
- experimentation lab to quickly implement and connect new strategies
- transitioning from proof of concept to efficient solver
- ultimately unlock more complex classes of problems (robust, MINLP, MPCC)

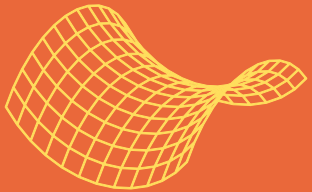
Argonne NATIONAL LABORATORY

ZIB ZUSE INSTITUTE BERLIN

Try out Uno on your favorite problem and give us some feedback
`https://github.com/cvanaret/Uno`

Special thanks to contributors

Oscar Dowson Marcel Jacobse Arnav Kapoor David Kießling Rujia Liu
Stefano Lovato Manuel Schaich Silvio Traversaro



UNO

UnoSolver.jl, a unified SQP/barrier solver for nonlinearly constrained optimization

Charlie Vanaret (ZIB) Alexis Montoison (ANL / AMD)

May 2026