

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/381522383>

Unifying nonlinearly constrained nonconvex optimization

Preprint · June 2024

DOI: 10.48550/arXiv.2406.13454

CITATIONS

0

READS

786

2 authors:



[Charlie Vanaret](#)

Argonne National Laboratory

61 PUBLICATIONS 415 CITATIONS

[SEE PROFILE](#)



[Sven Leyffer](#)

Argonne National Laboratory

168 PUBLICATIONS 8,352 CITATIONS


[SEE PROFILE](#)

Unifying nonlinearly constrained nonconvex optimization

Charlie Vanaret · Sven Leyffer

June 19, 2024


Abstract Derivative-based iterative methods for nonlinearly constrained nonconvex optimization usually share common algorithmic components, such as strategies for computing a descent direction and mechanisms that promote global convergence. Based on this observation, we introduce an abstract framework based on four common ingredients that describes most derivative-based iterative methods and unifies their workflows. We then present Uno, a modular C++ solver that implements our abstract framework and allows the automatic generation of various strategy combinations with no programming effort from the user. Uno is meant to (1) organize mathematical optimization strategies into a coherent hierarchy; (2) offer a wide range of efficient and robust methods that can be compared for a given instance; (3) enable researchers to experiment with novel optimization strategies; and (4) reduce the cost of development and maintenance of multiple optimization solvers. Uno’s software design allows user to compose new customized solvers for emerging optimization areas such as robust optimization or optimization problems with complementarity constraints, while building on reliable nonlinear optimization techniques. We demonstrate that Uno is highly competitive against state-of-the-art solvers filterSQP, IPOPT, SNOPT, MINOS, LANCELOT, LOQO, and CONOPT on a subset of 429 small problems from the CUTEst collection. Uno is available as open-source software under the MIT license at <https://github.com/cvanaret/Uno>.

C. Vanaret 

Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA

Department of Mathematical Optimization, Zuse-Institut Berlin, 14195 Berlin, Germany

E-mail: vanaret@zib.de

S. Leyffer 

Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA

E-mail: leyffer@anl.gov

Keywords Nonconvex optimization · interior-point methods · sequential quadratic programming methods · globalization techniques

Mathematics Subject Classification (2010) 49M15 · 65K05 · 90C30 · 90C51 · 90C55

Acknowledgments

This work was supported by the Applied Mathematics activity within U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

We would like to thank David Kiessling (KU Leuven) for fruitful discussions about nonconvex optimization and his extensive testing of Uno, Nils-Christian Kempke (Cardinal Operations) for his valuable help with technical questions in C++, and Gail Pieper (Argonne National Laboratory) for proofreading our manuscript.

1 Motivation and contributions

We consider nonlinearly constrained optimization problems of the form

$$\begin{aligned} \min_x & f(x) \\ \text{s.t. } & l \leq \begin{Bmatrix} c(x) \\ Ax \\ x \end{Bmatrix} \leq u, \end{aligned} \quad (1)$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, $c : \mathbb{R}^n \rightarrow \mathbb{R}^{m_c}$ are constraint functions, $A \in \mathbb{R}^{m_A \times n}$ is a constraint matrix, and $l \in (\mathbb{R} \cup \{-\infty\})^{m_c+m_A+n}$ and $u \in (\mathbb{R} \cup \{\infty\})^{m_c+m_A+n}$ are lower and upper bounds, respectively. f and c may be nonconvex, which results in a nonconvex optimization problem. This formulation allows for unbounded variables and equality constraints and explicitly separates general nonlinear, linear, and bound constraints, enabling solvers to readily exploit this structure. However, for the sake of simplicity of this presentation and without loss of generality, we consider the problem in the following form:

$$\begin{aligned} \min_x & f(x) \\ \text{s.t. } & c(x) = 0 \\ & x \geq 0, \end{aligned} \quad (\text{NLP})$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, and $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

Most derivative-based iterative methods for nonlinearly constrained nonconvex optimization (e.g., [15, 25, 30, 35]) share common algorithmic components. Based on this observation, we introduce an abstract framework structured around four generic ingredients that describes these methods in a unified fashion. We then present Uno (Unifying Nonconvex Optimization), a modular

open-source solver for nonlinearly constrained nonconvex optimization that (1) unifies most existing state-of-the-art methods and organizes existing strategies into a coherent hierarchy; (2) provides efficient and robust implementations of existing strategies as independent building blocks that can be combined at will and compared on a given instance; (3) promotes extensive code reusability; and (4) allows users to experiment with new algorithmic ideas by building upon Uno’s abstractions and interfaces to modeling languages and subproblem solvers. Uno was first introduced at the ISMP 2018 conference under the name Argonot [41].

Uno automatically generates various strategy combinations on the fly with no programming effort from the user, even though all combinations do not lead to convergent methods. We demonstrate that Uno is competitive against state-of-the-art solvers on a subset of 429 CUTEst test problems [24], while being extensible and lightweight. We believe that Uno has the potential to serve as an experimental laboratory for practitioners and optimizers and to accelerate research in nonconvex optimization. Our ultimate goal is to promote the extension of state-of-the-art nonlinear optimization techniques to new classes of problems such as problems with equilibrium constraints (see, e.g., [18, 28, 29, 32, 36]) and nonlinear robust optimization (see, e.g., [31]).

This paper is organized as follows. In Section 2 we introduce our notation and discuss relevant optimality conditions. In Section 3 we introduce our abstract framework with four ingredients for unifying derivative-based iterative methods. In Section 4 we briefly describe state-of-the-art optimization strategies through the prism of our unifying framework. In Section 5 we present the basic algorithmic design of Uno and show how various nonlinear optimization methods fit within the architecture. In Section 6 we illustrate with three concrete strategy combinations how the four ingredients interact with one another. In Section 7 we provide preliminary numerical results and compare Uno with state-of-the-art solvers.

2 Notation and stationarity conditions

In this section we define our notations and state first-order optimality conditions of (NLP).

2.1 Notation

We start by defining the scaled Lagrangian or Fritz John function [4] of (NLP) at (x, y, z, ρ) :

$$\mathcal{L}_\rho(x, y, z) \stackrel{\text{def}}{=} \rho f(x) - y^T c(x) - z^T x = \rho f(x) - \sum_{j=1}^m y_j c_j(x) - \sum_{i=1}^n z_i x_i,$$

where y are the Lagrange multipliers of the general constraints $c(x) = 0$, $z \geq 0$ are the Lagrange multipliers of the bound constraints $x \geq 0$, and $\rho \geq 0$ is a

multiplier of the objective that is introduced to handle infeasibility or lack of constraint qualification (CQ) [33] in a consistent way.

$\nabla_x \mathcal{L}_\rho(x, y, z)$ is the gradient of the scaled Lagrangian with respect to x :

$$\nabla_x \mathcal{L}_\rho(x, y, z) \stackrel{\text{def}}{=} \rho \nabla f(x) - \sum_{j=1}^m y_j \nabla c_j(x) - z.$$

$W_\rho(x, y)$ is the Hessian of the scaled Lagrangian with respect to x :

$$W_\rho(x, y) \stackrel{\text{def}}{=} \nabla_{xx}^2 \mathcal{L}_\rho(x, y, z) = \rho \nabla^2 f(x) - \sum_{j=1}^m y_j \nabla^2 c_j(x).$$

2.2 First-order stationarity conditions

We are primarily concerned with first-order stationary points. The first-order optimality conditions (aka Fritz John conditions) of problem (NLP) at a stationary point x^* state that there exist (ρ^*, y^*, z^*) such that

$$\begin{aligned} \text{(stationarity)} \quad \nabla_x \mathcal{L}_{\rho^*}(x^*, y^*, z^*) &= \rho^* \nabla f(x^*) - \sum_{j=1}^m y_j^* \nabla c_j(x^*) - z^* = 0 \end{aligned} \tag{2a}$$

$$\text{(primal feasibility)} \quad c(x^*) = 0, \quad x^* \geq 0 \tag{2b}$$

$$\text{(dual feasibility)} \quad \rho^* \geq 0, \quad z_i^* \geq 0, \quad (\rho^*, y^*, z^*) \neq (0, 0, 0) \tag{2c}$$

$$\text{(complementarity)} \quad x_i^* z_i^* = 0, \quad \forall i \in \{1, \dots, n\}. \tag{2d}$$

If $\rho^* > 0$, the optimality conditions are equivalent to the KKT conditions, which can be recovered by scaling Equation (2a) by $1/\rho^*$. If $\rho^* = 0$, they characterize Fritz John points, that is, feasible points at which constraint qualification is violated.

3 An abstract framework for unifying nonlinear optimization

In this section we introduce a unified view for describing iterative nonlinear optimization methods and argue that they can be assembled by combining the following four generic ingredients:

1. a **constraint relaxation strategy** constructs a feasible nonlinear problem by relaxing the general constraints;
2. a **subproblem method** computes a primal-dual direction by solving a local approximation of the nonlinear problem at the current iterate;
3. a **globalization strategy** determines whether a trial iterate makes sufficient progress toward a solution and accepts or rejects it; and,

4. a **globalization mechanism** defines the recourse action taken when a trial iterate is rejected.

This coloring will be used throughout to illustrate how these four ingredients interact with one another within an optimization algorithm. The role of each ingredient is summarized in Algorithm 1: the inner loop (**repeat**) generates and solves a feasible subproblem (possibly a sequence of feasible subproblems) until a trial iterate is accepted by the globalization strategy, and the outer loop (**while**) generates a sequence of acceptable iterates until termination.

Algorithm 1: Abstract framework for iterative methods.

```

Data: initial point  $(x^{(0)}, y^{(0)}, z^{(0)})$ 
Set  $k \leftarrow 0$ 
while termination criteria at  $(x^{(k)}, y^{(k)}, z^{(k)})$  not met do
    repeat globalization mechanism
        Solve a (sequence of) feasible subproblem (s) that approximate(s) NLP
        at  $(x^{(k)}, y^{(k)}, z^{(k)})$ 
        Assemble trial iterate  $(\hat{x}^{(k+1)}, \hat{y}^{(k+1)}, \hat{z}^{(k+1)})$ 
    until  $(\hat{x}^{(k+1)}, \hat{y}^{(k+1)}, \hat{z}^{(k+1)})$  is acceptable
    Update  $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) \leftarrow (\hat{x}^{(k+1)}, \hat{y}^{(k+1)}, \hat{z}^{(k+1)})$ 
     $k \leftarrow k + 1$ 
Result:  $(x^{(k)}, y^{(k)}, z^{(k)})$ 

```

Table 1 presents a unified view of state-of-the-art solvers ALGLIB MinNLC [3], CONOPT [12], FICO XSLP [2], filterSQP [17], IPOPT [45], KNITRO [6], LANCELOT [10], LOQO [42], MINOS [34], NAG e04uc/e04wdc [1], NLPQL [40], SLSQP [27], SNOPT [23], SQuID [4], and WORHP [5]. Each solver is characterized in terms of the four ingredients within the proposed abstract framework. AL is short for augmented Lagrangian, QP for quadratic problem, LP for linear problem, and EQP for equality-constrained quadratic problem. Note that FICO XSLP is the only solver that does not implement a proper globalization strategy.

Figure 1, albeit not comprehensive, shows how most existing methods fit within the abstract framework. Various strategies are listed under each ingredient (e.g., a line-search method is a globalization mechanism) since they share a common role within an optimization method. Grey edges connect ingredients that interact with each other. A more fine-grained representation of the dependencies between ingredients is given later on in Figure 4.

4 Unified view of state-of-the-art techniques

In this section we describe a set of strategies that fall into each of the four ingredients of our abstract framework. Our goal is to illustrate the wide variety

Table 1: Description of state-of-the-art solvers within the proposed abstract framework.

Solver	Constraint relaxation strategy	Globalization strategy	Globalization mechanism	Subproblem
ALGLIB MinNLC	ℓ_1 relaxation	ℓ_1 merit	trust region	strictly convex QP
CONOPT	feasible step method	objective merit	line search	generalized reduced-gradient method
FICO XSLP	ℓ_1 relaxation	none	ℓ_∞ trust region	LP
filterSQP	feasibility restoration	filter method	ℓ_∞ trust region	nonconvex QP
IPOPT	feasibility restoration	filter method	line search	primal-dual interior-point
KNITRO-ASM	ℓ_1 relaxation	ℓ_1 merit	trust region	LP-EQP
KNITRO-IPM	feasible mode	ℓ_2 merit	trust region / line search	primal-dual interior-point (full- or reduced-space methods)
LANCELOT	bound-constrained AL	AL merit	ℓ_2 / ℓ_∞ trust region	projected gradient and EQP \simeq LP-EQP
LOQO	ℓ_2^2 relaxation	ℓ_2^2 merit	line search	primal-dual interior-point
MINOS	linearly constrained AL + ℓ_1 relaxation	forcing sequences	line search	reduced-gradient method
NAG e04uc/e04wdc	ℓ_1 relaxation	AL merit	line search	strictly convex QP
NLPQL	right-hand-side relaxation	AL merit	line search	strictly convex QP
SLSQP	right-hand-side relaxation	ℓ_1 merit	line search	strictly convex QP
SNOPT	ℓ_1 relaxation	AL merit	line search	strictly convex QP
SQuID	ℓ_1 relaxation	ℓ_1 merit	line search	strictly convex QP
WORHP	right-hand-side relaxation	ℓ_1 merit / filter method	line search	strictly convex QP

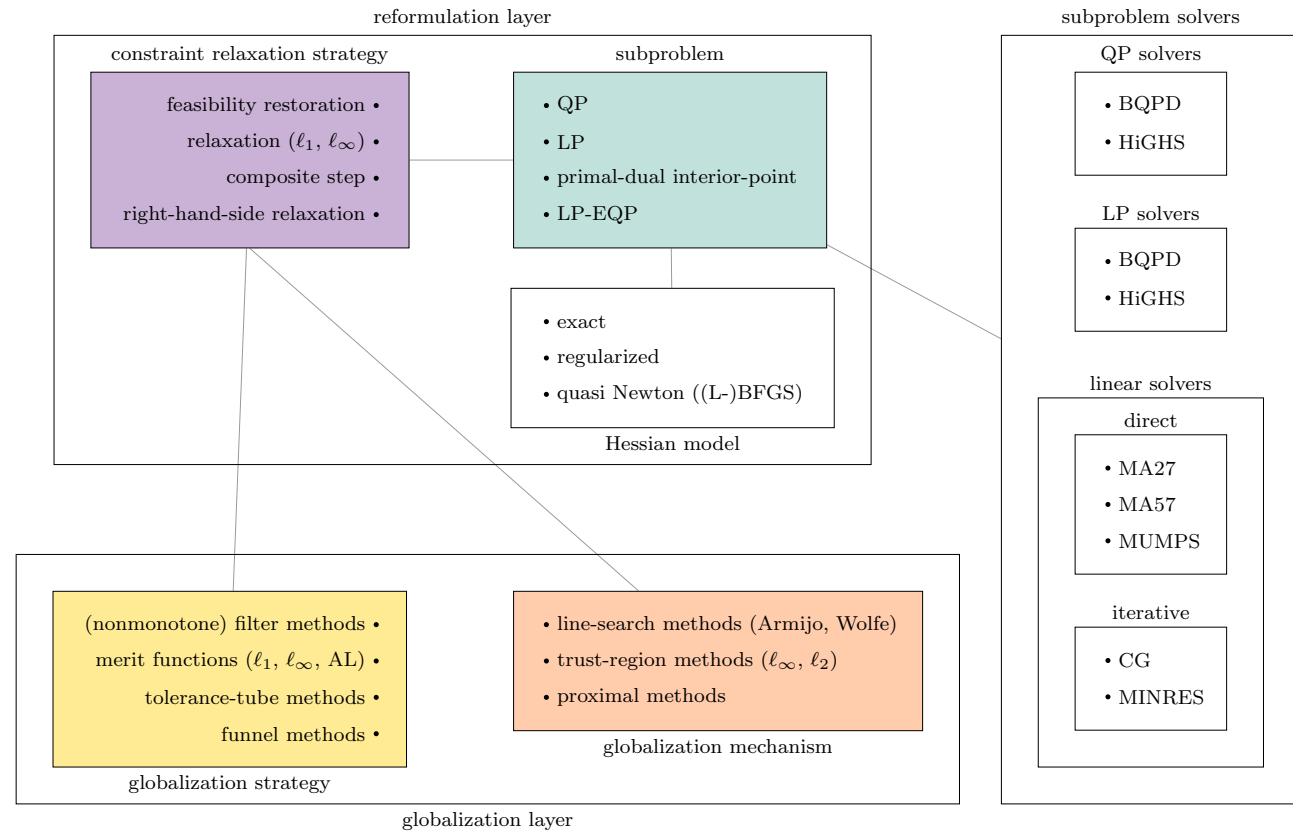


Fig. 1: Full abstract framework.

of methods within a common notation to motivate the design of our modular solver.

4.1 Constraint relaxation strategies

In general, we cannot assume that the nonlinear problem (NLP) or the subproblems are feasible. Hence, nonlinear optimization solvers must include provisions for infeasible problems. Moreover, nonlinear solvers may converge to points that violate standard constraint qualifications, and we must take these situations into account when defining optimality conditions. We review two constraint relaxation strategies: ℓ_1 relaxation and feasibility restoration.

4.1.1 ℓ_1 relaxation

The ℓ_1 relaxation strategy replaces a constrained optimization problem with a nonsmooth bound-constrained problem in which a penalty term is added to the objective:

$$\begin{aligned} \min_x \quad & \rho f(x) + \|c(x)\|_1 \\ \text{s.t.} \quad & x \geq 0, \end{aligned} \quad (3)$$

where $\rho \geq 0$ is an inverse penalty parameter. An appropriate value of ρ is obtained by solving a sequence of subproblems; efficient steering rules can be found in [4, 7, 8]. Note that the nonsmooth ℓ_1 relaxed problem can be reformulated as a smooth constrained problem with elastic variables. Other norms can be used; however, the ℓ_1 norm has emerged as the preferred option. In particular, the ℓ_1 relaxation is exact: one can show under mild conditions that for $\rho > 0$ sufficiently small, a second-order sufficient point of (3) is also a second-order sufficient point of (NLP) (see, e.g., Theorem 14.3.1 in [15]).

4.1.2 Feasibility restoration

An infeasible subproblem results from inconsistent linearized or bound constraints; it is an indication that (NLP) may be infeasible. In this case, the method may revert to the *feasibility restoration phase*: the original objective is temporarily discarded, and the following feasibility problem is solved instead:

$$\begin{aligned} \min_x \quad & \|c(x)\| \\ \text{s.t.} \quad & x \geq 0, \end{aligned} \quad (4)$$

for some norm $\|\cdot\|$ in \mathbb{R}^m . The aim of solving the feasibility problem is to compute a point as close as possible to the feasible region. Feasibility restoration improves feasibility until a minimum of the constraint violation is obtained or the subproblem becomes feasible again, in which case solving the original problem (the *optimality phase*) is resumed. Any (local) solution $x^* \geq 0$ of (4) with $\|c(x^*)\| > 0$ is a certificate that (NLP) is (locally) infeasible.

4.2 Subproblem strategies

Subproblem strategies construct a local approximation of the nonlinear reformulated problem at the current primal-dual iterate. We briefly review three classes of subproblem strategies: inequality-constrained methods, equality-constrained methods, and interior-point methods. Inequality-constrained methods invoke a (usually active-set) subproblem solver that handles inequality constraints. Equality-constrained methods compute a cheap estimate of the active set, then solve an equality-constrained subproblem to refine the solution. In contrast, interior-point methods delay the identification of the activities until the end. Note that a classification into inequality-constrained and equality-constrained SQP methods can be found in [35].

In the following, we abuse notation and denote the current evaluations of the nonlinear reformulated problem by $f^{(k)} \stackrel{\text{def}}{=} f(x^{(k)})$, $c^{(k)} \stackrel{\text{def}}{=} c(x^{(k)})$, $\nabla f^{(k)} \stackrel{\text{def}}{=} \nabla f(x^{(k)})$, $\nabla c^{(k)} \stackrel{\text{def}}{=} \nabla c(x^{(k)})$, and $W^{(k)} \stackrel{\text{def}}{=} W(x^{(k)}, y^{(k)})$.

4.2.1 Inequality-constrained subproblem

In inequality-constrained methods, the handling of inequality constraints is deferred to the subproblem solver. Traditionally, sequential quadratic programming (SQP) methods [26, 38, 39, 46] generate a sequence of quadratic problems (QPs) that are solved by means of an active-set QP solver: it maintains an estimate of the active set and solves a sequence of equality-constrained subproblems. The estimate of the active set is updated at each iteration using dual information, until the algorithm terminates with primal-dual feasibility. SQP methods converge quadratically under reasonable assumptions near a local minimizer, once the active set settles down. The local quadratic approximation at the current point is given by

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} d_x^T W^{(k)} d_x + (\nabla f^{(k)})^T d_x \\ \text{s.t.} \quad & c^{(k)} + (\nabla c^{(k)})^T d_x = 0 \\ & x^{(k)} + d_x \geq 0. \end{aligned}$$

For convex equality-constrained problems, an SQP iteration can be interpreted as taking a Newton step on the first-order optimality conditions of the problem.

Sequential linear programming [9] is a particular case of SQP in which the subproblems are linear problems (LPs); that is, no second-order information is exploited ($W^{(k)} = 0$).

4.2.2 Equality-constrained subproblem

Equality-constrained methods operate in two phases: the first phase solves a “low-fidelity” subproblem (such as an LP or a convex QP with a quasi-Newton Hessian), which provides an estimate of the active set \mathcal{A} . The second phase solves a “high-fidelity” (such as second-order) equality-constrained problem with exact Hessian in which the inequality constraints of the active set are

fixed to their active bounds and the inactive inequalities are dropped. This is illustrated in the following equality-constrained problem (using the notation of Problem 1):

$$\begin{aligned} \min_x & f(x) \\ \text{s.t.} & \begin{Bmatrix} c(x) \\ Ax \\ x \end{Bmatrix}_{\mathcal{A}} = b_{\mathcal{A}}, \end{aligned}$$

where each component of $b_{\mathcal{A}}$ is the active (lower or upper) bound of the corresponding constraint. A typical example of equality-constrained methods is SLPEQP [9] (aka SLQP): it estimates the active set by solving an LP, then solves an equality-constrained QP (EQP).

4.2.3 Interior-point subproblem

Primal-dual interior-point methods relax the complementarity equations (2d) by a factor $\mu > 0$:

$$x_i z_i = \mu, \quad \forall i \in \{1, \dots, n\},$$

which implies $x > 0$ and $z > 0$. Provided that the subproblem is convex, the primal-dual direction is the solution of a Newton linear system, the *primal-dual system* (or full-space system), in which the multipliers z are treated as independent variables:

$$\begin{pmatrix} W^{(k)} & \nabla c^{(k)} & I \\ (\nabla c^{(k)})^T & & \\ Z^{(k)} & -X^{(k)} & \end{pmatrix} \begin{pmatrix} d_x \\ -d_y \\ -d_z \end{pmatrix} = - \begin{pmatrix} \nabla f^{(k)} - \nabla c^{(k)} y^{(k)} - z^{(k)} \\ c^{(k)} \\ X^{(k)} Z^{(k)} e - \mu e \end{pmatrix}, \quad (5)$$

where $X^{(k)} = \text{diag}(x^{(k)})$, $Z^{(k)} = \text{diag}(z^{(k)})$, and e is a vector of ones of appropriate size. The primal-dual system can be made symmetric by eliminating d_z from the third row block. Positivity of x and z is then enforced by the so-called fraction-to-the-boundary rule [45]. Similarly to homotopy methods, the parameter $\mu > 0$ is asymptotically driven to 0 until the termination criteria are met. In contrast, the *primal system* (or barrier system) treats z as dependent variables.

4.3 Globalization strategies

Constrained optimization methods must achieve two competing goals: minimizing the objective function and minimizing the constraint violation. Globalization strategies determine whether a trial iterate $\hat{x}^{(k+1)} \stackrel{\text{def}}{=} x^{(k)} + \alpha d_x^{(k)}$ (given by a fraction $\alpha \in (0, 1]$ along a direction $d_x^{(k)}$) makes acceptable progress with respect to these goals. We consider strategies that ensure *global convergence*, that is, convergence to a local minimum, or (weaker) stationary point, from any starting point. In addition, ideally, the minimization of the measure of infeasibility takes precedence. However, in problems where no feasible point

exists, a (global) minimum of the constraint violation is a certificate that the problem is infeasible.

Three (possibly primal-dual) *progress measures* are monitored throughout the optimization process:

1. an **infeasibility measure** η (typically $\|c(x)\|$ for some norm);
2. an **objective measure** ω_ρ parameterized by the objective multiplier $\rho \geq 0$ (typically $\rho f(x)$); and
3. an **auxiliary measure** ξ (terms such as barrier or proximal terms).

In order to ensure convergence, these measures must be intimately linked to the problem reformulation: η and ω_ρ are defined by the constraint relaxation strategy (Table 2), while ξ is defined by the subproblem method (Table 3).

Table 2: Objective measure ω_ρ and infeasibility measure η .

Constraint relaxation strategy	$\omega_\rho(x)$	$\eta(x)$
ℓ_1 relaxation	$\rho f(x)$	$\ c(x)\ _1$
ℓ_1 feasibility restoration		

Table 3: Auxiliary measure ξ .

Subproblem method	$\xi(x)$
Primal-dual interior-point subproblem	$-\mu \sum_{i=1}^n \log(x_i)$
(In)equality-constrained subproblem	0

The local models of $\eta(x)$, $\omega_\rho(x)$, and $\xi(x)$ at iteration k about the current iterate are denoted by $\boldsymbol{\eta}^{(k)}(d_x)$, $\boldsymbol{\omega}_\rho^{(k)}(d_x)$, and $\boldsymbol{\xi}^{(k)}(d_x)$ for a given primal direction d_x . We define the respective *predicted reductions* $\Delta\boldsymbol{\eta}^{(k)}(d_x) \stackrel{\text{def}}{=} \boldsymbol{\eta}^{(k)}(0) - \boldsymbol{\eta}^{(k)}(d_x)$, $\Delta\boldsymbol{\omega}_\rho^{(k)}(d_x) \stackrel{\text{def}}{=} \boldsymbol{\omega}_\rho^{(k)}(0) - \boldsymbol{\omega}_\rho^{(k)}(d_x)$, and $\Delta\boldsymbol{\xi}^{(k)}(d_x) \stackrel{\text{def}}{=} \boldsymbol{\xi}^{(k)}(0) - \boldsymbol{\xi}^{(k)}(d_x)$ in Tables 4 and 5.

Table 4: Objective reduction model $\Delta\boldsymbol{\omega}_\rho^{(k)}$ and infeasibility reduction model $\Delta\boldsymbol{\eta}^{(k)}$.

Constraint relaxation strategy	$\Delta\boldsymbol{\omega}_\rho^{(k)}(d_x)$	$\Delta\boldsymbol{\eta}^{(k)}(d_x)$
ℓ_1 relaxation	$-\rho(\nabla f^{(k)})^T d_x - \frac{1}{2} d_x^T W_\rho^{(k)} d_x$	$\ c^{(k)}\ - \ c^{(k)} + (\nabla c^{(k)})^T d_x\ _1$
ℓ_1 feasibility restoration		

Note that the penalty parameter is often attached to $\eta(x)$ in the literature. We adopt the inverse notation as in [7] for several reasons: (i) it is numerically

Table 5: Auxiliary reduction model $\Delta\xi^{(k)}$.

Subproblem method	$\Delta\xi^{(k)}(d_x)$
Primal-dual interior-point subproblem	$\mu(X^{(k)})^{-1}e^T d_x - \frac{1}{2}d_x^T \left(W_\rho^{(k)} + (X^{(k)})^{-1}Z^{(k)} \right) d_x$
(In)equality-constrained subproblem	0

easier to drive ρ to 0 than to drive the penalty parameter of $\eta(x)$ to $+\infty$; (ii) in second-order methods, the inverse penalty parameter enters the Hessian as objective multiplier; and (iii) as $\rho \rightarrow 0$, only ω_ρ vanishes, and the implicit constraints in ξ (barrier or proximal terms) are still enforced.

The two main classes of globalization strategies are merit functions and filters (Figure 2). They typically enforce a sufficient decrease condition that forces some scalar combination of η , ω_ρ , and ξ to decrease by at least a fraction of the decrease predicted by the local model.

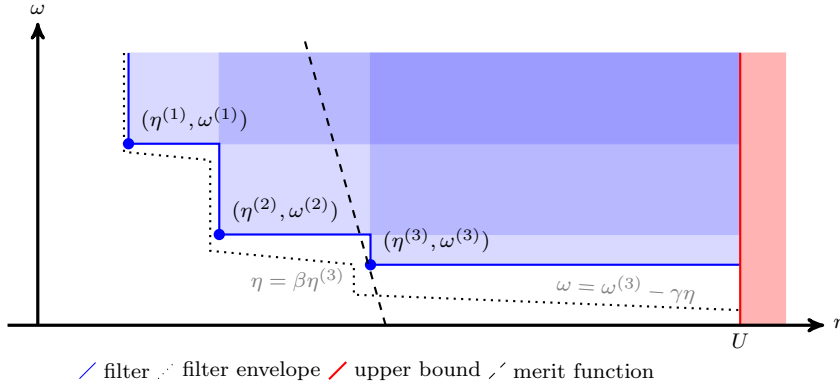


Fig. 2: Example of a filter and a merit function.

4.3.1 Merit functions

A merit (or penalty) function combines the three goals η , ω_ρ , and ξ into a single scalar value:

$$\psi_\rho(x) \stackrel{\text{def}}{=} \omega_\rho(x) + \eta(x) + \xi(x),$$

where $\rho \geq 0$ is an inverse penalty parameter. Its predicted reduction is given by

$$\Delta\psi_\rho^{(k)}(d_x) \stackrel{\text{def}}{=} \Delta\omega_\rho^{(k)}(d_x) + \Delta\xi^{(k)}(d_x) + \Delta\eta^{(k)}(d_x).$$

The trial iterate $\hat{x}^{(k+1)}$ is accepted if the actual reduction $\psi_\rho(x^{(k)}) - \psi_\rho(\hat{x}^{(k+1)})$ in the merit function is larger than a fraction $\sigma \in (0, 1)$ of its predicted reduction:

$$\psi_\rho(x^{(k)}) - \psi_\rho(\hat{x}^{(k+1)}) \geq \sigma \Delta\psi_\rho^{(k)}(\alpha d_x^{(k)}). \quad (6)$$

The sufficient decrease condition (6) (also known as Armijo condition) indicates how the subproblem solve is connected to the merit function to ensure global convergence.

4.3.2 Filter methods

Filter methods are motivated by the desire to decouple reduction in the objective function from progress toward feasibility. We can interpret a filter method as a mechanism to force iterates closer to the feasible region, so that unconstrained sufficient reduction conditions on the objective can be used to force convergence (see, e.g., [16, 19, 43, 44]). The decrease function is given by

$$\phi(x) \stackrel{\text{def}}{=} \omega_1(x) + \xi(x),$$

and its predicted reduction is given by

$$\Delta\phi^{(k)}(d_x) \stackrel{\text{def}}{=} \Delta\omega_1^{(k)}(d_x) + \Delta\xi^{(k)}(d_x).$$

Filter methods measure progress toward a solution by comparing the trial infeasibility measure η and objective measure ϕ to a filter \mathcal{F} , a list of pairs $(\eta^{(l)}, \phi^{(l)})$ (typically from previous iterates) such that no pair dominates another pair; that is, there exists no index l' such that $\eta^{(l')} < \eta^{(l)}$ and $\phi^{(l')} < \phi^{(l)}$ for all $(\eta^{(l)}, \phi^{(l)}) \in \mathcal{F}$. Formally, the trial iterate $\hat{x}^{(k+1)}$ is acceptable to the filter if and only if the following conditions hold:

$$\phi(\hat{x}^{(k+1)}) \leq \phi^{(l)} - \gamma\eta(\hat{x}^{(k+1)}) \quad \text{or} \quad \eta(\hat{x}^{(k+1)}) < \beta\eta^{(l)}, \quad \forall (\eta^{(l)}, \phi^{(l)}) \in \mathcal{F}^{(k)},$$

where $\gamma > 0$ and $0 < \beta < 1$ are constants that ensure that iterates cannot accumulate at infeasible limit points. These conditions are represented as the filter envelope in Figure 2.

The filter provides convergence only to a feasible limit because any infinite sequence of iterates must converge to a point where $\eta(x) = 0$, provided that $\phi(x)$ is bounded below. To ensure convergence to a local minimum, filter methods use a standard sufficient reduction (Armijo) condition from unconstrained optimization:

$$\phi(x^{(k)}) - \phi(\hat{x}^{(k+1)}) \geq \sigma \Delta\phi^{(k)}(\alpha d_x^{(k)}), \quad (7)$$

where $\sigma \in (0, 1)$. It makes sense to enforce this condition only if the model predicts a decrease in the objective function. Thus, filter methods use the switching condition

$$\Delta\phi^{(k)}(\alpha d_x^{(k)}) \geq \delta \eta(x^{(k)})^2,$$

where $\delta > 0$, to decide when (7) should be enforced. If the trial point is accepted, it is added to $\mathcal{F}^{(k)}$ if $\eta(x^{(k)}) > 0$ or if the switching condition fails (which automatically satisfies $\eta(x^{(k)}) > 0$).

4.4 Globalization mechanisms

When the methods are started far from a solution, directions may be unbounded or result in trial iterates that increase both the objective and the constraint violation. Globalization mechanisms provide a recourse action if a local approximation is deemed too poor to make progress toward a solution: line-search methods restrict the length of the step along a given direction, while trust-region methods restrict the length of the direction a priori.

4.4.1 Line-search methods

Line-search methods compute a trial step $\alpha d_x^{(k)}$ by determining a step length $\alpha \in (0, 1]$ along the direction $d_x^{(k)}$. Exact line-search methods compute the global solution of the one-dimensional problem $\min_{\alpha \in (0, 1]} f(x^{(k)} + \alpha d_x^{(k)})$, which is typically impractical to solve. Inexact line-search methods determine an approximate step length accepted by the globalization strategy (sufficient decrease for a merit function or acceptance by a filter). For instance, backtracking line-search methods generate a sequence of trial iterates $x^{(k)} + \alpha^{(k,l)} d_x^{(k)}$ for $l \in \mathbb{N}$ until acceptance, where $\alpha^{(k,l)} = c^l \alpha_{max}^{(k)}$, $\alpha_{max}^{(k)} \in (0, 1]$ (usually 1 in SQP methods and smaller than 1 in interior-point methods) and $c \in (0, 1)$ (Algorithm 2).

Algorithm 2: Backtracking line-search method.

Input: primal-dual iterate $(x^{(k)}, y^{(k)}, z^{(k)})$
 $(d_x^{(k)}, d_y^{(k)}, d_z^{(k)}, \alpha_{max}^{(k)}) \leftarrow$ solve subproblem at $(x^{(k)}, y^{(k)}, z^{(k)})$
 $\alpha^{(k,0)} \leftarrow \alpha_{max}^{(k)}$
Set inner iteration counter $l \leftarrow 0$
repeat
 Assemble trial iterate
 $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)}) \stackrel{\text{def}}{=} (x^{(k)}, y^{(k)}, z^{(k)}) + (\alpha^{(k,l)} d_x^{(k)}, \alpha^{(k,l)} d_y^{(k)}, \alpha^{(k,l)} d_z^{(k)})$
 if $\hat{x}^{(k+1,l)}$ *is not acceptable* **then**
 Decrease step size $\alpha^{(k,l)}$
 $l \leftarrow l + 1$
until $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$ *is acceptable*
Return $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$

Line-search methods require that the Hessian of the Lagrangian (or a regularization thereof) be positive definite on the nullspace of the Jacobian of the active constraints in order to guarantee that $d_x^{(k)}$ is a descent direction for the objective or merit function.

4.4.2 Trust-region methods

Trust-region methods limit the length of the direction d_x a priori by imposing the trust-region constraint $\|d_x\| \leq \Delta^{(l)}$ for some norm $\|\cdot\|$ in \mathbb{R}^m , where $\Delta^{(l)} > 0$ is the trust-region radius. The step $d_x^{(k,l)}$ is then computed by (approximately) solving the trust-region subproblem. If the trial iterate $\hat{x}^{(k+1,l)} \stackrel{\text{def}}{=} x^{(k)} + d_x^{(k,l)}$ is accepted by the globalization strategy (Algorithm 3), $\Delta^{(l)}$ is increased if the trust region is active at $d_x^{(k,l)}$. Otherwise, $\Delta^{(l)}$ is decreased to a value smaller than $\min(\Delta^{(l)}, \|d_x^{(k,l)}\|)$. Contrary to line-search methods, a positive definite Hessian is not required because directions of negative curvature are bounded by the trust region.

Algorithm 3: Trust-region method.

Input: primal-dual iterate $(x^{(k)}, y^{(k)}, z^{(k)})$
Set inner iteration counter $l \leftarrow 0$
Reset trust-region radius $\Delta^{(l)} \in [\underline{\Delta}, \overline{\Delta}]$
repeat
 $(d_x^{(k,l)}, d_y^{(k,l)}, d_z^{(k,l)}) \leftarrow$ solve trust-region subproblem at $(x^{(k)}, y^{(k)}, z^{(k)})$ with radius $\Delta^{(l)}$
 Assemble trial iterate
 $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)}) \stackrel{\text{def}}{=} (x^{(k)}, y^{(k)}, z^{(k)}) + (d_x^{(k,l)}, d_y^{(k,l)}, d_z^{(k,l)})$
 Reset the multipliers $\hat{z}^{(k+1,l)}$ corresponding to the active trust region
 if $\hat{x}^{(k+1,l)}$ *is acceptable* **then**
 Possibly increase radius $\Delta^{(l)}$
 else
 Decrease radius $\Delta^{(l)}$
 $l \leftarrow l + 1$
until $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$ *is acceptable*
return $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$

5 Uno: a modular solver for unifying nonconvex optimization

We have implemented our unifying framework for nonlinearly constrained nonconvex optimization within Uno, a novel modular solver written in C++17. A generic and flexible code, it supports a broad range of constraint relaxation strategies, subproblems, globalization strategies, and globalization mechanisms that can be combined automatically and on the fly with no programming effort from the user. In addition to the four basic ingredients, an optimization solver requires components that can be implemented independently, such as termination criteria, subproblem solvers, or preprocessing techniques, which results in a large amount of code reuse. The code is packaged in a lightweight library (around 7,400 lines of code for Uno 1.0.0) available as open-source software under the MIT license at <https://github.com/cvanaret/Uno>.

In the following, we list the features of the current version Uno 1.0.0, briefly present its generic architecture, describe how ingredients can be automatically combined, and discuss the current limitations.

5.1 General utilities of Uno 1.0.0

Uno 1.0.0 contains a number of general utilities that are described here.

5.1.1 Interfaces to modeling languages

Uno 1.0.0 is interfaced to the modeling language AMPL [21] through the ASL library [22]. It performs 1st- and 2nd-order automatic differentiation of the objective and constraints of the model. Gradients, Jacobian and Hessian, are stored in sparse data structures (sparse vectors and sparse matrix formats COO and CSC).

5.1.2 Interfaces with subproblem solvers

Interfaces with the following subproblem solvers are available in Uno 1.0.0:

- BQPD [14, 20], a null-space active-set solver for nonconvex QPs, and
- MA57 [13], a direct solver for sparse symmetric indefinite linear systems.

5.1.3 Preprocessing

The preprocessing techniques available in Uno 1.0.0 are the following:

- Feasibility with respect to the linear constraints $Ax = b$ and $x \geq 0$ is enforced at the initial point $x^{(0)}$ by solving a proximal QP:

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} \|d_x\|_2^2 \\ \text{s.t.} \quad & A(x^{(0)} + d_x) = b \\ & x^{(0)} + d_x \geq 0. \end{aligned}$$

Minimizing the quadratic objective $\frac{1}{2} \|d_x\|_2^2$ ensures that the point $x^{(0)} + d_x$ is not too distant from $x^{(0)}$. If the linear constraints are consistent, the point $x^{(0)} + d_x$ satisfies the linear constraints and is taken as the initial point. Otherwise, NLP is infeasible.

- An estimate of the initial multipliers $y^{(0)}$ can be obtained as a least-square solution to the stationarity equation (2a):

$$\begin{pmatrix} I & \nabla c^{(0)} \\ (\nabla c^{(0)})^T & 0 \end{pmatrix} \begin{pmatrix} w \\ y^{(0)} \end{pmatrix} = \begin{pmatrix} \nabla f^{(0)} - z^{(0)} \\ 0 \end{pmatrix},$$

where w is discarded after computation. If the multipliers $y^{(0)}$ exceed a given threshold y_{max} in the modulus (that is, $\|y^{(0)}\|_\infty > y_{max}$), we discard the least-square estimate and set $y^{(0)} = 0$ (similarly to the IPOPT implementation). This situation often arises if a constraint qualification fails to hold at $x^{(0)}$.

- The functions are scaled by coefficients that depend on the magnitudes of the function gradients at the initial iterate (similarly to the IPOPT implementation). The objective f is replaced with $s_f \cdot f$, and the constraint $c_j(x) = 0$ is replaced with $s_{c_j} \cdot c_j(x) = 0$, where

$$s_f = \min \left(1, \frac{s_{\max}}{\|\nabla f(x^{(0)})\|_{\infty}} \right) \in (0, 1]$$

$$s_{c_j} = \min \left(1, \frac{s_{\max}}{\|\nabla c_j(x^{(0)})\|_{\infty}} \right) \in (0, 1]$$

and s_{\max} is typically 100. Scaling helps improve the conditioning of the problem.

5.1.4 Automatic model reformulations

An optimization model may be automatically reformulated as follows:

- A model with nonlinear equality constraints. Slack variables are introduced to turn nonlinear inequality constraints into equality constraints. The corresponding bounds become bounds on the slack variables. This reformulation is required for our implementation of interior-point methods.
- A scaled model whose objective and constraints are scaled by coefficients that depend on the values of the function gradients at the initial point.

5.1.5 Automatic nonlinear reformulations

Constraint relaxation strategies replace the original problem with a (possibly nonsmooth) optimization problem that strategically balances the objective function and the constraint violation. Reformulations in Uno 1.0.0 include the ℓ_1 relaxed problem (see Section 4.1.1) and the ℓ_1 feasibility problem (see Section 4.1.2). Nonsmooth problems resulting from ℓ_1 or ℓ_{∞} relaxations are rewritten as smooth problems in which nonnegative elastic variables capture the positive and negative parts of the constraints, respectively.

5.1.6 Termination criteria

In practice, we cannot hope to drive the error in the first-order conditions to zero: optimization algorithms may terminate at locally infeasible points or at points that fail to satisfy constraint qualifications. We therefore check for termination, rather than for optimality. Uno terminates at the primal-dual iterate (x^*, y^*, z^*, ρ^*) if

- sufficient first-order optimality conditions are approximately satisfied:
 - a **feasible KKT point** (CQ holds) if it satisfies Equations 2a, 2b, 2c, and 2d with $\rho^* > 0$;
 - a **feasible FJ point** (CQ does not hold) if it satisfies Equations 2a, 2b, 2c, and 2d with $\rho^* = 0$;

- an **infeasible stationary point** (a minimum of the constraint violation) if it satisfies Equations 2a and 2c with $\rho = 0$, no primal feasibility $c(x^*) > 0$ and a complementarity condition on the violated constraints that depends on the norm used in the feasibility problem;
- primal feasibility is approximately satisfied and the trust-region radius is close to machine epsilon. This is an indication that the problem is poorly scaled or non-differentiable;
- the first-order optimality conditions cannot be satisfied for the user-defined tolerance ε but are satisfied for a looser tolerance (e.g., 100ε) for a certain number of consecutive iterations (e.g., 15).

5.1.7 Error handling

Uno throws an error if it encounters an IEEE exception at the initial point $x^{(0)}$. Otherwise, it tries to recover from IEEE exceptions during the optimization process by invoking the globalization mechanism in the following way:

- the current trust-region radius is reduced;
- the backtracking line search reduces the tentative step length.

5.1.8 Flexible parameterization

The values of the hyperparameters used by Uno are dynamically loaded from an option file. Values passed as command line arguments take precedence. The solver is therefore fully parameterizable at runtime.

5.2 Generic architecture

The modularity of Uno stems from its generic architecture: each ingredient is implemented independently of the others, which improves readability, promotes code reuse, and makes building blocks less prone to error and easier to maintain than monolithic codes. This results in a modern, flexible, and maintainable framework. The intricate distribution of responsibilities is presented in Figure 4: each ingredient is responsible for computing certain quantities (represented by the list of bullet points under the ingredient) that are passed on to other ingredients. Arrows point to the recipient of each responsibility.

Figure 3 represents Uno’s object-oriented architecture as a Unified Modeling Language (UML)¹ diagram based on inheritance (“is a”) and composition (“has a”). The four ingredients are modeled as abstract classes: they define interfaces, that is, generic actions and behaviors that must be implemented by concrete strategies modeled as subclasses. For example:

¹ <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

- the classes `BacktrackingLineSearch` and `TrustRegionStrategy` inherit the abstract class `GlobalizationMechanism` (inheritance is represented by dashed arrows) and thus must implement the purely virtual member function `compute_acceptable_iterate()`;
- the `GlobalizationMechanism` class possesses a `ConstraintRelaxationStrategy` member (composition is represented by solid diamond lines).

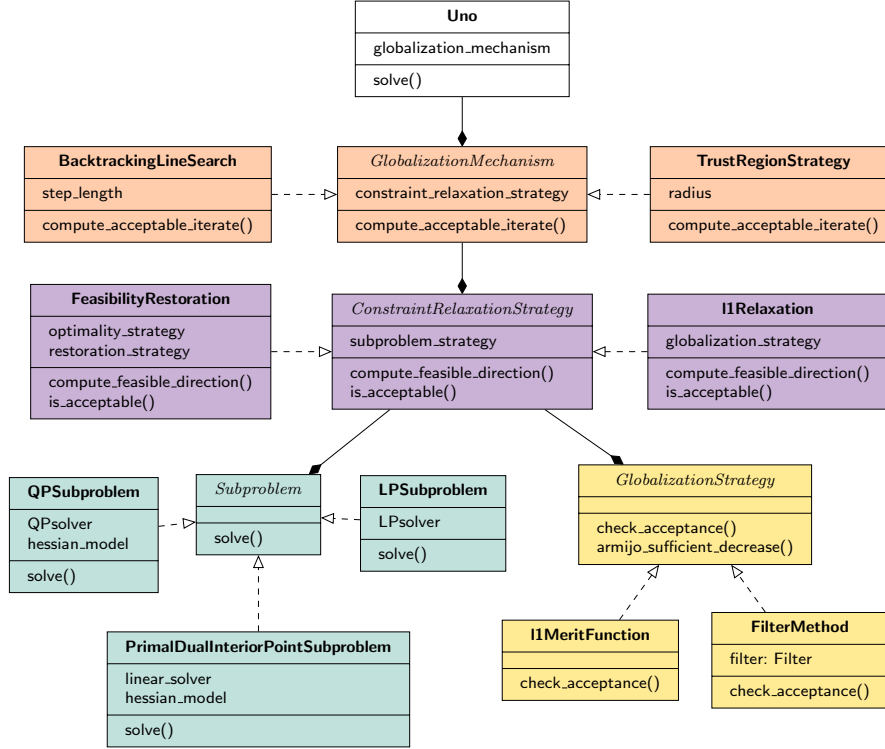


Fig. 3: Uno’s UML diagram: interactions and dependencies between ingredients, using the same color scheme as Figure 1.

The data members and member functions of each class are listed at the top and bottom of the blocks, respectively. For the sake of readability, the return types of the member functions and their arguments are omitted.

5.3 Automatic strategy combinations

Figure 5 shows the strategies implemented in Uno 1.0.0 and illustrates the full potential of our approach: strategies implemented as independent software components are agnostic of the other components as long as they comply

with the defined interfaces. Consequently, the number of possible strategy combinations is the size of the Cartesian product of the four ingredients. Note that all combinations do not necessarily result in sensible algorithms, or even convergent approaches.

The command line syntax to automatically assemble the four ingredients is

```
./uno_ampl -constraint_relaxation_strategy feasibility_restoration
          -subproblem QP -globalization_strategy leyffer_filter_method
          -globalization_mechanism TR ./model.nl
```

Some strategy combinations that correspond to state-of-the-art solvers are available as “presets”: the four ingredients are automatically connected, and the hyperparameters are set to values that can be found in the solvers’ documentations (they will be listed in the Uno user manual). The following presets are available in Uno 1.0.0:

- `filtersqp`: A trust-region restoration filter SQP method à la filterSQP [16]. Second-order correction steps were not implemented.
- `ipopt`: A line-search restoration filter interior-point method à la IPOPT [45]. Second-order correction steps, a proximal term in the feasibility problem, iterative refinement, iterative bound relaxations, non-monotone techniques, and soft feasibility restoration were not implemented.
- `byrd`: A line-search ℓ_1 -merit $S\ell_1$ QP method à la Byrd et al. [7].

The command line syntax of presets is

```
./uno_ampl -preset filtersqp ./model.nl
```

5.4 Current limitations

Uno 1.0.0 implements state-of-the-art strategies that have proven efficient and robust and can be combined automatically thanks to the modular software architecture. Two elementary combinations require additional work and, as is, do not result in sensible algorithms.

- `feasibility restoration` and `ℓ_1 merit function`: the inverse penalty parameter is steered by the ℓ_1 relaxation strategy but not by the feasibility restoration strategy. Consequently, the direction obtained by solving the subproblem may not be a descent direction for the merit function. In that case, Uno issues a warning.
- `interior-point methods` and `trust-region strategies`: using an l_∞ trust region in the definition of the subproblem would introduce additional bound constraints, which would defeat the purpose of interior-point methods. An alternative is to use an iterative linear solver (such as MINRES on the KKT system or CG on the normal equations) with an ℓ_2^2 trust region. At the moment, Uno prohibits this combination.

These limitations will be resolved in later Uno versions.

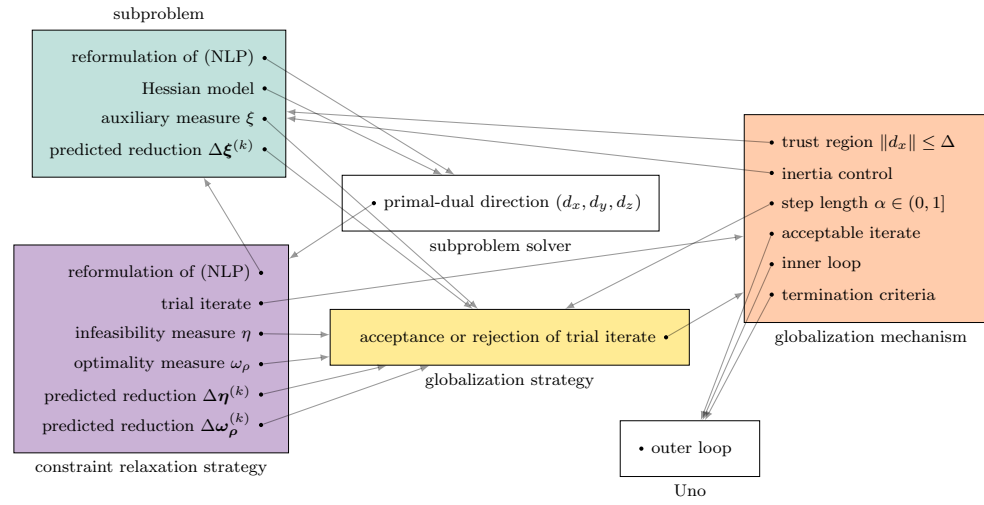


Fig. 4: Responsibilities of each ingredient. Arrows point to the recipient of each responsibility.

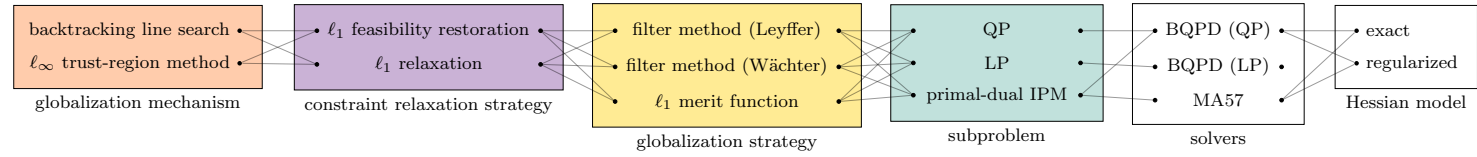


Fig. 5: Uno 1.0.0: hypergraph of strategy combinations.

6 Combining the ingredients

We show below how the four ingredients naturally arise in two popular SQP methods—namely, a trust-region restoration filter SQP method and a line-search ℓ_1 -merit Sl_1 QP method—and a line-search restoration filter interior-point method.

6.1 Trust-region restoration filter SQP

Convergence of SQP filter methods has been proven under mild conditions in [16, 19] in the context of trust-region methods and in [43, 44] in the context of line-search methods. The trust-region optimality QP subproblem about $(x^{(k)}, y^{(k)})$ is defined as

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} d_x^T W_1^{(k)} d_x + 1 (\nabla f^{(k)})^T d_x \\ \text{s.t.} \quad & c^{(k)} + (\nabla c^{(k)})^T d_x = 0 \\ & x^{(k)} + d_x \geq 0 \\ & \|d_x\|_\infty \leq \Delta^{(l)}, \end{aligned} \quad (QP^{(k)}(\Delta^{(l)}))$$

where $\Delta^{(l)} > 0$ is the current trust-region radius and $W_1^{(k)}$ is the Lagrangian Hessian (following the notation introduced in Section 2). If $QP^{(k)}(\Delta^{(l)})$ is infeasible (the linearized constraints are inconsistent), we switch to feasibility restoration and solve a smooth reformulation of the ℓ_1 feasibility problem with elastic variables $u^+ \in \mathbb{R}^m$ and $u^- \in \mathbb{R}^m$:

$$\begin{aligned} \min_{d_x, u^+, u^-} \quad & \frac{1}{2} d_x^T W_0^{(k)} d_x + e^T u^+ + e^T u^- \\ \text{s.t.} \quad & c^{(k)} + (\nabla c^{(k)})^T d_x - u^+ + u^- = 0 \\ & x^{(k)} + d_x \geq 0 \\ & \|d_x\|_\infty \leq \Delta^{(l)} \\ & u^+ \geq 0, u^- \geq 0. \end{aligned} \quad (FQP^{(k)}(\Delta^{(l)}))$$

If the trial iterate $x^{(k)} + d_x$ makes sufficient progress with respect to the filter method, it is accepted. If it was active at the solution of the QP ($\|d_x^*\|_\infty = \Delta^{(l)}$), we enlarge the trust region and start a new iteration. If the trial iterate is rejected, we resolve the trust-region subproblem with a smaller trust-region radius. It can be shown that this mechanism either generates an acceptable iterate or results in an infeasible QP. The complete pseudocode of the method is given in Algorithm 4. Note that in [16] the authors do not solve the feasibility problem with elastic variables but exploit the partition into satisfied and violated linearized constraints provided by the Phase I method of the QP solver BQPD.

6.2 Line-search ℓ_1 -merit $S\ell_1$ QP

We solve the ℓ_1 relaxed problem (3) using an SQP method with a line search, and we build a smooth convex QP subproblem about $(x^{(k)}, y^{(k)})$:

$$\begin{aligned} \min_{d_x, u^+, u^-} \quad & \frac{1}{2} d_x^T (W_\rho^{(k)} + \delta_w I) d_x + \rho (\nabla f^{(k)})^T d_x + e^T u^+ + e^T u^- \\ \text{s.t.} \quad & c^{(k)} + (\nabla c^{(k)})^T d_x - u^+ + u^- = 0 \\ & x^{(k)} + d_x \geq 0 \\ & u^+ \geq 0, u^- \geq 0, \end{aligned} \quad (\ell_1 QP^{(k)})$$

where $\delta_w > 0$ is a regularization coefficient chosen such that $W_\rho^{(k)} + \delta_w I$ is positive definite; this guarantees that the $(\ell_1 QP^{(k)})$ steps are descent directions for the ℓ_1 merit function and the line search does not fail. The complete pseudocode of the method is given in Algorithm 5. It implements the steering rule described in [7]; during the penalty parameter update, the search for a suitable ρ may involve several QP solves. The algorithm makes use of the following error measure that aggregates the dual FJ residuals:

$$\begin{aligned} E_\rho^{(k)}(x, y) \stackrel{\text{def}}{=} & \|\nabla_x \mathcal{L}_\rho(x, y)\|_1 + \sum_{j \in \mathcal{S}^{(k)}} |y_j c_j(x)| + \sum_{j \in \mathcal{V}_+^{(k)}} |(y_j + 1) c_j(x)| \\ & + \sum_{j \in \mathcal{V}_-^{(k)}} |(y_j - 1) c_j(x)|, \end{aligned}$$

where $\mathcal{S}^{(k)}$, $\mathcal{V}_+^{(k)}$, and $\mathcal{V}_-^{(k)}$ are the sets of satisfied ($c(x^{(k)}) = 0$), upper violated ($c(x^{(k)}) > 0$), and lower violated ($c(x^{(k)}) < 0$) constraints evaluated at $x^{(k)}$, respectively.

6.3 Line-search filter restoration interior-point method

We adopt the primal-dual approach described in Section 4.2.3 and solve a smaller, symmetrized version of Problem 5:

$$\begin{pmatrix} W^{(k)} + (X^{(k)})^{-1} Z^{(k)} + \delta_w I & \nabla c^{(k)} \\ (\nabla c^{(k)})^T & -\delta_c I \end{pmatrix} \begin{pmatrix} d_x \\ -d_y \end{pmatrix} = - \begin{pmatrix} 1 \nabla f^{(k)} - \nabla c^{(k)} y^{(k)} - \mu (X^{(k)})^{-1} e \\ c^{(k)} \end{pmatrix}, \quad (IPSP_\mu)$$

where δ_w and δ_c are primal and dual regularization coefficients, respectively. The dual direction for the bound constraints is given by $d_z = (X^{(k)})^{-1}(\mu e - Z^{(k)} d_x) - z^{(k)}$. The fraction-to-boundary rule determines primal and dual step lengths that maintain positivity of x and z :

$$\begin{aligned} \alpha_x^{(k)} & \stackrel{\text{def}}{=} \max\{\alpha \in (0, 1] \mid x^{(k)} + \alpha d_x \geq (1 - \tau) x^{(k)}\} \\ \alpha_z^{(k)} & \stackrel{\text{def}}{=} \max\{\alpha \in (0, 1] \mid z^{(k)} + \alpha d_z \geq (1 - \tau) z^{(k)}\}, \end{aligned} \quad (8)$$

where τ is a parameter close to 1. A filter line search assesses whether the trial iterate makes sufficient progress with respect to the filter method. If so, the trial iterate is accepted; otherwise the line search backtracks and tries again with a smaller step length. If the step length ultimately falls below a given threshold (e.g., 10^{-7}), we switch to feasibility restoration and solve the ℓ_1 feasibility problem with elastic variables.

Note that by construction the filter entries depend on the barrier parameter μ through the auxiliary measure ξ . Consequently, the filter must be flushed whenever μ is updated. The complete pseudocode of the method is given in Algorithm 6. An alternative that we plan to explore in the future is to update the filter whenever the barrier parameter is updated.

7 Numerical results

We compare Uno 1.0.0 against state-of-the-art solvers filterSQP (20010817) (with the QP solver BQPD), IPOPT 3.14.11 (with the linear solver MUMPS 5.5.1), SNOPT 7.5-1.2, MINOS 5.51, LANCELOT, LOQO 7.03, and CONOPT 3.17A on 429 small test problems of the CUTEst benchmark [24]. The log files of all solvers are available at the following repository:
https://github.com/cvanaret/nonconvex_solver_comparison.

7.1 Validation of Uno presets

In this section we demonstrate that the Uno presets closely mimic the corresponding solvers and are competitive against all state-of-the-art solvers. Table 6 Table 7 and Table 8 summarize the number of objective evaluations required by each solver for linear, quadratic, and nonlinear instances, respectively. The lowest count is shown in bold. IPOPT terminated with the status “EXIT: Problem has too few degrees of freedom” on the instances argauss and lewispol; this is interpreted as failure.

Figure 6 portrays a performance profile [11] of all solvers: the y axis is the fraction of solved problems, and the x axis represents the relative budget of objective evaluations compared with the (virtual) best solver for each instance. The higher and more to the left, the better. This performance profile conveys three important messages:

1. the Uno presets mimic the corresponding state-of-the-art solvers well;
2. the Uno presets `filtersqp` and `ipopt` outperform most state-of-the-art solvers;
3. 12 instances (`aljazzaf`, `hatfldf`, `himmelbd`, `hs085`, `hs109`, `hs114`, `launch`, `polak6`, `powellsq`, `snake`, `spiral`, and `vanderm4`) are solved by IPOPT but not by the Uno `ipopt` preset, most likely because of the IPOPT features that were not implemented (see Section 5.3).

These results validate the implementation of off-the-shelf strategies within Uno and demonstrate that Uno is a strong new contender on the nonlinear optimization scene.

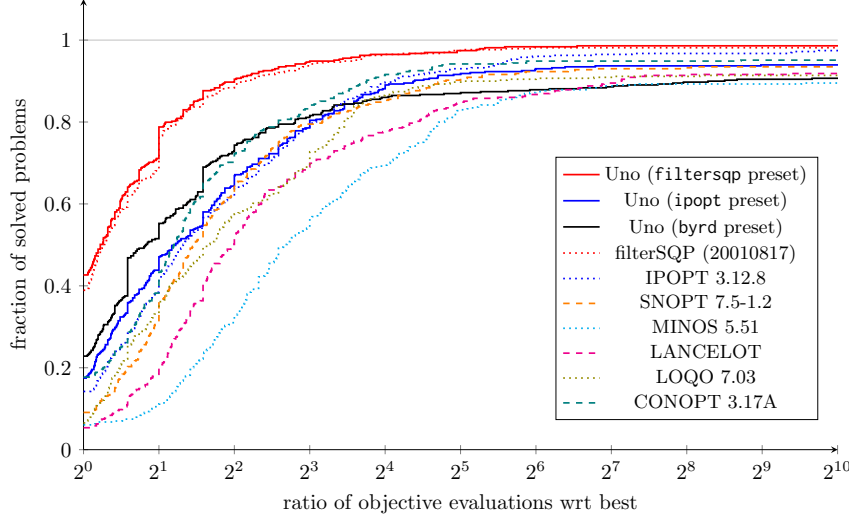


Fig. 6: Performance profile (number of objective evaluations) of state-of-the-art solvers and Uno presets on 429 small CUTEst problems.

7.2 Performance of a novel Uno combination: a trust-region ℓ_1 -merit Sl_1QP method

We now assess the performance of a novel Uno combination, a trust-region ℓ_1 -merit Sl_1QP method, against the state-of-the-art solvers on the 429 small CUTEst instances. The method builds on the byrd preset and is obtained via the following command line:

```
./uno_ampl -preset byrd -globalization_mechanism TR ./model.nl
```

where `-globalization_mechanism TR` overwrites byrd's line-search method with a trust-region method.

Figure 7 portrays the performance profile of the state-of-the-art solvers, the new Uno combination, and the Uno byrd preset (for comparison). It shows that the trust-region Sl_1QP method (byrd preset + TR) outperforms the original line-search Sl_1QP method (byrd preset) in terms of objective evaluations and robustness, as well as most state-of-the-art solvers except for filterSQP. In fact, it turns out to be as robust as filterSQP (both solvers solve 428 of the 429 instances), albeit requiring more objective evaluations.

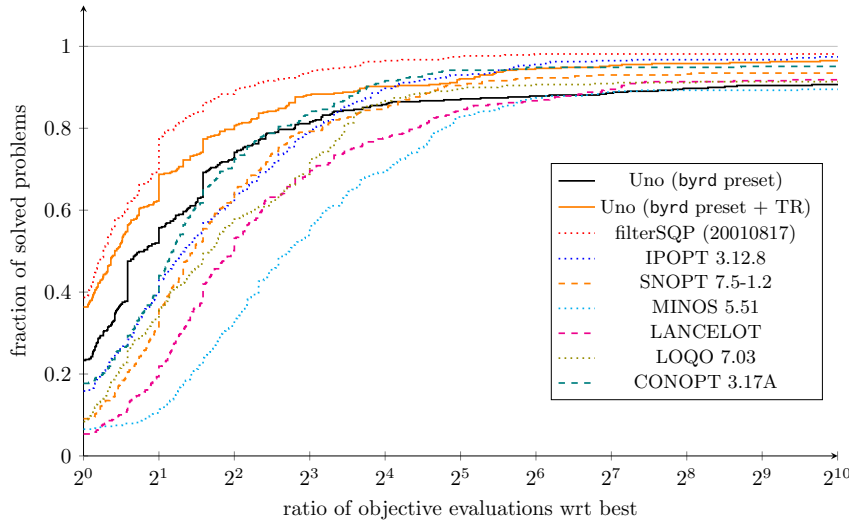


Fig. 7: Performance profile (number of objective evaluations) of state-of-the-art solvers and a novel Uno combination, a trust-region ℓ_1 -merit $S\ell_1$ QP method, on 429 small CUTEst problems.

These results demonstrate the modularity and versatility of Uno: with a single command line and with no programming effort from the user, Uno can generate combinations of strategies on the fly that perform well. In particular, individual ingredients can be replaced in existing presets effortlessly.

8 Conclusion and future developments

We have introduced an abstract framework for unifying nonlinearly constrained nonconvex optimization based on four ingredients common to most methods: a constraint relaxation strategy, a subproblem, a globalization strategy, and a globalization mechanism. We have shown that our abstract framework provides a unified view of most state-of-the-art solvers, as well as common notation and abstractions for the unified description of well-known optimization strategies.

We then presented Uno, a C++ implementation of the abstract framework. A modular solver, Uno provides efficient implementations of off-the-shelf strategies and facilitates the development of new algorithmic ideas by making building blocks and abstractions readily available. In particular, newly developed strategies (e.g., a new type of line search) can be immediately deployed and tested within a wide range of strategy combinations. Entirely novel strategy combinations (such as a trust-region ℓ_1 -merit $S\ell_1$ QP method) can be generated on the fly with no programming effort and tested against more traditional approaches for a given instance. We believe that Uno has the potential

to serve as an experimentation laboratory for the optimization community and accelerate research in nonconvex optimization.

Future releases of Uno will include the following features:

- quasi-Newton methods: L-BFGS and DFP;
- iterative linear solvers: MINRES [37];
- equality-constrained subproblems: SLPEQP;
- globalization strategies: funnel method;
- globalization mechanisms: parallel line search [40];
- interfaces to subproblem solvers: HiGHS and ProxQP;
- interfaces to programming languages: Python, Julia, and Matlab.

References

1. NAG library manual, mark 26. Tech. rep., The Numerical Algorithms Group, https://support.nag.com/numeric/cl/nagdoc_cl26/pdf/frontmatter/manconts.pdf (2017)
2. FICO Xpress nonlinear manual. Tech. rep., FICO, <https://www.fico.com/fico-xpress-optimization/docs/latest/solver/nonlinear/HTML> (2023)
3. Bochkkanov, S., Bystritsky, V.: ALGLIB – a cross-platform numerical analysis and data processing library. ALGLIB Project. Novgorod, Russia (2011)
4. Burke, J.V., Curtis, F.E., Wang, H.: A sequential quadratic optimization algorithm with rapid infeasibility detection. *SIAM Journal on Optimization* **24**(2), 839–872 (2014)
5. Büskens, C., Wassel, D.: The ESA NLP solver WORHP. *Modeling and optimization in space engineering* pp. 85–110 (2013)
6. Byrd, R., Nocedal, J., Waltz, R.: KNITRO: An integrated package for nonlinear optimization. In: G. di Pillo, M. Roma (eds.) *Large-Scale Nonlinear Optimization*, pp. 35–59. Springer-Verlag, New York (2006)
7. Byrd, R.H., Curtis, F.E., Nocedal, J.: Infeasibility detection and SQP methods for nonlinear optimization. *SIAM Journal on Optimization* **20**(5), 2281–2299 (2010)
8. Byrd, R.H., Nocedal, J., Waltz, R.A.: Steering exact penalty methods for nonlinear programming. *Optimization Methods and Software* **23**(2), 197–213 (2008)
9. Chin, C., Fletcher, R.: On the global convergence of an SLP-filter algorithm that takes EQP steps. *Mathematical Programming* **96**(1), 161–177 (2003)
10. Conn, A., Gould, N., Toint, P.: LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A). Springer Verlag, Heidelberg, New York (1992)
11. Dolan, E.D., Moré, J.: Benchmarking optimization software with performance profiles. *Mathematical Programming* **91**(2), 201–213 (2002)
12. Drud, A.S.: CONOPT – a large-scale GRG code. *ORSA Journal on Computing* **6**(2), 207–216 (1994)
13. Duff, I.S.: MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)* **30**(2), 118–144 (2004)
14. Fletcher, R.: Stable reduced Hessian updates for indefinite quadratic programming. *Mathematical Programming* **87**(2), 251–264 (2000)
15. Fletcher, R.: *Practical methods of optimization*. John Wiley & Sons (2013)
16. Fletcher, R., Gould, N.I., Leyffer, S., Toint, P.L., Wächter, A.: Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization* **13**(3), 635–659 (2002)
17. Fletcher, R., Leyffer, S.: User manual for filterSQP. Numerical Analysis Report NA/181, University of Dundee (1998)
18. Fletcher, R., Leyffer, S.: Solving mathematical program with complementarity constraints as nonlinear programs. *Optimization Methods and Software* **19**(1), 15–40 (2004)
19. Fletcher, R., Leyffer, S., Toint, P.: On the global convergence of a filter-SQP algorithm. *SIAM J. Optimization* **13**(1), 44–59 (2002)
20. Fletcher, R.: An optimal positive definite update for sparse Hessian matrices. *SIAM Journal on Optimization* **5**(1), 192–218 (1995)

21. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modelling Language for Mathematical Programming, 2nd edn. Books/Cole—Thomson Learning (2003)
22. Gay, D.M.: Hooking your solver to AMPL. Tech. rep., Technical Report 93-10, AT&T Bell Laboratories, Murray Hill, NJ, 1993, revised (1997)
23. Gill, P., Murray, W., Saunders, M.: SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization* **12**(4), 979–1006 (2002)
24. Gould, N.I., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications* **60**(3), 545–557 (2015)
25. Gould, N.M., Leyffer, S.: An introduction to algorithms for nonlinear optimization. In: J. Blowey, A. Craig, T. Shardlow (eds.) *Frontiers in Numerical Analysis*, pp. 109–197. Springer Verlag, Berlin (2003)
26. Han, S.: A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications* **22**(3), 297–309 (1977)
27. Kraft, D.: A software package for sequential quadratic programming. *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt* (1988)
28. Leyffer, S.: Complementarity constraints as nonlinear equations: Theory and numerical experience. In: S. Dempe, V. Kalashnikov (eds.) *Optimization and Multivalued Mappings*, pp. 169–208. Springer (2006)
29. Leyffer, S., Lopez-Calva, G., Nocedal, J.: Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization* **17**(1), 52–77 (2006)
30. Leyffer, S., Mahajan, A.: *Software for nonlinearly constrained optimization*. Wiley Encyclopedia of Operations Research and Management Science (2010)
31. Leyffer, S., Menickelly, M., Munson, T., Vanaret, C., Wild, S.M.: A survey of nonlinear robust optimization. *INFOR: Information Systems and Operational Research* (2020). DOI 10.1080/03155986.2020.1730676
32. Luo, Z.Q., Pang, J.S., Ralph, D.: *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, Cambridge, UK (1996)
33. Mangasarian, O.L., Fromovitz, S.: The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications* **17**(1), 37–47 (1967)
34. Murtagh, B., Saunders, M.: MINOS 5.4 user’s guide. Report SOL 83-20R, Department of Operations Research, Stanford University (1993)
35. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer (2006)
36. Outrata, J., Kocvara, M., Zowe, J.: *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*. Kluwer Academic Publishers, Dordrecht (1998)
37. Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* **12**(4), 617–629 (1975)
38. Powell, M.J.D.: A fast algorithm for nonlinearly constrained optimization calculations. In: G. Watson (ed.) *Numerical Analysis, 1977*, pp. 144–157. Springer-Verlag, Berlin (1978)
39. Powell, M.J.D., Yuan, Y.: A recursive quadratic programming algorithm that uses differentiable exact penalty functions. *Mathematical Programming* **35**, 165–278 (1986)
40. Schittkowski, K.: NLPQLP: A new Fortran implementation of a sequential quadratic programming algorithm for parallel computing. Report, Department of Mathematics, University of Bayreuth (2001)
41. Vanaret, C., Leyffer, S.: Argonot: An open-source software framework for nonlinear optimization (2018). 23rd International Symposium on Mathematical Programming (ISMP 2018)
42. Vanderbei, R.J.: LOQO: An interior point code for quadratic programming. *Optimization Methods and Software* **11**(1-4), 451–484 (1999)
43. Wächter, A., Biegler, L.: Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization* **16**(1), 32–48 (2005)
44. Wächter, A., Biegler, L.: Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization* **16**(1), 1–31 (2005)
45. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming* **106**(1), 25–57 (2006)
46. Wilson, R.: A simplicial algorithm for concave programming. Ph.D. thesis, Harvard University Graduate School of Business Administration (1963)

A Statistics

Table 6: Number of objective evaluations on a subset of linear CUTEst problems.

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
booth	2	2	3	2	2	2	1	3	7	1
degenlpa	2	32	3	2	29	26	15	25	29	1
degenlpb	2	35	3	2	41	26	23	45	30	1
extrasm	2	6	3	2	6	1	1	3	11	1
goffin	3	9	3	3	10	25	25	10	13	1
himmelba	2	2	3	2	2	2	1	3	7	1
linspanh	2	19	4	2	54	5	14	13	12	1
makela4	3	8	3	3	8	1	1	20	12	1
model	2	15	3	2	15	23	34	33	15	1
res	3	9	2	2	10	1	5	1	12	1
simpllpa	2	13	3	2	14	3	3	5	12	1
simpllpb	2	11	3	2	11	1	1	4	13	1
supersim	2	7	3	2	2	1	1	7	11	1
zangwil3	2	2	3	2	2	3	2	3	8	1

Table 7: Number of objective evaluations on a subset of quadratic CUTEst problems.

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
3pk	7	12	2	7	12	41	209	47	20	24
arglinb	2	3	3	2	3	1	8	2	3	20
arglinc	2	3	3	2	3	1	8	2	3	18
avgasa	2	10	3	2	10	9	19	12	11	16
avgasb	2	13	3	2	13	9	15	11	16	19
biggsc4	2	26	5	2	35	13	16	15	21	4
bqp1var	2	6	2	2	6	1	5	3	10	2
bqpgabim	2	13	2	2	21	36	107	6	14	17
bqpgasim	2	13	2	2	21	40	121	6	14	15
bt3	2	2	3	2	2	5	12	6	3	9
deconvb	42	–	32	30	–	103	30	31	45	23
dixon3dq	2	2	2	2	2	10	43	4	3	9
dual1	2	17	3	2	17	221	–	9	22	29
dual2	2	15	3	2	14	116	–	9	19	27
dual4	2	14	3	2	14	31	–	8	21	23
dualc1	2	28	3	2	30	7	24	17	26	14
dualc2	2	26	3	2	28	5	14	19	21	17
dualc5	2	11	3	2	11	7	24	8	16	15
dualc8	2	13	3	2	13	9	16	24	19	25
fcu	4	2	4	4	2	19	48	13	3	11
genhs28	2	2	3	2	2	11	12	5	3	9
hatfldc	5	6	5	5	6	14	19	9	9	15
hatfldh	2	14	3	2	19	3	8	13	16	5
hilberta	2	2	2	2	2	2	12	60	3	9
hilbertb	2	2	2	2	2	50	–	5	3	10
hs003	2	6	31	2	5	2	12	16	11	5
hs021	2	7	3	2	9	1	8	3	12	12
hs028	2	2	3	2	2	4	12	4	3	8
hs035	2	8	3	2	8	5	12	8	10	12
hs044	2	20	10	2	20	2	10	11	16	5
hs048	2	2	3	2	2	6	16	3	3	9

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
hs051	2	2	3	2	2	6	12	10	3	7
hs052	2	2	3	2	2	5	11	6	3	9
hs053	2	7	3	2	7	2	12	6	12	15
hs054	2	8	3	2	8	5	16	9	12	12
hs076	2	8	3	2	8	4	13	9	11	16
hs118	3	12	4	3	12	21	41	19	17	4
hs21mod	2	16	3	2	17	1	8	3	20	12
hs268	2	19	4	2	17	6	36	27	27	19
hs35mod	2	16	3	2	16	1	6	3	16	6
hs3mod	2	6	3	2	6	5	13	4	12	10
hs44new	2	14	6	2	14	4	10	10	21	4
lotschd	3	15	4	3	15	8	5	9	19	7
lsqfit	2	8	3	2	8	3	11	7	12	10
maratosb	10	33	20	13	33	7	10	8	21	14
nasty	2	2	2	2	2	1	6	5	3	10
obstclal	2	11	2	2	15	37	90	7	23	15
obstclbl	2	12	2	2	13	44	87	3	13	47
obstclbu	2	13	2	2	13	36	74	2	15	26
oslbqp	2	15	2	2	15	6	11	3	19	2
palmer1c	7	2	2	7	2	8	61	145	–	–
palmer1d	6	2	2	6	2	7	50	34	–	–
palmer2c	5	2	2	5	2	8	62	298	–	–
palmer3c	5	2	2	5	2	8	61	206	–	–
palmer4c	6	2	2	6	2	8	61	176	–	–
palmer5c	4	2	2	4	2	6	26	2	–	–
palmer5d	5	2	2	5	2	4	25	2	–	–
palmer6c	6	2	2	6	2	8	60	159	–	–
palmer7c	8	2	2	8	2	8	64	189	–	–
palmer8c	7	2	2	7	2	8	61	152	–	–
portfl1	2	10	3	2	10	12	48	20	17	20
portfl2	2	9	3	2	9	12	50	14	17	23
portfl3	2	11	3	2	11	13	52	14	17	32
portfl4	2	10	3	2	10	11	49	19	16	19
portfl6	2	9	3	2	9	11	48	20	16	31
qudlin	2	25	7	2	26	11	18	2	20	3
sim2bqp	2	8	2	2	8	2	9	3	14	5
simbqp	2	8	2	2	8	2	10	2	13	4
tame	2	6	3	2	6	1	8	2	9	6
tointqor	2	2	2	2	2	50	–	8	3	12
zangwil2	2	2	2	2	2	2	11	4	3	3
zecevic2	2	9	3	2	9	2	9	7	11	7

Table 8: Number of objective evaluations on a subset of nonlinear CUTEst problems.

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
aircrfta	4	4	4	4	4	–	1	10	6	2
aircftb	21	15	15	21	19	58	65	27	21	32
airport	13	16	10	13	16	58	527	69	19	101
aljazaf	12	–	37	15	82	145	64	24	79	9
allinitc	29	44	16	24	44	105	55	76	–	17
allinit	11	15	10	11	19	17	29	13	18	16
allinitu	12	15	10	12	15	14	20	15	11	17
alsotame	5	9	5	5	9	6	11	11	11	4
argauss	1	–	3	1	–	7	–	–	–	3
avion2	23	78	3	19	143	19	17	787	65	25
bard	11	9	19	11	9	23	36	15	19	17

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
batch	9	31	8	9	34	33	379	–	64	32
beale	19	11	12	10	19	15	24	21	10	16
biggs3	11	16	18	11	28	24	31	40	12	15
biggs5	54	24	94	50	36	107	38	64	32	48
biggs6	74	38	67	83	50	120	119	103	59	139
box2	9	9	12	9	9	10	10	20	10	10
box3	8	11	9	8	15	24	15	31	11	13
brkmcc	4	4	4	4	4	10	13	7	8	10
brownal	8	8	8	8	8	21	77	24	10	17
brownbs	48	8	10	48	8	32	35	7	49	1
brownden	9	9	9	9	9	40	41	9	57	15
bt10	7	8	7	7	7	1	1	20	11	3
bt11	7	9	8	7	9	12	56	22	11	21
bt12	5	5	5	5	5	10	59	11	11	11
bt13	58	25	24	48	25	34	69	–	21	590
bt1	2	15	13	1	15	12	16	19	24	1
bt2	13	13	13	13	13	18	385	36	18	20
bt4	8	11	80	11	10	10	47	25	9	21
bt5	9	8	27	9	8	11	172	20	8	10
bt6	11	11	31	12	18	14	117	25	12	28
bt7	17	26	–	19	30	36	85	49	18	8
bt8	12	29	32	12	52	14	21	30	–	9
bt9	19	14	40	23	14	19	34	22	14	17
byrdsphr	8	28	–	11	19	59	35	43	11	17
camel6	9	11	7	8	11	19	24	8	12	19
cantilvr	13	12	18	16	12	23	69	27	16	45
catena	12	7	16	13	7	87	125	56	26	80
cb2	7	9	7	7	9	7	31	18	11	14
cb3	7	10	7	7	10	1	26	18	11	3
chaconn1	5	7	5	5	7	9	22	12	11	10
chaconn2	5	7	5	5	7	1	20	11	11	4
chebyqad	77	127	31	50	168	–	–	62	12	87
chnrosnb	56	60	58	59	92	170	–	68	58	101
cliff	28	24	28	28	24	28	46	28	32	32
cluster	10	10	9	10	10	–	3	45	11	3
concon	7	10	6	5	10	14	–	676	–	40
coniglmz	5	33	–	4	33	10	16	30	36	3
coolhans	3	10	–	3	10	–	4	281	25	2
core1	6	–	–	6	105	44	86	–	52	5
coshfun	86	–	–	303	1039	273	1091	154	24	140
cresc4	84	189	–	52	269	93	864	–	111	47
csfi1	20	12	1737	18	12	36	–	155	16	7
csfi2	13	331	–	8	86	60	–	180	17	3
cube	39	38	38	41	58	42	66	52	41	36
dallass	18	23	77	56	29	109	140	–	–	–
deconvc	24	73	12	58	99	81	87	43	31	30
deconvu	–	454	150	971	687	152	30	69	76	86
demymalo	8	12	16	8	12	8	34	28	15	15
denschna	7	7	7	7	7	12	23	13	9	10
denschnb	10	21	20	10	25	10	17	11	9	10
denschnb	11	11	11	11	11	21	30	13	14	13
denschnb	41	27	35	43	27	77	111	65	44	64
denschnb	11	17	20	11	25	44	34	16	15	34
denschnf	7	7	7	7	7	12	21	8	11	12
dipigri	12	18	19	13	22	23	129	63	12	30
disc2	5	48	217	25	48	800	–	–	28	3
discs	–	–	1088	40	186	–	–	422	59	3
dixchlng	10	11	153	12	11	31	–	44	26	43
djtl	31	3	122	29	861	–	–	100	132	1
dnieper	4	31	594	4	31	13	36	75	25	6

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
egl	8	8	15	8	8	9	14	9	10	13
eigencco	20	13	21	29	14	34	158	17	22	26
eigmaxc	5	9	13	7	7	18	—	21	14	3
eigminc	5	12	10	7	8	21	182	11	13	4
engval2	19	29	19	20	33	34	66	30	28	33
errinros	53	42	41	53	70	267	—	76	60	90
expfita	13	34	19	13	31	24	28	54	23	14
expfit	13	9	9	13	9	18	28	11	13	19
extrosnb	2	2	2	2	1	2	5	1	1	1
fletcher	2	26	3	1	28	2	—	28	14	19
genhumps	139	212	165	188	321	77	169	134	146	71
gigomez1	8	18	17	8	19	9	51	33	16	14
gottfr	14	8	8	13	9	—	1	35	12	3
gridnetg	4	16	5	4	11	22	42	21	12	21
gridneth	5	9	5	5	7	36	114	20	12	12
gridneti	5	14	5	5	14	47	111	22	16	24
growthls	101	104	—	106	171	184	258	178	122	212
growth	101	109	—	106	171	187	258	178	143	206
gulf	26	28	25	26	44	66	695	63	26	32
hadamals	13	117	70	13	128	19	266	20	13	38
haifas	12	10	58	13	10	28	59	27	13	23
hairy	45	71	84	84	96	35	59	102	64	92
haldmads	25	249	—	41	23	71	94	45	31	12
hart6	15	10	9	11	14	16	36	9	23	20
hatflda	10	11	24	15	11	30	44	47	9	114
hatfldb	9	11	21	11	11	28	32	28	11	138
hatfldd	31	23	22	23	27	29	48	66	25	37
hatflde	36	28	36	26	32	31	63	57	29	31
hatfldf	8	—	—	15	1335	—	2	113	13	3
hatfldg	5	20	18	15	20	—	2	29	16	3
heart6ls	2596	1109	1970	—	1588	—	—	—	—	3067
heart6	5	1516	—	16	—	—	49	—	327	3
heart8ls	248	133	105	217	189	—	474	238	108	369
heart8	5	37	—	12	40	—	—	359	39	3
helix	18	17	21	19	25	28	53	18	13	21
himmelbb	62	12	12	25	12	8	20	11	16	32
himmelbc	5	8	7	8	9	—	1	11	8	3
himmelbd	4	—	—	4	79	12	—	—	—	3
himmelbe	2	3	4	2	3	—	1	8	8	2
himmelbf	8	11	17	8	11	53	47	29	24	15
himmelbg	10	10	6	10	14	12	17	16	7	13
himmelbh	8	20	20	8	24	10	10	7	9	10
himmelbk	6	19	573	6	19	82	109	206	23	11
himmelp1	9	12	10	9	12	19	20	29	11	21
himmelp2	9	21	10	9	19	32	151	275	20	18
himmelp3	5	9	5	5	13	8	119	870	17	9
himmelp4	5	11	6	5	25	8	115	737	17	9
himmelp5	12	26	16	12	543	44	75	273	90	23
himmelp6	2	11	2	2	12	2	5	2	45	1
hong	5	13	8	5	13	4	14	6	20	4
hs001	33	33	34	36	53	48	9	41	35	36
hs002	9	13	10	9	17	15	12	7	21	10
hs004	3	6	3	3	6	4	6	2	8	2
hs005	8	9	11	11	9	9	13	9	10	14
hs006	3	8	12	3	7	9	90	63	10	4
hs007	12	51	11	13	28	30	64	26	12	13
hs008	6	6	6	6	6	—	1	13	8	3
hs009	5	5	5	5	6	10	11	22	9	8
hs010	10	13	26	10	13	20	22	18	15	19
hs011	6	9	6	6	9	15	46	16	12	12

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
hs012	8	9	9	8	9	11	158	26	11	16
hs013	25	1083	22	34	79	17	53	60	—	19
hs014	6	8	6	6	8	10	9	13	11	3
hs015	4	21	6	7	21	11	85	47	31	3
hs016	5	10	6	5	23	5	9	19	18	3
hs017	8	20	9	8	18	19	11	20	30	6
hs018	7	14	7	7	27	32	93	117	15	16
hs019	7	16	6	7	16	9	56	45	18	9
hs020	5	8	5	5	7	5	8	23	24	3
hs022	2	7	5	2	7	7	47	10	9	3
hs023	7	11	7	7	12	7	53	51	18	6
hs024	3	11	16	3	13	8	8	14	13	3
hs025	31	40	2	27	44	2	5	1	20	1
hs026	18	26	19	18	26	27	76	41	14	33
hs027	21	125	1042	8	143	21	136	31	16	43
hs029	8	9	25	8	9	14	173	18	10	13
hs030	2	24	3	2	26	5	29	8	9	20
hs031	6	8	6	6	8	11	28	12	17	20
hs032	2	16	3	2	20	5	14	7	24	3
hs033	5	11	16	5	16	9	38	9	11	3
hs034	8	10	14	8	10	5	9	21	15	16
hs036	3	13	7	3	13	10	8	7	20	3
hs037	6	13	81	6	13	10	16	13	11	15
hs038	53	50	51	54	78	101	88	56	13	94
hs039	19	14	40	23	14	19	34	22	14	17
hs040	5	4	19	5	4	9	21	11	8	11
hs041	2	9	44	2	12	7	5	7	16	14
hs042	6	7	6	6	7	10	21	13	9	13
hs043	9	10	8	11	10	11	102	25	12	27
hs045	2	24	2	2	48	2	5	1	25	1
hs046	19	20	18	19	20	32	121	28	18	37
hs047	21	25	19	21	21	28	125	29	24	31
hs049	17	20	19	17	20	34	61	38	21	21
hs050	9	10	10	9	10	20	24	11	16	14
hs055	2	8	3	2	4	2	5	7	11	1
hs056	15	35	72	19	40	52	52	12	12	27
hs057	5	24	26	5	28	—	28	2	14	28
hs059	11	182	14	11	72	22	127	340	27	22
hs060	7	8	7	7	8	13	120	18	9	23
hs061	1	30	7	1	10	39	70	19	10	2
hs062	8	9	7	10	9	16	18	37	13	19
hs063	8	8	32	1	8	18	119	21	8	19
hs064	12	18	—	13	18	28	98	38	26	20
hs065	5	114	5	5	91	11	319	42	19	22
hs066	9	7	9	14	8	8	8	11	15	15
hs067	12	12	—	12	12	32	60	287	17	19
hs070	40	12	25	42	36	34	66	39	23	31
hs071	6	9	48	6	9	8	55	16	13	14
hs072	15	17	210	15	17	27	52	65	28	20
hs073	4	10	3	4	9	11	8	18	21	5
hs074	6	10	26	6	10	15	27	13	16	15
hs075	5	10	9	5	10	12	18	110	18	6
hs077	11	11	12	14	13	16	123	27	12	28
hs078	5	5	36	5	5	7	57	12	8	13
hs079	5	5	5	5	5	12	54	14	7	24
hs080	8	7	8	8	7	9	48	13	10	17
hs081	29	8	—	38	8	11	55	17	16	17
hs083	5	17	4	5	15	8	11	16	14	6
hs084	11	12	—	6	12	58	45	47	44	4
hs085	—	—	—	—	127	—	—	—	63	7

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
hs086	5	13	4	5	11	16	13	15	13	19
hs087	7	18	33	7	18	16	23	32	25	11
hs088	21	27	12	19	18	59	66	56	27	17
hs089	23	28	15	31	38	85	197	61	30	19
hs090	74	22	–	2	28	55	92	58	29	54
hs091	51	15	–	337	15	73	215	62	28	12
hs092	40	22	33	2	25	56	110	58	22	103
hs093	3	11	1619	2	10	33	–	–	13	22
hs095	3	16	3	3	18	1	1	24	17	4
hs096	3	21	3	3	24	1	1	23	22	4
hs097	7	24	69	7	24	13	63	19	19	9
hs098	7	21	69	7	21	13	47	19	93	9
hs099	12	6	–	9	7	19	60	–	22	29
hs100lnp	12	21	19	14	21	33	133	32	12	36
hs100mod	12	21	28	14	27	32	123	137	15	32
hs100	12	18	19	13	22	23	129	63	12	30
hs101	22	98	23	34	273	530	–	–	64	49
hs102	17	31	18	42	36	238	980	–	154	46
hs103	37	56	14	28	64	177	1418	–	88	38
hs104	17	9	36	23	11	29	85	–	14	3
hs105	9	24	80	9	31	89	114	–	17	24
hs106	16	15	66	17	15	34	–	–	27	33
hs107	6	11	8	6	12	14	20	26	35	21
hs108	25	17	5286	36	17	152	164	43	20	32
hs109	6	–	59	7	44	349	–	–	45	13
hs110	5	7	5	5	7	11	43	5	8	15
hs111lnp	40	16	42	31	16	64	388	57	17	30
hs111	40	16	42	31	16	70	388	46	15	29
hs112	12	18	12	12	18	35	92	47	19	58
hs113	6	12	6	6	12	28	146	97	17	30
hs114	1	–	–	1	73	9	–	664	–	4
hs116	12	27	–	14	26	75	52	–	24	25
hs117	6	30	8	6	23	20	157	66	19	46
hs119	7	14	8	7	15	22	29	28	29	18
hs99exp	12	25	–	12	30	42	212	–	256	4
hubfit	2	8	3	2	9	8	11	8	–	–
humps	114	186	260	–	571	257	193	–	307	259
hypcir	6	6	6	8	8	–	1	10	8	3
jensmp	11	10	11	11	10	36	55	10	14	13
kiwcresc	11	11	45	11	11	13	30	23	14	17
kowosb	18	15	17	18	23	33	39	24	11	26
lakes	–	74	–	63	20	39	–	–	288	1
launch	1	–	–	1	673	246	–	–	–	3
lewispol	1	–	5	1	–	6	–	–	–	1
loadbal	8	17	15	8	18	58	130	62	23	26
loghair	90	–	224	–	–	249	402	–	64	150
logros	50	323	35	50	358	109	147	66	398	84
lootsma	5	11	16	5	16	9	38	9	12	3
lsnnodoc	7	13	7	7	15	8	7	11	21	3
madsen	14	23	12	25	25	14	28	26	26	28
makela1	12	19	17	15	19	8	27	19	15	20
makela2	5	8	14	5	8	16	21	40	12	7
makela3	22	17	29	25	17	288	96	125	18	15
maratos	10	5	19	10	5	9	20	9	7	11
matrix2	12	21	21	12	21	14	65	13	26	161
maxlika	9	24	80	9	31	89	114	–	17	24
mconcon	7	10	6	5	10	14	–	676	–	40
mdhole	3	62	51	56	106	70	116	68	98	6
methanb8	47	9	24	47	9	306	175	221	155	28
methanl8	105	50	164	96	82	499	669	640	80	84

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
mexhat	10	5	5	10	5	37	18	4	7	8
meyer3	255	525	—	280	494	—	775	559	522	716
mifflin1	10	7	18	23	7	10	15	18	9	8
mifflin2	10	16	20	10	16	14	53	50	13	13
minmaxbd	22	114	20	9	78	115	160	592	31	38
minmaxrb	3	11	29	3	11	4	34	81	14	5
minsurf	14	13	13	10	21	24	196	15	13	15
mistake	22	15	9	18	15	19	159	30	15	36
mwright	9	11	18	12	11	12	37	19	11	19
nonmsqrt	786	—	228	716	—	—	1490	167	—	—
nuffield ²	5	7	5	5	7	9	15	15	10	17
odfits	7	11	10	7	11	17	28	49	15	16
optcntrl	4	190	3	4	134	5	15	352	58	6
optmass	9	25	812	18	23	2	331	—	15	27
optprloc	16	19	32	6	19	12	685	438	23	19
orthregb	2	3	3	2	3	9	262	64	8	2
orthrege	2896	70	—	180	77	31	857	795	482	13038
osbornea	—	92	—	—	152	120	127	57	—	52
osborneb	20	21	24	20	25	82	127	45	19	42
palmer1a	42	47	46	51	71	205	—	102	113	200
palmer1b	21	22	17	21	26	87	—	55	65	69
palmer1e	105	82	295	74	122	186	149	353	187	100
palmer1	33	1003	12	33	1854	30	39	28	41	21
palmer2a	68	147	102	68	392	115	197	211	169	83
palmer2b	16	22	15	16	34	61	—	77	52	49
palmer2e	99	36	278	86	52	191	345	133	171	111
palmer2	32	39	11	33	63	44	—	34	24	57
palmer3a	78	115	94	82	199	136	—	201	172	172
palmer3b	21	15	11	21	15	54	—	36	37	49
palmer3e	453	78	92	117	133	293	426	—	234	118
palmer3	11	285	17	12	553	13	11	58	32	23
palmer4a	52	77	55	52	133	109	—	93	110	42
palmer4b	21	19	10	21	31	52	—	64	35	48
palmer4e	25	30	74	24	38	123	210	123	87	65
palmer4	12	617	18	12	1182	14	11	142	35	32
palmer5a	—	—	—	—	—	—	321078	—	—	—
palmer5b	837	125	66	855	208	—	3728	961	—	—
palmer5e	3	—	—	3	—	—	25501	8	—	—
palmer6a	132	157	122	137	263	202	270	277	—	—
palmer6e	22	39	25	38	59	198	259	47	—	—
palmer7a	—	—	—	—	—	—	—	—	—	—
palmer7e	1486	—	—	—	—	—	924	39	—	—
palmer8a	49	58	35	50	102	127	103	61	—	—
palmer8e	29	27	39	29	31	92	121	86	—	—
pentagon	8	19	9	12	20	15	22	48	38	23
pfit1ls	365	369	304	566	678	480	717	453	337	1064
pfit1	365	369	304	566	678	480	717	453	337	1064
pfit2ls	178	105	125	210	184	175	237	217	115	2586
pfit2	178	105	125	210	184	175	237	217	115	2586
pfit3ls	166	188	149	139	337	289	474	272	148	3428
pfit3	166	188	149	139	337	289	474	272	148	3428
pfit4ls	84	298	288	126	548	470	754	410	286	4925
pfit4	84	298	288	126	548	470	754	410	286	4925
polak1	8	7	15	8	7	15	30	37	14	24
polak2	46	29	450	10	15	106	138	321	24	2
polak3	21	—	18	24	—	183	—	234	24	2
polak4	5	10	9	5	10	6	22	16	11	37
polak5	82	33	192	45	33	43	16	7	69	17

² full name: nuffield.continuum, which this table is too narrow to contain.

Problem	Uno presets			State-of-the-art solvers						
	filtersqp	ipopt	byrd	filterSQP	IPOPT	SNOPT	MINOS	LANCELOT	LOQO	CONOPT
polak6	43	–	47	29	301	62	108	644	27	40
powellbs	10	12	196	16	12	–	1	–	17	3
powellsq	80	–	–	4	109	78	–	23	–	2
prodpl0	9	16	12	9	16	60	42	35	24	34
prodpl1	7	17	16	7	17	59	56	27	21	5
pspdoc	9	11	9	7	15	15	25	10	12	12
recipe	2	3	4	2	3	–	1	12	10	2
rk23	7	11	19	9	12	18	28	54	12	6
robot	14	11	604	45	10	18	377	33	18	64
rosenbr	29	29	29	29	45	45	9	36	26	29
rosenmmx	36	41	24	37	22	41	66	226	17	75
s365mod	23	–	–	86	43	31	539	–	28	57
sineali	10	–	10	–	–	–	7333	–	16	24
sineval	57	66	66	62	110	94	123	75	57	77
sisser	19	21	15	19	21	15	17	38	17	19
snake	3	–	99	3	14	2	9	–	–	4
spanhyd	4	24	6	11	24	13	–	26	–	24
spiral	113	–	527	152	64	130	148	96	135	642
ssnlbeam	5	31	–	5	22	36	124	39	60	44
stancmin	2	11	3	2	11	5	5	9	20	4
swopf	5	17	–	6	17	160	100	290	21	22
synthes1	5	10	5	5	10	10	21	13	17	13
try-b	8	20	9	8	20	10	10	13	16	3
twobars	8	10	12	8	10	11	31	13	10	14
vander4	1	–	–	1	51	86	–	36	–	3
watson	22	14	71	21	14	172	117	44	18	18
weeds	38	27	4	39	32	51	–	3	29	44
womflet	42	13	–	9	12	14	40	97	12	25
yfit	47	116	49	48	185	95	130	103	45	74
yfitu	47	45	57	48	69	95	130	103	43	74
zecevic3	10	22	346	9	22	11	38	19	12	19
zecevic4	6	10	6	6	10	8	22	12	15	16
zigzag	11	27	76	11	23	23	218	43	29	74
zy2	5	12	16	5	10	9	43	9	14	4

B Combining the ingredients

Algorithm 4: Uno: trust-region filter restoration SQP.

Input: initial primal-dual iterate $(x^{(0)}, y^{(0)}, z^{(0)})$, initial trust-region radius Δ
 $k \leftarrow 0$

$(\eta(x), \omega_\rho(x), \xi(x)) \stackrel{\text{def}}{=} (\|c(x)\|_1, f(x), 0)$ *constraint relaxation*

$(\Delta\eta^{(k)}(d_x), \Delta\omega_\rho^{(k)}(d_x), \Delta\xi^{(k)}(d_x)) \stackrel{\text{def}}{=} \left(\|c^{(k)}\|_1 - \|c^{(k)}\|_1 + (\nabla c^{(k)})^T d_x \|_1, -(\nabla f^{(k)})^T d_x, 0 \right)$

$\text{phase} \leftarrow \text{Optimality}$

$\phi \stackrel{\text{def}}{=} \omega_1 + \xi$ *globalization strategy*

$\Delta\phi^{(k)} \stackrel{\text{def}}{=} \Delta\omega_1^{(k)} + \Delta\xi^{(k)}$

Initialize \mathcal{F}

repeat

Set inner iteration counter $l \leftarrow 0$ *globalization mechanism*

Reset trust-region radius $\Delta^{(l)} \in [\underline{\Delta}, \overline{\Delta}]$

repeat

if $\text{phase} = \text{Optimality}$ then *constraint relaxation*

$(d_x^{(k,l)}, d_y^{(k,l)}, d_z^{(k,l)}) \leftarrow \text{solve } QP^{(k)}(\Delta^{(l)})$

 if $QP^{(k)}(\Delta^{(l)})$ infeasible then

$\text{phase} \leftarrow \text{Restoration}$

$y^{(k)} \leftarrow 0$

$\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$ *globalization strategy*

 if $\text{phase} = \text{Restoration}$ then

$(d_x^{(k,l)}, d_y^{(k,l)}, d_z^{(k,l)}) \leftarrow \text{solve } FQP^{(k)}(\Delta^{(l)})$ starting from $d_x^{(k,l)}$

Assemble trial iterate $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)}) \stackrel{\text{def}}{=} (x^{(k)}, y^{(k)}, z^{(k)}) + (d_x^{(k,l)}, d_y^{(k,l)}, d_z^{(k,l)})$

Reset the bound multipliers corresponding to the active trust region

$\text{acceptable} \leftarrow \text{false}$

if $\|d_x^{(k,l)}\| = 0$ then *constraint relaxation*

$\text{acceptable} \leftarrow \text{true}$

else

 if $\text{phase} = \text{Restoration}$ and $QP^{(k)}(\Delta^{(l)})$ feasible and

$\eta(\hat{x}^{(k+1,l)}) < \eta_{\min}(\mathcal{F})$ then

$\text{phase} \leftarrow \text{Optimality}$

$\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$

 if $\text{phase} = \text{Restoration}$ then *globalization strategy*

 if $\Delta\eta(x^{(k)}) - \eta(\hat{x}^{(k+1,l)}) \geq \sigma\Delta\eta^{(k)}(d_x^{(k,l)})$ then

$\text{acceptable} \leftarrow \text{true}$

 else if $\hat{x}^{(k+1,l)}$ acceptable to \mathcal{F} and improves upon $x^{(k)}$ then

 if $\Delta\phi^{(k)}(d_x^{(k,l)}) \geq \delta\eta(x^{(k)})^2$ then

 if $\phi(x^{(k)}) - \phi(\hat{x}^{(k+1,l)}) \geq \sigma\Delta\phi^{(k)}(d_x^{(k,l)})$ then

$\text{acceptable} \leftarrow \text{true}$ (f-type)

 else

$\text{acceptable} \leftarrow \text{true}$ (h-type)

$\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$

 if acceptable then

 if trust region is active at $d_x^{(k,l)}$ then

 Increase radius $\Delta^{(l)}$

 else

 Decrease radius $\Delta^{(l)}$

$l \leftarrow l + 1$

until $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$ is acceptable

Update $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) \leftarrow (\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$

$k \leftarrow k + 1$

until termination criteria are satisfied

return $(x^{(k)}, y^{(k)}, z^{(k)})$

Algorithm 5: Uno: line-search ℓ_1 -merit SL_1QP .

Input: initial primal-dual iterate $(x^{(0)}, y^{(0)}, z^{(0)})$, initial penalty parameter ρ
 $k \leftarrow 0$

$$\begin{aligned} (\eta(x), \omega_\rho(x), \xi(x)) &\stackrel{\text{def}}{=} (\|c(x)\|_1, \rho f(x), 0) && \text{constraint relaxation} \\ (\Delta\eta^{(k)}(d_x), \Delta\omega_\rho^{(k)}(d_x), \Delta\xi^{(k)}(d_x)) &\stackrel{\text{def}}{=} \left(\|c^{(k)}\|_1 - \|c^{(k)}\|_1 + \right. \\ &\quad \left. (\nabla c^{(k)})^T d_x \|_1, -\rho(\nabla f^{(k)})^T d_x, 0 \right) \end{aligned}$$

$$\begin{aligned} \psi_\rho &\stackrel{\text{def}}{=} \omega_\rho + \eta + \xi && \text{globalization strategy} \\ \Delta\psi_\rho^{(k)} &\stackrel{\text{def}}{=} \Delta\omega_\rho^{(k)} + \Delta\eta^{(k)} + \Delta\xi^{(k)} \end{aligned}$$

repeat

$$\begin{aligned} (d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) &\leftarrow \text{solve } (\ell_1 \text{QP}_\rho^{(k)}) && \text{constraint relaxation} \\ \text{if } l^{(k)}(d_x^{(k)}) > 0 \text{ then} && \\ \quad (\bar{d}_x^{(k)}, \bar{d}_y^{(k)}, \bar{d}_z^{(k)}) &\leftarrow \text{solve } (\ell_1 \text{QP}_\rho^{(k)}) \text{ with } \rho = 0 && \triangleright \text{Eq. } (\ell_1 \text{QP}^{(k)}) \\ \quad \text{Decrease } \rho \text{ until the solution } (d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) \text{ to } (\ell_1 \text{QP}_\rho^{(k)}) \text{ satisfies} && \\ \quad \quad \begin{cases} l^{(k)}(d_x^{(k)}) = 0 & \text{if } l^{(k)}(\bar{d}_x^{(k)}) = 0 \\ l^{(k)}(0) - l^{(k)}(d_x^{(k)}) \geq \varepsilon_1 (l^{(k)}(0) - l^{(k)}(\bar{d}_x^{(k)})) & \text{otherwise} \end{cases} && \\ \quad \text{Further decrease } \rho \text{ until the solution } (d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) \text{ to } (\ell_1 \text{QP}_\rho^{(k)}) \text{ satisfies} && \\ \quad \quad \Delta\psi_\rho^{(k)}(d_x^{(k)}) \geq \varepsilon_2 \Delta\psi_0^{(k)}(\bar{d}_x^{(k)}) && \\ \quad \quad \rho \leftarrow \min \left(\rho, \left(\frac{E_0^{(k)}(x^{(k)}, y^{(k)} + \bar{d}_y^{(k)})}{\max(1, \|c(x^{(k)})\|_1)} \right)^2 \right) && \\ \quad \text{if } \rho \text{ has decreased then} && \\ \quad \quad (d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) &\leftarrow \text{solve } (\ell_1 \text{QP}_\rho^{(k)}) \end{aligned}$$

$$\alpha^{(0)} \leftarrow 1$$

Set inner iteration counter $l \leftarrow 0$

repeat

$$\begin{aligned} \text{Assemble trial iterate } (\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)}) &\stackrel{\text{def}}{=} (x^{(k)}, y^{(k)}, z^{(k)}) + \\ &\quad (\alpha^{(l)} d_x^{(k)}, \alpha^{(l)} d_y^{(k)}, \alpha^{(l)} d_z^{(k)}) \\ \text{acceptable} &\leftarrow \text{false} \\ \text{if } \|d_x^{(k,l)}\| = 0 \text{ then} && \text{constraint relaxation} \\ \quad \text{acceptable} &\leftarrow \text{true} \\ \text{else} && \\ \quad \text{if } \psi_\rho(x^{(k)}) - \psi_\rho(\hat{x}^{(k+1,l)}) \geq \sigma \Delta\psi_\rho^{(k)}(\alpha^{(l)} d_x^{(k)}) && \text{globalization strategy} \\ \quad \quad \text{acceptable} &\leftarrow \text{true} \end{aligned}$$

$$\begin{aligned} \text{if not acceptable then} && \\ \quad \text{Decrease step length } \alpha^{(l)} && \\ \quad l &\leftarrow l + 1 \end{aligned}$$

until $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$ is acceptable

$$\text{Update } (x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) \leftarrow (\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$$

$$k \leftarrow k + 1$$

until termination criteria are satisfied

return $(x^{(k)}, y^{(k)}, z^{(k)})$

Algorithm 6: Uno: line-search filter restoration interior-point method.

Input: initial primal-dual iterate $(x^{(0)}, y^{(0)}, z^{(0)})$, initial barrier parameter $\mu > 0$
 $k \leftarrow 0$

$(\eta(x), \omega_\rho(x), \xi(x)) \stackrel{\text{def}}{=} (\|c(x)\|_1, f(x), -\mu \log(X)e)$ *constraint relaxation*
 $(\Delta\eta^{(k)}(d_x), \Delta\omega_\rho^{(k)}(d_x), \Delta\xi^{(k)}(d_x)) \stackrel{\text{def}}{=} (\|c^{(k)}\|_1 - \|c^{(k)}\|_1 + (\nabla c^{(k)})^T d_x, -(\nabla f^{(k)})^T d_x, \mu(X^{(k)})^{-1} e^T d_x)$
 $phase \leftarrow \text{Optimality}$

$\phi \stackrel{\text{def}}{=} \omega_1 + \xi$ *globalization strategy*
 $\Delta\phi^{(k)} \stackrel{\text{def}}{=} \Delta\omega_1^{(k)} + \Delta\xi^{(k)}$
Initialize \mathcal{F}

repeat

Possibly update the barrier parameter μ *subproblem*

$(d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) \leftarrow \text{solve } \begin{cases} IPSP_\mu^{(k)} & \text{if } phase = \text{Optimality} \\ FIPSP_\mu^{(k)} & \text{if } phase = \text{Restoration} \end{cases}$

Scale primal-dual direction according to (Eq. 8) *subproblem*

$\alpha^{(0)} \leftarrow 1$
Set inner iteration counter $l \leftarrow 0$

repeat

Assemble trial iterate $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)}) \stackrel{\text{def}}{=} (x^{(k)}, y^{(k)}, z^{(k)}) + (\alpha^{(l)} d_x^{(k)}, \alpha^{(l)} d_y^{(k)}, \alpha^{(l)} d_z^{(k)})$
 $acceptable \leftarrow false$

if $\|d_x^{(k)}\| = 0$ **then** *constraint relaxation*
 $acceptable \leftarrow true$
else

if $phase = \text{Restoration}$ **then** *globalization strategy*
if $\Delta\eta(x^{(k)}) - \eta(\hat{x}^{(k+1,l)}) \geq \sigma \Delta\eta^{(k)}(d_x^{(k,l)})$ **then**
 $acceptable \leftarrow true$
else if $\hat{x}^{(k+1,l)}$ *acceptable to* \mathcal{F} **then**
if $\eta(x^{(k)}) \leq \theta_{min}$ and $0 < \Delta\phi^{(k)}(\alpha^{(l)} d_x^{(k)})$ and $\Delta\phi^{(k)}(\alpha^{(l)} d_x^{(k)}) \geq \delta\eta(x^{(k)})^2$ **then**
if $\phi(x^{(k)}) - \phi(\hat{x}^{(k+1,l)}) \geq \sigma \Delta\phi^{(k)}(\alpha^{(l)} d_x^{(k)})$ **then**
 $acceptable \leftarrow true$
else
 $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$
else if $\hat{x}^{(k+1,l)}$ *improves upon* $x^{(k)}$ **then**
 $acceptable \leftarrow true$
if $\neg (0 < \Delta\phi^{(k)}(\alpha^{(l)} d_x^{(k)}) \text{ and } \Delta\phi^{(k)}(\alpha^{(l)} d_x^{(k)}) \geq \delta\eta(x^{(k)})^2)$ **then**
 $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$

if $acceptable$ and $phase = \text{Restoration}$ and $\hat{x}^{(k+1,l)}$ *acceptable to* \mathcal{F} and $\eta(\hat{x}^{(k+1,l)}) \leq \kappa\eta(x^{(k)})$ **then**
 $phase \leftarrow \text{Optimality}$
 $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$

if not acceptable then
Decrease step length $\alpha^{(l)}$
 $l \leftarrow l + 1$
if $\alpha^{(l)}$ *too small* **then**
 $phase \leftarrow \text{Restoration}$ *constraint relaxation*
 $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\eta(x^{(k)}), \phi(x^{(k)}))\}$ *globalization strategy*
 $(d_x^{(k)}, d_y^{(k)}, d_z^{(k)}) \leftarrow \text{solve } FIPSP_\mu^{(k)}$
Scale primal-dual direction according to (Eq. 8) *subproblem*

until $(\hat{x}^{(k+1,l)}, \hat{y}^{(k+1,l)}, \hat{z}^{(k+1,l)})$ *is acceptable*
Update $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) \leftarrow (\hat{x}^{(k+1)}, \hat{y}^{(k+1)}, \hat{z}^{(k+1)})$
 $k \leftarrow k + 1$

until *termination criteria are satisfied*
return $(x^{(k)}, y^{(k)}, z^{(k)})$